

Lecture

# 2D & 3D Manipulations & Operations

Amit Zoran

## Advanced Topics in Digital Design

67682

The Rachel and Selim Benin School of Computer Science and Engineering  
The Hebrew University of Jerusalem, Israel

# Outline

- 2.5D & Extrude
- Parametric Curves & Surfaces
- Point Collection & Vector Field
- Mapping & Recursion
- The Loft Challenge

# **2.5D & Extrude**

## 2.5D

Two and a half dimensional (shortened to 2.5D, nicknamed three-quarter perspective and pseudo-3D) is a term used to describe either 2D graphical projections and similar techniques used to cause images or scenes to simulate the appearance of being three-dimensional (3D) when in fact they are not...  
(wikipedia)



# Extrude curves and surfaces

Extrudes use a base curve or surface to make an open or closed shape. Extrusions can be polysurfaces or lightweight extrusion objects.

Light-weight extrusion objects use only a profile curve and a length as input instead of the network of isocurves normally needed for NURBS objects. The Box, Cylinder, Tube, ExtrudeCrv, and ExtrudeSrf commands create extrusion objects. Extrusion objects can be closed with a planar cap or open. These objects will be converted to polysurfaces by some commands if necessary to add additional information for editing. Lightweight extrusion objects use less memory, mesh faster, and save smaller than the traditional polysurfaces.

(mcneel)

## 2.5D Fabrication

- Laser cutting & engraving
- 2.5D CNC milling



# **Parametric Curves & Surfaces**

# Parametric equation

In mathematics, parametric equations define a group of quantities as functions of one or more independent variables called parameters.

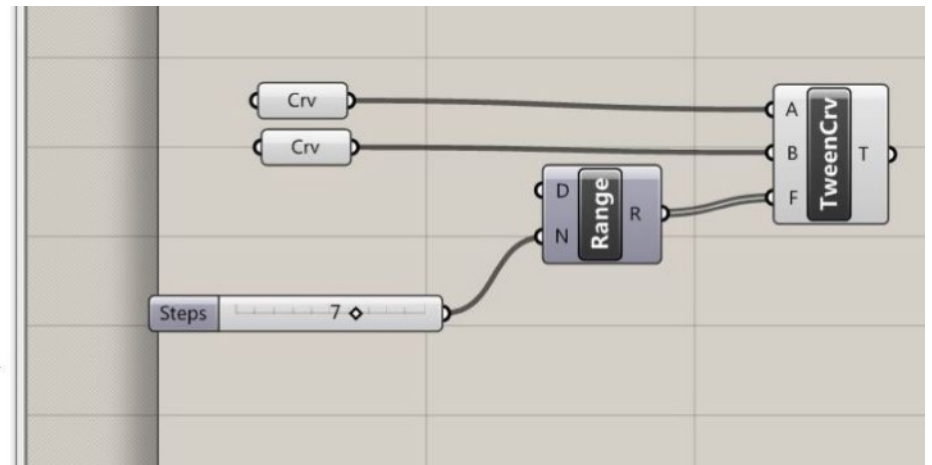
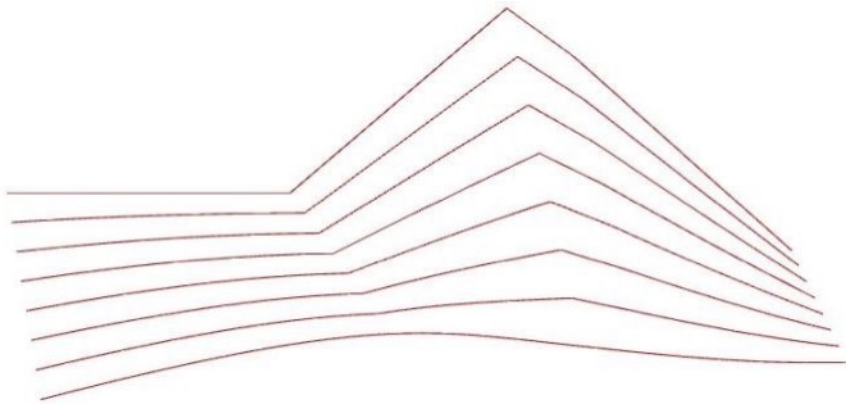
Weisstein, Eric W. "Parametric Equations". MathWorld.

# Parametric curves

The fundamental constructor for many kinds of curves employs the concept of linear interpolation or tweening. Parametric curves results from linearly interpolating a parameter in an equation to generate the points on the curve. In a parametric line, the point and the parameter have a direct relationship: equal increments between parameter values produce corresponding equal increments between points on the line. In curves, this relation becomes indirect. Identical parameter changes can yield unequal spacing between points - the implications deeply affect the form-making process.

# Parametric curves

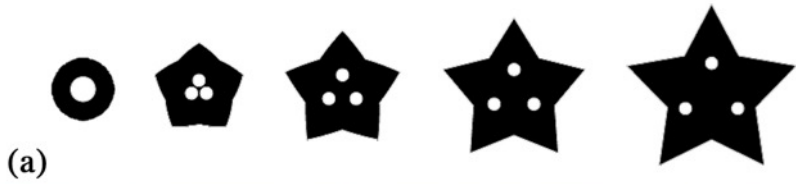
- Tweening & offsetting



<http://www.grasshopper3d.com/forum/topics/tween-curves>

# Parametric curves

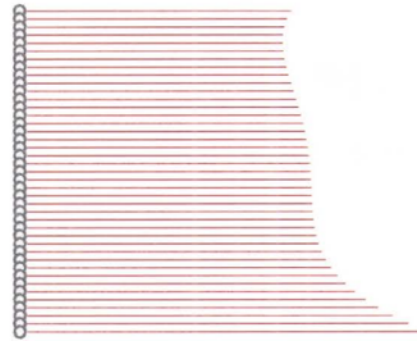
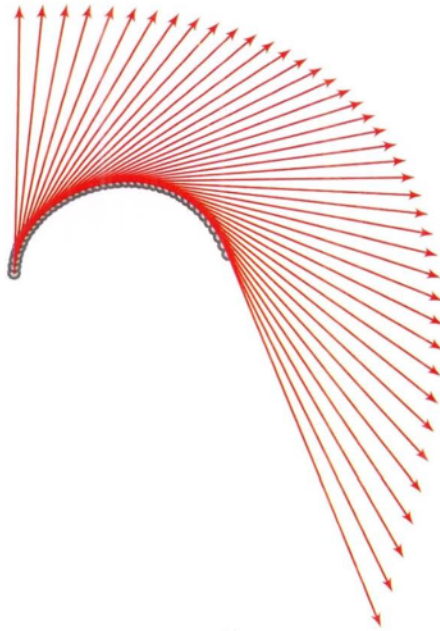
- Tweening & offsetting



# Parametric curves

- Tweening & offsetting
- Tangent vector

The tangent vector is the first derivative of the parametric curve at the given point. The tangent vectors vary in length along the curve.

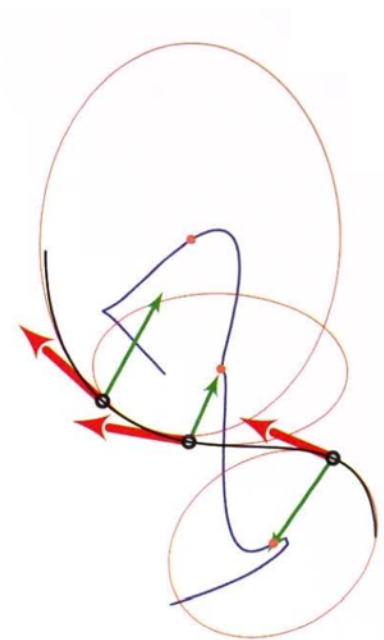




# Parametric curves

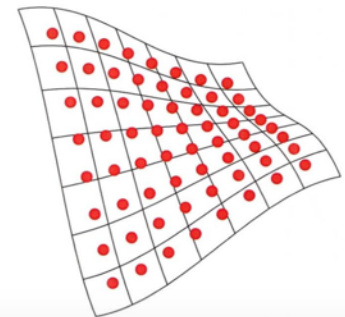
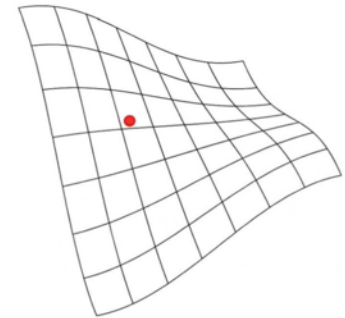
- Tweening & offsetting
- Tangent vector
- Normal vector

The *normal vector* of the curve at  $p(t)$  is of unit length and lies in what is called the *osculating plane* of the curve at  $p(t)$ . Its direction is approximated by the second derivative. This is the plane that most closely approximates the curve at  $p(t)$ . Lying in this plane is the *osculating circle*, which is the circle that is both tangent to and has the same curvature as the curve at  $p(t)$ . The centre points of the osculating circles at each point along the curve define another curve called the *evolute*.



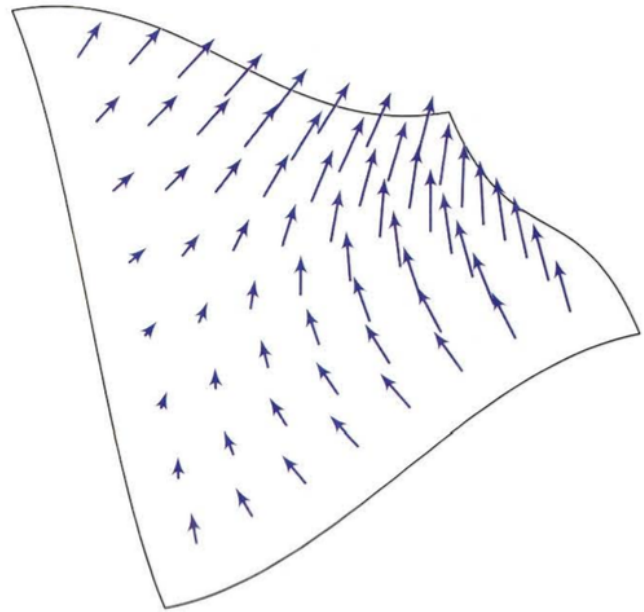
# Parametric surfaces

Parametric surfaces and curves share mathematical structure. Surfaces are more complex than curves so, naturally, their representation must be more involved. Parametric surfaces comprise a point  $p(u, v)$  that moves along the surface as the parameters  $u$  and  $v$  change.



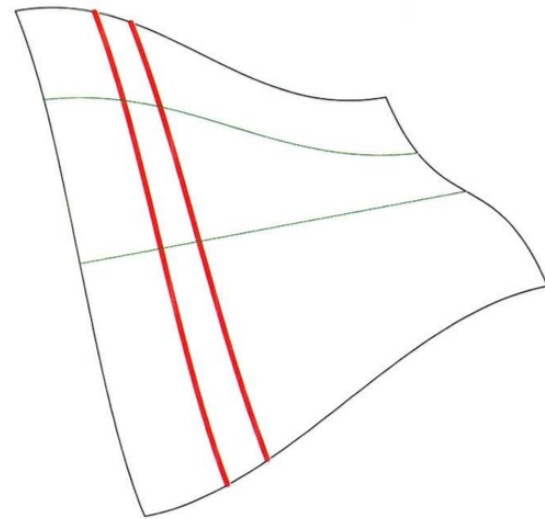
# Parametric surfaces

With exceptions similar (but more complex) to those for curves, every point on a surface has a unique unit surface normal.



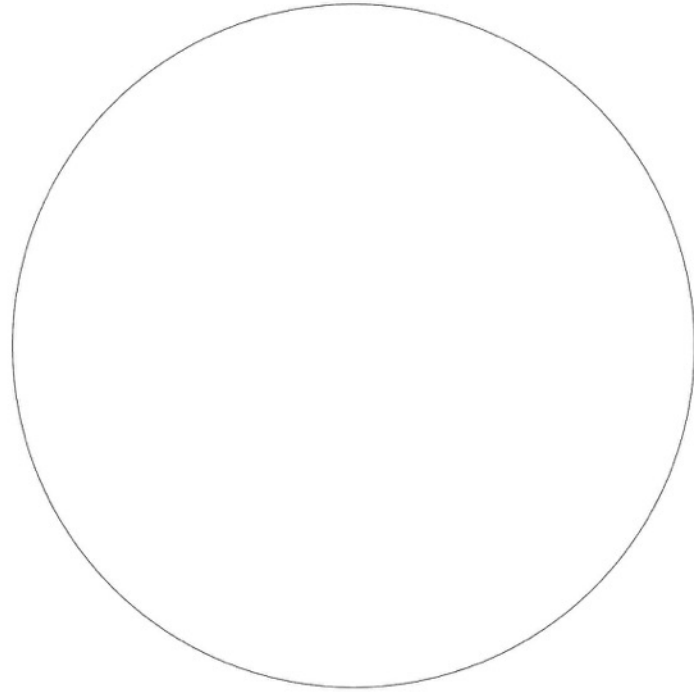
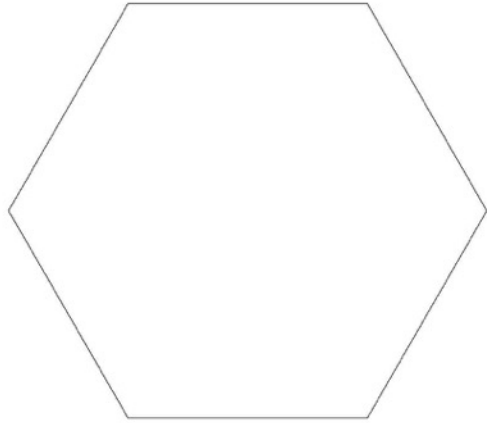
# Parametric surfaces

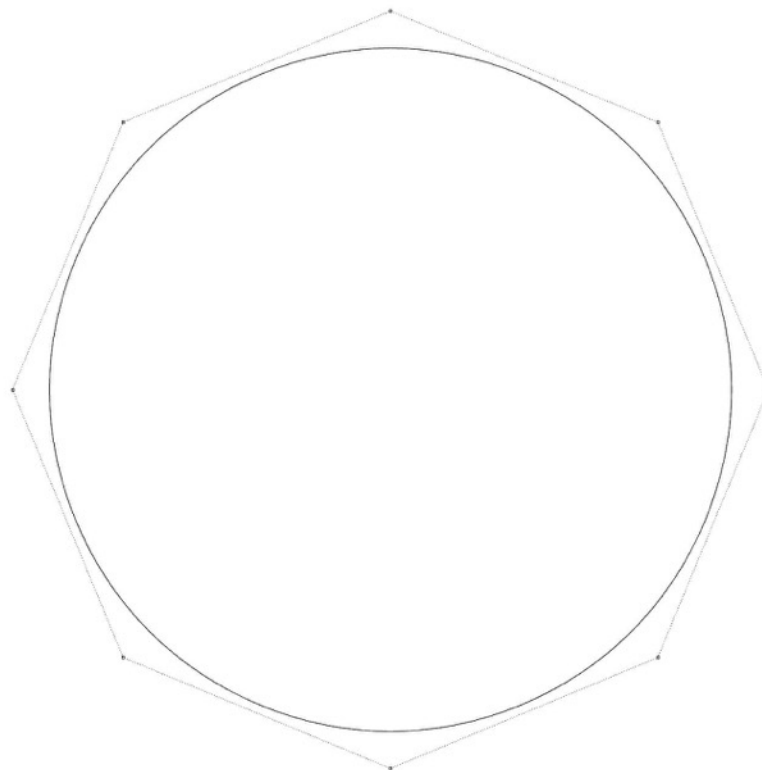
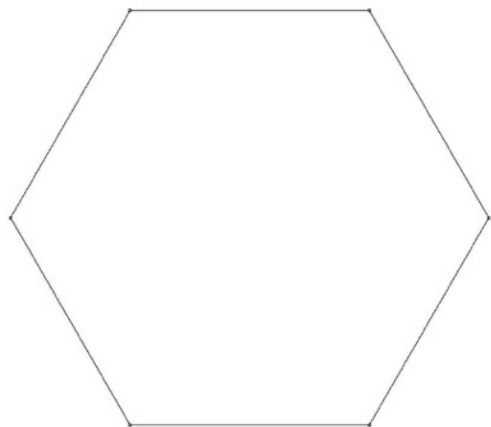
Lines in  $u$  and  $v$  parameter space map to curves on the surface. When either  $u$  or  $v$  is held constant, the line in parameter space is parallel to the parameter axes. The square  $uv$  parameter space is mapped to the surface, stretching like a rubber sheet in the process. The curve in geometry space stretches with the sheet, so lies on the surface in rough proportion to its position in parameter space. Such curves, where one of  $u$  or  $v$  is held constant, are called *isocurves*.

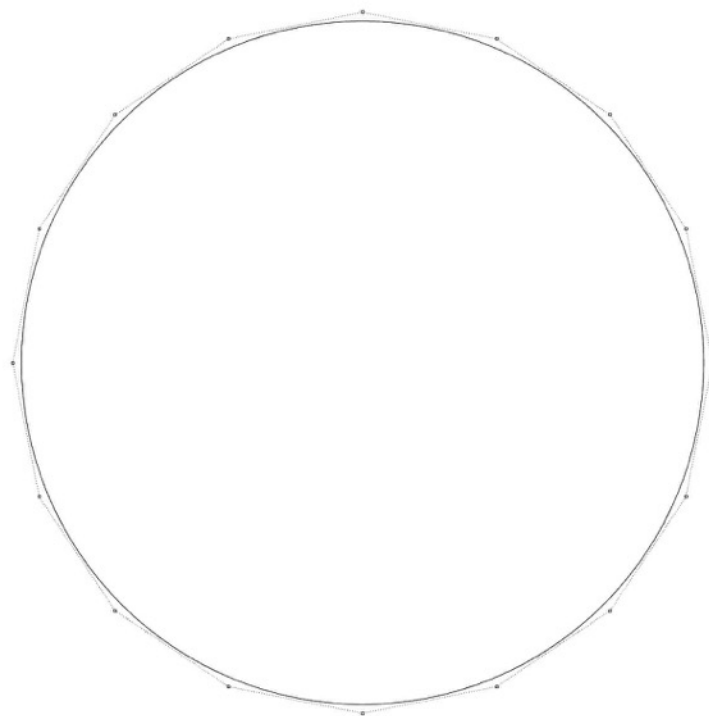
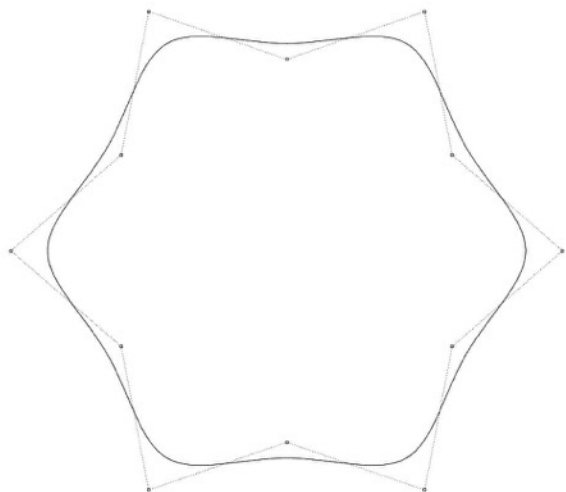


# From curves to surfaces

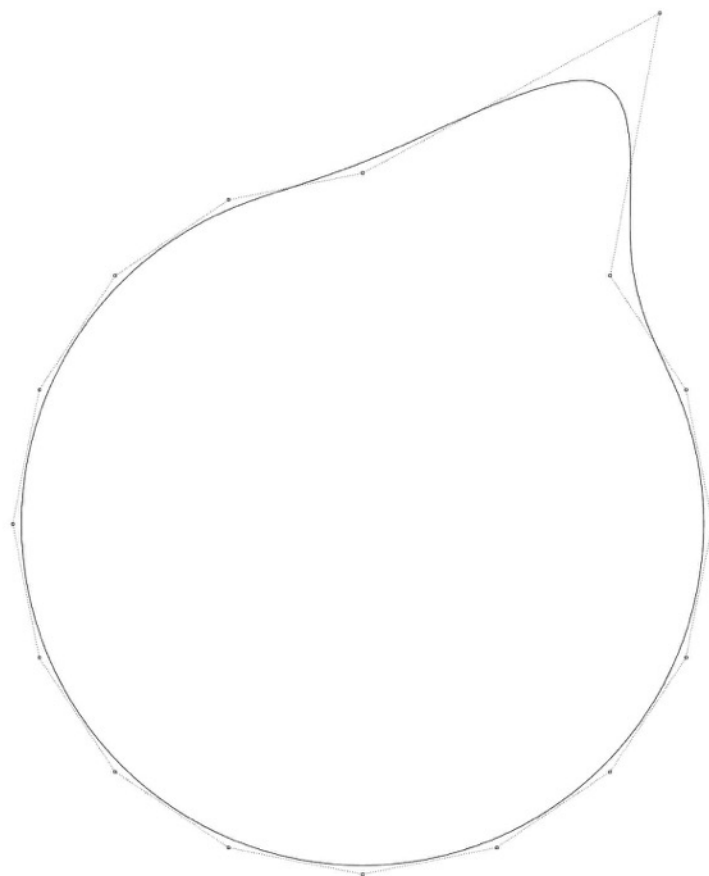
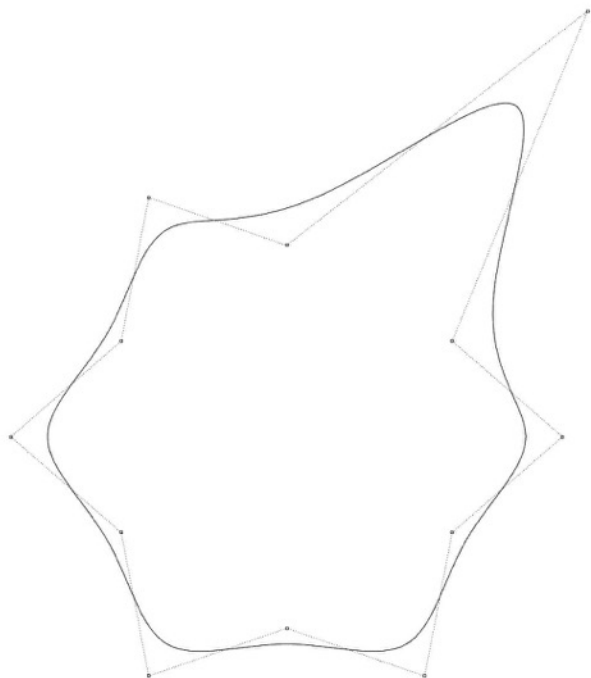
- Extrude
- Revolve
- Rail
- Patch, Plane, Edge, etc.
- (Rebuild)

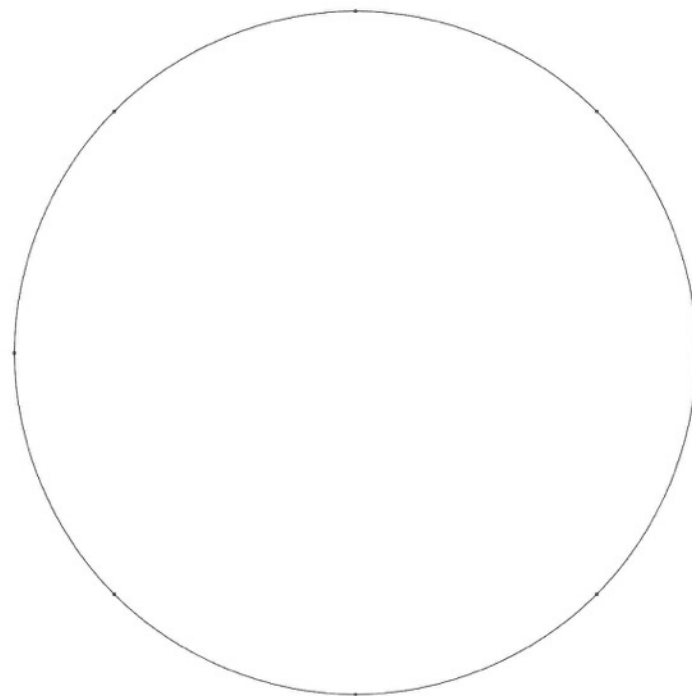
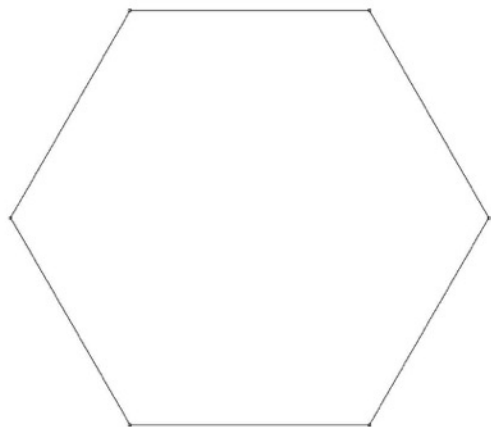


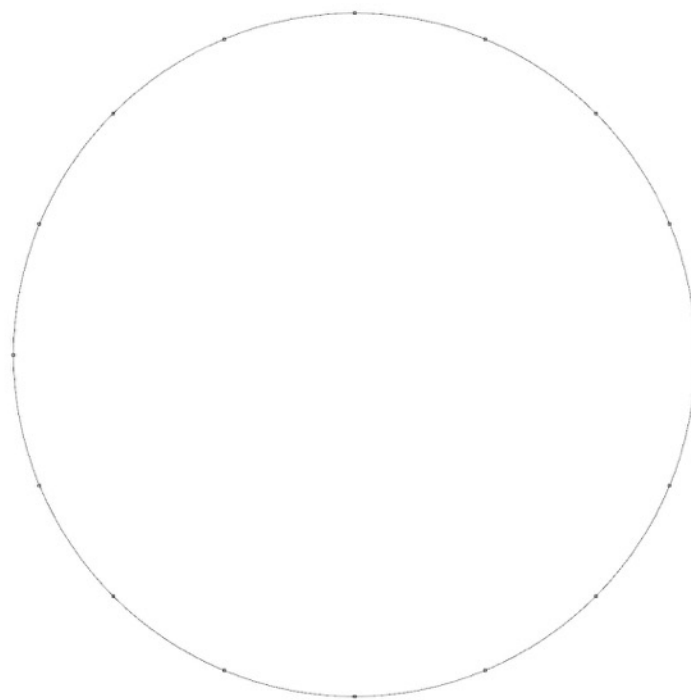
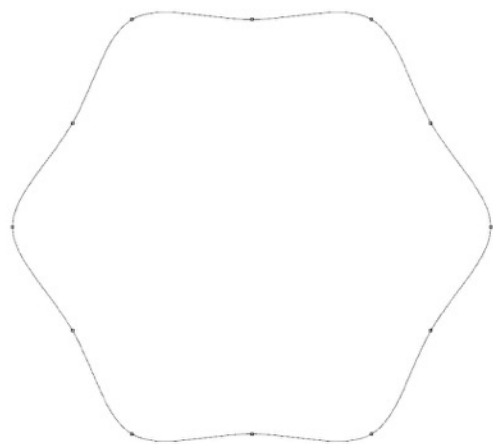


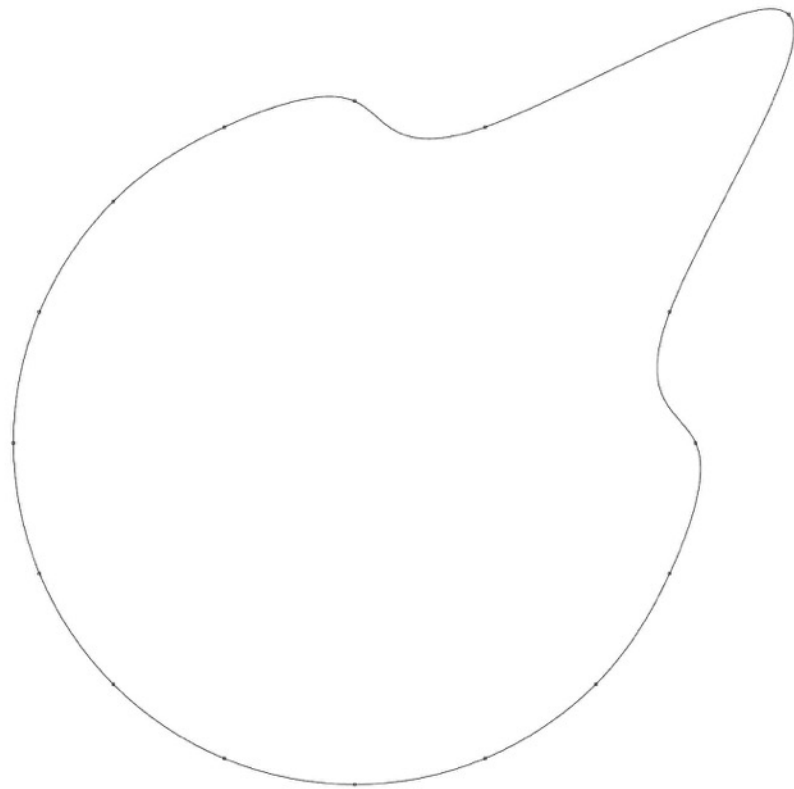
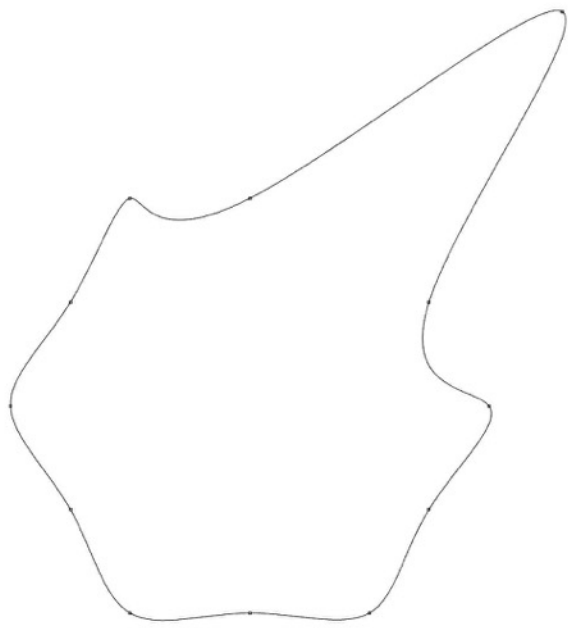


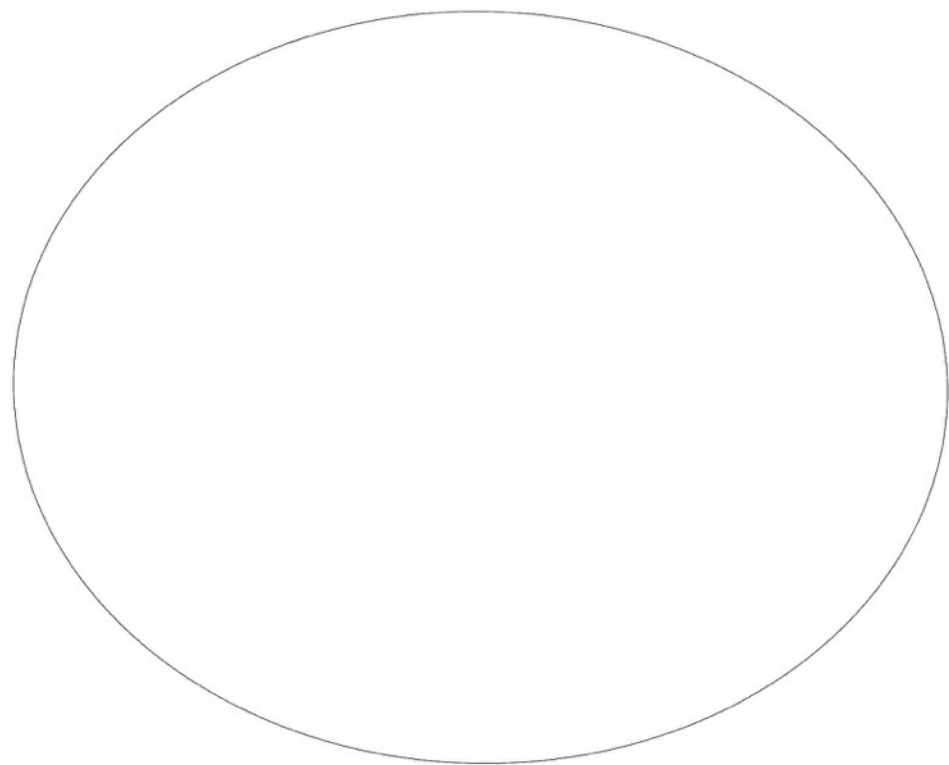


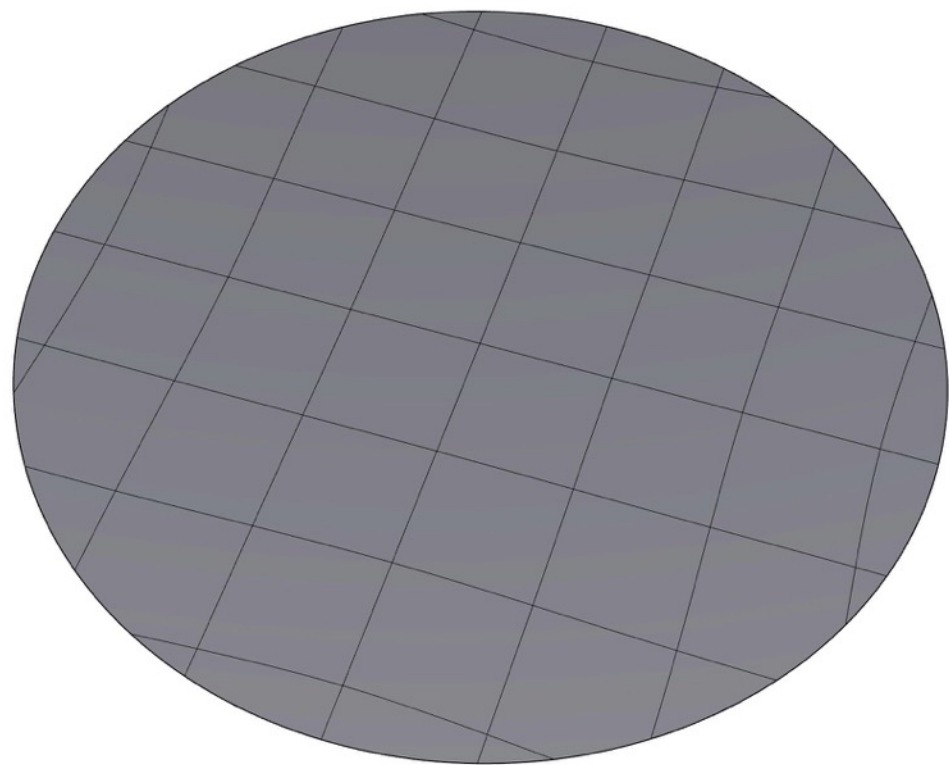


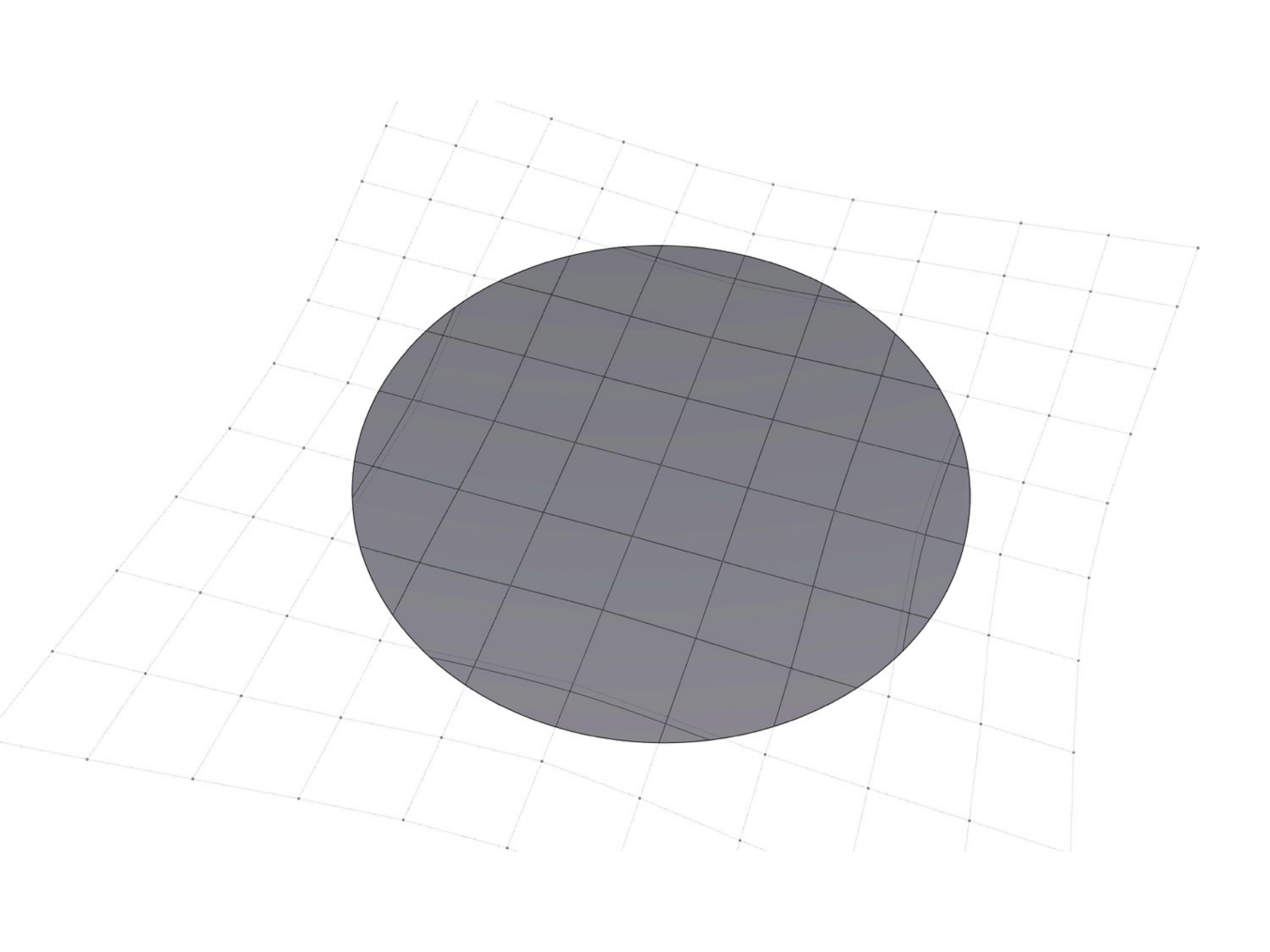


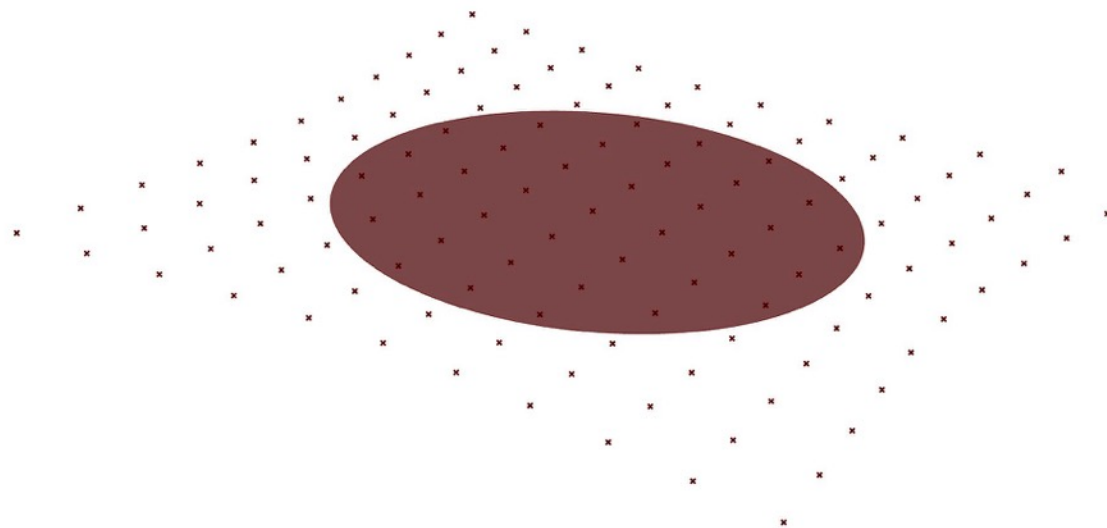




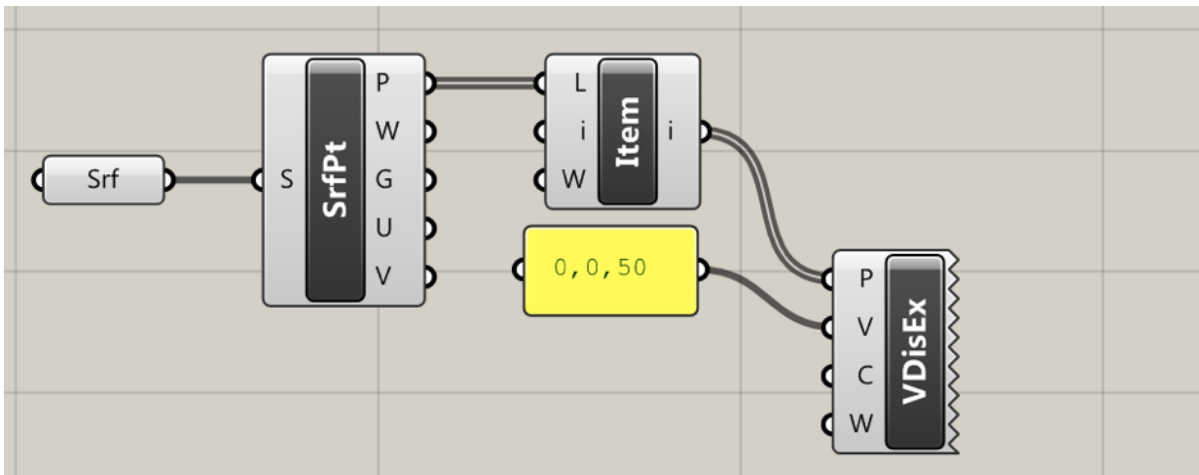
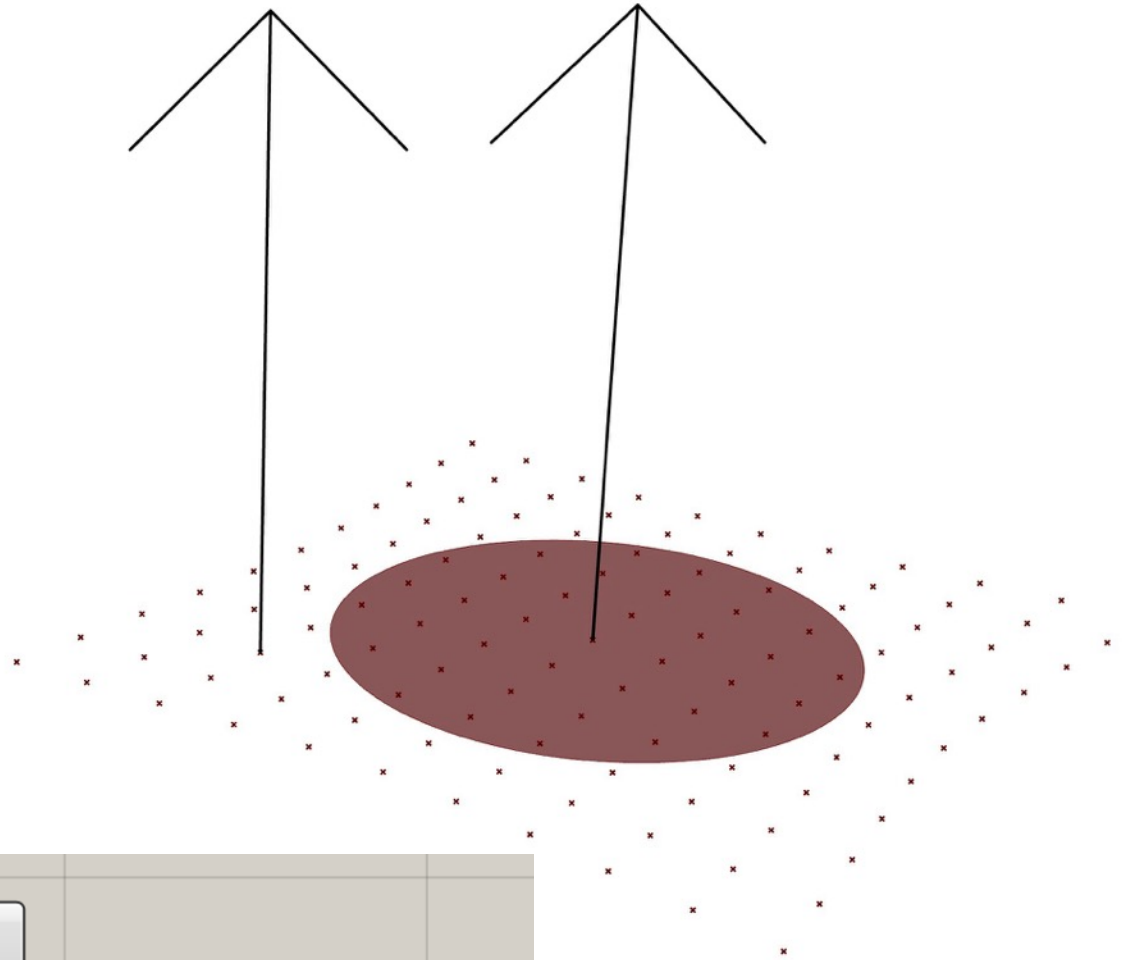


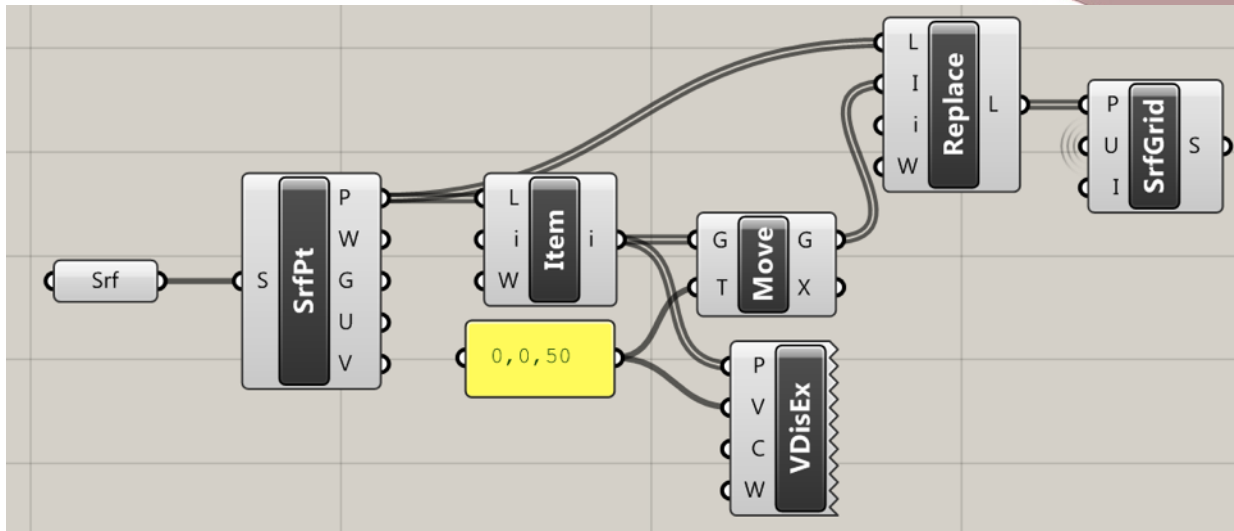
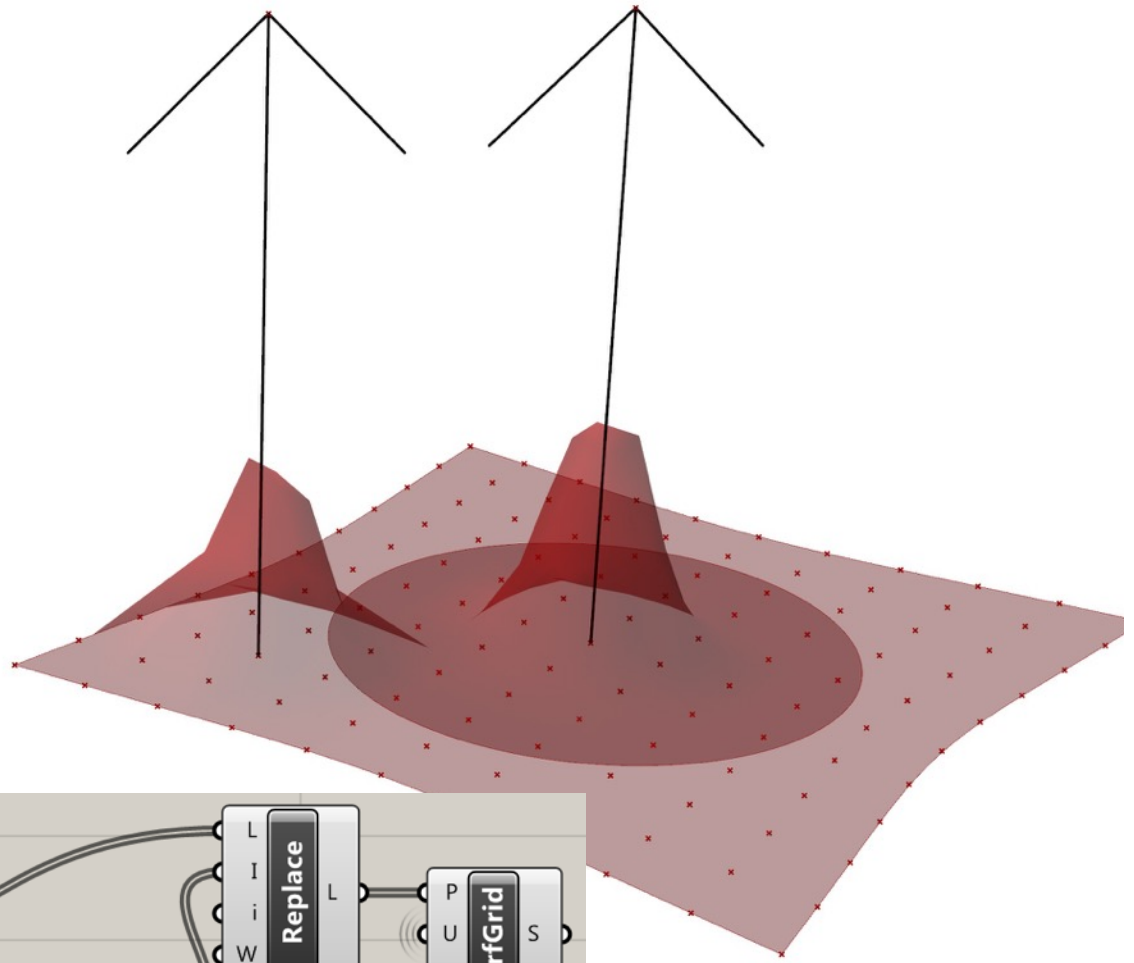












# **Point Collection & Vector Field**

# Point collection

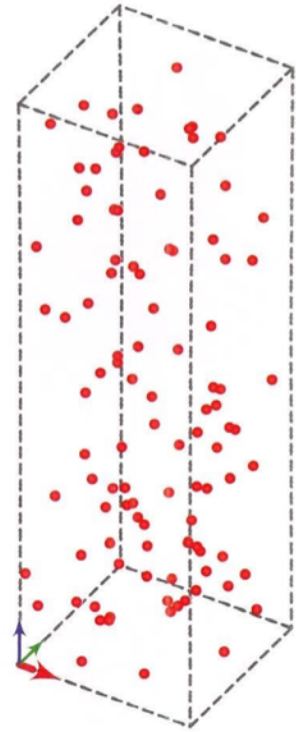
Point-like objects may be located in Euclidean space or parametric space, so a collection can be specified in either space. Euclidean space can be represented through Cartesian, cylindrical or spherical coordinates. Most curves and surfaces (those defined internally by parametric equations) define a moving frame that gives locations on the curve or surface. Unlike those of Cartesian space, these parametric formulations may not preserve constant distance, either geometrically or along the defining object.

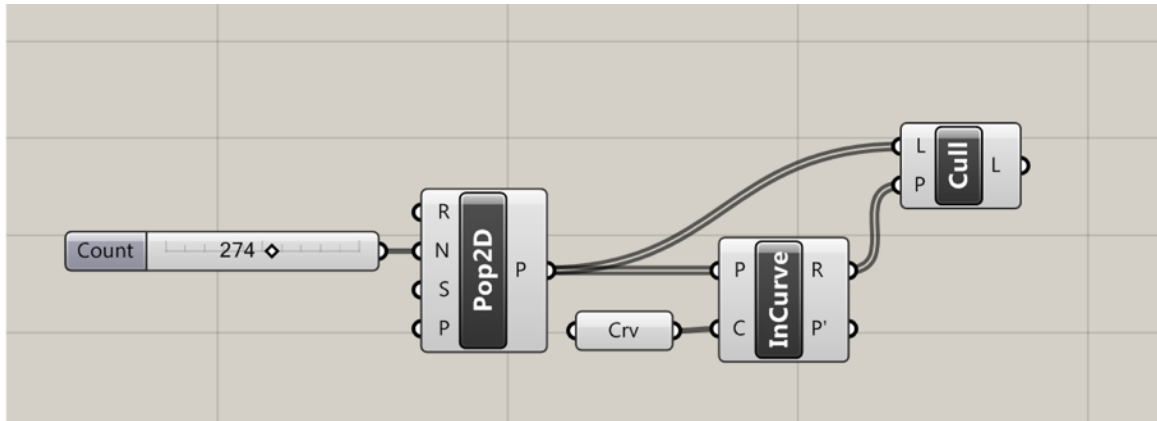
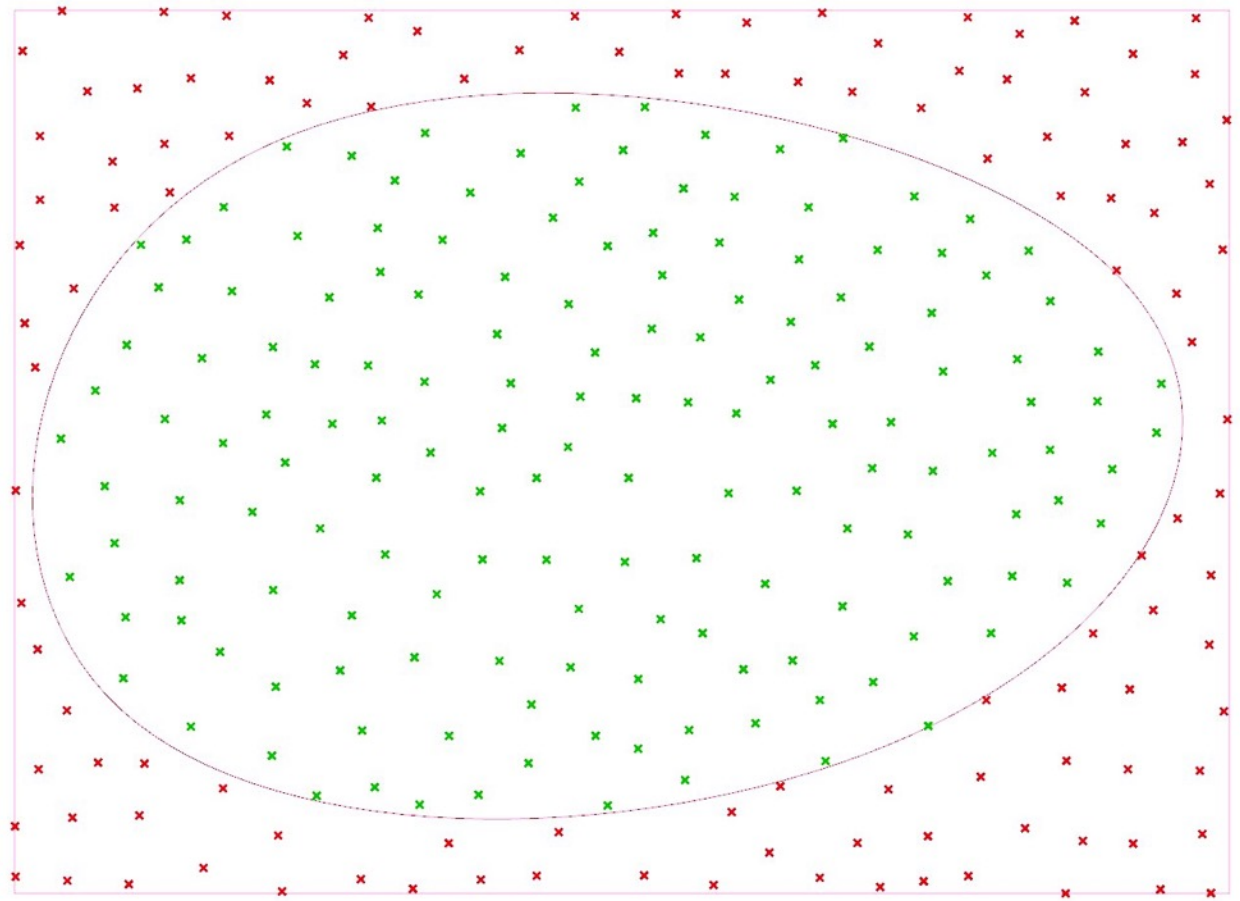
Use a collection of point-like objects as the input to define repeating elements. The logical structure of a collection is important- it provides the relationships through which points can be used to define objects.



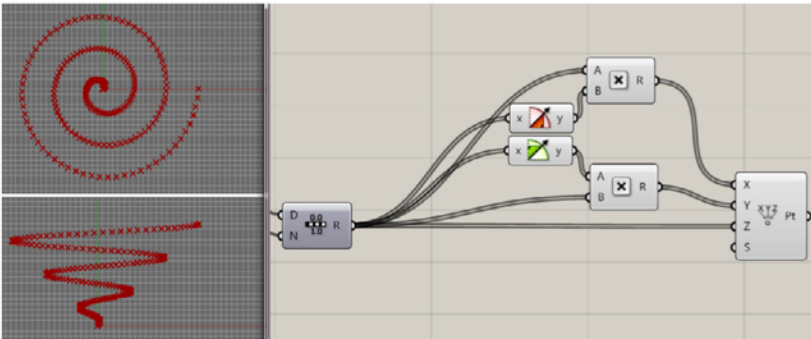
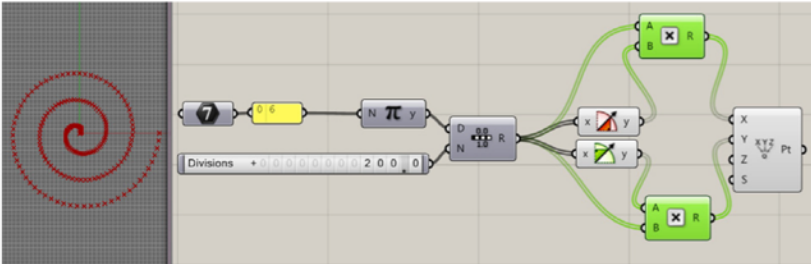
# Point cloud

The geometric and symbolic structures of collections need not be the same. Here, the symbolic structure is a sequence and there is no geometric structure, just randomness. This sample places uniformly-distributed random points within a rectangular bounding box. Its parameterization gives count, the number of points. In this case, the range of the function is a rigid body transformation of the domain. This means that the uniform distribution defined in the domain will persist into the range.

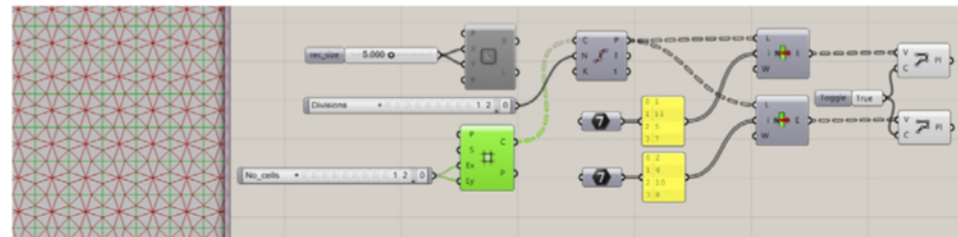
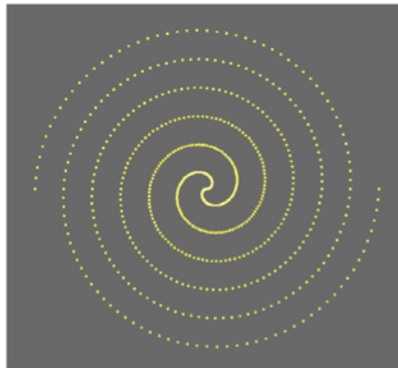




From Generative Algorithms | Zubin Khabazi



There are various types of spirals like Fermat's spiral, Cornu spiral, Hyperbolic spiral, Lituus spiral and so on. All can be formulated and implemented in functions to draw with Grasshopper. Playing around math functions could be endless. You can find various mathematical resources to match your data sets with them. Here the idea of math operations and functions which can change the original set of data is important for further design purposes.

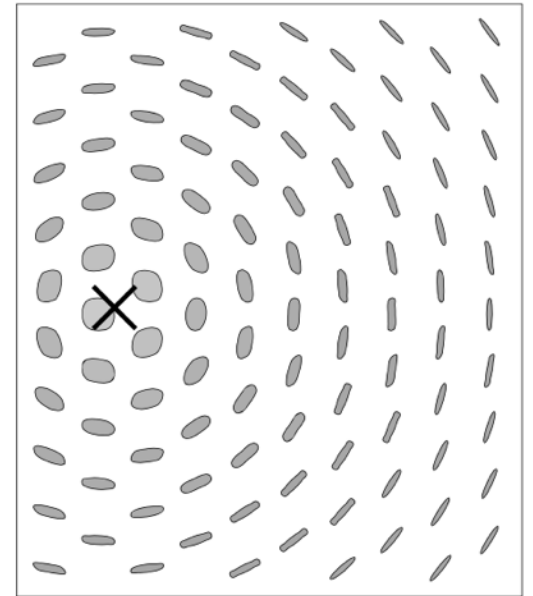
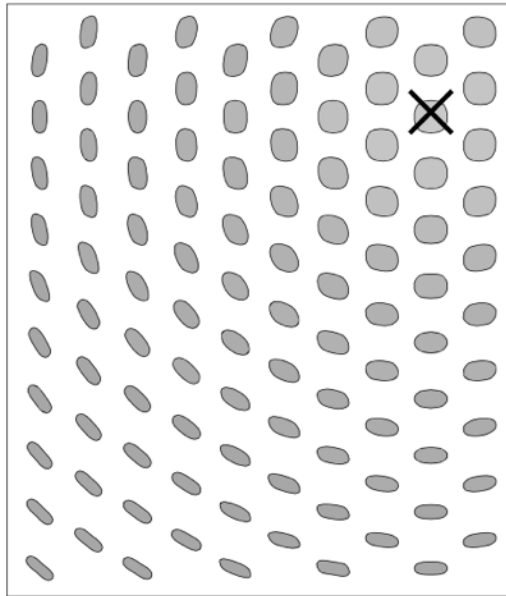
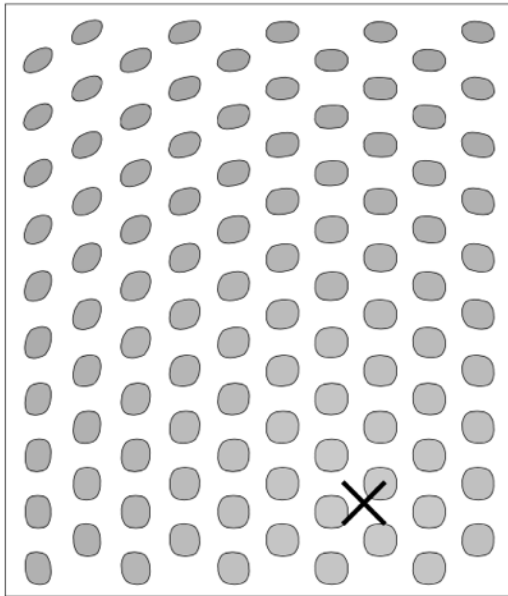


# Attractors

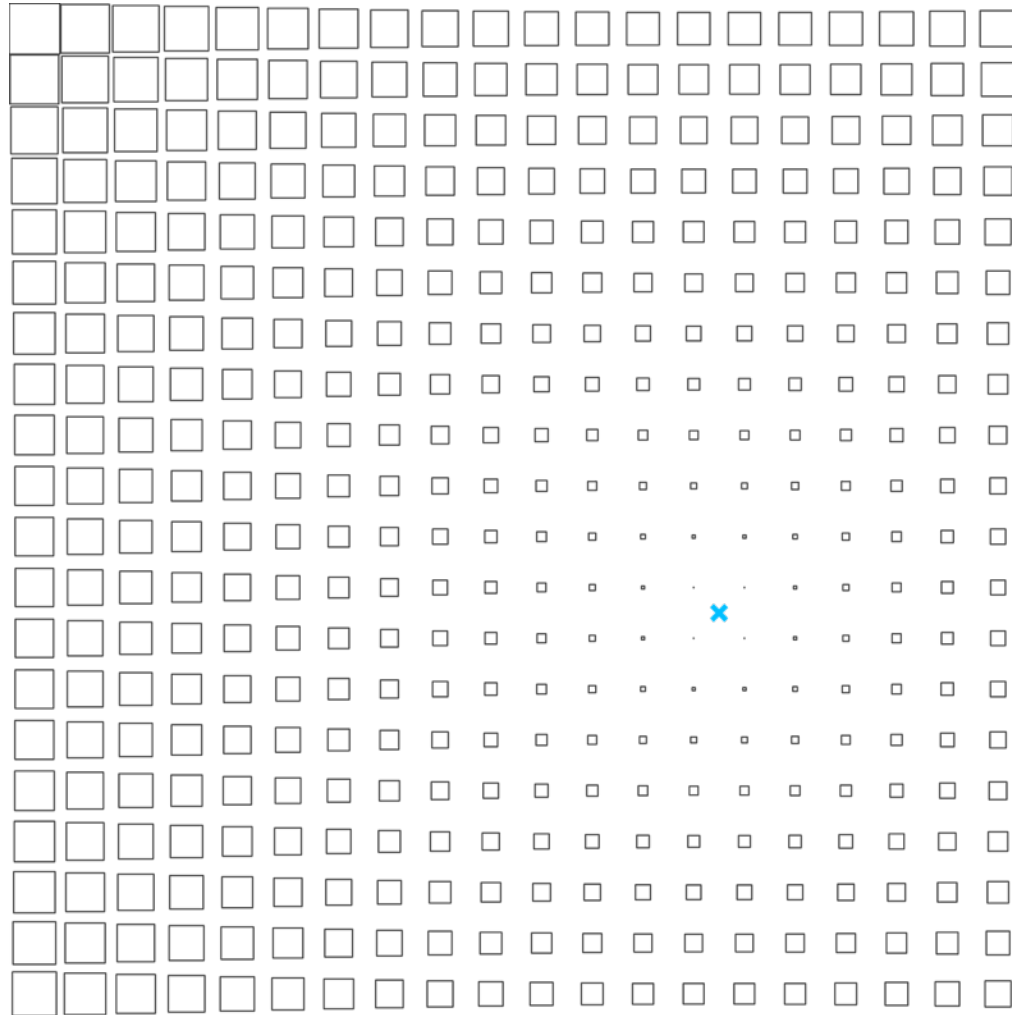
Attractors are points that act like virtual magnets - either attracting or repelling other objects. In Grasshopper, any geometry referenced from Rhino or created within Grasshopper can be used as an attractor. Attractors can influence any number of parameters of surrounding objects including scale, rotation, color, and position. These parameters are changed based on their relationship to the attractor geometry.



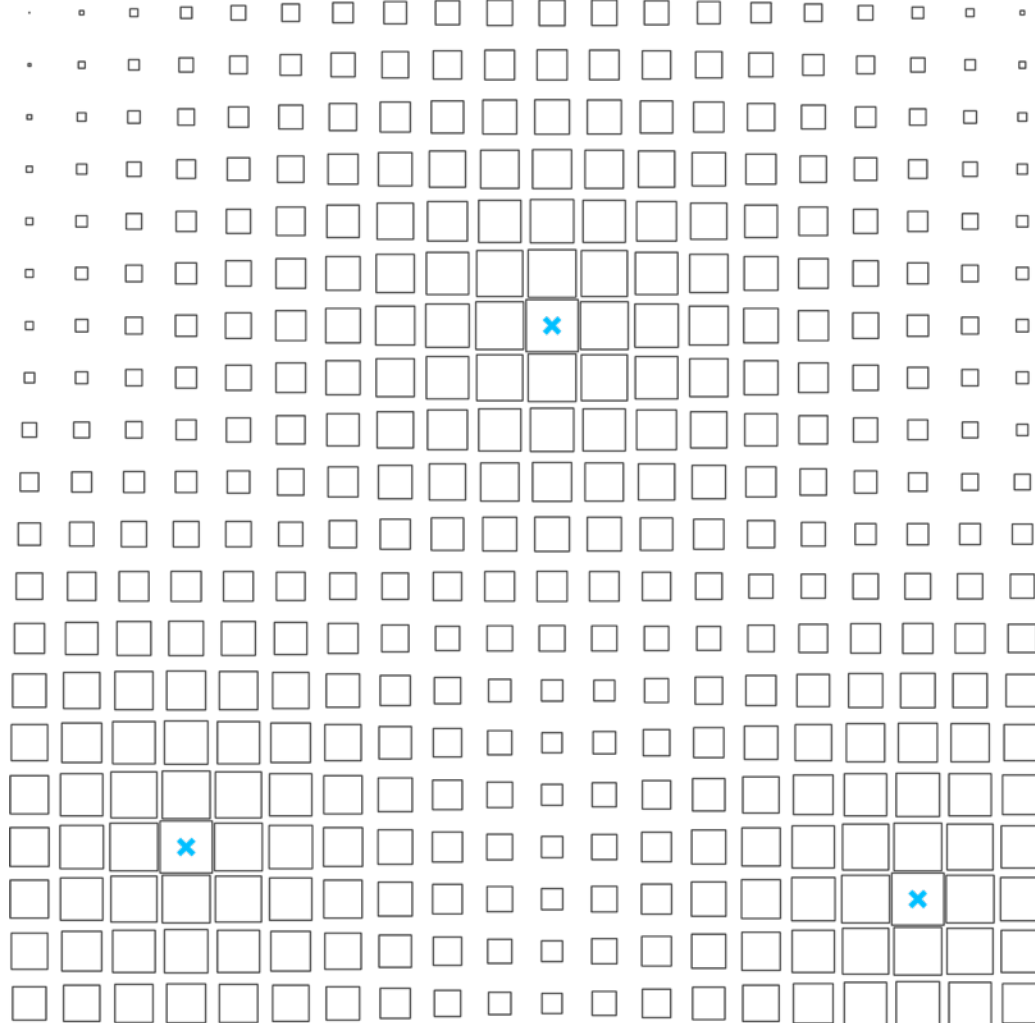
# Attractors



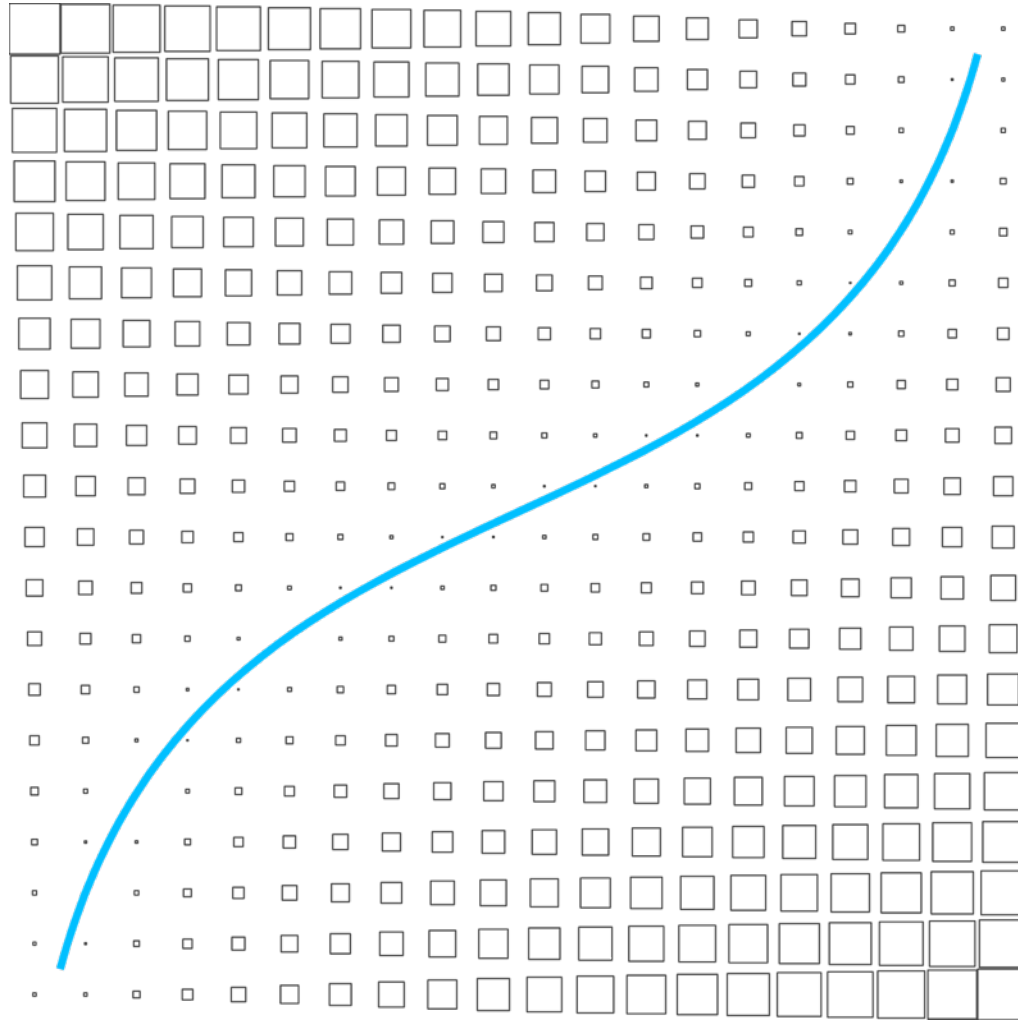
# Attractors



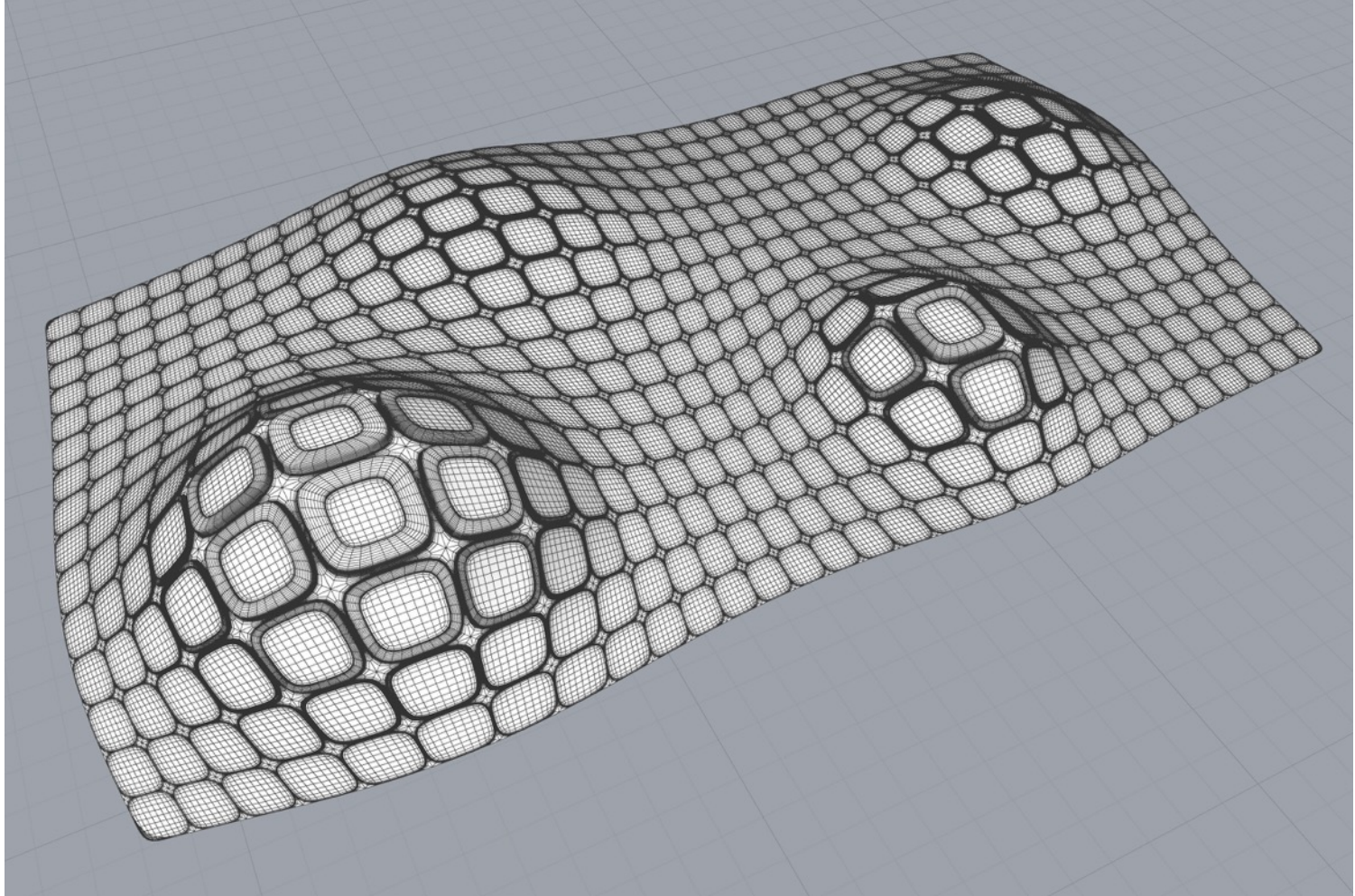
# Attractors



# Attractors



# Attractors



<https://gozourworkshops.wordpress.com/2014/07/09/grasshopper-attractor-essentials/>

# Attractors



<https://s-media-cache-ak0.pinimg.com/originals/b3/5d/f1/b35df131038ca02a2062991136a91b12.jpg>









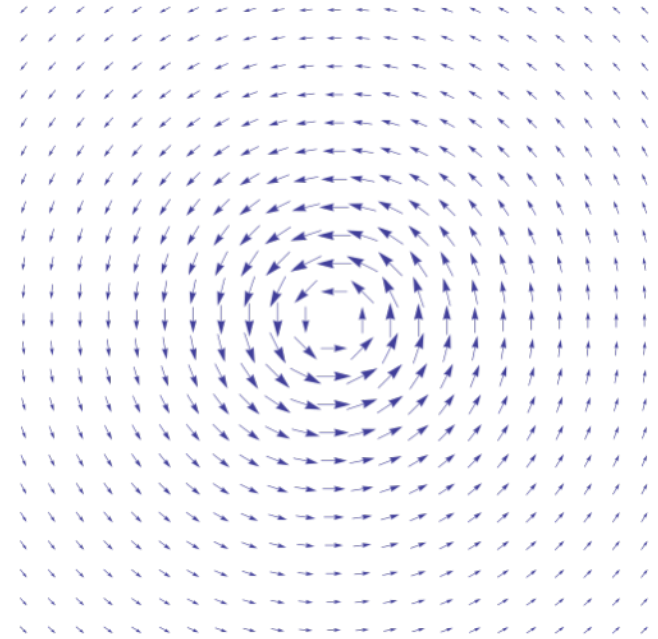




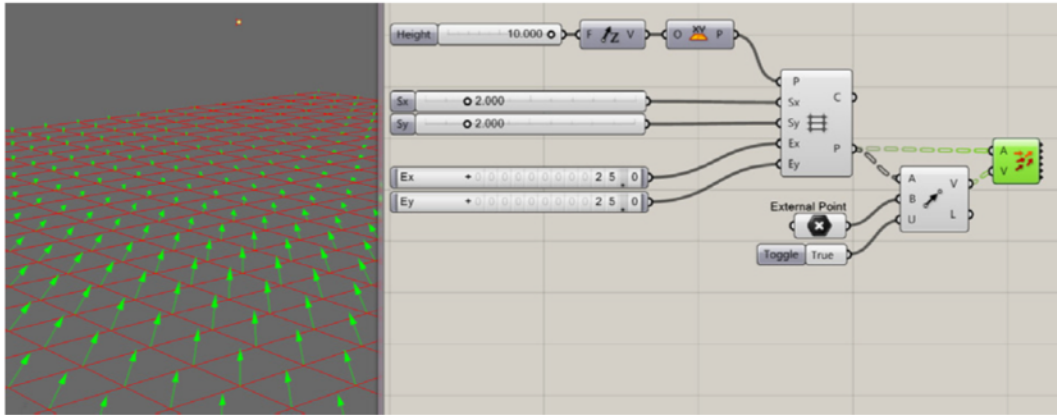
One Ocean, Thematic Pavilion in Yeosu, South Korea for Expo 2012 by SOMA

# Vector field

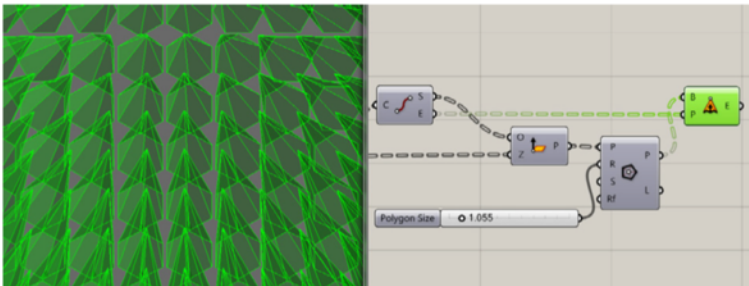
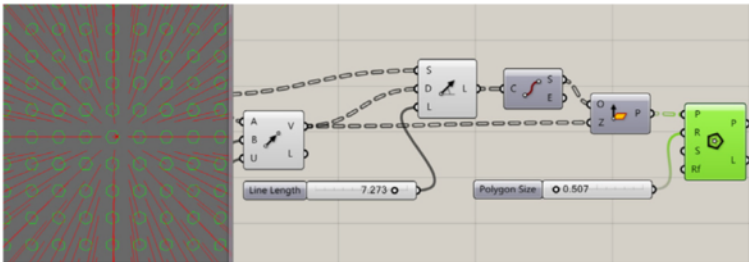
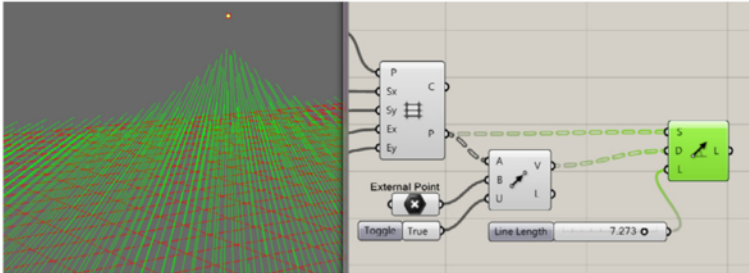
A vector field is an assignment of a vector to each point in a subset of space. A vector field in the plane (for instance), can be visualised as: a collection of arrows with a given magnitude and direction, each attached to a point in the plane. Vector fields are often used to model, for example, the speed and direction of a moving fluid throughout space, or the strength and direction of some force, such as the magnetic or gravitational force, as it changes from point to point.



## From Generative Algorithms | Zubin Khabazi



## From Generative Algorithms | Zubin Khabazi



## From Generative Algorithms | Zubin Khabazi

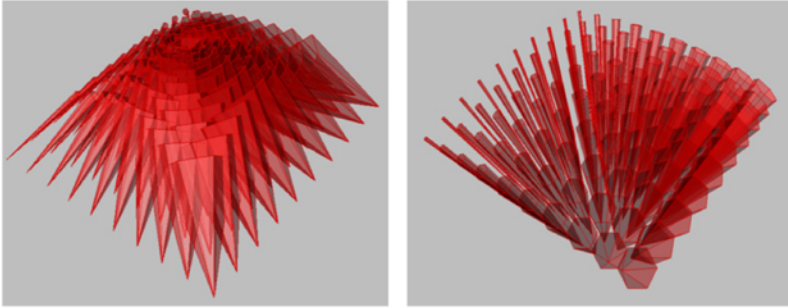


Fig.4.8. It is possible to change the size of polygons gradually using series of numbers instead of <number slider>. It is also possible to change the limit of line length (<Line SDL>) from  $-x$  to  $x$  so you can change the direction of the lines and also pyramids with negative values. Finally you can bake and render a star shape geometry after these simple experimentations with curves and vectors.

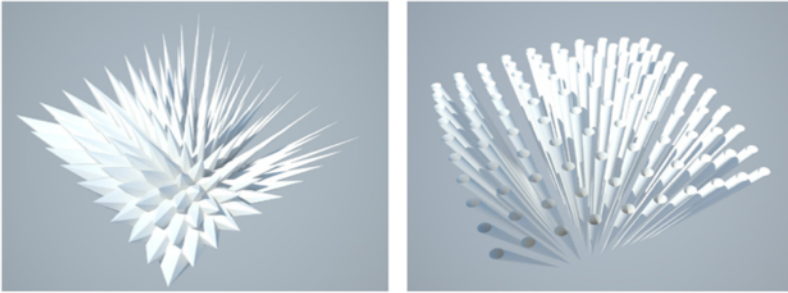


Fig.4.9. Final model (Rendered with Polygon and also circle as the profile curve for extrusion).



# Mapping & Recursion





# Mapping

Much form-making comes directly from relatively simple functions, for many reasons. The repetitive use of simple functions can unify a design across its parts and across design scales. If the ease and cost of fabrication is a concern, simple functions can help control complexity and cost (but do not necessarily do so). In contrast, the so-called *freeform* curves and surfaces provide interfaces to more complex functions, but cede some control of the form-making process to the underlying algorithms.

Simple functions come with natural domains and ranges. It is much easier to think about a function in its natural domain and range than in a transformed version.

# Mapping

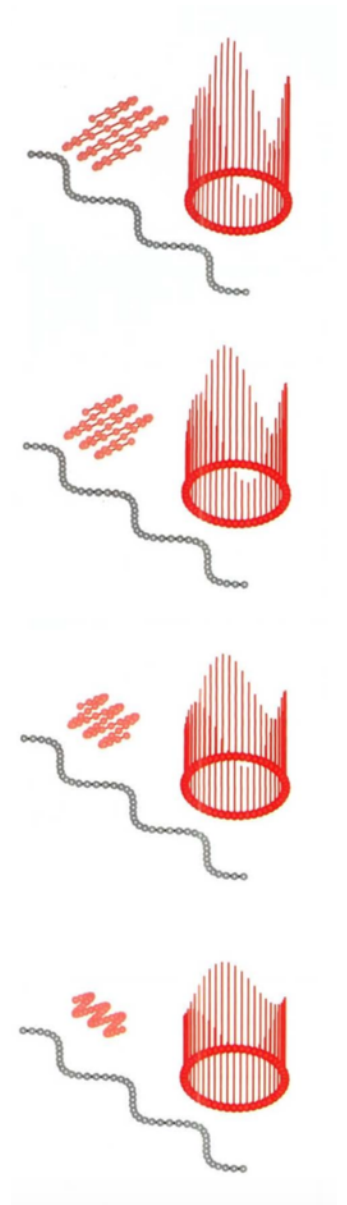
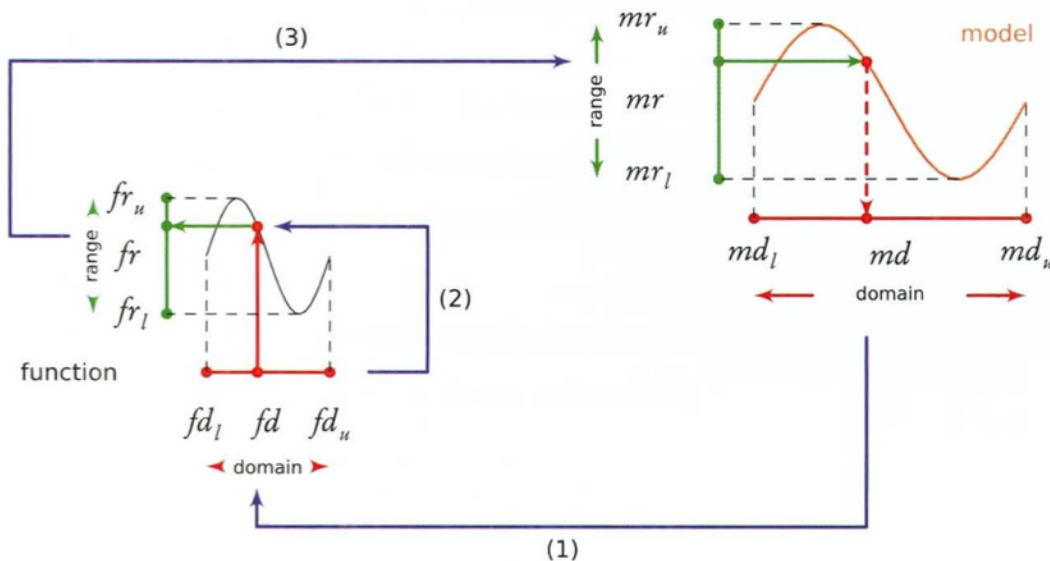
To use a function in a model requires reframing it so that it makes sense in the model. There is a universal method of precisely seven parameters that works for almost all reframing tasks. Further, this method is based on one simple equation - essentially the same equation that defines a one-dimensional *affine map* or, equivalently, an *affine function* .

Affine Transform	Example	Transformation Matrix
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

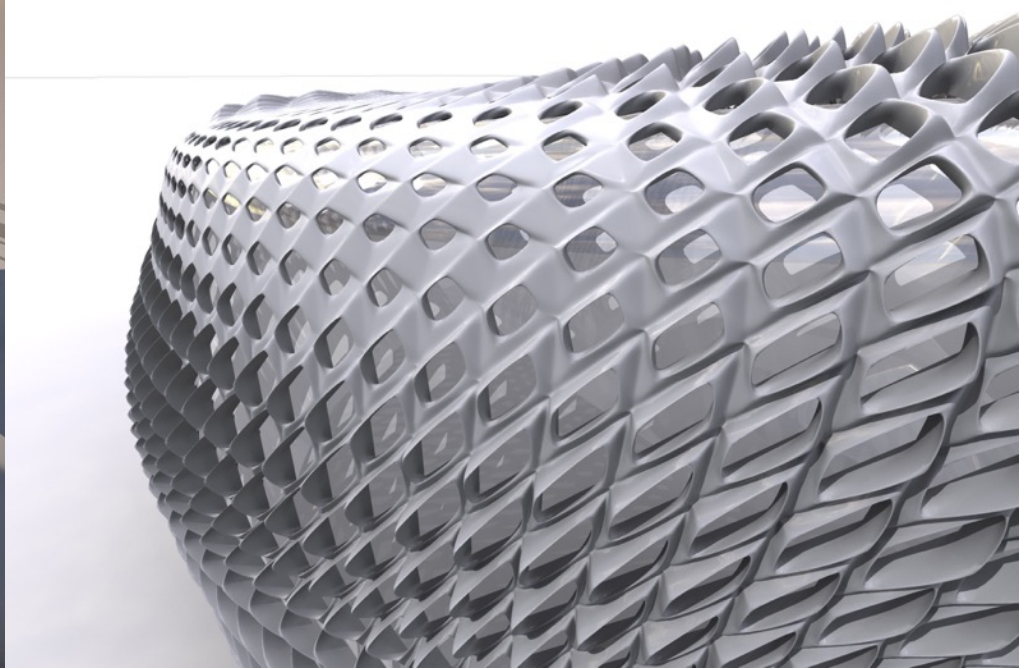
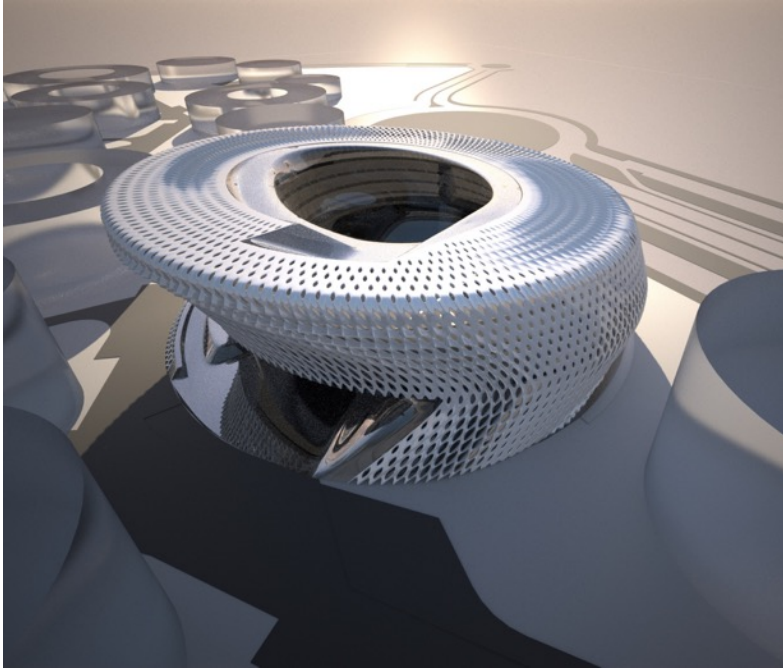


# Mapping

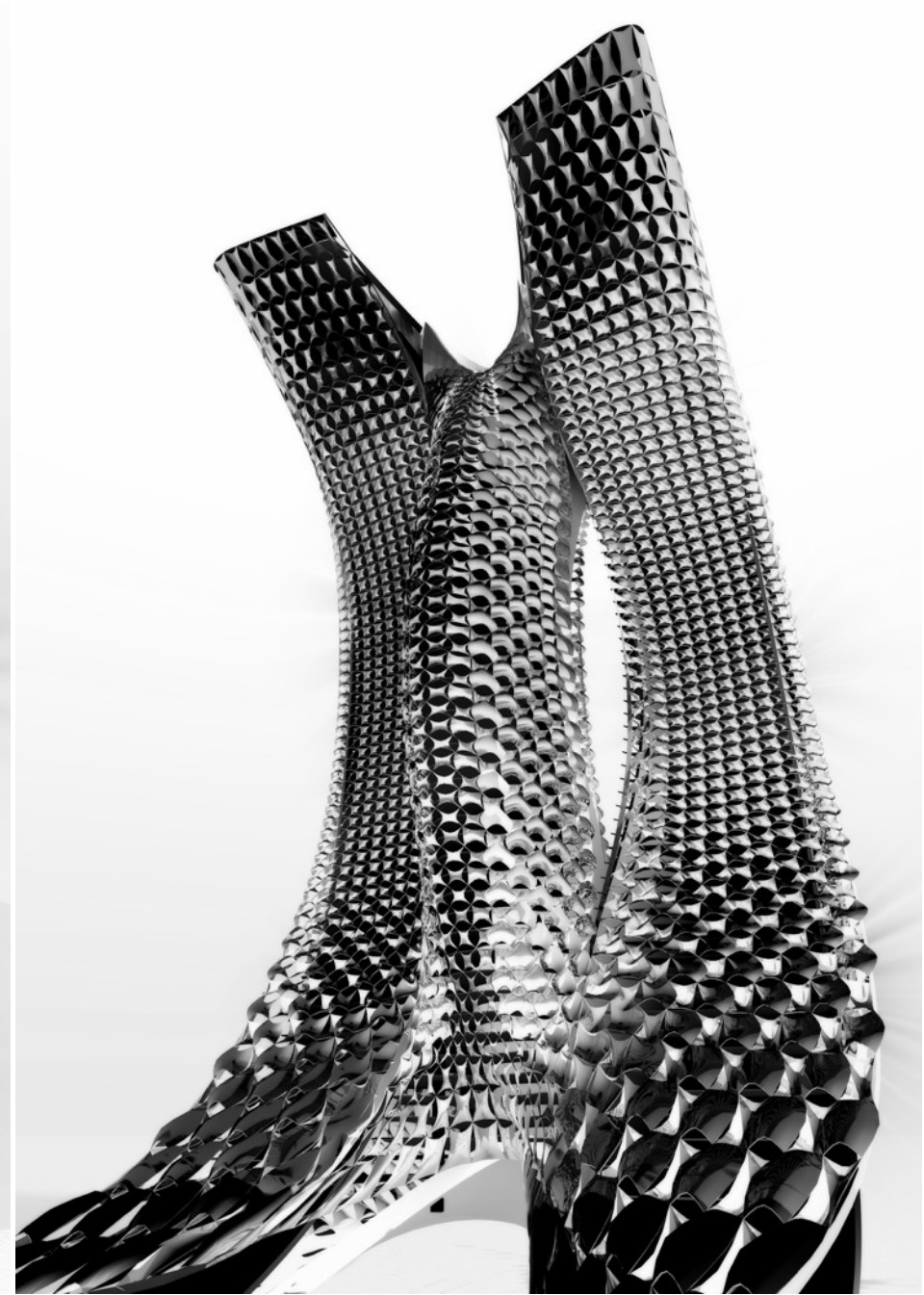
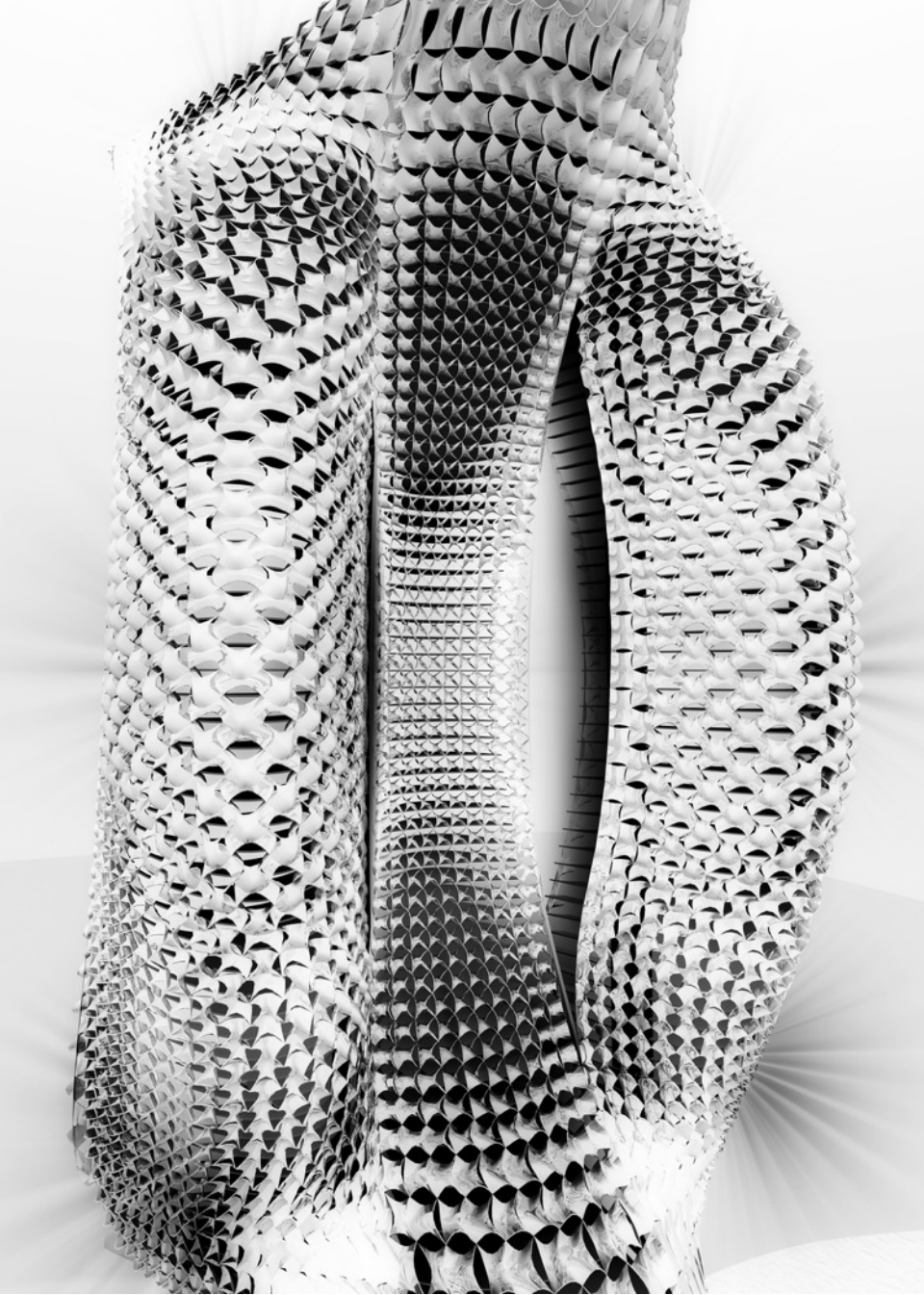
An affine map is a *linear transformation* followed by a translation. In turn, a linear transformation preserves vector addition and scalar multiplication- you can add or multiply before or after the transformation and the result is the same.



# Mapping



Zaha Hadid Architects, Civil Courts – Madrid, 2007, Accentuating Environmentally Adaptive Façade



Parametric Figuration Project, Masterclass Zaha Hadid, tutored by Patrik Schumacher, University of Applied Arts, Vienna 2007



Rear View



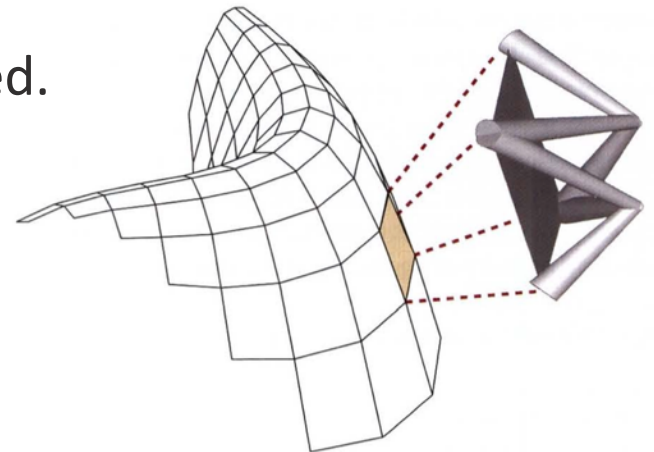
ERNESTO F. OLMEDO CRESPO

3D\_HONEYCOMB  
PARAMETRIC DESIGN + DIGITAL FABRICATION



# Place Holder

A very common scenario arrays a module across a target surface or along a set of curves. If this module requires point-like inputs themselves defined on the target, organizing these inputs is sure to be complex and error prone. If you can define the inputs to the complex module through a simple construct such as a polygon, it is often much easier to place the module. An arrangement of polygons on the goal surface creates proxies on which the module can be later (and easily) placed.



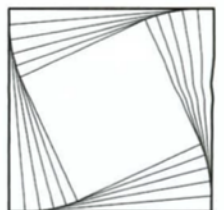
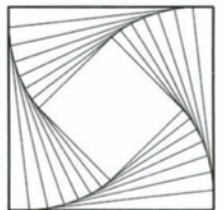
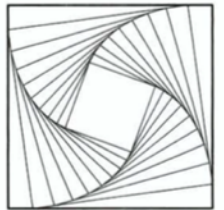
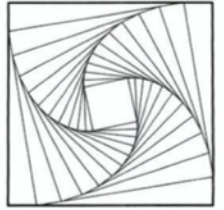
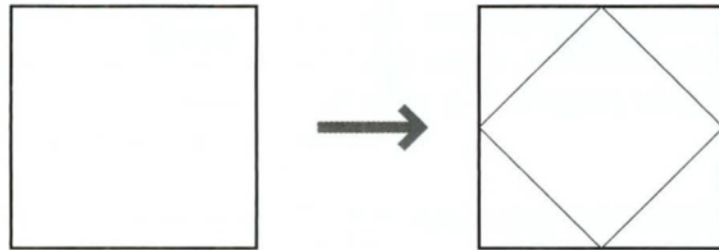
# Recursion

Some complex models such as spirals, trees or space-filling curves can be elegantly represented with a recursive function. A recursive function typically takes a motif and a replication rule and calls itself on the copies of the motif that the rule generates. Recursive algorithms as update methods can be slow. Typically, limiting the depth of the recursion is the only way to maintain an adequate interactive update rate.

RECURSION requires a motif (a geometric object) and a replication rule. The simplest rule is merely a coordinate system with some combination of translation, rotation, scale and shear. Other more complex rules are possible, including ones that change or abstract the motif itself.

# Recursion

The recursive function uses the replication rule to generate a collection of clones of the motif. It then calls itself on each clone, typically (but not necessarily) with the same replication rule. So, in each step, the recursive function takes an existing motif, replicates it and calls itself. Every recursive function requires a termination condition to specify when to stop this process.



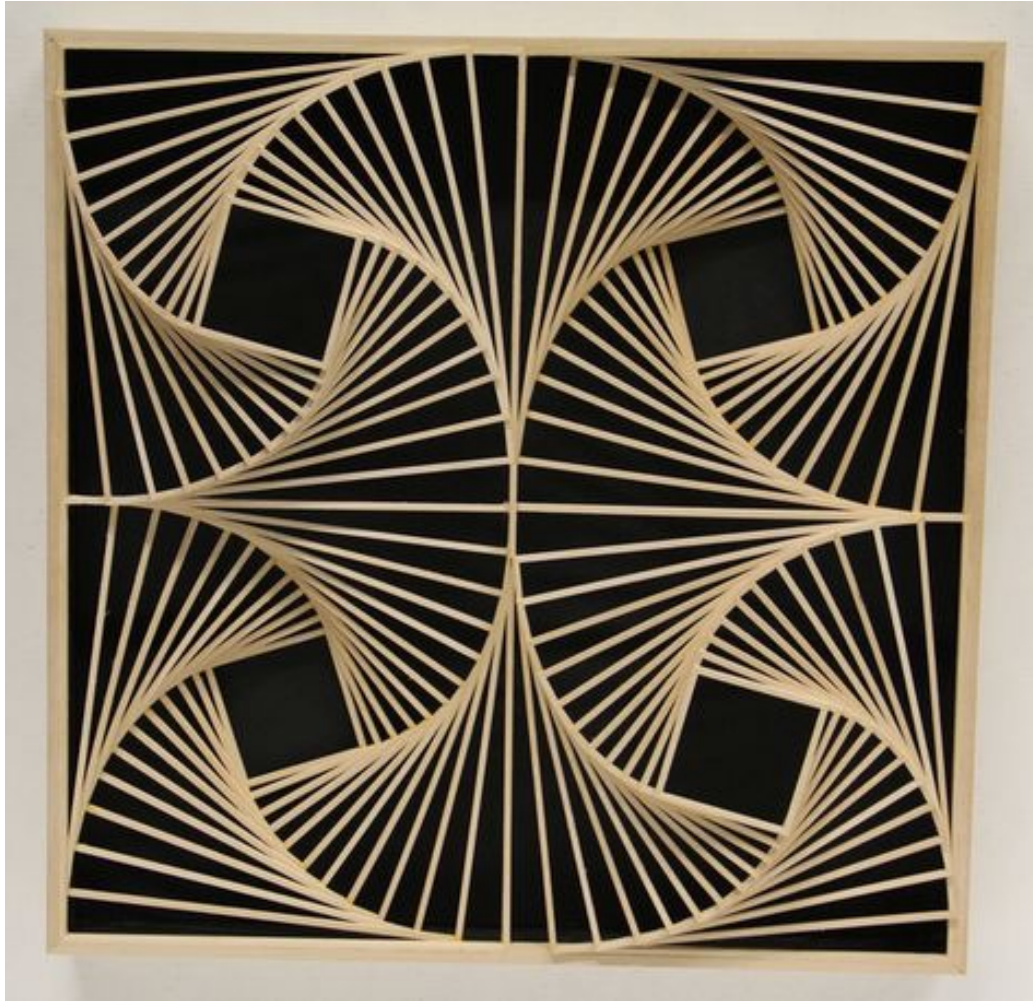
# Recursion

LINK:

[http://www.elementsofparametricdesign.com/  
view.php?hash=&dir=files%2FPatterns%2FRecursion](http://www.elementsofparametricdesign.com/view.php?hash=&dir=files%2FPatterns%2FRecursion)



# Recursion



# **The Loft Challenge**

