Introduction to Grasshopper for Rhinoceros
<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRIDAY (7th Sep)</td>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Storage I: Objects, Scalars, Data Trees</td>
<td>Examples: Spiral Function, Curve paths</td>
</tr>
<tr>
<td>FRIDAY (14th Sep)</td>
<td>Data Storage II: Vectors</td>
<td>Unit Vectors&lt;br&gt;Examples: Shift Data, Attractor scaled boxes</td>
</tr>
<tr>
<td></td>
<td>Curves (NURBS)</td>
<td>Definition &amp; type of curves&lt;br&gt;Curve Analysis (Curve Domain)&lt;br&gt;Examples: Curve Domain &amp; reparametrize, Curves in between two curves</td>
</tr>
<tr>
<td>FRIDAY (21st Sep)</td>
<td>Surfaces (NURBS)</td>
<td>Definition &amp; Surface Analysis&lt;br&gt;Surface Domain &amp; Evaluate Surface&lt;br&gt;Examples: Surface Connect &amp; Panelling of a surface</td>
</tr>
<tr>
<td>FRIDAY (28th Sep)</td>
<td>Surfaces (NURBS):</td>
<td>Manipulation of Curves &amp; Surfaces II&lt;br&gt;Example: Surface Diagrid Uniform and Non-uniform Distribution</td>
</tr>
</tbody>
</table>
What can Grasshopper do for you?

Surface DiaGrid

Paneling a Surface
What can Grasshopper do for you?
1. Data Storage

1.1 Objects

a) Params:
   • Import and store data
   • Used to introduced objects previously created in Rhino Interface

   e.g. point, surface, plane, curve, etc

b) Components:
   • Generate and store data
   • Created within Grasshopper

   e.g. logic, scalar, vector, curve, surface, mesh, etc
1. Data Storage

1.1 Objects

[Diagram image]
1. Data Storage

1.1 Objects

A: Input parameters
B: Name of Component
C: Output parameters
A) **Constants**: Used to represent a constant value (e.g. Pi, Golden Ratio, etc).

B) **Operators**: used for mathematical operations such as Add, Subtract, Multiply, etc. Example: `scalar_operators.ghx`

C) **Intervals**: used to divide numeric extremes (or domains) into interval parts. Apart from Intervals, Ranges and Series are included within this group. Example: `scalar_intervals.ghx`

D) **Utility**: evaluate two or more numerical values. Examples: `conditional_statements.ghx` and `If_Elsetest.ghx`

- Conditional Statements: Equality, Similarity, Larger than, Smaller than
- Booleans: ‘If/Else’ statements in grasshopper

E) **Polynomials**: used to define a polynomial function. Example: `Function_Spiral.ghx`

F) **Trigonometry**: used to define a trigonometric function such as sine, cosine, tangent, etc. Example: `Trigonometric_curves.ghx`
1. Data Storage

1.2 Scalars:
(B) Operators
1. Data Storage

1.2 Scalars:

(C) Intervals

Range

Series

Data Storage
1. Data Storage

1.2 Scalars:

(C) Intervals
Range & Series

Interval

Slider 10
Slider 20
Slider 10

A B Int
D N Range

10.0 To 20.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
1. Data Storage

1.2 Scalars:

(D) Conditional Statements:
1. Data Storage

1.2 Scalars:

(D) Conditional Statements:

\[ x > 5 \]
1. Data Storage

1.2 Scalars:

(E)

Polynomials:

Spiral Function
1. Data Storage

1.2 Scalars:

(F)

Trigonometric:
1. Data Storage

1.3 Data Trees

- Path {0;0;0;0} (N=6)
- Path {0;0;0;1} (N=9)
- Path {0;0;1;0} (N=9)
- Path {0;0;1;1} (N=5)
- Path {0;1;0;0} (N=7)
- Path {0;1;0;1} (N=4)
- Path {0;1;1;0} (N=9)
- Path {0;1;1;1} (N=9)
1. Data Storage

1.3 Data Trees

Lists & data management

- List Item
- List length
- List Reverse
- List Shift
- Split list
- Cull nth
- Cull Pattern
- Weaving Data
Vector Definition:

magnitude: length of the vector; distance between the start and end points (scalar value)
direction: the line (−∞ to +∞) on which the vector is based
sense: whether vector is oriented towards −∞ or +∞
1.4 Vectors:

Vector Unit:

- magnitude: 1 (the unity)
- direction: x, y or z cartesian system axis
- sense: cartesian system agreed senses
1. Data Storage

1.4 Vectors: Manipulation

<table>
<thead>
<tr>
<th>Component</th>
<th>Location</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B Dist</td>
<td>Vector/Point/Distance</td>
<td>Compute the Distance between two points (A and B inputs)</td>
<td><img src="image" alt="Distance" /></td>
</tr>
<tr>
<td>pComp XYZ</td>
<td>Vector/Point/Decompose</td>
<td>Break down a point into its X, Y, and Z components</td>
<td><img src="image" alt="Decomposition" /></td>
</tr>
<tr>
<td>A, B Angle</td>
<td>Vector/Vector/Angle</td>
<td>Compute the angle between two vectors Output computed in Radians</td>
<td><img src="image" alt="Angle" /></td>
</tr>
<tr>
<td>VLen L</td>
<td>Vector/Vector/Length</td>
<td>Compute the length (amplitude) of a vector</td>
<td><img src="image" alt="Vector Length" /></td>
</tr>
<tr>
<td>VComp XYZ</td>
<td>Vector/Vector/Decompose</td>
<td>Break down a vector into its component parts</td>
<td><img src="image" alt="Decomposition" /></td>
</tr>
</tbody>
</table>
### 1. Data Storage

### 1.4 Vectors: Manipulation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sum</strong></td>
<td>Add the components of vector 1 (A input) to the components of vector 2 (B input)</td>
</tr>
<tr>
<td><strong>Vec2Pt</strong></td>
<td>Creates a vector from two defined points</td>
</tr>
<tr>
<td><strong>Reverse</strong></td>
<td>Negate all the components of a vector to invert the direction. The length of the vector is maintained</td>
</tr>
<tr>
<td><strong>Unit</strong></td>
<td>Divide all components by the inverse of the length of the vector. The resulting vector has a length of 1.0 and is called the unit vector. Sometimes referred to as 'normalizing'</td>
</tr>
<tr>
<td><strong>Multiply</strong></td>
<td>Multiply the components of the vector by a specified factor</td>
</tr>
</tbody>
</table>
1. Data Storage

1.4 Vectors:
Manipulation of Vectors & Data Trees
Example ‘Shift Data’
1. Data Storage

1.4 Vectors:
Manipulation of Vectors & Data Trees & Grids of Points.

Example ‘Scaled Boxes’
2. Curves

2.1 Definition & Type of Curves

NURBS Curve

- Curve Control Points
- Curve’s Degree
- Periodic /Non Periodic
- Resultant Curve
- Length of the curve
- Domain of the curve
2. Curves

2.1 Definition & Type of Curves

Degree of a NURBS curve

NURBS curve knot vectors as a result of varying degree

A $D^1$ nurbs curve behaves the same as a polyline. It follows from the knotcount formula that a $D^1$ curve has a knot for every control point. Thus, there is a one-to-one relationship.

$D^2$ nurbs curve is in fact a rare sighting. It always looks like it is over-stressed, but the knots are at least in straightforward locations. The spline intersects with the control polygon halfway each segment. $D^2$ nurbs curves are typically only used to approximate arcs and circles.

$D^3$ is the most common type of nurbs curve and -indeed- the default in Rhino. You are probably very familiar with the visual progression of the spline, even though the knots appear to be in odd locations.

$D^4$ is technically possible in Rhino, but the math for nurbs curves doesn't work as well with even degrees. Odd numbers are usually preferred.

$D^5$ is also quite a common degree. Like the $D^3$ curves it has a natural, but smoother appearance. Because of the higher degree, control points have a larger range of influence.

$D^7$ and $D^9$ are pretty much hypothetical degrees. Rhino goes all the way up to $D^{11}$, but these high-degree splines bear so little resemblance to the shape of the control polygon that they are unlikely to be of use in typical modeling applications.
2. Curves

2.1 Definition & Type of Curves

Other Types:

Interpolated Curve

Kinky Curve

Polyline

PolyArc

Curve through control points

Similar to interpolated; kinks at each vertex; requires angle (A) of kink

Curve made of straight segments

Curve made of arched segments
## 2. Curves

### 2.2 Curve Analytics

<table>
<thead>
<tr>
<th>Component</th>
<th>Location</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cen</td>
<td>Curve/Analysis/Centre</td>
<td>Find the center point and radius of arcs and circles</td>
<td>{x,y,z}</td>
</tr>
<tr>
<td>Cls</td>
<td>Curve/Analysis/Closed</td>
<td>Test if a curve is closed or periodic</td>
<td></td>
</tr>
<tr>
<td>Crv CP</td>
<td>Curve/Analysis/Closest Point</td>
<td>Find the closest point on a curve to any sample point in space</td>
<td></td>
</tr>
<tr>
<td>End</td>
<td>Curve/Analysis/End Points</td>
<td>Extract the end points of a curve.</td>
<td></td>
</tr>
<tr>
<td>Explode</td>
<td>Curve/Analysis/Explode</td>
<td>Decompose a curve into its component parts</td>
<td></td>
</tr>
</tbody>
</table>
## 2. Curves

### 2.2 Curve Analytics

<table>
<thead>
<tr>
<th>Curve Type</th>
<th>Description</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curve/Utility/Join Curves</td>
<td>Join as many curve segments together as possible</td>
<td><img src="curve_utility_join_curves.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Curve/Analysis/Length</td>
<td>Measure the length of a curve</td>
<td><img src="curve_analysis_length.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Curve/Division/Divide Curve</td>
<td>Divide a curve into equal length segments</td>
<td><img src="curve_division_divide_curve.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Curve/Division/Divide Distance</td>
<td>Divide a curve with a preset distance between points</td>
<td><img src="curve_division_divide_distance.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Curve/Division/Divide Length</td>
<td>Divide a curve with a preset length</td>
<td><img src="curve_division_divide_length.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
# 2. Curves

## 2.2 Curve Analytics

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curve/Utility/Flip</td>
<td>Flip the direction of a curve using an optional guide curve</td>
</tr>
<tr>
<td>Curve/Utility/Offset</td>
<td>Offset a curve with a specified distance</td>
</tr>
<tr>
<td>Curve/Utility/Fillet</td>
<td>Fillets the sharp corners of a curve with an input radius</td>
</tr>
<tr>
<td>Curve/Utility/Project</td>
<td>Project a curve onto a Brep (a Brep is a set of joined surfaces like a polysurface in Rhino)</td>
</tr>
<tr>
<td>Intersect/Region/Split with Brep(s)</td>
<td>Split a curve with one or more Breps</td>
</tr>
<tr>
<td>Intersect/Region/Trim with Brep(s)</td>
<td>Trim a curve with one or more Breps. The <strong>Ci</strong> (Curves Inside) and <strong>Co</strong> (Curves Outside) outputs indicate the direction in which you would like the trim to occur.</td>
</tr>
</tbody>
</table>
### 2. Curves

#### 2.2 Curve Analytics

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trim</strong></td>
<td>Trim a curve with one or more Regions. The <strong>Ci</strong> (Curves Inside) and <strong>Co</strong> (Curves Outside) outputs indicate the direction in which you would like the trim to occur.</td>
</tr>
<tr>
<td><strong>Union</strong></td>
<td>Finds the outline (or union) of two planar closed curves.</td>
</tr>
<tr>
<td><strong>Intersection</strong></td>
<td>Finds the intersection of two planar closed curves.</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>Finds the difference between two planar closed curves.</td>
</tr>
</tbody>
</table>
2. Curves

2.2 Curve Analytics

Curve Divide & Curve Domain

Divide curve by number of Divisions

Curve Domain

3D Points at Division
Tangent at Divisions
Domain parameter at divisions

T1 (tangent vector) = (0.1, 0.1, 0.4)

P1 = (0.2, 1.4, 5.7)
T1 = (0.1, 0.1, 0.4)

P2 = (1.3, 6.1, 7.4)
T2 = (0.2, -1.0, 0.7)

P3 = (1.3, 6.1, 7.4)
T3 = (-1.0, 0.4, 0.0)
2.2 Curve Analytics

Curve Domain

Other Options:

- Divide curve by Distance
  - 3D Points at Division
  - Tangent at Divisions
  - Domain parameter at divisions

- Divide curve by length
  - 3D Points at Division
  - Tangent at Divisions
  - Domain parameter at divisions
3. Surfaces

3.1 Type of Surfaces

**Primitives**

- Sphere primitive (Plane; Radius)
- Cylinder primitive (Plane; Radius; Height)
- Plane primitive (Plane; Width; Height)
- Cone primitive (Plane; Radius; Height)

**Free-Form**

- Surface from 4 corner points
- Extrude Curves
- Extrude linear
- Pipe
- Loft Curves
- Offset Surface
3. Surfaces

3.1 Surface Analytics:

Surface Domain

Tangent Plane and Normal Vector

Surface Divide (to obtain coordinates in u,v directions)

- Surface
- u and v coordinates to evaluate
- 3D Points at u,v coordinates
- Normal Vectors at u,v
- Tangent Plane (Frame)

Divisions in V

- Surface
- u,v coordinates from division of surface
- 3D Points at u,v coordinates
- Normal Vectors at u,v
- u,v coordinates from division of surface
3. Surfaces

3.1 Surface Analytics:

Surface Divide + Evaluate Surface

Normal Vectors & Tangent Planes:

Surface Divide + Evaluate Surface
3. Surfaces
3.1 Surface Analytics:
Example Surface Connect
3. Surfaces

3.1 Surface Analytics:

Example Panelisation
3. Surfaces

3.1 Surface Analytics:

Surface Diagrid (Uniform)
3. Surfaces

3.1 Surface Analytics:

Surface Diagrid (Uniform)
3. Surfaces

3.1 Surface Analytics:

Surface Diagnostic (Non-uniform)
3. Surfaces

3.1 Surface Analytics:

Surface Diagrid (Non-uniform)

Non-Uniform Distortion:

Alter u & v coordinate grid by enlarging or reducing the length of the u,v spaces

U divisions = larger
V divisions = maintained