Rhinoceros®
NURBS modeling for Windows

Paneling Tools
Note from the Author

Development on PanelingTools plug-in for Rhino 4.0 started in 2008. Some of its functionality was included in SectionTools plug-in (previously called ArchCut). The plug-in facilitates conceptual and detailed design of paneling patterns using NURBS geometry. PanelingTools is closely integrated with Rhino environment using standard Rhino geometry. PanelingTools also extends RhinoScript for completely customized paneling.

This manual describes the two-step process of generating panels. The first is to represent base geometry in terms of a two-dimensional point grid, and the second step is to define and apply patterns to that grid. This manual has detailed description of all commands with examples and options.

PanelingTools is under rapid development. New functionality is added frequently and like any other McNeel products, your feedback continuously shapes and steers its development.

I hope using the plug-in will be a fun and useful experience. I am always happy to hear from you and learn how you are using the plug-in. If you have any questions or suggestions to further its development, feel free to contact me.

Rajaa Issa
Robert McNeel & Associates
raja@mcneel.com
### Table of Contents

Note from the Author........................................................................................................... ii

1 **Getting Started** ..................................................................................................... 1
   Toolbars and Menu........................................................................................................ 2
   Check for Updates......................................................................................................... 2
   Help and Support......................................................................................................... 2
   History Support .......................................................................................................... 3
   How does history work............................................................................................... 3

2 **Introduction** ........................................................................................................ 4
   Paneling Process.......................................................................................................... 4

3 **Create a Paneling Grid** ....................................................................................... 5
   Create Paneling Grid Directly .................................................................................. 5
      ptGridArray........................................................................................................... 5
      ptGridArrayPolar................................................................................................... 6
   Grid from Predefined Points .................................................................................... 7
      ptGridPoints........................................................................................................... 7
      ptGridPointsOnSurface......................................................................................... 8
   Grid from Curves ....................................................................................................... 9
      ptGridExtrude1..................................................................................................... 9
      ptGridExtrude2..................................................................................................... 11
      ptGridUCurves..................................................................................................... 12
      ptGridUVCurves................................................................................................. 13
   Grid from Surfaces .................................................................................................... 14
      ptGridSurfaceDomain.......................................................................................... 14
      ptGridSurfaceDomainExact................................................................................ 15
      ptGridSurfaceDomainVariable............................................................................ 17
      ptGridSurfaceDistance....................................................................................... 18
   Grid from Projected Curves on Surface or Polysurface ............................................ 19
      ptGridCurve (one directional curve)................................................................. 19
      ptGridCurve2 (two directional curves) ............................................................. 21
   Grid from RhinoScript.............................................................................................. 23
   Grid from Grasshopper.............................................................................................. 25
   Create Paneling Grid for Polysurfaces .................................................................... 26
      Use ptGridSurfaceUV with ArcLength method............................................... 26
      Use ptGridCurve command ............................................................................. 27
      Use ptGridCurve2 command .......................................................................... 28
      Use approximate surface and Project or Pull command.................................... 30
4 Generate Paneling ........................................................................................31
  Connecting Grid Points ..............................................................................31
  Mapping to Unit Grid ..................................................................................31
  2-D Connecting Patterns ............................................................................31
    ptManage2DPatterns ..................................................................................32
    Save and load custom 2D patterns ..........................................................34
  3-D Connecting Pattern ............................................................................35
    ptManage3DPattern ..................................................................................36
    Save and load 3D custom patterns ..........................................................37
    ptPanelGridCustom ..................................................................................38
    ptPanelGridCustomVariable ...................................................................39
    ptPanel3DCustom ....................................................................................43
    ptOrientToGrid ........................................................................................45
    ptPanel3DCustomVariable .......................................................................46
  Paneling Planar Quadangles .......................................................................49
    ptPanelGridQuads ...................................................................................49
  Paneling with a Grid ..................................................................................50
    ptPanelGrid ................................................................................................50
    ptPanel3D ..................................................................................................52
  Paneling without a Grid ............................................................................52
    ptPanelSubDivide .....................................................................................52
    ptPanelRandomPoints ...............................................................................54
    ptTriangulatePoints ..................................................................................55

5 Paneling Output .........................................................................................56
  Paneling format ..........................................................................................56
  Paneling shape ...........................................................................................57
  Trimmed surfaces .......................................................................................58

6 Utility Functions ........................................................................................59
  Grid Utility Functions ................................................................................59
    ptDirection ...............................................................................................59
    ptSwapGridUV ........................................................................................59
    ptRowsDirection .......................................................................................60
    ptCompactGrid ........................................................................................61
    ptCloseGrid .............................................................................................61
    ptGridSeam .............................................................................................62
    ptCleanOverlap .......................................................................................63
    ptTrimGrid ...............................................................................................64
    ptOffsetPoints ........................................................................................65
    ptChangeGridDensity ...............................................................................66
    ptExtendGrid ...........................................................................................67
    ptShiftGrid ..............................................................................................68
    ptShuffleGrid ..........................................................................................68
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptConvertToDiagonal and ptConvertToDiamond</td>
<td>69</td>
</tr>
<tr>
<td>ptWeaveGrids</td>
<td>70</td>
</tr>
<tr>
<td>ptExtractCenterGrids</td>
<td>70</td>
</tr>
<tr>
<td>ptMeanGrid</td>
<td>70</td>
</tr>
<tr>
<td>ptOffsetGridByHeightfield</td>
<td>71</td>
</tr>
<tr>
<td><strong>Paneling Utility Functions</strong></td>
<td></td>
</tr>
<tr>
<td>ptExtrudeEdges</td>
<td>72</td>
</tr>
<tr>
<td>ptOffsetEdges</td>
<td>73</td>
</tr>
<tr>
<td>ptFinEdges</td>
<td>74</td>
</tr>
<tr>
<td>ptUnifyFacesDirection</td>
<td>75</td>
</tr>
<tr>
<td>ptAnalyzeFlatFaces</td>
<td>75</td>
</tr>
<tr>
<td>ptGroupSimilarPanels</td>
<td>76</td>
</tr>
<tr>
<td>ptUnrollFaces</td>
<td>76</td>
</tr>
<tr>
<td>ptTriangulateFaces</td>
<td>77</td>
</tr>
<tr>
<td>ptUnrollEdges</td>
<td>77</td>
</tr>
<tr>
<td>ptUnrollEdges</td>
<td>78</td>
</tr>
<tr>
<td>ptUnrollPoints</td>
<td>79</td>
</tr>
<tr>
<td>ptOffsetBorder</td>
<td>79</td>
</tr>
<tr>
<td>ptPlanarLips</td>
<td>80</td>
</tr>
<tr>
<td><strong>General Utility Functions</strong></td>
<td></td>
</tr>
<tr>
<td>ptDivideCurveSpan</td>
<td>81</td>
</tr>
<tr>
<td>ptDivideCurveByChordLength</td>
<td>81</td>
</tr>
<tr>
<td>ptSurfaceFromGridOfEditPoints</td>
<td>82</td>
</tr>
<tr>
<td>ptSurfaceFromGridOfControlPoints</td>
<td>83</td>
</tr>
<tr>
<td>ptUnifyCurvesDirection</td>
<td>83</td>
</tr>
<tr>
<td>ptTagObjects</td>
<td>84</td>
</tr>
<tr>
<td>ptSerializeObjects</td>
<td>85</td>
</tr>
<tr>
<td>ptMeanCurves</td>
<td>85</td>
</tr>
<tr>
<td>ptMeanSurfaces</td>
<td>86</td>
</tr>
<tr>
<td>ptRemoveOverlapedPoints</td>
<td>87</td>
</tr>
<tr>
<td><strong>Serializing joints and connections for FE analysis</strong></td>
<td></td>
</tr>
<tr>
<td>ptSerializePoints</td>
<td>87</td>
</tr>
<tr>
<td>ptSerializeEdges</td>
<td>87</td>
</tr>
<tr>
<td>ptTagSerializedData</td>
<td>87</td>
</tr>
<tr>
<td>ptExportPointsSerializeData</td>
<td>88</td>
</tr>
<tr>
<td>ptExportEdgesSerializeData</td>
<td>88</td>
</tr>
<tr>
<td><strong>7 Extending RhinoScript</strong></td>
<td>90</td>
</tr>
</tbody>
</table>
1 Getting Started

PanelingTools is a plug-in for Rhino 4.0 that supports designing and modeling paneling patterns. It also rationalizes NURBS surfaces and polysurfaces. Many of the commands in the plug-in are history based.

To load the PanelingTools plug-in:

1. Save PanelingTools.rhp in any location. Normally this is C:\Program Files\Rhinoceros 4.0\Plug-ins.

2. Use the Rhino PluginManager command (Tools menu > Options > Plug-ins) to install PanelingTools.rhp.

Or,

Drag-and-drop the PanelingTools.rhp file into Rhino.

Note: Load the plug-in only once using either of the above methods. After that, with every new Rhino session, PanelingTools and its menu loads when you call a plug-in command for the first time in that session. If you like to have the menu loaded at startup, you need to type an empty command such as "!_ptGridArray _Enter" in the "General" section of "Rhino Options" as follows:

![Rhino Options](image)
Toolbars and Menu
The PanelingTools toolbar file (PanelingTools.tb) includes all plug-in commands.

To load toolbars:
1. Start the Toolbar command (Tools >Toolbar Layout).
2. From the File menu, click Open, and browse to PanelingTools.tb.

The PanelingTools plug-in also adds the PanelingTools menu when it is loaded.

Check for Updates
PanelingTools is under active development. For new versions and up-to-date documentation go to http://en.wiki.mcneel.com/default.aspx/McNeel/PanelingTools.html.

Suggestions, bug reports and comments are welcome. Please share your stories, examples, and experiences with us. Post questions to our newsgroup or email us directly. Visit http://www.rhino3d.com/support.htm for more details, or contact the developer, Rajaa Issa.

Help and Support

For tutorial examples and video clips go to: http://en.wiki.mcneel.com/default.aspx/McNeel/PanelingExamples.html

Post your questions to the Rhino newsgroup: news://news.rhino3d.com/rhino
Email support is also available: tech@mcneel.com
History Support

Most PanelingTools commands support history. However, history can be expensive to use. It works great for smaller grids, but can be very time consuming for larger ones.

Here are few useful things to keep in mind when using history:

Although history is very useful in some cases, it can be slow and counter-intuitive when dealing with a large set of data.

History in the Rhino SDK is not designed to handle problems typical to paneling where there is a lot of input (grid of hundreds of points) and even more output data (all those panels) that typically expands and shrinks with each update. The PanelingTools history implementation affects speed.

There is the possibility to cancel calculation in most cases.

How does history work

Just like other Rhino commands, you need to activate history recording before running the command you intend to record history for. When referenced input geometry changes, the command will be replayed to update output. Here is an example that generate paneling grid with history.

Modifying input curves update grid points which in turn updates paneling triggering a chain update effect.
2 Introduction

The PanelingTools plug-in supports conceptualizing with 2-D and 3-D design patterns and helps rationalize NURBS surfaces and polysurfaces for analysis and fabrication.

Forms that can be paneled with PanelingTools can be represented with a 2-dimensional point grid. PanelingTools provides many functions to turn base geometry of points, curves, surfaces, and polysurfaces into an ordered 2-dimensional grid. The grid is then used as basis to apply 2-D and 3-D patterns.

Paneling Process

Paneling is done in two steps: first, create a paneling grid, and then generate the paneling geometry of curves, surfaces and polysurfaces.

Creating a paneling grid results in points that can be manipulated with any Rhino command or PanelingTools grid utility commands.

Generating the paneling creates patterns and applies the patterns to a valid paneling grid of points. The resulting paneling is standard Rhino geometry in the form of curves, surfaces, or a mesh. To further process panels (with the Unroll, Offset, Pipe, or Fin commands, for instance) use paneling utility functions and other Rhino commands.

The two-step process gives more flexibility and better control over the result. Normally, the initial grid is generated interactively and is a good indicator of scale. The grid can be generated using the many grid-generating commands or with scripting. The grid can be directly edited and refined before any paneling is applied. Panels can be created using built-in patterns or user-defined free-form patterns.

The following sections illustrate various methods for generating a base grid of points and how paneling is applied using that base grid.
3 Create a Paneling Grid

A paneling grid is a group of Rhino point objects. Each paneling point has a name consisting of its row and column location in the grid.

For example:

Points have name tags S0(0)(0), S0(0)(1), .... These names are object properties. The naming convention is as follows:

- S0 = paneling grid name
- (first number) = row location
- (second number) = column location

Since the names can be edited directly using the Rhino Properties command, points that are valid input for paneling can also be created using the Rhino Points command, and names can be assigned to reflect their locations with the Properties command. PanelingTools grid creation commands do the naming automatically.

Create Paneling Grid Directly

The ptGridArray and ptGridArrayPolar commands create a two-dimensional array of points.

ptGridArray

The ptGridArray command creates an array of parallel points.
Command flow

1. Start the `ptGridArray` command.
2. Pick a base point.
3. Press Enter to accept options.

Options

**U_Number**
Number of points in the u-direction.

**U_Spacing**
Distance between points in the u-direction.

**U_direction**
Pick two points to set the u-direction.

**V_Number**
Number of points in the v-direction.

**V_Spacing**
Distance between points in the v-direction.

**V_Direction**
Pick two points to set the v-direction.

**Group**
If Yes, group the resulting points.

**NameOfGrid**
Grid name prefix attached to each point. The row and column location complete the point name.

**ptGridArrayPolar**
The `ptGridArrayPolar` command creates a polar array of points.

Command flow

1. Start the `ptGridArrayPolar` command.
2. Pick a center and first point of the rotation axis.
3 Pick a second point of the rotation axis.
   Press Enter to rotate normal to active construction plane.
4 Pick the base and first point of the grid direction.
5 Pick the second direction point.
   Press Enter if parallel to rotation axis.
6 Press Enter to accept options.

Options

U_Number
Number of points in the u-direction.

U_Spacing
Distance between points in the u-direction.

U_Direction
Pick two points to specify a u-direction.

V_Number
Number of points in the v-direction (polar direction).

V_Angle
Angle between points in v direction.

Group
If Yes, group the resulting points.

NameOfGrid
Grid name prefix attached to each point. The row and column location complete the point name.

Grid from Predefined Points
The ptGridPoints and ptGridPointsOnSurface commands use a pre-defined set of points to create a paneling grid or to order these points into rows and columns.

These commands define a distance tolerance to identify points that belong to one row or one column. The result may not always be desirable. It is best to define the paneling grid points using plug-in commands whenever possible.

ptGridPoints
The ptGridPoints command uses u- and v-values from a base surface as a parallel reference grid. Input points take the row/column location from the closest reference grid point. The result might have fewer points than the input since more than one point could be rounded to same index. Reference spacing is critical to the result as illustrated in the image:
Command flow

1. Start the `ptGridPoints` command.
2. Pick the base point.
3. Press `Enter` to accept options.

Options

- **U_Spacing**
  Reference grid spacing in the u-direction.

- **V_Spacing**
  Reference grid spacing in the v-direction.

- **U_Direction**
  Pick two points to specify a u-direction.

- **V_Direction**
  Pick two points to specify a v-direction.

- **Group**
  If Yes, group the resulting points.

- **DeleteInput**
  If Yes, delete the input points.

- **AlignPoints**
  If Yes, shift the points to align with reference grid.

- **NameOfGrid**
  Grid name prefix attached to each point. The row and column location complete the point name.

**ptGridPointsOnSurface**

The `ptGridPointsOnSurface` command turns points existing on a surface into a valid grid of paneling points. The algorithm creates isocurves using (2*`Tolerance`). Points within tolerance of any one curve are added to the grid as one row of points. Points in one row are ordered relative to their parametric location on the curve.
Command flow

1. Start the ptGridPointsOnSurface command.
2. Select the points.
3. Select the surface.
4. Press Enter to accept options.

Options

**Tolerance**

Spacing between isocurves is equal to double the tolerance value. Points should be within tolerance from a particular row base line to be included in that row.

**Group**

If Yes, group the resulting points.

**DeleteInput**

If Yes, delete input points.

**AlignPoints**

If Yes, shift the points to align with reference grid.

**NameOfGrid**

Grid name prefix attached to each point. The row and column location complete the point name.

**Grid from Curves**

The ptGridExtrude1, ptGridExtrude2, ptGridUCurves, and ptGridUVCurves commands use input curves to create a paneling grid.

**ptGridExtrude1**

The ptGridExtrude1 command uses one curve and extrudes division points in parallel or polar directions.
Command flow

1. Start the `ptGridExtrude1` command.
2. Select the base curve.
3. Press `Enter` to accept options.

Options

**U_Method**
Base curve division method.

<table>
<thead>
<tr>
<th>Number</th>
<th>U_NumberOfSpans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of spaces between points.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ArcLength</th>
<th>U_Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Along-curve distance between points.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U_Round</th>
</tr>
</thead>
<tbody>
<tr>
<td>If Yes, round the distance up or down to fill the whole curve.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U_RoundingMethod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up</td>
</tr>
<tr>
<td>Down</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ChordLength</th>
<th>U_ChordLength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight-line distance between points.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U_AddEndPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>If Yes, add a point at the end.</td>
</tr>
</tbody>
</table>

**V_Number**
Number of points in the extrusion v-direction.

**V_Method**
Array curve divide points in parallel or polar direction.

<table>
<thead>
<tr>
<th>Parallel</th>
<th>V_Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance between points.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V_Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick two points to specify a v-direction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polar</th>
<th>V_Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle between rows of points.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V_RotationAxis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick two points to specify a rotation axis.</td>
</tr>
</tbody>
</table>

**Group**
If Yes, group the resulting points.
**NomeOfGrid**
Grid name prefix attached to each point. The row and column location complete the point name.

**ptGridExtrude2**
The *ptGridExtrude2* command extrudes points of a base curve along a path curve.

**Command flow**
1. Start the *ptGridExtrude2* command.
2. Select the first curve.
3. Select the second curve.
4. Press **Enter** to accept options.

**Options**

**U_Method**
First curve division method.
- **Number**
  - U\_NumberOfSpans
    - Number of spaces between points.
- **ArcLength**
  - U\_ArcLength
    - Along-curve distance between points.
  - U\_Round
    - If **Yes**, round the distance up or down to fill the whole curve.
  - U\_RoundingMethod
    - Up
    - Down
- **ChordLength**
  - U\_ChordLength
    - Straight-line distance between points.
  - U\_AddEndPoint
    - If **Yes**, add a point at the end.

**V_Method**
Second curve division method.
- **Number**
  - V\_NumberOfSpans
    - Number of spaces between points.
ArcLength  V_ArcLength
Along-curve distance between points.

V_Round
If Yes, round the distance up or down to fill the whole curve.

V_RoundingMethod
  Up
  Down

ChordLength  V_ChordLength
Straight-line distance between points.

V_AddEndPoint
If Yes, add a point at the end.

Group
If Yes, group the resulting points.

NameOfGrid
Grid name prefix attached to each point. The row and column location complete the point name.

SwitchCurves
Change which of the two curves to copy along the other curve points.

ptGridUCurves
The ptGridUCurves command uses an existing array of curves to create a paneling grid. It divide the curves, that are usually parallel or non-intersecting, by number or distance. It is best to select curves in the order of desired rows (first selected curve become row0, etc). The option to automatically order curves and unify their direction might not yield desired result in all cases.

Command flow

1  Start the ptGridUCurves command.

2  Select curves in order or group select curves and have the command order curves internally using each curves midpoint.

3  Press Enter to accept options.
Options

SortCurvesOrder
If Yes, the following option appears:

SortMethod
   StartPoint Curves start point
   MidPoint Curves mid point
   Centroid Curves centroid

Method
Division method.
   Number NumberOfSpans
      Number of spaces between points.
   ArcLength ArcLength
      Along-curve distance between points.
      Round
         If Yes, round the distance up or down to fill the whole curve.
   RoundingMethod
      Up
      Down
   ChordLength ChordLength
      Straight-line distance between points.
   AddEndPoint
      If Yes, add a point at the end.

Group
If Yes, group the resulting points.

ptGridUVCurves
The ptGridUVCurves creates paneling points at curve intersections. Select the curves in each direction. Selection order defines order of rows and columns in the grid. An option orders the curves automatically.

Command flow
1. Start the ptGridUVCurves command.
2. Select u-direction curves in order or group select curves and have the command order curves internally using each curve midpoint.
3. Select v-direction curves in order or group select curves and have the command order curves internally using each curve midpoint.
4 Press Enter to accept options.

Options

SortCurvesOrder
Sort curves relative to their midpoints.

Group
If Yes, group the resulting points.

Grid from Surfaces
The ptGridSurfaceDomain, ptGridSurfaceDomainExact, and ptGridSurfaceDistance use a base NURBS surface to generate a paneling grid.

ptGridSurfaceDomain
The ptGridSurfaceDomain command divides a surface following its u- and v-directions. Division can be by number, distance, or chord length using any combination in u and v-directions. The grid name references the surface name. You can name the input surface prior to calling the ptGridSurfaceDomain command using Properties command.

Command flow

1 Start the ptGridSurfaceDomain command.
2 Select a surface.
3 Press Enter to accept options.

The algorithm:

1 Extracts an isocurve in the v and u directions at the minimum domain.
2 Divides the curve using U and V method set in the options to compile a list of parameters.
3 Use these parameters to divide the surface domain.
Options

**U_Method**
U-direction division method.

- **Number**
  U_NumberOfSpans
  Number of spaces between points.

- **ArcLength**
  U_Length
  Along-curve distance between points.

- **U_Round**
  If Yes, round the distance up or down to fill the whole curve.

- **U_RoundingMethod**
  Up
  Down

- **ChordLength**
  U_Length
  Straight-line distance between points.

**V_Method**
V-direction division method.

- **Number**
  V_NumberOfSpans
  Number of spaces between points.

- **ArcLength**
  V_Length
  Along-curve distance between points.

- **V_Round**
  If Yes, round the distance up or down to fill the whole curve.

- **V_RoundingMethod**
  Up
  Down

- **ChordLength**
  V_Length
  Straight-line distance between points.

**Group**
If Yes, group the resulting points.

**ptGridSurfaceDomainExact**
The *ptGridSurfaceDomainExact* command divides a surface domain. Division can be by number, arc length, or chord length using any combination in u- and v-directions. The grid name references the surface name. You can name the input surface prior to calling the *ptGridSurfaceUVDomainExact* command using the **Properties** command.

**Command flow**

1. Start the *ptGridSurfaceDomainExact* command.
2. Select a surface.
3. Allow selecting a base point on surface when dividing by distance or chord length.
4. Press **Enter** to accept options.
The algorithm:

Literary divides the parametric domain of the surface. Result can be similar to ptGridSurfaceDomain depending on how the parametric space of the surface looks like.

See the following comparison that shows an evenly spaced parametric space compared to one that is not and the effect of each using the two surface commands:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptGridSurfaceDomain</td>
<td>Generate equal distances in 3D space and therefore it tends to distribute points evenly regardless of the 2D parameterization</td>
</tr>
<tr>
<td>ptGridSurfaceDomainExact</td>
<td>Uses equal domain distances in parameter space and therefore it basically follows exact surface domain</td>
</tr>
</tbody>
</table>

Options

**Number**  
U_NumberOfSpans / V_NumberOfSpans  
Number of spans.
### ArcLength
**U_Length / V_Length**
Along-curve distance between points.

### ChordLength
**UVDirection**
Change grid base point through changing u-, v- or both directions.
- Ureverse
- Vreverse
- ReverseBoth
- Default

**SelectBasePoint**
Select a base point on surface.

### Group
If **Yes**, group the resulting points.

### ptGridSurfaceDomainVariable
The **ptGridSurfaceDomainVariable** command divides a surface domain with variable distance grid using surface curvature or other constraints.

![Grids with different methods](image)

**Command flow**

1. Start the **ptGridSurfaceDomainVariable** command.
2. Select a surface.
3. Press **Enter** to accept options.

**The algorithm:**

1. Divide with uniform distribution along surface domain (similar to **ptGridSurfaceDomainExact** command)
2. Adjust the grid using one of the methods.
Options

**UNumber**
Number of spans in u-direction..

**VNumber**
Number of spans in V direction.

**DistanceMethod**
Distribution method.

- **GaussianCurvature** Use surface gaussian curvature values
- **MeanCurvature** Use surface mean curvature values.
- **AttractorPoints** Sift towards/away from attractor points
- **AttractorCurves** Sift towards/away from attractor curves
- **SunVector** Use dot product between a vector and normal on surface at each point.
- **Random** Shift points by random amount
- **Bitmap** Use heightfield of an input image

**AttractMethod**
Either away or towards attractor points or curves. If distance method is Mean or Gaussian, then attract towards or away from the highest curvature.

**Magnitude**
Use to reduce or magnify distance variations.

**Group**
If Yes, group the resulting points.

---

**ptGridSurfaceDistance**
The **ptGridSurfaceDistance** command divides a surface by specified distances in the first and second directions. Since, this command uses an algorithm where every new point depends on previously created points; it might not give complete coverage. The **Extend** option might help creating better coverage. The grid name references the surface name. A point on the surface can be used as a base.

**Command flow**

1. Start the **ptGridSurfaceDistance** command.
2. Select a surface.
3. Select a base point on surface, or press **Enter** to use the surface minimum.
4. Press **Enter** to accept options.

**The algorithm:**

1. Extracts a u- and v-isocurve at the selected base point.
2. Divides the isocurves by the chord length.
3. Takes second point on v-isocurve and second point on u-isocurve and finds a point on the surface that is equal to u-distance and v-distance from v and u points. If the point is valid, that new point is used with the third point on the u-isocurve to find the next point, and so on until no valid point is found.
Options

**U_Distance**
Distance in first direction.

**V_Distance**
Distance in second direction.

**Extend**
Extend surface before dividing to possibly get better converge.

**Group**
If Yes, group the resulting points.

Grid from Projected Curves on Surface or Polysurface
This method is suitable for gridding a polysurface or a surface without using its uv-directions. The general algorithm:

1. Arrays the input curve in a parallel or polar direction.
2. User defines the distance/angle and direction of the array.
3. User defines the direction to project curves towards base surface or polysurface.
4. Curves are then projected and the resulting curves are joined and cleaned.
5. Uses the new curves to divide by number or distance.
6. If the curves are in both directions, grid points are extracted from their intersections.

**ptGridCurve (one directional curve)**
The ptGridCurve command generates a grid based on an object and direction curve. Curves can be open or closed. An open curve is copied in the extrusion direction by the Spacing/Angle distance. A closed curve is offset by that distance.
Options

**Line**
Option to define direction curve with two points.

**CurveOptions**
- **NumberOfCuts**
  Number of curves to be projected to the object.
- **Spacing/Angle**
  Distance or angle between curves.
- **ExtrudeMethod**
  Parallel or polar.
- **ExtrudeDirection**
  Extrusion direction (for parallel).
- **ProjectionDirection**
  Direction the curves are projected.

**GridOptions**
- **Method**
  Curves division method.
- **Number**  **NumberOfSpans**
  Number of spaces between points.
**ArcLength**

Length
Along-curve distance between points.

**Round**
If Yes, round distances to fill the span of the curve.

**RoundingMethod**
- Up
- Down

**ChordLength**

Distance
Straight-line distance between points.

**AddEndPoint**
If Yes, add a point at the end.

**Group**
If Yes, group the resulting points.

**NameOfGrid**
Grid name prefix attached to each point. The row and column location complete the point name.

**ptGridCurve2 (two directional curves)**
The `ptGridCurve2` command generates a grid using two direction curves. If there is an undesirable surface uv or surface seam, this command can apply the desired directions. In the example below, the surface has twisted seam, this is how the paneling looks using the `ptGridCurve` command (`ptGridSurfaceUV` gives a similar result):

Using two direction curves, the first in polar and the second in parallel direction, gives desired result in this case:
Options

Line
Pick two points to define the direction curve.

FirstDirCurvesOptions / SecondDirCurvesOptions
   NumberOfCuts
   Number of curves to be projected to object.
   Spacing/Angle
   Distance or angle between curves.
   ExtrudeMethod
   Parallel or polar.
   ExtrudeDirection
   Extrusion direction (for parallel).
   ProjectionDirection
   Direction the curves are projected.

Group
If Yes, group the resulting points.

NameOfGrid
Grid name prefix attached to each point. The row and column location complete the point name.
Grid from RhinoScript

A paneling grid can be generated using RhinoScript. Many of the grid generation commands are accessible directly from RhinoScript. Even without using these commands, a custom point grid can be generated and each point location can be appended to point names.

The following example shows how to create a paneling grid using RhinoScript:

```
Sub Main()
Dim i, j, PTObj, strPt
Dim x, y, z
Dim doubleA, doubleB, doubleStep
doubleA = 2
doubleB = 20
doubleStep = 0.5
On Error Resume Next
'Get PanelingTools Object
Set PTObj = Rhino.GetPluginObject("PanelingTools")
If Err Then
    MsgBox Err.Description
    Exit Sub
End If
End If
For i = 0 To 8 Step 1
    j = 0
    For x = doubleA To doubleB Step doubleStep
        y = (2*i)+Sin(x)
        z = Sin(y)
        strPt = PTObj.InsertPointInGrid(Array(x, y, z),i,j )
        j = j+1
    Next
Next
End Sub
```

This example uses RhinoScript to generate a variable distance grid, and panel it using diamond pattern:
Sub AddGrid( arrPoints, PTObj )

dim i, j, arrRow, arrPt, strPt

i = 0
j = 0
If IsArray(arrPoints) Then
    for Each arrRow In arrPoints
        j = 0
        If IsArray( arrRow ) Then
            For Each arrPt In arrRow
                If IsArray( arrPt ) Then
                    strPt = PTObj.InsertPointInGrid( arrPt, i, j )
                    End If
                Next
            Next
            i = i+1
        Next
    End If
End Sub

Call Main()

Sub Main()

dim i, PTObj, arrPoints, strSrf, dbl_cost, dbl_t

Dim arr_uList(10)
Dim arr_vList(10)

On Error Resume Next
*Get PanelingTools Object
Set PTObj = Rhino.GetPluginObject("PanelingTools")
If Err Then MsgBox Err.Description Exit Sub

*Get surface
strSrf = Rhino.GetObject("Select surface", 8)
If IsNull(strSrf) Then Exit Sub

For i = 0 To 10 Step 1
    dbl_t = 0.1*i
    dbl_cost = cos( dbl_t* (Rhino.PI/2) )
    arr_uList(i) = dbl_cost
    arr_vList(i) = dbl_cost
Next

arrPoints = PTObj.DivideSurfaceByVariableDistances(strSrf, arr_uList, arr_vList, False)

*Inset points as grid
Call AddGrid( arrPoints, PTObj )

Set PTObj = Nothing
End Sub
Grid from Grasshopper

Following is an example of a grid generated with Grasshopper and baked with a custom scripting component to attach point locations.

This is how the code inside the VB script component looks:

```vbnet
Sub RunScript(ByVal Points As List(Of Object), ByVal GridName As String, ByVal Bake As Boolean, ByVal Num As Integer)
    If Not Bake Then
        m_count = 0
        Return
    End If

    'Iterate through points by row
    Dim i As Integer
    Dim j As Integer

    j = 0
    For i = 0 To Points.Count() - 1
        If j > Num + 1 Then
            j = 0
        End If
        Dim pt As On3dPoint
        pt = Points(i)
        'Name the point
        Dim att As New On3dmObjectAttributes
        doc.GetDefaultObjectAttributes(att)
        att.m_name = GridName & "(" & m_count & ")" & j & ""
        print(m_count)
        'increment index
        j += 1
        'Add to document
        doc.AddPointObject(pt, att)
    Next
    m_count += 1
    A = m_count
End Sub

#Region "Additional methods and Type declarations"
    Private Shared m_count As Integer
#End Region
```
Create Paneling Grid for Polysurfaces

Polysurfaces are joined NURBS surfaces that might not have matching isocurve directions. It is not always possible to define a continuous paneling grid for polysurfaces, but there are few commands that are geared towards dealing with polysurfaces. The main commands are `ptPanelCurve` and `ptPanelCurve2`.

Use `ptGridSurfaceUV` with ArcLength method

If all surfaces in a polysurface have their isocurves aligned, then using `ptGridSurfaceUV` with Method=ArcLength will result in aligned grids. However, these grids will be separate and paneling has to be applied separately to each of them. One way to combine all grids is to use `ptShiftGrid` command to redefine starting row and column for these groups of grids to act as one grid.

If isocurves of faces are not aligned, then using any of the commands that use surface uv-parameterization will probably not be useful. See the following example.
Use ptGridCurve command

The ptGridCurve command is useful if there is a general projection plane that covers target polysurface, such as a wiggly roof that runs generally horizontal. Using the previous example, we can generate a continuous grid using ptGridCurve and divide by number as in the following:
Use *ptGridCurve2* command

In cases when the polysurface has no linear edge along the extrusion direction, grid can be undesirable. To force a second direction, you can use *ptGridCurve2*. The following example shows result using one curve (*ptGridCurve*).
The paneling does not run right to the edge because the grid does not extend far enough. In general, it is best to create a grid that bleeds out of the boundaries of base polysurface. The following example uses the untrimmed base polysurface to generate the initial grid, then uses the trimmed polysurface as a reference when paneling.

Using one or two curves is also useful when dealing with polysurfaces created from revolved surfaces. The `ptGridCurve` and `ptGridCurve2` polar extrusion direction option is useful in these cases. See the following example.
Use approximate surface and Project or Pull command

Sometimes it is possible to define an approximate surface that is close enough to the target polysurface. In this case, you can grid the approximate surface and then use the Rhino **Pull** and **Project** commands to move the grid points to the polysurface.
4 Generate Paneling

The PanelingTools plug-in supports generating paneling patterns either by connecting paneling grid points or by mapping a given unit pattern to a unit grid.

Connecting paneling points is faster and does not involve time-consuming mapping.

Connecting Grid Points

`ptPanelGrid` command generates 2-D patterns from a base grid and allows access to pre-defined connecting patterns and user-defined patterns. The `ptPanel3D` command creates 3-D paneling.

The built-in connecting patterns are optimized for speed. 2-D connecting patterns include Box, BoxX, Triangular, TriBasic, Dense, Diamond, AngleBox, Wave, and Brick. Built-in 3-D patterns include WireBox, Partition, Box, Wedge, Pyramid1, and Pyramid2.

`ptManage2DPattern` and `ptManage3DPattern` commands create, edit, and delete custom patterns.

Mapping to Unit Grid

`ptPanelGridCustom` command supports mapping to populate 2-D free-form patterns. `ptPanel3DCustom` and `ptOrientToGrid` create 3-D patterns.

2-D Connecting Patterns

The `ptPanelGrid` command, Pattern option provides built-in connecting patterns. Built-in patterns are optimized for speed and should cover many of the common cases. The following illustration shows these patterns:
ptManage2DPatterns

Custom patterns are created by:
- Defining a unit pattern with any number of grid points.
- Drawing polylines to connect the points.
- Defining how many units the pattern shifts.
- Defining a unique pattern name or reference in paneling command (ptPanelGrid).

Here is an example using one closed polyline with $\text{GridWidth}=7$, $\text{GridHeight}=7$, $\text{ShiftX}=6$ and $\text{ShiftY}=6$.

Multiple polylines can be included in the same pattern. The following example used two closed polylines with $\text{GridWidth}=5$, $\text{GridHeight}=6$, $\text{Shift}_x=2$ and $\text{Shift}_y=4$. 
Command flow

1. Start the `ptManage2DPatterns` command.
2. Create a New pattern, Edit, or Delete existing patterns.
3. When selecting the New option, pick two points to define the scale of the preview grid.
4. Pick the grid points to define connecting pattern.
5. Press Enter to define additional connections.
6. Press Enter to accept that pattern.
7. Press Enter to save patterns and exit.

Options

New
Distance in first direction.

GridWidth
Number of grid points in the u-direction to connect.

GridHeight
Number of grid points in the v-direction to connect.

Shift_x
Shift of connecting pattern in the u-direction.

Shift_y
Shift of connecting pattern in the v-direction.

Reset
Clear all connections created for the current pattern.

Undo
Clear all last selection point.

Name
Pattern name.
**Edit**  
Select a pattern name from the list. The command flow and options are similar to when creating new pattern.

**Delete**  
Select the pattern name to be deleted from the list.

**Save and load custom 2D patterns**  
Created 2D patterns persist in a document and are saves with it, but if you like to share patterns across files, then it is best to save these patterns to an external text file.  
**ptSave2DPatterns** creates a test file of all 2D patterns in a given document. There is an option to append patterns to end of file.

Saved patterns can be loaded using **ptLoad2DPatterns**. If pattern name already exist in the custom patterns list, then it will be overwritten by the newly loaded pattern.

This is what the typical file with saved 2D patterns look like:

```
my2D.txt - Notepad
File Edit Format View Help

!-- Custom 2D patterns of PanelingTools plugin for Rhinoceros
!-- Recorded on Wednesday, November 18, 2009 at 14:46:52
!-- Pattern format:
!--   Name
!--   Shift: (x_shift, y_shift)
!--   Connections: (x0,y0)(x1,y1);(x2,y2)(x3,y3)(x4,y4);...

My_First2D
{1,1}
{0,0}(1,1)(1,0)(0,0)

My_Second2D
{2,1}
{0,0}(0,1)(1,0)(0,0);(1,0)(2,0)(2,1)(1,0)
```

**Options for ptSave2DPatterns**

**Append**  
If set to “Yes” then patterns are saved at the end of the file.

**SetTargetFile**  
Select target file. Accept default path or user may type the path at the command line.
3-D Connecting Pattern

`ptPanel3D` command comes with built-in 3-D patterns. The command help create edges, faces, and solids. The following illustration shows built-in patterns:

Built-in 3-D connecting patterns

**WireBox**
Curves connecting grid points.

**Partition**
Faces generated from connecting points in first and second bounding grids.

**Box**
Closed boxes connecting four points from each bounding grid.

**Wedge**
Closed trapezoid connecting three points from each bounding grid.

**Pyramid1**
Connects four points from the first grid with the corresponding minimum point from the second grid.

**Pyramid2**
Connect four points from first grid that extend over two spans with the corresponding middle point in the second grid.
**ptManage3DPattern**

3-D pattern require two bounding grids where a pattern is defined with polylines. Closed polylines result in faces that can be joined into a polysurface when pattern is later applied.

**Command flow**

1. Start the **ptManage3DPattern** command.
2. Create a **New** pattern, **Edit**, or **Delete** existing patterns.
3. When selecting the **New** option, pick two points to define the scale of the preview grid.
4. Pick the grid points to define connecting pattern. Closed polylines define faces.
5. Press **Enter** to define additional connections.
6. Press **Enter** to accept that pattern.
7. Press **Enter** to save patterns and exit.

**Options**

**New**
Distance in first direction.

**GridWidth**
Number of grid points in the u-direction to connect.

**GridHeight**
Number of grid points in the v-direction to connect.

**Shift_x**
Shift of connecting pattern in the u-direction.

**Shift_y**
Shift of connecting pattern in the v-direction.

**Reset**
Clear all connections created for the current pattern.
**Undo**
Clear all last selection point.

**Name**
Pattern name.

**Edit**
Select a pattern name from the list. The command flow and options are similar to when creating new pattern.

**Delete**
Select the pattern name to be deleted from the list.

### Save and load 3D custom patterns

Created 2D patterns persist in a document and are saved with it, but if you like to share patterns across files, then it is best to save these patterns to an external text file. 

**ptSave3DPatterns** creates a test file of all 2D patterns in a given document. There is an option to append patterns to end of file.

Saved patterns can be loaded using **ptLoad3DPatterns**. If pattern name already exist in the custom patterns list, then it will be overwritten by the newly loaded pattern.

This is what the typical file with saved 3D patterns look like:

```
my3D.txt - Notepad
File       Edit  Format  View  Help

!-------------------------------------------------------
!-- Custom 3D patterns of PanelingTools plugin for Rhinoceros
!-- Recorded on Wednesday, November 18, 2009 at 14:47:13
!-- Pattern format:
!--    Name
!--    Shift: (x_shift,y_shift)
!--    Connections: (x0,y0,z0)(x1,y1,z1),(x2,y2,z2)(x3,y3,z3);
!-------------------------------------------------------

My_First3D
(1,1)
(0,0,0)(0,0,1)(1,1,1)(1,1,0)(0,0,0);(0,1,1)(0,0,1)(1,0,0)(0,0,0)[0,1,1]

My_Second3D
(1,1)
(0,0,0)(0,1,1)(1,1,1)(1,1,0)(0,1,0)(0,0,0);(1,1,1)(1,0,1)[1,0,0]
```

**Options for ptSave3DPatterns**

**Append**
If set to “Yes” then patterns are saved at the end of the file.

**SetTargetFile**
Select target file. Accept default path or user may type the path at the command line
ptPanelGridCustom

The ptPanelGridCustom command uses a free-form pattern that cannot be represented by connecting grid points. The command scales a given pattern within a unit size then maps it to unit grid. The GridWidth and GridHeight options scale the pattern. Other options add spacing between each unit pattern and the next in the u- and v-directions.

Command flow:
1. Start the ptPanelGridCustom command.
2. Select base paneling grid.
3. Select base surface (optional).
4. Select pattern curves and points.
5. Press Enter to accept options.

Options

- **Base_u**
  Starting u index of the pattern in the grid.

- **Base_v**
  Starting v index of the pattern in the grid.

- **Shift_u**
  An integer value that defines packing density in the u direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

- **Shift_v**
  An integer value that defines packing density in the v direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

- **Length**
  Length of module or its x dimension that is mapped to one full length of the unit grid. If set the length to be bigger than pattern length then morphed module will be scaled down in the x direction to fit within same relative unit dimension.
Width
Width of module or its y dimension that is mapped to one full width of the unit grid. If set the width to be bigger than pattern width then morphed module will be scaled down in the y direction to fit within same relative unit dimension.

Group
If Yes, group the resulting pattern.

Name
Name of resulting paneling. A new layer is created with that name and each paneling object name starts with this name.

ptPanelGridCustomVariable
The ptPanelGridCustomVariable command is similar to ptPanelgridCustom command except that it allows user to scale, rotate, translate, define a list of shapes or generate mean curves between two shapes. Variation responds to surface curvature, attractors, vector or randomly. This command supports history, so changing the location of attractor points for example, updates the pattern.

Command flow:
1. Start the ptPanelGridCustomVariable command.
2. Select base paneling grid.
3. Select base surface (optional).
4. Press Enter to accept options.
5. (if applicable) Select attractor points or curves.
6. Select pattern curves and points.
7. Select bounding objects (optional – if press Enter, then pattern bounding box is used)
8. Press Enter to accept options.

Options

Base_u
Starting u index of the pattern in the grid.
**Base_v**
Starting $v$ index of the pattern in the grid.

**Shift_u**
An integer value that defines packing density in the $u$ direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**Shift_v**
An integer value that defines packing density in the $v$ direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**PatternMethod**
Method of varying input pattern.

**Scale**
Scale input unit pattern using base reference point and scale factor range.

**Scale1D**
Scale input unit pattern in one direction using base reference point, target reference point and scale factor range.

**Rotate**
Rotate input unit pattern using base reference point and angle range.
**Translate**
Move selected pattern between 2 points.

**List**
Enable selecting a list of patterns in order.

**Mean**
Calculate mean curves between two sets of input curves. The first curve of the first set is matched with the first curve of the second set.
**DistributionMethod**
Method of calculating distance factor for each square unit pattern. Distance factor is a normalized number between 0 and 1.

**GaussCurvature**
Use normalized surface (or grid) Gaussian curvature to define a factor between 0.0 and 1.0 for each square grid unit.

**MeanCurvature**
Use normalized surface (or grid) Mean curvature to define a factor between 0.0 and 1.0 for each square grid unit.

**PointAttractors**
The factor is based on the normalized distance from a set of points.
CurveAttractors
The factor is based on the normalized distance from a set of curves.

Vector
The factor is based on the normalized angle with the input vector.

Random
Randomly assign a factor between 0.0 and 1.0 for each square grid unit.

Bitmap
Use heightfield of an input image.

PullCurves
Pull resulting pattern curves to base surface (if available).

Group
If Yes, group the resulting pattern.

Name
Name of resulting paneling. A new layer is created with that name and each paneling object name starts with this name.

ptPanel3DCustom
The ptPanel3DCustom command scales a given 3-D-pattern bounding box to a unit grid box. (A unit grid is a box bounded by four points from first bounding grid and four points from second bounding grid.) Options scale the pattern and add spacing between each unit pattern and the next in u and v directions. The following is an example paneling uses two bounding grids that do not have to be parallel and maps the bounding box of the 3-D module to each of the one unit grid box:

Default

Increase length in y
**Command flow**

1. Start the `ptPanel3DCustom` command.
2. Select first bounding paneling grid.
3. Select second bounding paneling grid.
4. Select two bounding surfaces (optional).
5. Select 3D pattern (any type object).
6. Press **Enter** to accept options.

**Options**

**Base_u**
Starting `u` index of the pattern in the grid.

**Base_v**
Starting `v` index of the pattern in the grid.

**Shift_u**
An integer value that defines packing density in the `u` direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**Shift_v**
An integer value that defines packing density in the `v` direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**Length**
Length of module or its x dimension that is mapped to one full length of the unit grid. If set the length to be bigger than pattern length then morphed module will be scaled down in the x direction to fit within same relative unit dimension.

**Width**
Width of module or its y dimension that is mapped to one full width of the unit grid. If set the width to be bigger than pattern width then morphed module will be scaled down in the y direction to fit within same relative unit dimension.
**Height**
Height of module or its z dimension that is mapped to one full height of the unit grid. If set the height to be bigger than the module height then morphed module will be scaled down in the z direction to fit within same relative unit dimension.

**ptOrientToGrid**
The `ptOrientToGrid` command populates 3-D pattern objects to one paneling grid (and one surface if available). This command gives additional control to define input pattern base point, scale, and whether the mapping is rigid or deformed. Following is a rigid transform that uses three reference points (base, x and y):

If a fourth reference point is set, the object will be deformed when populating the grid.

**Command flow:**
1. Start the `ptOrientToGrid` command.
2. Select module objects. Press **Enter** when done.
3. Select module base point, x reference point and y reference point.
4. The command prompts for an optional fourth point if the module should be deformed. If the module needs to maintain its size, press **Enter**.
5. Select paneling grid.
6. Press **Enter** to accept options.
Options

**Base_u**
Starting \( u \) index of the pattern in the grid.

**Base_v**
Starting \( v \) index of the pattern in the grid.

**Shift_u**
An integer value that defines packing density in the \( u \) direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**Shift_v**
An integer value that defines packing density in the \( v \) direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**Extend_u**
Number of grid units in the \( v \) direction to use in the paneling grid.

**Extend_v**
Number of grid units in the \( v \) direction to use in the paneling grid.

**ptPanel3DCustomVariable**
The **ptPanel3DCustomVariable** command is similar to ptPanel3DCustom command except that it allows define a list of objects or generate mean surfaces between two input surfaces. Variation responds to surface curvature, attractors, vector or randomly.

**Command flow:**

1. Start the **ptPanel3DCustomVariable** command.
2. Select bounding paneling grids (second grid is optional).
3. Press Enter to accept options.
4. (If applicable) select attractor points or curves.
5. Select list of modules (of start and end surface for mean option).
6. Press **Enter** to accept options.

Options

**Base_u**
Starting \( u \) index of the pattern in the grid.

**Base_v**
Starting \( v \) index of the pattern in the grid.

**Shift_u**
An integer value that defines packing density in the \( u \) direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.

**Shift_v**
An integer value that defines packing density in the \( v \) direction of the paneling grid. Set the shift to 1 to pack the pattern. Set the spacing to 2 to map every second paneling pattern unit grid, etc.
**PatternMethod**  
Method of varying input pattern.

**List**  
Enable selecting a list of patterns in order.

**MeanSurfaces**  
Calculate mean surfaces between two input surfaces. Surfaces have to per-matched by the user or use one of the matching methods. The command generate mean surfaces is a fation similar to ptMeanSurfaces command.
MeanCurves
Calculate mean curves between two input curves. Curves have to per-matched by the user or use one of the matching methods. The command generate mean surfaces is a function similar to ptMeanCurves command.

DistributionMethod
Method of calculating distance factor for each square unit pattern. Distance factor is a normalized number between 0 and 1.

GaussCurvature
Use normalized surface (or grid) Gaussian curvature to define a factor between 0.0 and 1.0 for each square grid unit.

MeanCurvature
Use normalized surface (or grid) Mean curvature to define a factor between 0.0 and 1.0 for each square grid unit.

PointAttractors
The factor is based on the normalized distance from a set of points.

CurveAttractors
The factor is based on the normalized distance from a set of curves.

Vector
The factor is based on the normalized angle with the input vector.

Random
Randomly assign a factor between 0.0 and 1.0 for each square grid unit.

Bitmap
Use heightfield of an input image.

Group
If Yes, group the resulting pattern.
Name
Name of resulting paneling. A new layer is created with that name and each paneling object name starts with this name.

Paneling Planar Quadrangles
Creating planar panels from a free-form base surface is an active area of research. The PanelingTools plug-in provides very basic functionality to approximate an input paneling grid to maximize number of planar quadrangular panels. Following is the details of the command that supports this functionality.

ptPanelGridQuads
The ptPanelGridQuads command adjusts paneling grid to create maximum number of quads within tolerance.

One way to have better quadrangle coverage is to increase the maximum deviation, but that increases distortion. Another way is to use a dense grid.

Command flow:
1. Start the ptPanelGridQuads command.
2. Select paneling grid.
3. Select base surface or polysurface.
4. Press Enter to accept options.

Options

MaxDeviation
Deviation from base grid. Higher deviation=better coverage and more distortion.

Triangulate
If Yes, quadrangles that are not made planar are split into two triangles.

Group
If Yes, group the resulting points.

Name
Grid name prefix attached to each point. The row and column location complete the point name.
Paneling with a Grid

**ptPanelGrid**

The **ptPanelGrid** command creates panels with ordered grid of points. A reference surface or polysurface is optional. A list of patterns includes built-in and connecting patterns created with **ptManage2DPatterns**. Panels are added in the form of edges, surface borders, surfaces (EdgeSrf or Patch), flat surfaces, and a mesh.

**Command flow**

1. Start the **ptPanelGrid** command.
2. Select paneling grid.
3. Select base surface (optional).
4. Press **Enter** to accept options.

**Options**

**Pattern**

- Box, BoxX, Triangular, TriBasic, Dense, Diamond, AngleBox, Wave, Brick, and user-defined patterns.

**PanelShape**

- **Straight**
  - Line curve connection between points.
- **Pull**
  - Pull line connection to base surface, if available.
- **ShortPath**
  - Shortest path on the surface between grid points being connected.
- **Iso**
  - Isocurve between points if possible, otherwise pull back.
- **Projected**
  - Project line connection to surface using direction option.

**ProjectionDirection**

- **Z_dir**
  - Projection direction = world z-axis.
**X_dir**
Projection direction = world x-axis

**Y_dir**
Projection direction = world y-axis.

**CPlaneNormal**
Projection direction = normal to active construction plane.

**PickPoints**
Pick two points to define projection direction.

**AddEdges**
Add edge panels to a new layer. Edges are serialized.

**AddFacesBorder**
Add border curves panels to a new layer. Borders are serialized.

**AddFaces**
Add face panels (Patch or EdgeSrf) to a new layer. Faces are serialized.

**AddFlatFaces**
Create planar faces. Faces might not join. Faces are serialized and added to a new layer.

**FlatFaceMethod**
Specify how flat panels are calculated.

  **BestFit**
  Best-fit plane through all unit grid points.

  **FitBasePt0**
  Best-fit plane through three of the fours unit grid points starting from min u, min v and going clockwise.

  **FitBasePt1**
  Best-fit plane through three of the fours unit grid points starting from min u, max v and going clockwise.

  **FitBasePt2**
  Best-fit plane through three of the fours unit grid points starting from max u, max v and going clockwise.

  **FitBasePt3**
  Best-fit plane through three of the fours unit grid points starting from max u, min v and going clockwise.

  **TangentToCenter**
  Best-tangent plane to surface center.

**AddMesh**
Add a mesh.

**Group**
If **Yes**, group the resulting points.

**Name**
Grid name prefix attached to each point. The row and column location complete the point name.
**ptPanel3D**

The **ptPanel3D** command creates a panel from two grids. A list of patterns includes built-in and user-defined panels. Panels are added in the form of edges, surfaces, and polysurfaces. Spacing of unit patterns depends on grid spacing.

![Paneling Tool](image)

**Command flow:**

1. Start the **ptPanel3D** command.
2. Select first paneling grid. Set options for pattern in this step.
3. Select second paneling grid.

**Options**

- **Pattern**
  - WireBox, Partition, Box, Wedge, Pyramid1, Pyramid2, and user-defined patterns.
- **Group**
  - If **Yes**, group the resulting points.
- **Name**
  - Grid name prefix attached to each point. The row and column location complete the point name.

**Paneling without a Grid**

The **ptPanelSubDivide**, **ptPanelRandomPoints**, and **ptTriangulatePoints** commands do not create a base grid, but use either polylines or points on surface as an input to panel.

**ptPanelSubDivide**

The **ptPanelSubDivide** command recursively subdivides any number of polylines on a base surface. Each new polyline connects the midpoints of parent polyline segments.
Command flow

1. Start the **ptPanelSubDivide** command.
2. Select base surface.
3. Select polylines.
4. Press **Enter** when done.

Options

**Degree**
Number of subdividing steps

**Method**
If **Yes**, group the resulting points.

- **All**
  Subdivide all resulting sub polylines.
- **SubOnly**
  Subdivide sub curves only.
- **Main Only**
  Subdivide main curve only.

**PanelShape**
Straight or Pull.
ptPanelRandomPoints

The `ptPanelRandomPoints` command triangulates points on a base surface. Select any number of points on surface or have the command generate random points. The command solves triangulation of points on surface and solves shortest distance. It can be time consuming for big set of points. The `ptTriangulatePoints` command is more appropriate for big set of data.

Command flow

1. Start the `ptPanelRandomPoints` command.
2. Select base surface.
3. Select points on surface.
4. Press Enter when done.

Options

- **GenerateRandomly**
  If Yes, internally generate random points.

- **PointCount**
  Number of points to be generated.

- **PanelShape**
  Straight or Pull.
ptTriangulatePoints

The `ptTriangulatePoints` command uses Delauney triangulation to create a mesh from points. The command generates planar surfaces as an output.

Command flow

1. Start the `ptTriangulatePoints` command.
2. Select points.
3. Press Enter when done.
5 Paneling Output

Most paneling commands support multiple output formats: edges, surfaces, planar surfaces, patch surfaces and meshes. If a reference surface is used, panels can be pulled back to the surface. Using a base surface can trim panels to that surface. The following sections discuss paneling format and shape and address how panels are trimmed when using a reference surface or polysurface.

Paneling format

Panels can be curves, surfaces, or meshes. Each format might be desirable for a different situation. Here are few things to remember when deciding which format to use:

- Panels are labeled (serialized) and grouped in separate layers. One layer for each format.
- Straight edges and meshes are processed fastest.
- Creating surfaces can be time consuming especially when they are not planar and need to be trimmed. It is best to start with edges or a mesh when exploring design ideas and use surfaces during the final stages of design.
Rhino geometry used for each format

**Grid points**
ON_3dPoint

**Straight edges**
ON_Line or ON_LineCurve.

**Straight face borders**
ON_PolyLine or ON_PolylineCurve.

**Pulled edges**
ON_NurbsCurve.

**Flat faces**
Trimmed ON_Plane.

**Faces from straight edges**
ON_NurbsSurface from an EdgeSurface.

**Faces from pulled edges**
ON_NurbsSurface from an a Patch.

**Mesh**
ON_Mesh

Paneling shape

The paneling shape can only be straight when there is no reference surface or reference surface is planar. For example, edges are represented by lines connecting paneling grid, and custom patterns are mapped to a bilinear surface connecting unit grid. A straight shape can generate quickly and is recommended in intermediate stages of design.

If there is a reference surface for the paneling, Pulled, Iso and Projected shapes can be used. For example, when generating a simple box pattern, line edges connecting the grid are either pulled to the base surface, made to follow its isocurves (if applicable), or are projected to surface.

Here is an example that compares few of the paneling shapes:
Paneling shape

**Straight**
Line curve connection between points.

**Pull**
Pull straight line connection to base surface.

**ShortPath**
Shortest path on the surface between points being connected.

**Iso**
Isocurve between points if possible, otherwise pull back the straight line curve.

**Projected**
Project line connection to surface. Projection direction is user-defined.

Trimmed surfaces

When paneling uses a base surface or a polysurface, the panels are trimmed to the edges of that base. Selecting a base is optional in most cases and therefore it can be skipped if the user doesn’t wish to trim output.

Panels are trimmed regardless of what paneling shape is used. For example with "straight" shape, if a line curve happens to connect two point one within base and the other outside it, then the line is pulled to the base and the intersection point with the boundary is used as the new second point. In some cases this process fails to find useful result and few panels might need to be trimmed manually.
6 Utility Functions

Grid Utility Functions

The paneling grid can be modified directly using Rhino commands to project a grid on a surface, pull back, delete parts of it, or transform it (Move, Scale, Rotate, SoftMove, etc.). In general, any modification that does not change names of the grid of points is acceptable. Utility commands `ptDirection`, `ptRowsDirection`, `ptCompactGrid`, `ptCloseGrid`, `ptGridSeam`, `ptCleanOverlap`, `ptTrimGrid`, `ptOffsetPoints`, `ptChangeGridDensity`, `ptExtendGrid`, and `ptShiftGrid` help manipulate the paneling grid.

**ptDirection**

The `ptDirection` command flips u- and v-directions of the grid. This changes the names of point in the grid.

![Image of grid direction](image)

**Command flow**

1. Start the `ptDirection` command.
2. Select a grid.
3. Press Enter to accept options.

**Options**

- **UReverse**
  - If Yes, reverse u-direction of the grid.
- **VReverse**
  - If Yes, reverse v-direction of the grid.
- **Group**
  - If Yes, group the resulting points.
- **NameOfGrid**
  - Name of paneling grid. Default to selected grid point.

**ptSwapGridUV**

This command swaps the uv direction of a point grid. In other words, the u and v indices of each point are swapped.
**ptRowsDirection**

The `ptRowsDirection` command reverses rows directions of a paneling grid.

Flipped row direction causes flipped paneling

Use `ptRowsDirection` command to flip rows direction

**Command flow**

1. Start the `ptRowsDirection` command.
2. Select a grid.
3. Select a row base point to flip its direction.

**Options**

**Group**
If Yes, group the resulting points.

**NameOfGrid**
Name of paneling grid. Defaults to selected grid point.
**ptCompactGrid**

The *ptCompactGrid* command removes holes in the selected grid. The command compacts rows and columns of points.

**Command flow**

1. Start the *ptCompactGrid* command.
2. Select a grid.
3. Press Enter to accept.

**ptCloseGrid**

The *ptCloseGrid* command closes selected grid in u-, v- or both directions.

**Command flow:**

1. Start the *ptCloseGrid* command.
2. Select a grid.
3. Press Enter to accept.

**Options**

**Direction**
Close in U, V or Both directions

**Group**
If **Yes**, group the resulting points.

**NameOfGrid**
Name of paneling grid. Defaults to selected grid point.

**Overlap**
Number of rows/columns to overlap.

**StartIndex**
Index of the first row/column to overlap.

**ptGridSeam**
The **ptGridSeam** command changes the grid seam in a closed paneling grid. This command is useful when seam points do not align.

![Diagrams showing the effect of ptGridSeam command](image)

Shift=3, All=Yes to move all seam points three steps

**Command flow:**

1. Start the **ptGridSeam** command.  
2. Select a grid.  
3. Select seam points to shift. 
4. Press **Enter** when done

**Options**

**Direction (U/V)**
Appears when grid is closed in two directions.

**Shift**
Can be positive or negative number. Represents the number of steps a selected seam point moves by.

**All**
move all seam points together.

**Group**
If **Yes**, group the resulting points.
**NameOfGrid**
Name of paneling grid. Defaults to selected grid point.

**ptCleanOverlap**

The `ptCleanOverlap` command removes overlapped points in a paneling grid or merges them within tolerance. The command behaves different if points to be merged are in one row versus in one column. The command deletes grid points that are within tolerance in the u-direction. It moves points within tolerance in the v-direction to overlap.

![Grid and panels after merging near grid points with ptGridOverlap](image)

**Command flow:**

1. Start the `ptCleanOverlap` command.
2. Select a grid.
3. Press *Enter* to accept tolerance option.

**Options**

**Tolerance**
Maximum distance between points to be merged.
ptTrimGrid

The ptTrimGrid command trims a grid to a base surface or polysurface. It is possible to keep inside points, inside and points immediately outside, or move outside points to closest points on the edge.

Command flow:

1. Start the ptTrimGrid command.
2. Select a grid and select options.
3. Select base surface or polysurface.

Options

Mode

- **Inside**
  Select only inside points

- **Outside**
  Select inside points and ones immediately outside.

- **Edge**
  Move the nearest outside to the closest edge.

Group

If Yes, group the resulting points.
**NameOfGrid**  
Name of paneling grid. Defaults to selected grid point.

**ptOffsetPoints**  
The `ptOffsetPoints` command offsets points on surface or polysurface by a specified amount normal to that surface. An option connects input points with offset points.

![Diagram of ptOffsetPoints command](image)

**Command flow**

1. Start the `ptOffsetPoints` command.
2. Select a points (not necessarily part of a paneling grid) and pick options.
3. Select base surface or polysurface.

**Options**

- **DistanceMethod**
  - Fixed: Offset with fixed distance
  - GaussianCurvature: Use surface gaussian curvature values
  - MeanCurvature: Use surface mean curvature values.
  - AttractorPoints: Sift towards attractor points
  - SunVector: Use dot product between a vector and normal on surface at each point.
  - Random: Shift points by random amount

- **Distance** (when DistanceMethod=fixed)  
  Offset distance.

- **MinDistance** (when DistanceMethod is not fixed)  
  Minimum offset distance.

- **MaxDistance** (when DistanceMethod is not fixed)  
  Maximum offset distance.

- **Group**  
  If Yes, group the resulting points.

- **Connect**  
  Draw lines between points and their corresponding offset points.
**ptChangeGridDensity**

The `ptChangeGridDensity` command increases or decreases grid density by adding or removing points in the u-, v- or both directions. It is also possible to keep the same density in either direction.

**Command flow**

1. Start the `ptChangeGridDensity` command.
2. Select a grid and pick options.
3. Select base surface or polysurface.

**Options**

- **UDensity**
  U-direction density mode.
  - Increase
  - Decrease
  - Same

- **UNumber**
  Number of points to add/remove between each two original grid points.

- **VDensity**
  V-direction density mode.
  - Increase
  - Decrease
  - Same

- **VNumber**
  Number of points to add/remove between each two original grid points.

- **DeleteInput**
  Delete input grid.

- **Group**
  If Yes, group the resulting points.
NomeOfGrid
Name of paneling grid. Defaults to selected grid point.

ptExtendGrid
The ptExtendGrid command adds grid points in u-, v- or both directions at the specified distance. The extension direction is calculated relative to the direction between the last two points in a row or column.

Command flow
1. Start the ptExtendGrid command.
2. Select a grid.
3. Press Enter to accept options.

Options
UExtend
If Yes, extend the grid in the u-direction.
UNumber
Number of points to extend in the u-direction.
UDistance
Distance between points extended in the u-direction.
VExtend
If Yes, extend the grid in the v-direction.
VNumber
Number of points to extend in the v-direction.
VDistance
Distance between points extended in the v-direction.
Group
If Yes, group the resulting points.
NomeOfGrid
Name of paneling grid. Defaults to selected grid point.
ptShiftGrid

The **ptShiftGrid** command shifts the index of selected grid points by the specified amount. This helps space out a grid and create holes. It is also useful for combining existing grids.

![Shift in u direction by 1](image)

**Command flow**

1. Start the **ptShiftGrid** command.
2. Select a grid.
3. Press **Enter** to accept options.

**Options**

- **RowShift**
  Number of steps shifted in the u-direction.

- **ColShift**
  Number of steps shifted in the v-direction.

ptShuffleGrid

The **ptShuffleGrid** command uses reference surface and some parameters to redistribute a given grid. If a surface is not available, then one is created from the input grid on the fly and its parameteric space is used to shuffle the grid.

![variable distribution away from attractor points and curves](image)

![variable distribution towards attractor points and curves](image)
Command flow

1. Start the `ptShuffleGrid` command.
2. Select a grid.
3. Select a surface or press Enter to accept options.
4. If using attractor points or curves, then select reference points or curves.

Options

**DistanceMethod**
- **GaussianCurvature** Use surface gaussian curvature values
- **MeanCurvature** Use surface mean curvature values.
- **AttractorPoints** Sift towards/away from attractor points
- **AttractorCurves** Sift towards/away from attractor curves
- **SunVector** Use dot product between a vector and normal on surface at each point.
- **Random** Shift points by random amount
- **Bitmap** Use heightfield of an input image

**AttractorMethod**
Either away or towards attractor points or curves. If distance method is Mean or Gaussian, then attract towards or away from the highest curvature.

**Magnitude**
The default=1. Increasing the magnitude exadurates the effect.

**Group**
Option to group resulting grid.

**DeleteInput**
Option to delete input grid.

**ptConvertToDiagonal and ptConvertToDiamond**
These commands take a rectangular paneling grid and turn it into diagonal or diamond grids. This is useful when populating patterns that need to go in specific direction other than rectangular. Note that the result grid would not necessarily follow UV surface directions and therefore might not be able to properly modify them with commands like ptShuffleGrid.

In the following example, a circle is populated using `ptPanelGridCustom` command on a rectangular, diagonal and diamond grids respectively.
ptWeaveGrids
Create new grid by weaving rows from 2 input grids.

ptExtractCenterGrids
Extracts center point grid of the input grid. Center points are relevant to each four grid unit. There is an option to select a base surface to pull center points to.

ptMeanGrid
Make intermediate surfaces between two input grids. The command does not align seam location for closed surfaces or match UV direction or parameterization.

Command flow
1. Start the ptMeanGrid command.
2. Select start grid.
3. Select end grid.
4. Set number of intermediate grids or press Enter to accept default number.
Options

**NumberOfGrids**
Number of mean grids

**CreateSrf**
Use intermediate grid to create a surface

**Group**
Group output.

**ptOffsetGridByHeightfield**
Offset a paneling grid variably by some min and max offset distance using an image heightfield.

Command flow

1. Start the `ptOffsetGridByHeightfield` command.
2. Select a paneling grid and pick options.
3. Select base surface or polysurface if available.

Options

**MinDistance** (when DistanceMethod is not fixed)
Minimum offset distance.

**MaxDistance** (when DistanceMethod is not fixed)
Maximum offset distance.

**CreateSurface**
If Yes, the a surface that goes through the new grid is created.

**Group**
If Yes, group the resulting points.

**DeleteInput**
Delete input grid.
Paneling Utility Functions

Panels generated with paneling commands are standard Rhino geometry. They can be curves, surfaces, polysurfaces, or meshes. Any Rhino command can be used to manipulate them. In addition, `ptExtrudeEdges`, `ptOffsetEdges`, `ptFinEdges`, `ptUnifyFacesDirection`, `ptAnalyzeFlatFaces`, `ptGroupSimilarPanels`, `ptUnrollFaces`, `ptUnrollEdges`, `ptUnrollPoints`, `ptOffsetBorder`, and `ptPlanarLips` commands can help special situations.

**ptExtrudeEdges**

The `ptExtrudeEdges` command extrudes paneling edges normal to a base surface or a specified direction.

**Command flow**

1. Start the `ptExtrudeEdges` command.
2. Select curve panels.
3. Select base surface (optional), or pick two points for direction.

**Options**

**HeightMethod**

- **Fixed**: Extrude with fixed distance
- **GaussianCurvature**: Use surface gaussian curvature values
- **MeanCurvature**: Use surface mean curvature values.
- **AttractorPoints**: Sift towards attractor points
- **SunVector**: Use dot product between a vector and normal on surface at each point.
- **Random**: Shift points by random amount

When `HeightMethod=Fixed`, following option appear:

- **Height**: Extrude distance

When `HeightMethod= [all other options]`, following options appear:

- **MinHeight**: Minimum extrude distance
- **MaxHeight**: Maximum extrude distance
- **AttractMethod**: Towards or Away

**AddNotch**

Option to add notches to extuded edges. If set to **Yes**, the following options appear:

- **NotchWidth**: Width of the notch
**NotchHeight**  
Height of the notch  

**NotchSide**  
Which side to apply the notch (*Start, End, Both*).

**NameEnding**  
Suffix added to edge name to serialize extruded parts.

**ptOffsetEdges**

The `ptOffsetEdges` command offsets paneling edges using base surface. If no base is available, use `ptSurfaceFromGridOfEditPoints` command to create one.

```
connect=Yes  
connect=No
```

**Command flow**

1. Start the `ptOffsetEdges` command.  
2. Select curve panels.  
3. Select base surface.

**Options**

**DistanceMethod**

- **Fixed**  
  Offset with fixed distance  
- **GaussianCurvature**  
  Use surface gaussian curvature values  
- **MeanCurvature**  
  Use surface mean curvature values.  
- **AttractorPoints**  
  Sift towards attractor points  
- **SunVector**  
  Use dot product between a vector and normal on surface at each point.  
- **Random**  
  Shift points by random amount  

**Distance** (when `DistanceMethod`=fixed)  
Offset distance.

**MinDistance** (when `DistanceMethod` is not fixed)  
Minimum offset distance.

**MaxDistance** (when `DistanceMethod` is not fixed)  
Maximum offset distance.

**Connect**  
If *Yes*, connect offset edges with input edges.

**NameEnding**  
Prefix added to edge name to serialize extruded parts.
ptFinEdges
The **ptFinEdges** command extrudes paneling edges using base surface. The fin can be on one or both sides.

**Command flow**
1. Start the **ptFinEdges** command.
2. Select curve panels.
3. Select base surface.

**Options**

- **DistanceMethod**
  - **Fixed** Extrude with fixed distance
  - **GaussianCurvature** Use surface gaussian curvature values
  - **MeanCurvature** Use surface mean curvature values.
  - **AttractorPoints** Sift towards attractor points
  - **SunVector** Use dot product between a vector and normal on surface at each point.
  - **Random** Shift points by random amount

When **DistanceMethod=Fixed**, following option appear:

- **Distance** Extrude distance

When **DistanceMethod= [all other options]**, following options appear:

- **Min Distance** Minimum extrude distance
- **Max Distance** Maximum extrude distance
- **AttractMethod** Towards or Away

**AddNotch**
Option to add notches to extruded edges. If set to **Yes**, the following options appear:

- **NotchWidth** Width of the notch
- **NotchHeight** Height of the notch
- **NotchSide** Which side to apply the notch (Start, End, Both).
- **FlipNotch** Define if the notch is applied near the edge or away from it.

**NameEnding**
Suffix added to edge name to serialize extruded parts.

**BothSides**
If **Yes**, fin both sides.
**ptUnifyFacesDirection**

The **ptUnifyFacesDirection** command orients u-, v-, and normal directions of the input faces relative to a base surface.

![Few input faces have flipped normal and swapped uv compared to a base surface](image)

**Command flow**

1. Start the **ptUnifyFacesDirection** command.
2. Select input faces.
3. Select base surface to use its u-, v-, and normal direction as a reference.

**Options**

- **UnifyUV**
  Match reference surface u- and v-directions.

**ptAnalyzeFlatFaces**

The **ptAnalyzeFlatFaces** command creates an analysis mesh to show amount of deviation of flat surfaces from their base surface. When creating flat surfaces with the **ptPanelGrid** command, the deviation amount is saved as user data on each surface. This information is used to create the analysis mesh.
Command flow

1 Start the **ptAnalyzeFlatFaces** command.
2 Select flat faces (faces created with **ptPanelGrid FlatFaces** option).
3 Press **Enter** when finished.

**ptGroupSimilarPanels**

The **ptGroupSimilarPanels** command groups similar curves together within a given tolerance. Similar curves have similar edge length within tolerance.

Command flow

1 Start the **ptGroupSimilarPanels** command.
2 Select input paneling curves.
3 Press **Enter** to complete.

Options

**Tolerance**
Difference in edge length allowed for panels to be considered similar.

**ptUnrollFaces**

The **ptUnrollFaces** command unrolls surfaces that do not have to be joined in one polysurface and keeps input surface attributes (name, user-data, etc.).

Command flow

1 Start the **ptUnrollFaces** command.
2 Select faces.
3 Press **Enter** to complete.
ptTriangulateFaces
The *ptTriangulateFaces* command processes paneling faces that have 4 or more edges and triangulate them into triangular nurbs faces that can be joined into one polysurface.

**Command flow**

1. Start the *ptTriangulateFaces* command.
2. Select paneling faces.
3. Press Enter to complete.

**Options**

*DeleteInput*
Difference in edge length allowed for panels to be considered similar.

*Join*
Difference in edge length allowed for panels to be considered similar.

ptUnrollFaces
The *ptUnrollFaces* command unrolls surfaces that do not have to be joined in one polysurface and keeps input surface attributes (name, user-data, etc.).

**Command flow**

1. Start the *ptUnrollFaces* command.
2  Select faces.
3  Press **Enter** to complete.

**Options**

**Explode**
If **Yes**, explode the unrolled faces.

**Label**
Places matching numbered dots on the edges of the original polysurface and the flattened surfaces.

**Layer**
- **Current**
  Add unrolled faces to current layer.
- **NewSubLayer**
  Add unrolled faces to a new sub layer.
  - **SubLayerName**
    Name of the sub layer.

**ptUnrollEdges**
The **ptUnrollEdges** command unrolls edges using a base surface or polysurface. Attributes of input edges are passed to unrolled ones.
Command flow

1. Start the **ptUnrollEdges** command.
2. Select curves on a surface or a polysurface.
3. Select the base surface or polysurface.

Options

Layer
- **Current**
  Add unrolled edges to current layer.
- **NewSubLayer**
  Add unrolled edges to a new sub layer.
- **SubLayerName**
  Name of the sub layer.

**ptUnrollPoints**

The **ptUnrollPoints** command unrolls points using a base surface or polysurface. Attributes of input points are passed to unrolled ones.

Command flow

1. Start the **ptUnrollPoints** command.
2. Select points on a surface or a polysurface.
3. Select the base surface or polysurface.

Options

Layer
- **Current**
  Add unrolled points to current layer.
- **NewSubLayer**
  Add unrolled points to a new sub layer.
- **SubLayerName**
  Name of the sub layer.

**ptOffsetBorder**

The **ptOffsetBorder** command offsets faces borders inwards with an option to create a hole.
Command flow

1. Start the `ptOffsetBorder` command.
2. Select paneling faces.
3. Press Enter to accept.

Options

**DistanceMethod**
- **Fixed** Offset with fixed distance
- **GaussianCurvature** Use surface gaussian curvature values
- **MeanCurvature** Use surface mean curvature values
- **AttractorPoints** Sift towards attractor points
- **SunVector** Use dot product between a vector and normal on surface at each point.
- **Random** Shift points by random amount

**Distance** (when DistanceMethod=fixed)
Offset distance on surface.

**MinDistance** (when DistanceMethod is not fixed)
Minimum offset distance.

**MaxDistance** (when DistanceMethod is not fixed)
Maximum offset distance.

**MakeHole**
If Yes, use offset curve to drill a hole in the face.

**ptPlanarLips**
The `ptPlanarLips` command generates edge extrusions (lips) to planar surfaces. This is useful for unrolled faces that will be used for fabrication. The command uses a reference polyline to define the offset directions and the offset distance for each direction.
Command flow

1. Start the `ptPlanarLips` command.
2. Select a polyline to use its edges as a direction reference.
3. Select faces to add lips.
4. Press `Enter` to accept.

Options

**Output**
- **Curve**
- **Surface**

**ConnectEdges**
- If `Yes`, extend and connect offset edges of input face.

**TypeOfDistance**
- **Uniform**
  - Distance is the same in all directions.
- **Variable**
  - Each direction can be set to a different offset distance.

**DeleteInput**
- If `Yes`, delete input faces.

**Distance**
- Offset distance. If `TypeOfDistance = Variable`, each tagged edge will have a separate distance value to set (D1, D2, etc.)

General Utility Functions

The utility functions `ptDivideCurveSpan`, `ptDivideCurveByChordLength`, and `ptSurfaceFromGridOfControlPoints` divide curves, create NURBS surfaces from paneling grid, and label data.

**ptDivideCurveSpan**

The `ptDivideCurveSpan` command finds curve division points by number or distance along curve.
Command flow

1. Start the `ptDivideCurveSpan` command.
2. Select curves.
3. Select Enter to accept.

Options

Method
- **Number**
  - **NumberOfSpans**
  - Number of spaces between points on curve.
- **ArcLength**
  - **Length**
  - Along-curve distance between divide points.
- **Round**
  - If Yes, round the distance up or down to fill the whole curve.
- **RoundingMethod**
  - Up
  - Down

Group
- If Yes, group resulting points.

`ptDivideCurveByChordLength`

The `ptDivideCurveByChordLength` command finds curve divide points by chord length (straight-line distance) between points. The algorithm uses sphere intersections with the curve to find points.
**Command flow**

1. Start the `ptDivideCurveByChordLength` command.
2. Select curves.
3. Select Enter to accept.

**Options**

- **Distance**
  Straight-line distance between points.

- **AddEndPoint**
  If Yes, add a point to the end of the curve.

- **Group**
  If Yes, group resulting points.

---

**ptSurfaceFromGridOfEditPoints**

The `ptSurfaceFromGridOfEditPoints` command creates a NURBS surface through the grid points using the grid points as surface edit points.

---

**Command flow**

1. Start the `ptSurfaceFromGridOfEditPoints` command.
2. Select grid points.
3. press Enter to complete.

**ptSurfaceFromGridOfControlPoints**

The `ptSurfaceFromGridOfControlPoints` command creates a NURBS surface using the point grid as surface control points.
Command flow

1. Start the `ptSurfaceFromGridOfControlPoints` command.
2. Select grid points.
3. Press `Enter` to complete.

**ptUnifyCurvesDirection**

The `ptUnifyCurvesDirection` command unifies the direction of curves to point in the same general direction.

**ptTagObjects**

The `ptTagObjects` command tags objects with their names as text or dots.

<table>
<thead>
<tr>
<th>Tag with dot</th>
<th>Tag with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>G(2)[0]</td>
<td>G(2)[1]</td>
</tr>
<tr>
<td>G(1)[0]</td>
<td>G(1)[1]</td>
</tr>
<tr>
<td>G(0)[0]</td>
<td>G(0)[1]</td>
</tr>
</tbody>
</table>
**Options**

**TagMode**
- **Dot**
  Tag with dots.
- **Text**
  Tag with text.
- **Height**
  Text height.

**ptSerializeObjects**
The **ptSerializeObjects** command adds a serialized name to objects (points, curves, and surfaces).

**Options**

**SortMethod**
Sort using one of the following four methods.
- **OrderOfSelection**
  Selection order.
- **Coordinates**
  World coordinates.
- **Direction**
  User-defined direction.
- **Surface**
  Reference surface.

**Prefix**
Name prefixed to serial number.

**StartIndex**
Starting number.

**ptMeanCurves**
Make intermediate curves between two input curves. The command does not align seam location for closed curves. It is recommended to run CrvSeam command first to check and align seam location.
Command flow

1. Start the **ptMeanCurves** command.
2. Select start curve.
3. Select end curve.
4. Set number of intermediate curves or press Enter to accept default number.

Options

**NumOfCurves**
Number of mean curves.

**MakeCompatible**
If set to “No”, then the nurbs structure of the input curves is not changed. The user has to check that both curves have same number of control points before running the command.

**ptMeanSurfaces**
Make intermediate surfaces between two input surfaces. The command does not align seam location for closed surfaces or match UV direction or parameterization. T works best if create start surface, then copy and edit the end surface.

Command flow

1. Start the **ptMeanSurfaces** command.
2 Select start surface (untrimmed).
3 Select end surface (untrimmed).
4 Set number of intermediate surfaces or press Enter to accept default number. There is an option to generate a surface with some distance factor between the two surfaces. The factor is between 0 and 1.

Options

Method
Sort using one of the following four methods.

ByNumber
Specify number of mean surfaces

ByDisFactor
Specify

CreateSrf
Use intermediate grid to create a surface

Group
Group output.

ptRemoveOverlapedPoints
This is a cleaning function to remove overlapped points. Can be used before calling one of the serialize objects or serialize points commands to avoid duplicates.

Serializing joints and connections for FE analysis
There is a set of commands to add user data to serialize and store information on points and edges. Also to label geometry and export a formatted text.

ptSerializePoints
Take a set of points and serialize relative to coordinates and add serial numbers with user string to the point object. ptTagSerializedData can be used to visualize the data in Rhino.

Options

DataString
User may store data associated with the selected points. Set to null if no data is available.

StartIndex
Starting index for serial numbers.

ptSerializeEdges
Take a set of edges and serialize relative to coordinates and add serial numbers with user string to the point object. ptTagSerializedData can be used to visualize the data in Rhino. This command should be called after serializing end points of the edges using ptSerializePoints. The information about serial numbers of end points is also stored in the edges user data. If end points are not serialized, then this command will serialize them first.
### Options

**DataString**
User may store data associated with the selected edges. Set to null if no data is available.

**StartIndex**
Starting index for serial numbers.

### ptTagSerializedData
Tag objects (points and edges) with the user data created with ptSerializePoints and ptSerializeEdges commands.

### Options

**TagMode**
- **Dot**
  Tag with dots.
- **Text**
  Tag with text.

**Height**
Text height.

**DisplaySerialOnly**
Set to "Yes" to ignore edge end points serial number and point/edge data string.

### ptExportPointsSerializeData
Create a text file with serialize data created with ptSerializePoints command.

#### Command flow

1. Start the `ptExportPointsSerializeData` command.
2. Select points and set options.
3. Set target file

#### Options

**AddPointsCoordinates**
Output points coordinates (x, y and z).
Description
User title. Set to “null” if none is available.

StartString
Prefix string to be added before each point data

EndString
Suffix string to be added after each point data

NewLine
If set to “Yes” then separate points output by a new line

PrintPointSerial
If set to “No” then the serial number will not show

Append
Append to existing file

SetTargetFile
Click this option to set the file

ptExportEdgesSerializeData
Create a text file with serialize data created with ptSerializeEdges command.

Command flow
1. Start the ptExportEdgesSerializeData command.
2. Select points and set options.
3. Set target file

Options
Description
User title. Set to “null” if none is available.

StartString
Prefix string to be added before each edge data

EndString
Suffix string to be added after each edge data

NewLine
If set to “Yes” then separate edges output by a new line

PrintEdgeSerial
If set to “No” then the serial number will not show

Append
Append to existing file

SetTargetFile
Click this option to set the file
7 Extending RhinoScript

For a full description of PanelingTools methods exposed to RhinoScript go to
http://en.wiki.mcneel.com/default.aspx/McNeel/PanelingScripting.html

In addition, a few samples are included in PanelingTools toolbar.

To access and edit sample codes

- In the Paneling Tools toolbar, Shift+right-click the About toolbar button.