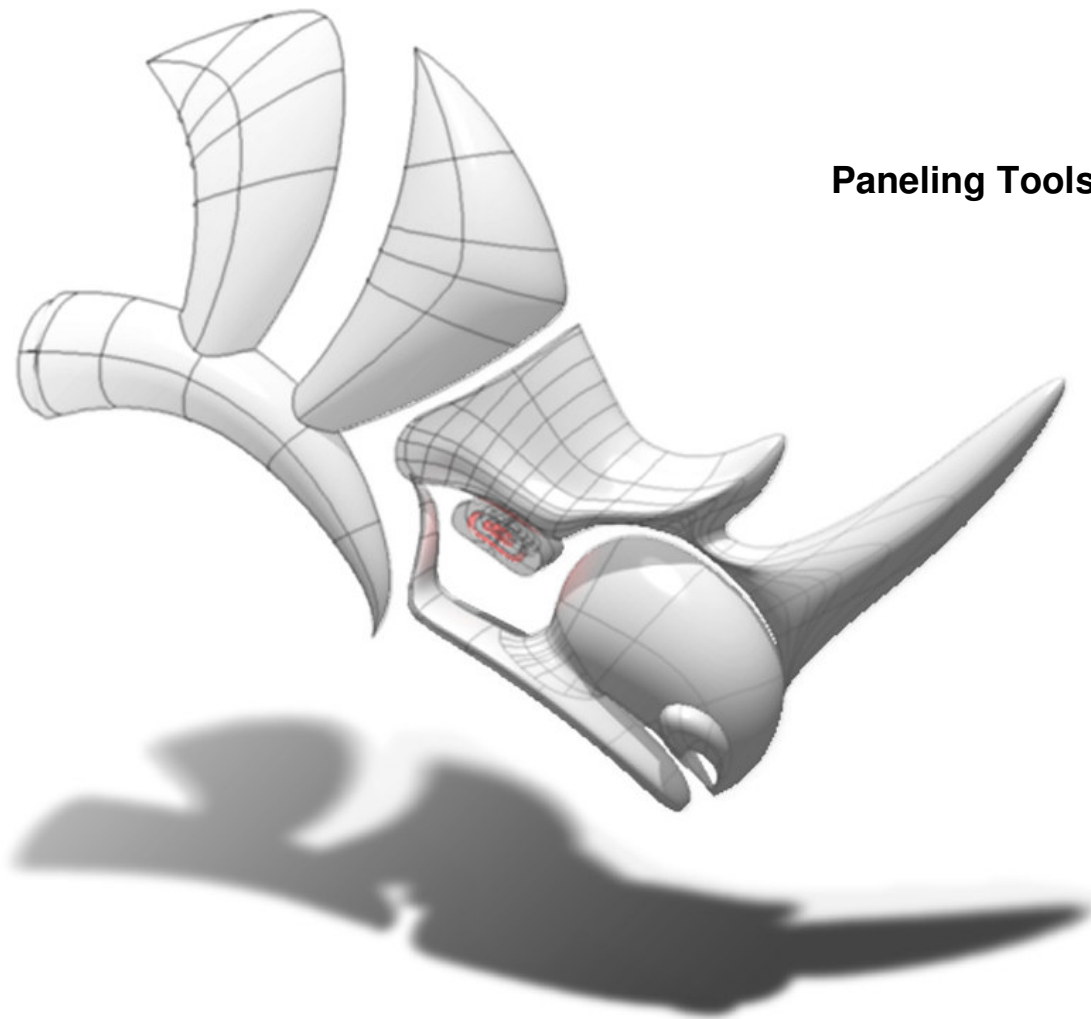


Rhinoceros®

NURBS modeling for Windows

Paneling Tools



Copyright © 2009 Robert McNeel & Associates. All rights reserved.

Rhinoceros is a registered trademark and Rhino is a trademark of Robert McNeel & Associates.

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
rajaa@mcneel.com

Table of Contents

Note from the Author.....	ii
1 Getting Started	1
Toolbars and Menu.....	1
Check for Updates.....	2
Help and Support.....	2
History Support	2
2 Introduction	3
Paneling process.....	3
3 Create a Paneling Grid.....	4
Create Paneling Grid Directly	4
ptGridArray.....	4
ptGridArrayPolar	5
Grid from Predefined Points	6
ptGridPoints.....	6
ptGridPointsOnSurface.....	7
Grid from Curves	8
ptGridExtrude1.....	8
ptGridExtrude2.....	10
ptGridUCurves.....	11
ptGridUVCurves.....	12
Grid from Surfaces.....	13
ptGridSurfaceUV	13
ptGridSurfaceDomain	14
ptGridSurfaceDistance	16
Grid from Projected Curves on Surface or Polysurface	17
ptGridCurve (one directional curve).....	17
ptGridCurve2 (two directional curves)	19
Grid from RhinoScript.....	21
Grid from Grasshopper	23
Create Paneling Grid for Polysurfaces	24
Use ptGridSurfaceUV with ArcLength method.....	24
Use ptGridCurve command	25
Use ptGridCurve2 command	26
Use approximate surface and Project or Pull command	28
4 Generate Paneling.....	29
Connecting Grid Points.....	29
Mapping to Unit Grid	29
2-D Connecting Patterns	29
ptManage2DPatterns	30

3-D Connecting Pattern	32
ptManage3DPattern	33
ptPanelGridCustom	34
ptPanel3DCustom	35
ptOrientToGrid	36
Paneling Planar Quadrangles	37
ptPanelGridQuads	38
Paneling with a Grid	38
ptPanelGrid	38
ptPanel3D	41
Paneling without a Grid.....	41
ptPanelSurface	42
ptPanelCurve	44
ptPanelSubDivide	47
ptPanelRandomPoints	48
ptTriangulatePoints	49
5 Paneling Output.....	50
Paneling format.....	50
Paneling shape.....	51
Trimmed surfaces	52
6 Utility Functions	53
Grid Utility Functions.....	53
ptDirection	53
ptRowsDirection	54
ptCompactGrid	55
ptCloseGrid	55
ptGridSeam	56
ptCleanOverlap	57
ptTrimGrid.....	58
ptOffsetPoints	59
ptChangeGridDensity	60
ptExtendGrid	61
ptShiftGrid	62
Paneling Utility Functions	62
ptExtrudeEdges.....	62
ptOffsetEdges	63
ptFinEdges	64
ptUnifyFacesDirection	64
ptAnalyzeFlatFaces	65
ptGroupSimilarPanels	66
ptUnrollFaces	66
ptUnrollEdges	67

ptUnrollPoints.....	68
ptOffsetBorder.....	68
ptPlanarLips.....	69
General Utility Functions	70
ptDivideCurveSpan.....	70
ptDivideCurveByChordLength.....	71
ptSurfaceFromGridOfEditPoints.....	72
ptSurfaceFromGridOfControlPoints.....	72
ptUnifyCurvesDirection	73
ptTagObjects.....	73
ptSerializeObjects	74
7 Extending RhinoScript.....	75

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 load and install **PanelingTools.rhp**.

Or,

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

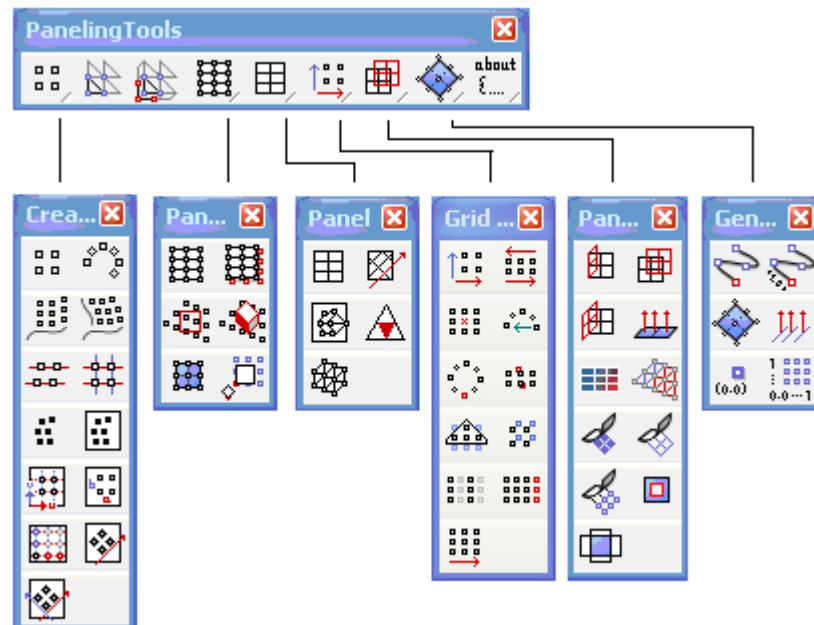
Note: Install the plug-in only once. After the initial installation, PanelingTools automatically loads when Rhino starts.

Toolbars and Menu

The PanelingTools toolbar file (**PanelingTools.tb**) file includes all plug-in commands.

To load toolbars:

- 1 Start the **Toolbars** 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, up-to-date documentation, and a command-by-command description, 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

- The PanelingTools Wiki page contains the latest download, documentation, and links to tutorials and short videos:
<http://en.wiki.mcneel.com/default.aspx/McNeel/PanelingTools.html>
- 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 small grids, but can be very time consuming for larger ones.

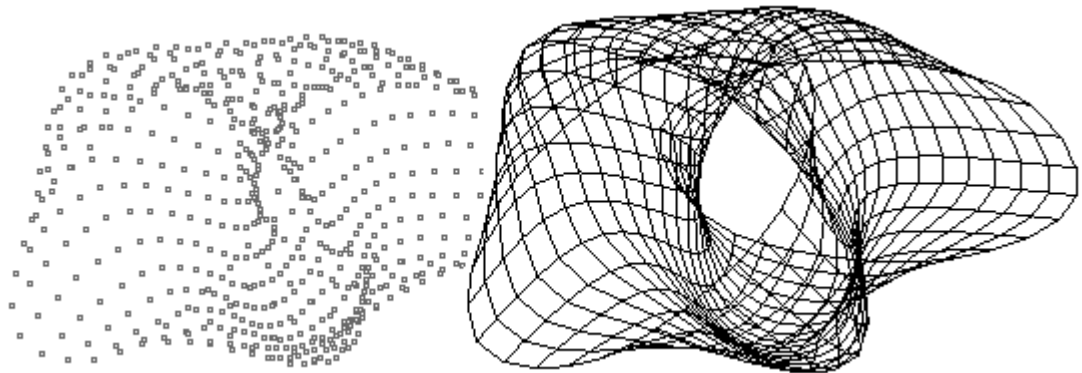
Here are few useful things to keep in mind when using history:

- Although history can be useful, it can be slow and counter-intuitive when dealing with a large set of data.
- History 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 recalculation in most cases.

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.

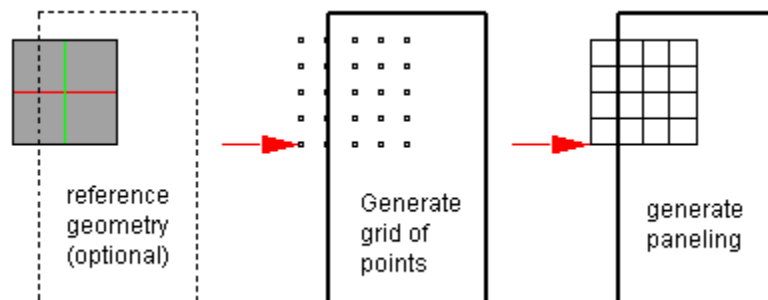


Paneling Process

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.



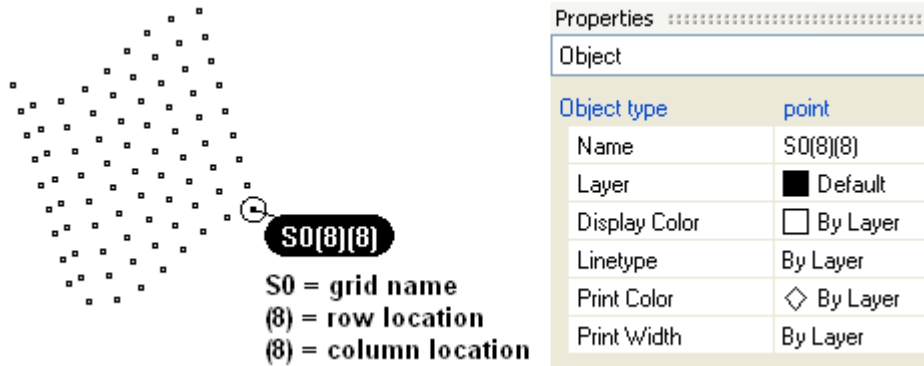
The two-step process gives more flexibility and better control over the result. Normally, the initial grid is generated interactively and is a good indicator of scale. The grid can be generated using the many grid-generating commands or with scripting. The grid can be directly edited and refined before 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

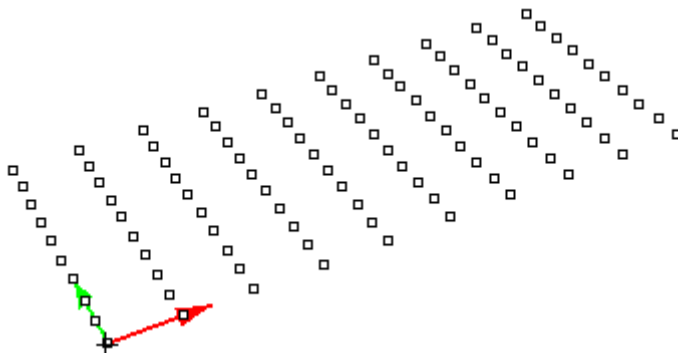
Since the names can be edited directly using the Rhino **Properties** command, points that are valid input for paneling can also be created using the Rhino **Points** command, and names can be assigned to reflect their locations with the **Properties** command. PanelingTools grid creation commands do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

Indirection

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

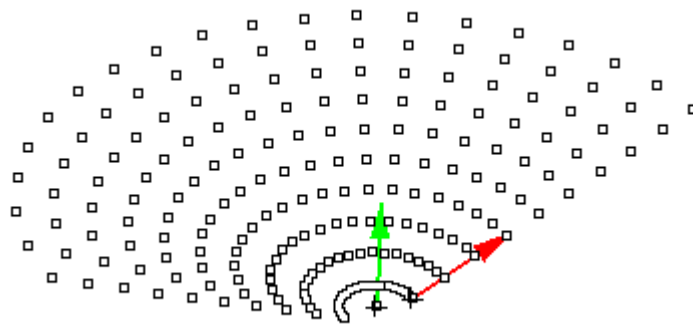
If **Yes**, group the resulting points.

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).

Angle

Angle between points.

Group

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

NameOfGrid

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

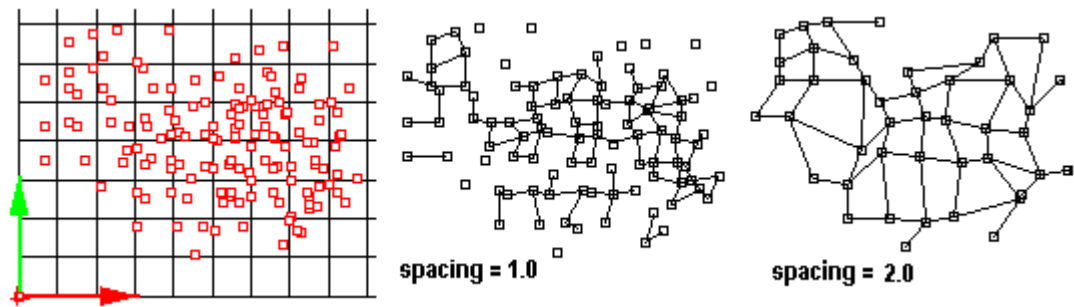
Grid from Predefined Points

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

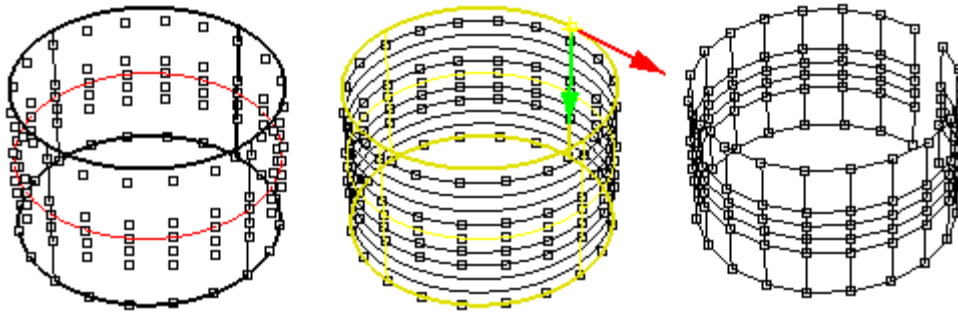
If **Yes**, shift the points to align with reference grid.

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 ($2 \times \text{**Tolerance**}$). 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 uses 2 times this value. Points should be within **Tolerance** to be included in the grid.

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

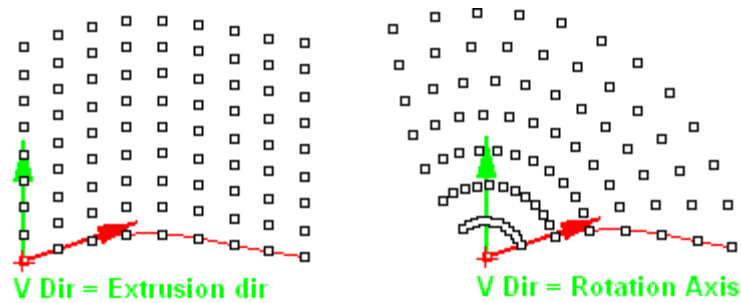
Grid name prefix attached to each point. The row and column location complete the point name.

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.

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_ChordLength Straight-line distance between points.
	U_AddEndPoint If Yes , add a point at the end.

V_Number

Number of points in the extrusion v-direction.

V_Method

Array curve divide points in parallel or polar direction.

Parallel	V_Distance Distance between points.
	V_Direction Pick two points to specify a v-direction.
Polar	V_Angle Angle between rows of points.
	V_RotationAxis Pick two points to specify a rotation axis.

Group

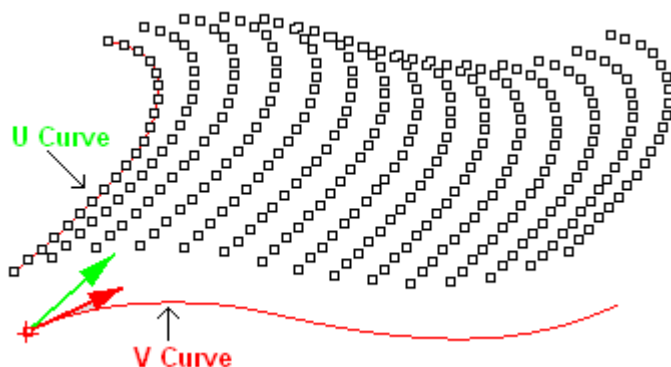
If **Yes**, group the resulting points.

NameOfGrid

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_RoundMethod

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_RoundMethod

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

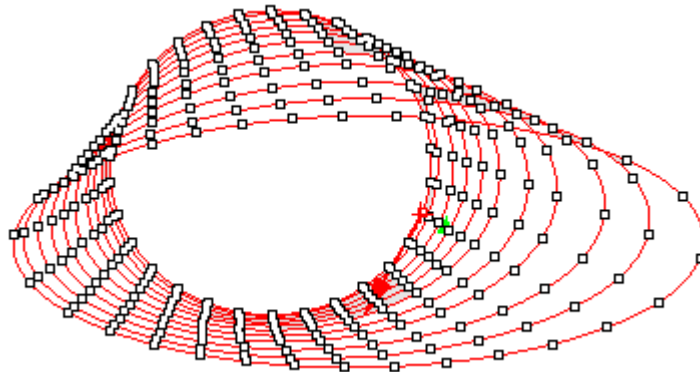
Grid name prefix attached to each point. The row and column location complete the point name.

SwitchCurves

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

ptGridUCurves

The **ptGridUCurves** command uses an existing array of parallel curves to create a paneling grid. Divide curves (usually parallel or non-intersecting). 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**, sort curves relative to their midpoint.

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.

RoundMethod

Up

Down

ChordLength

ChordLength

Straight-line distance between points.

AddEndPoint

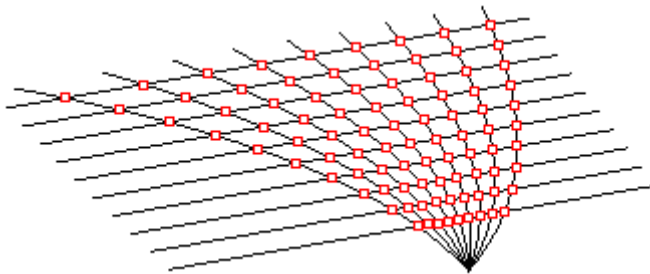
If **Yes**, add a point at the end.

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

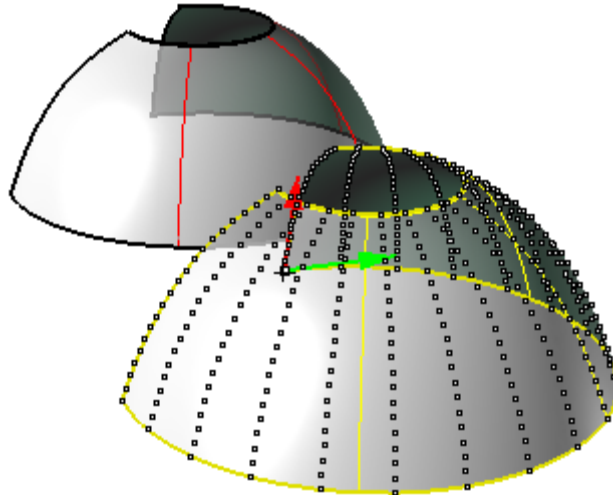
If **Yes**, group the resulting points.

Grid from Surfaces

The **ptGridSurfaceUV**, **ptGridSurfaceUV**, and **ptGridSurfaceDistance** use a base NURBS surface to generate a grid.

ptGridSurfaceUV

The **ptGridSurfaceUV** 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 **ptGridSurfaceUV** command using **Properties** command.



Command flow

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

The algorithm:

- 1 Extracts an isocurve in the v-direction at the minimum u-domain.
- 2 Divides the curve using V_Method.
- 3 At each divide point, extracts the isocurve in u-direction and divides that curve using U_Method.
- 4 Appends resulting points as a row.

Options

U_Method

U-direction 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_RoundMethod

Up

Down

ChordLength**U_ChordLength**

Straight-line distance between points.

V_Method

V-direction 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_RoundMethod

Up

Down

ChordLength**V_ChordLength**

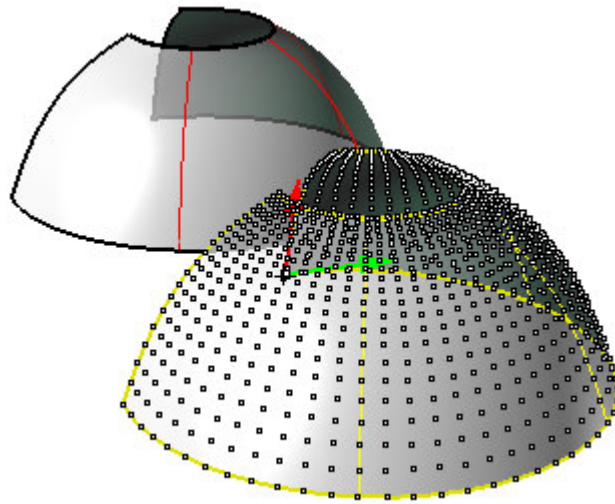
Straight-line distance between points.

Group

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

ptGridSurfaceDomain

The **ptGridSurfaceDomain** 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 **ptGridSurfaceUVDomain** command using the **Properties** command.



Command flow

- 1 Start the **ptGridSurfaceDomain** 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:

- 1 Extracts isocurves in the u- and v-directions at the selected base point.
- 2 Divides isocurves using U_Method and V-Method and gets a parameter value at each point.
- 3 Uses u- and v-parameters list to divide the surface domain.

Options

U_Method

U-direction 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_RoundMethod Up Down
ChordLength	U_ChordLength Straight-line distance between points.

V_Method

V-direction 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_RoundMethod Up Down UVDirection Change grid base point through changing u-, v- or both directions. Ureverse Vreverse ReverseBoth Default SelectBasePoint Select a base point on surface.
ChordLength	V_ChordLength Straight-line distance between points. UVDirection Change grid base point by changing the u-, v- or both directions. Ureverse Vreverse ReverseBoth Default SelectBasePoint Select a base point on surface.
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.

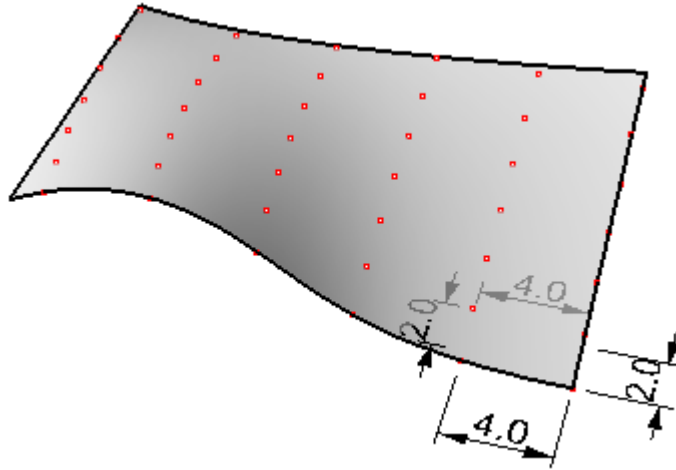
Command flow

- 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.

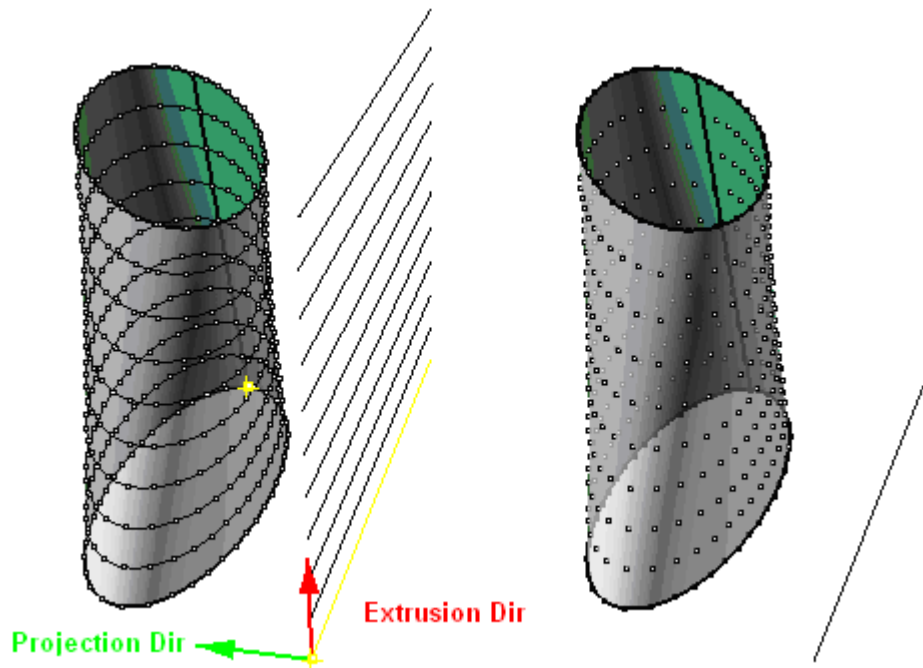
Grid from Projected Curves on Surface or Polysurface

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

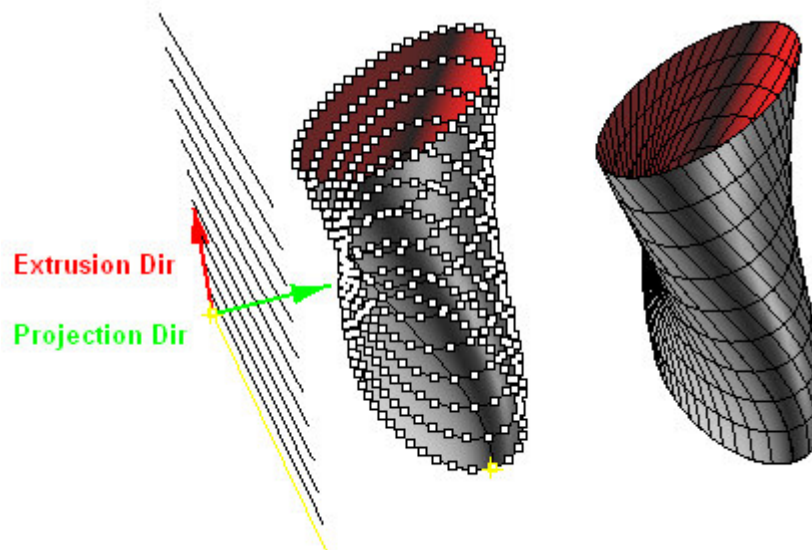
If **Yes**, group the resulting points.

NameOfGrid

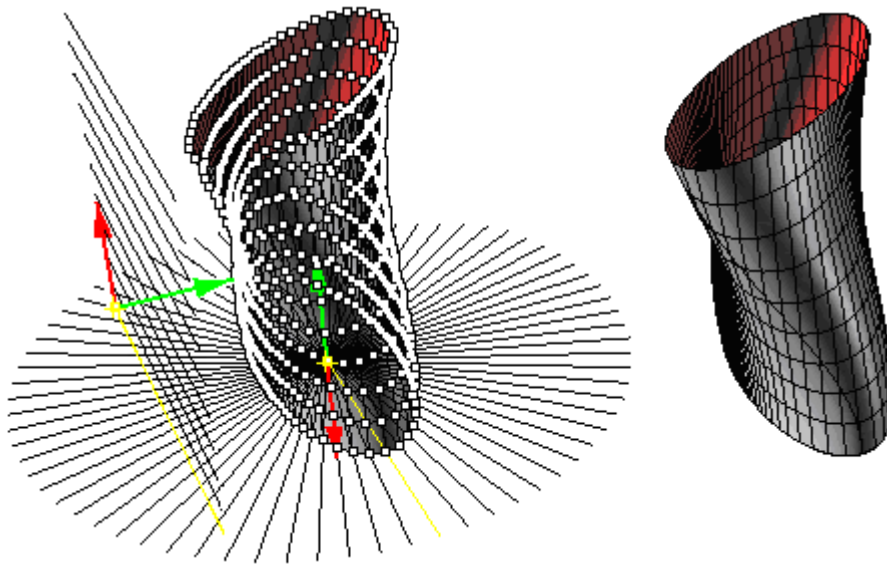
Grid name prefix attached to each point. The row and column location complete the point name.

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.

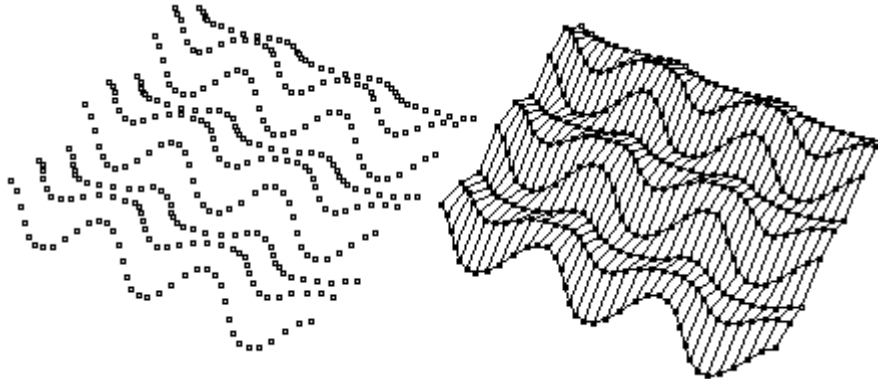
NameOfGrid

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

Grid from RhinoScript

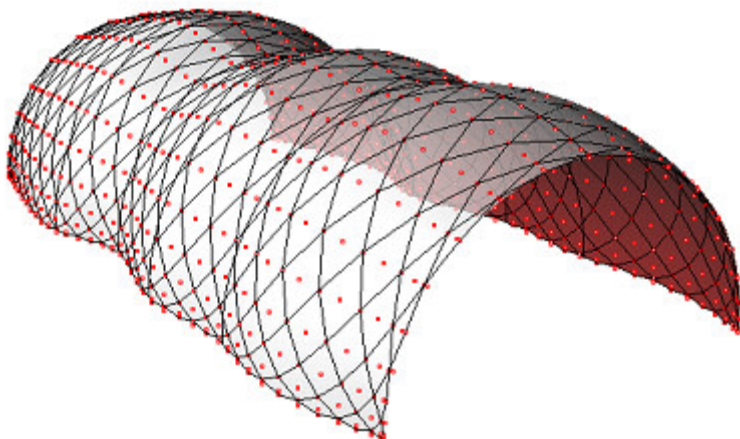
A paneling grid can be generated using RhinoScript. Many of the grid generation commands are accessible directly from RhinoScript. Even without using these commands, a custom point grid can be generated and each point location can be appended to point names.

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
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:



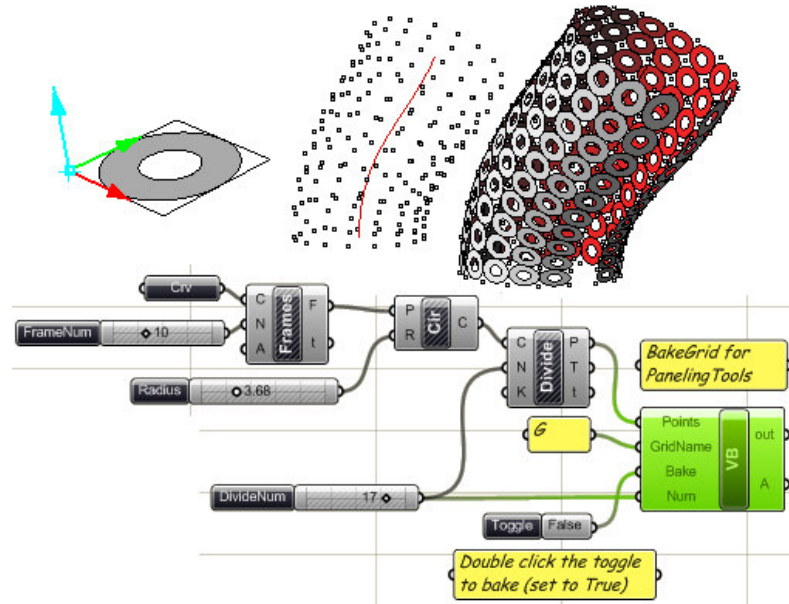
```

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
          j = j+1
        Next
        End If
        i = i+1
      Next
    End If
  End Sub
Call Main()
Sub Main()
  Dim i, PTObj, arrPoints, strSrf, dbl_cost, dbl_t
  Dim arr_u1ist(10)
  Dim arr_v1ist(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_u1ist(i) = dbl_cost
    arr_v1ist(i) = dbl_cost
  Next
  arrPoints = PTObj.DivideSurfaceByVariableDistances(strSrf, arr_u1ist, arr_v1ist, 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:

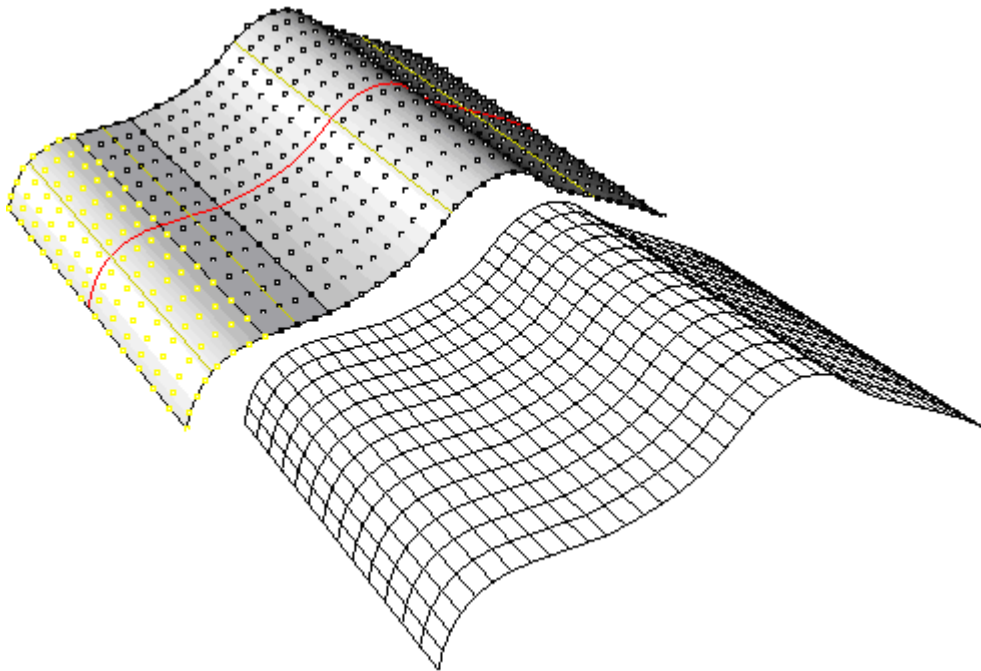
```
Sub RunScript(ByVal Points As List(Of Object), ByVal GridName As String, ByVal Bake As Boolean,
              ByVal Num As Integer)
    If( Not Bake ) Then
        m_count = 0
        Return
    End If
    'Iterate through points by row
    Dim i As Integer
    Dim j As Integer
    j = 0
    For 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

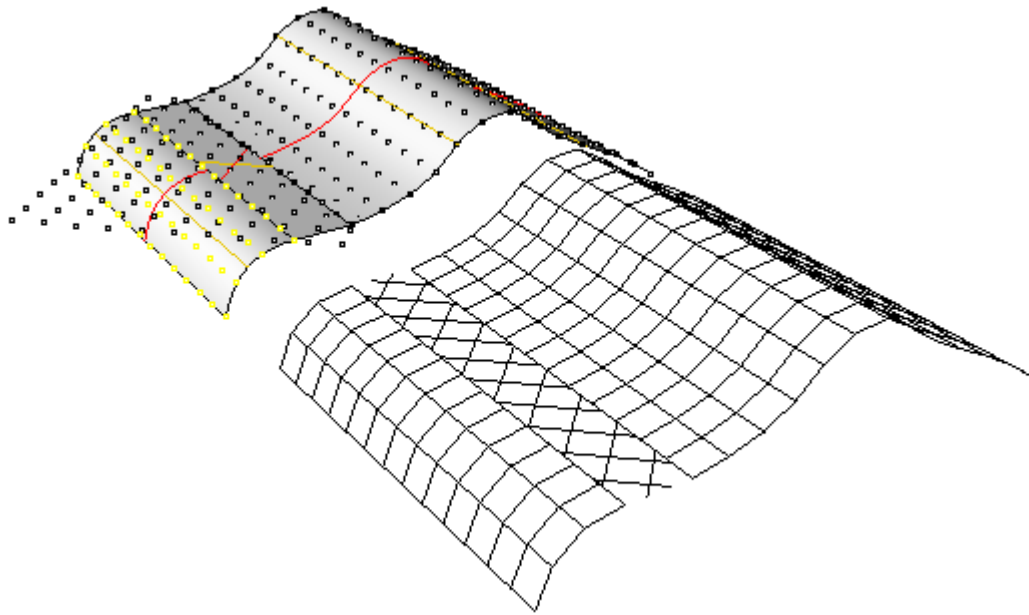
Polysurfaces are joined NURBS surfaces that might not have matching isocurve directions. It is not always possible to define a continuous paneling grid for polysurfaces, but there are few commands that are geared towards dealing with polysurfaces. The main commands are **ptPanelCurve** and **ptPanelCurve2**.

Use ptGridSurfaceUV with ArcLength method

If all surfaces in a polysurface have their isocurves aligned, then using **ptGridSurface** 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.

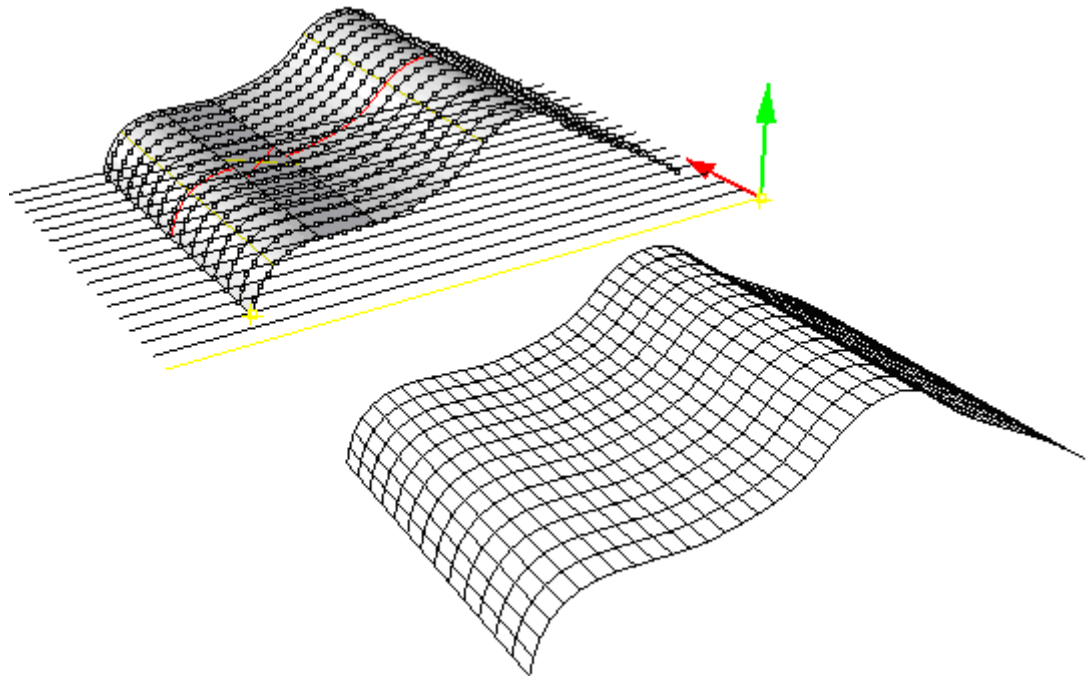


If isocurves of faces are not aligned, then using any of the commands that use surface uv-parameterization will probably not be useful. See the following example.



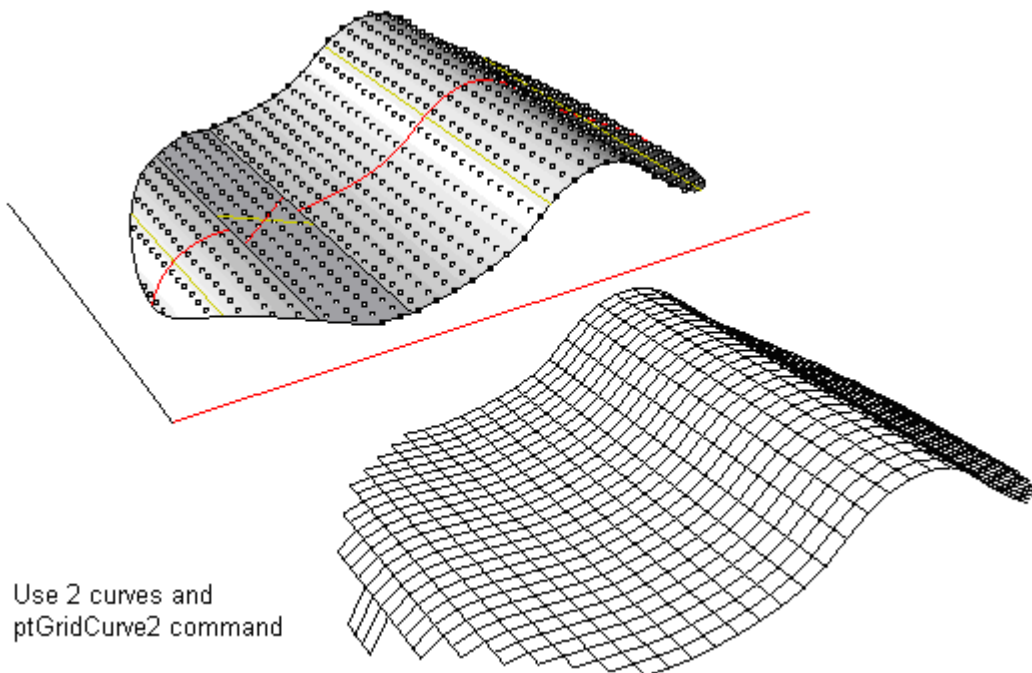
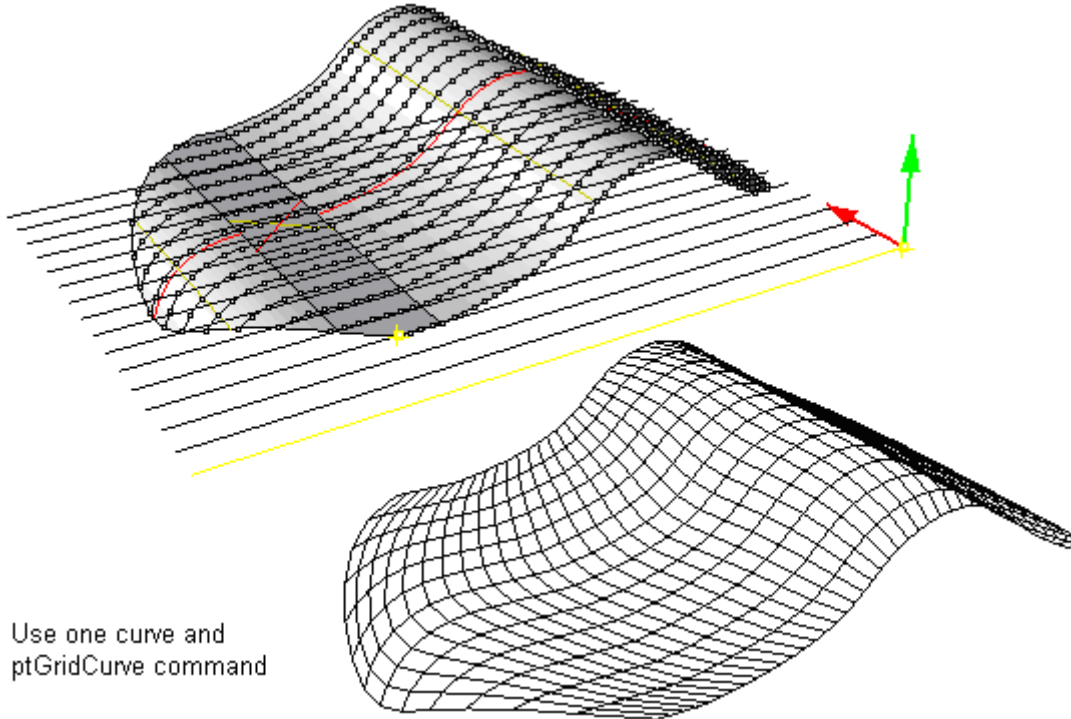
Use ptGridCurve command

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:

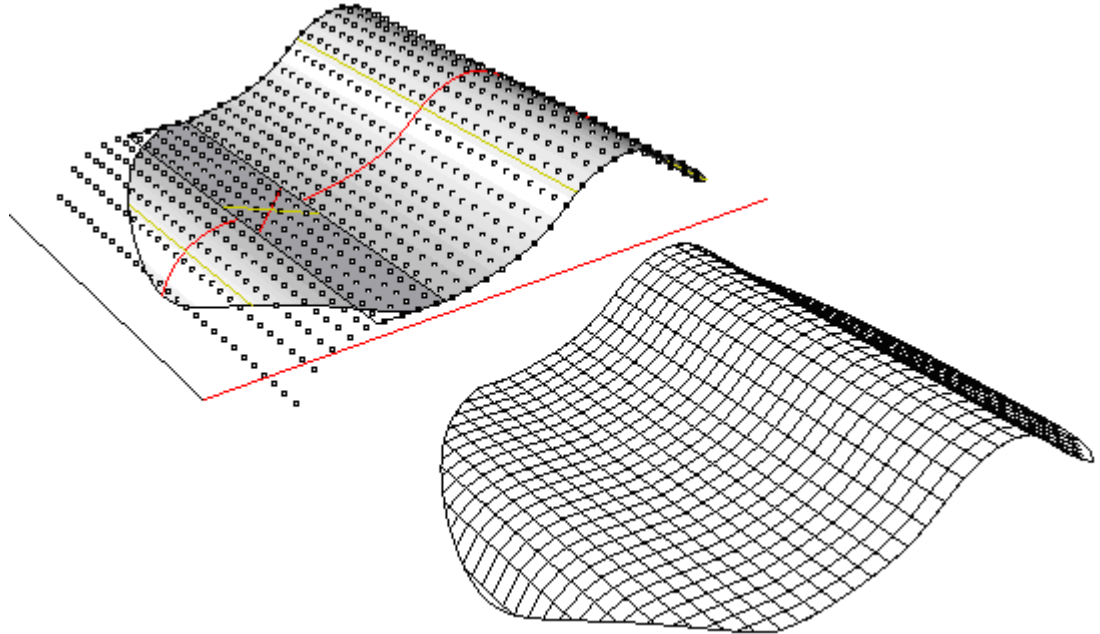


Use ptGridCurve2 command

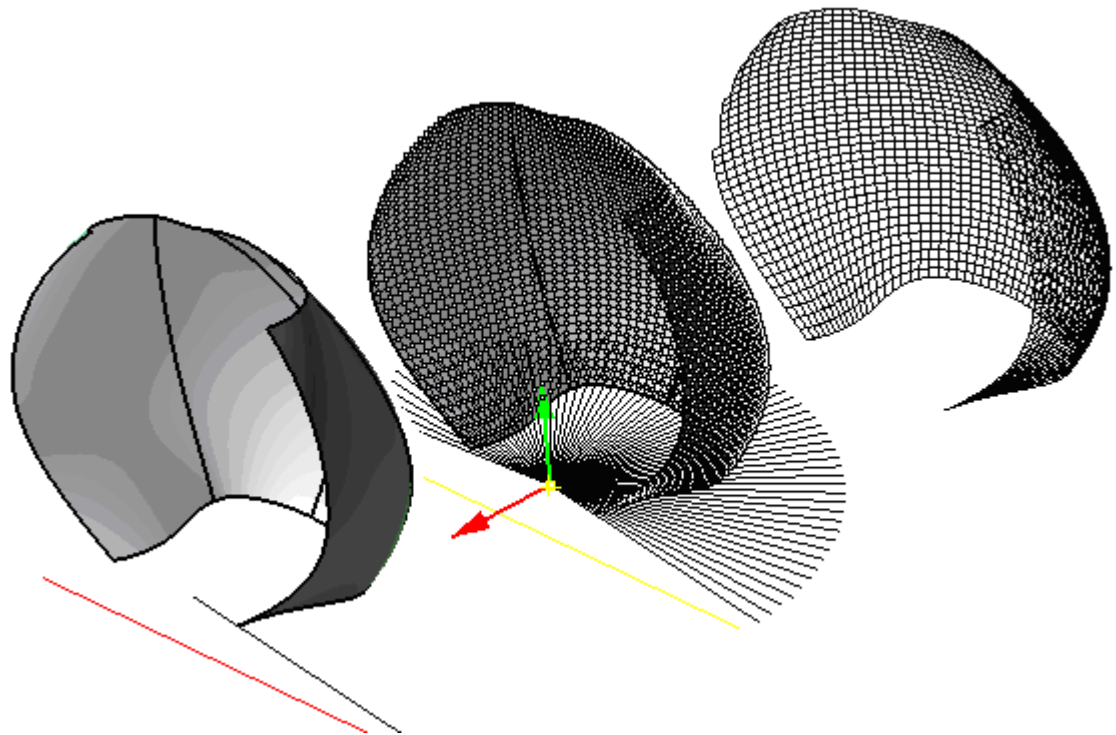
In cases when the polysurface has no linear edge along the extrusion direction, grid can be undesirable. To force a second direction, you can use **ptGridCurve2**. The following example shows result using one curve (**ptGridCurve**).



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.

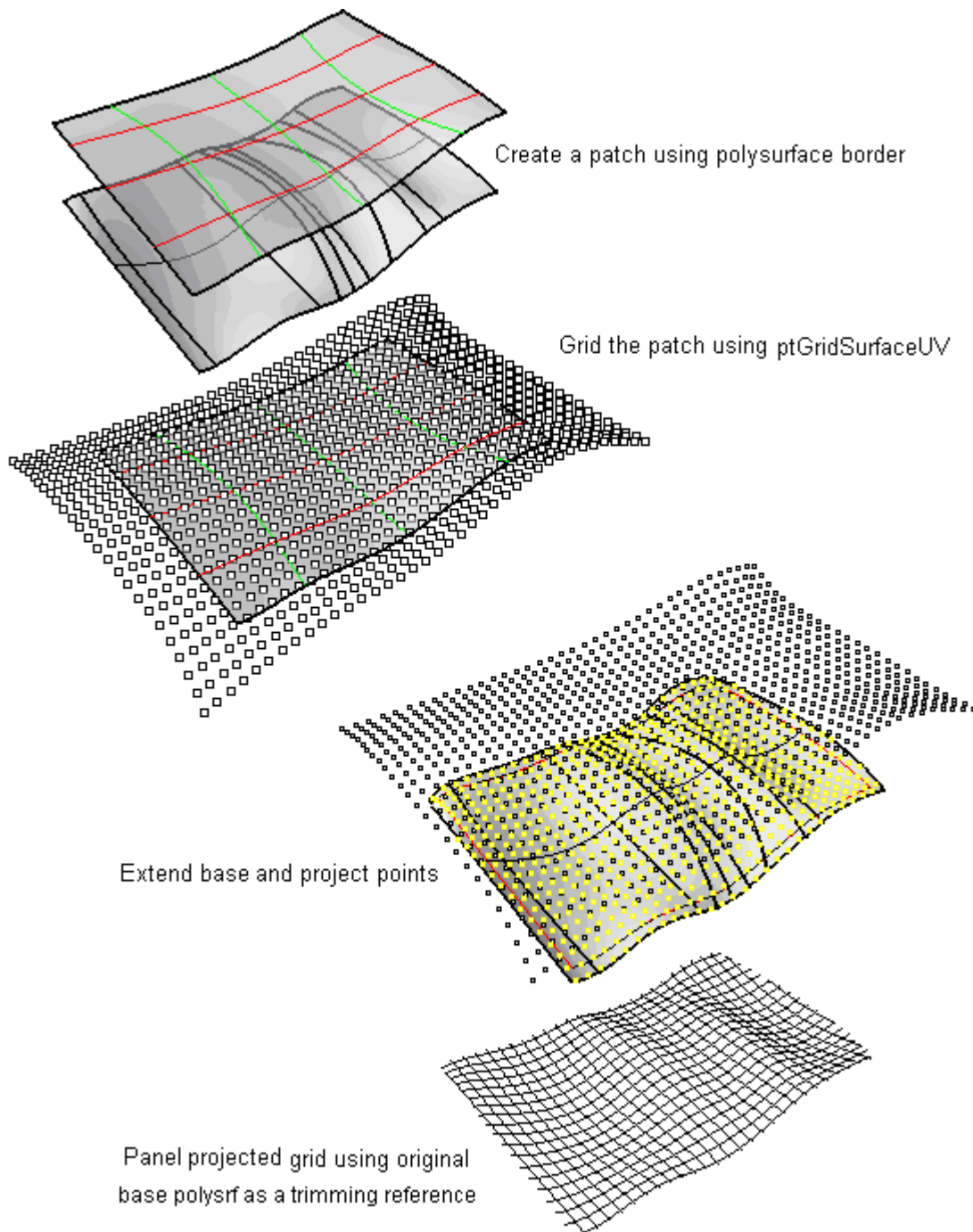


Using one or two curves is also useful when dealing with polysurfaces created from revolved surfaces. The **ptGridCurve** and **ptGridCurve2** polar extrusion direction option is useful in these cases. See the following example.



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

The **ptPanelGrid** command generates 2-D pattern 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. The 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.

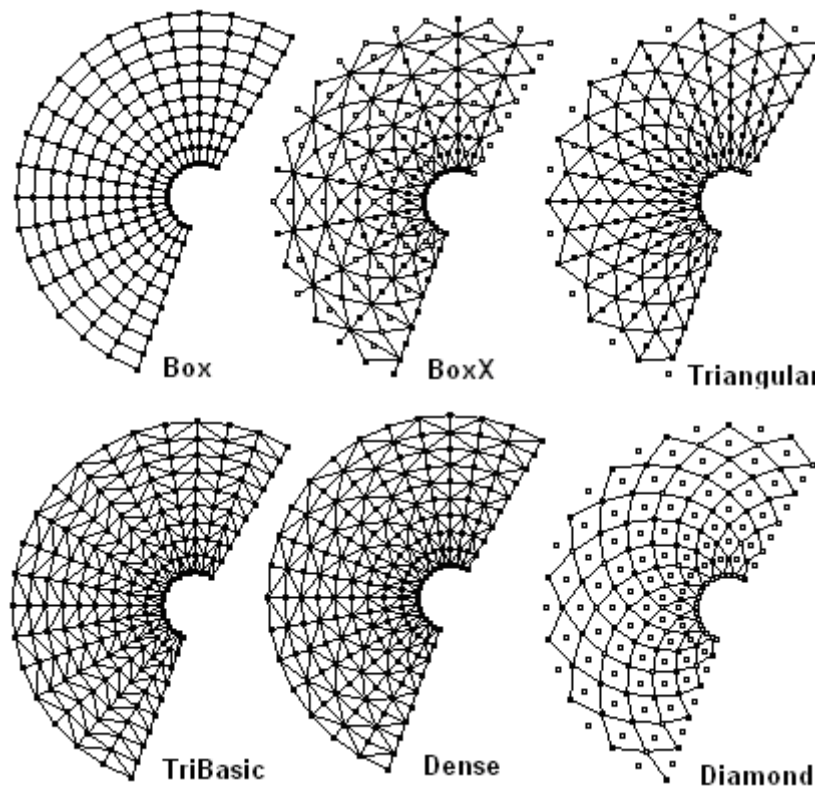
The **ptManage2DPattern** and **ptManage3DPattern** commands create, edit, and delete custom patterns.

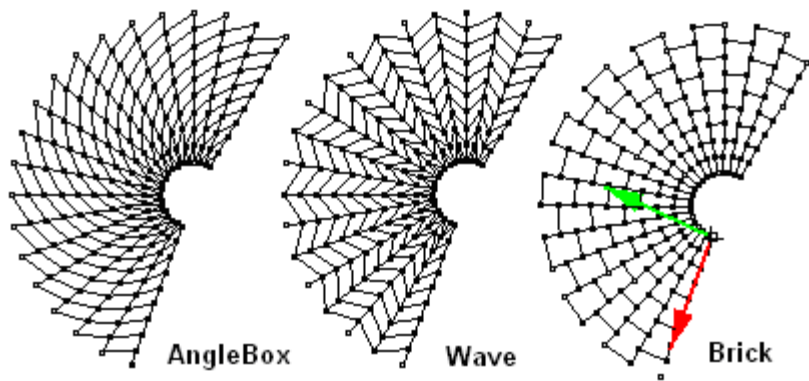
Mapping to Unit Grid

The **ptPanelGridCustom** command supports mapping to populate 2-D free-form patterns. The **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:



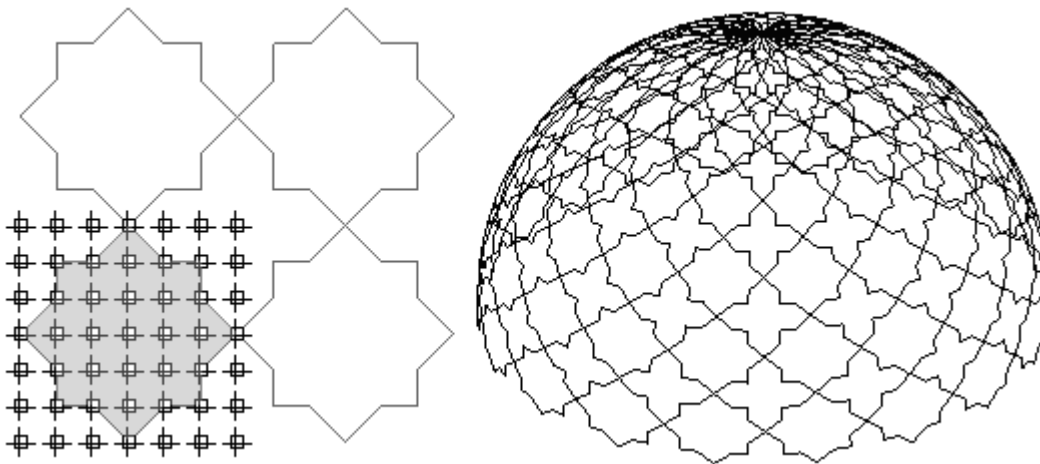


ptManage2DPatterns

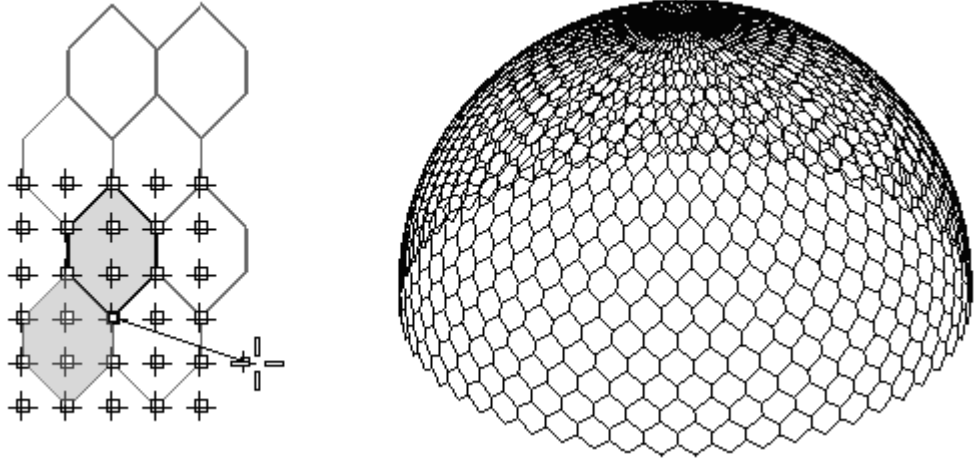
Custom patterns are created by:

- 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 **GridWidth=7**, **GridHeight=7**, **ShiftX=6** and **ShiftY=6**.



Multiple polylines can be included in the same pattern. The following example used two closed polylines with **GridWidth=5**, **GridHeight=6**, **Shift_x=2** and **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.

Name

Pattern name.

Edit

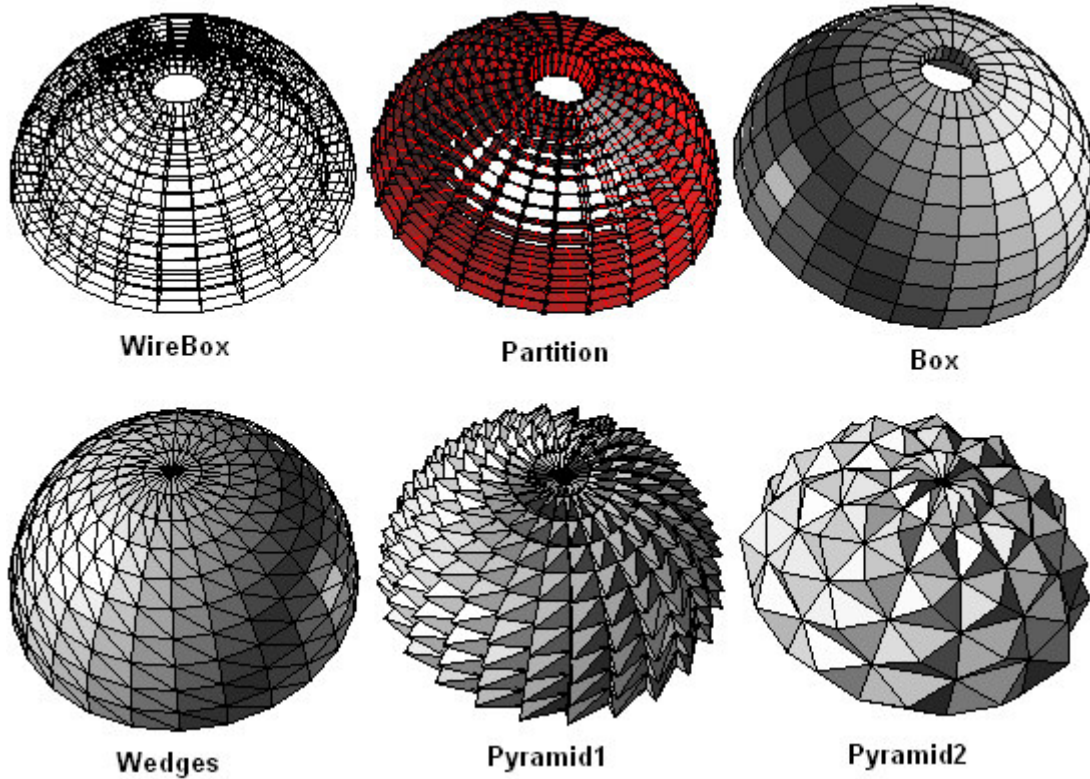
Select a pattern name from the list. The command flow and options are similar to when creating new pattern.

Delete

Select the pattern name to be deleted from the list.

3-D Connecting Pattern

The **ptPanel3Dgrid** command, **Pattern** option provides built-in 3-D connecting patterns that create edges, faces, or solids. The following illustration shows these 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.

Wedges

Closed trapezoid connecting three points from each bounding grid.

Pyramid1

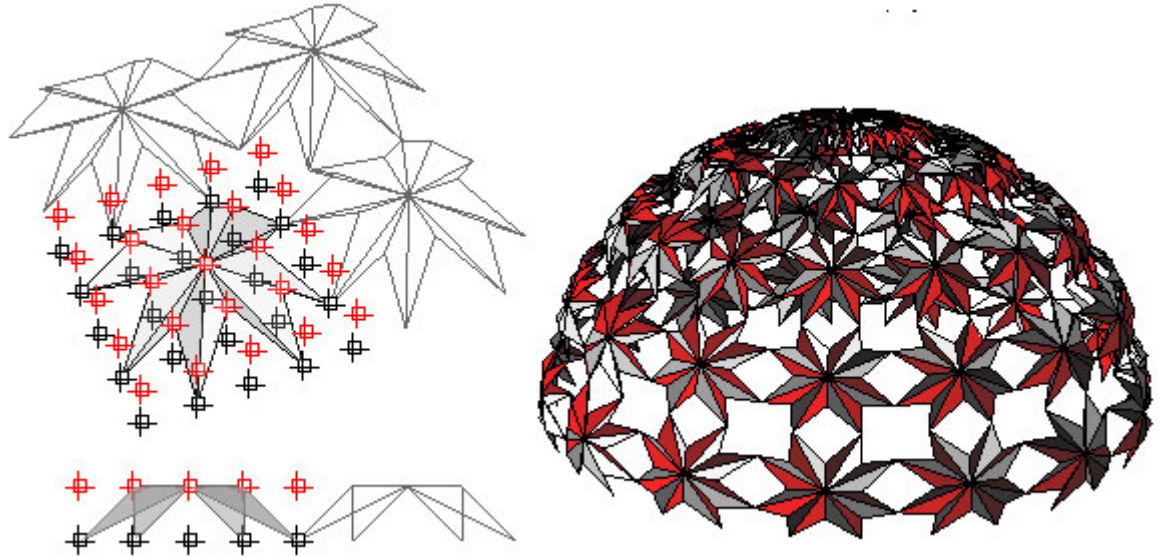
Connects four points from the first grid with the corresponding minimum point from the second grid.

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.

Name

Pattern name.

Edit

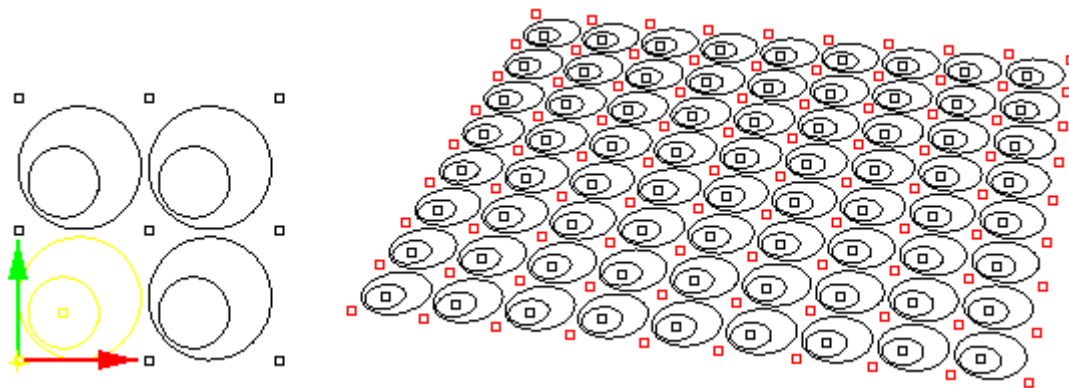
Select a pattern name from the list. The command flow and options are similar to when creating new pattern.

Delete

Select the pattern name to be deleted from the list.

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

GridWidth

Unit grid width. A width that is bigger than pattern width means that the pattern will be scaled down to fit within same relative unit dimension.

GridHeight

Unit grid height.

U_Spacing

Set the spacing to **1** to pack the pattern.

Set the spacing to **2** to map every second paneling pattern unit grid.

V_Spacing

Spacing in v-direction.

Group

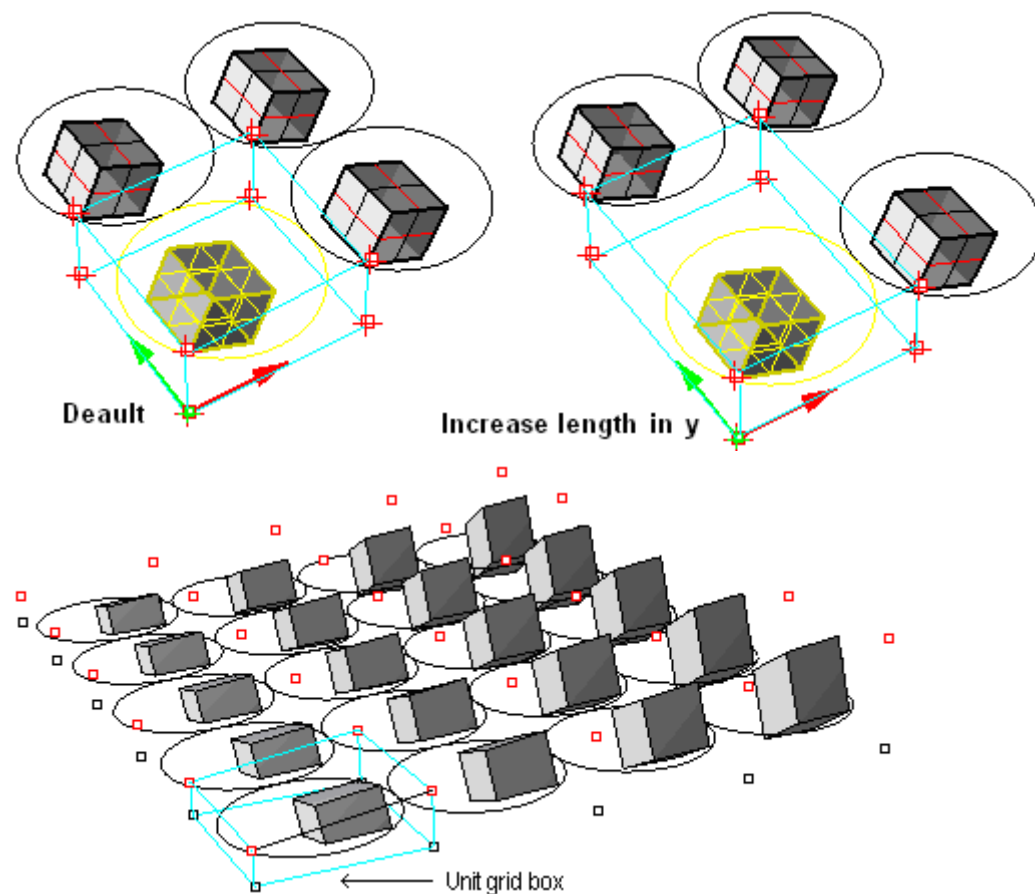
If **Yes**, group the resulting points.

NameOfGrid

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

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:



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_x

Starting grid index in the u-direction. If set to **0**, start at minimum x.

Base_y

Starting grid index in the v-direction. If set to **0**, start at minimum y.

Shift_x

Set the spacing to **1** to pack the pattern.

Set the spacing to **2** to map every second paneling pattern unit grid.

X_Length

Module x-length. If increased, pattern is scaled down in the x-direction.

Y_Length

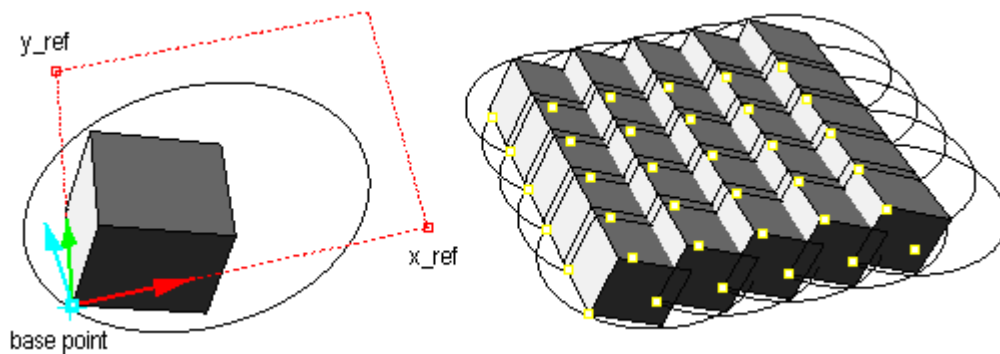
Module y length.

Z_Length

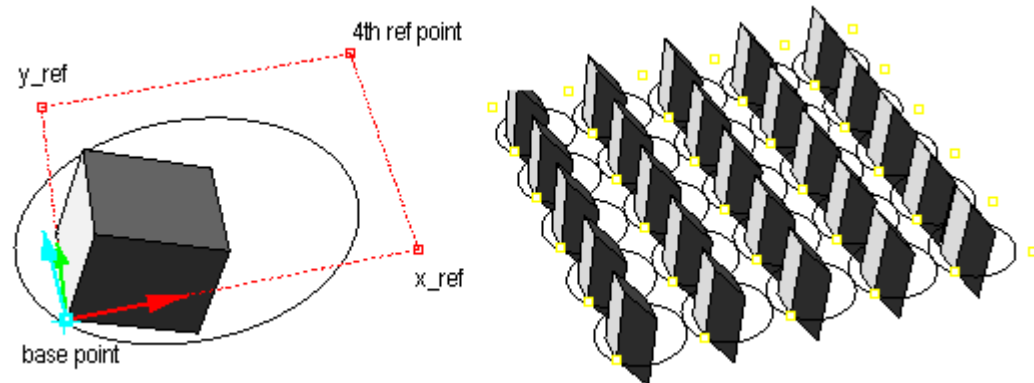
Unit grid z length.

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_x

Starting grid index in the u-direction. If set to **0**, start at minimum x.

Base_y

Starting grid index in the v-direction. If set to **0**, start at minimum y.

Shift_x

Set the spacing to **1** to pack the pattern.

Set the spacing to **2** to map every second paneling pattern unit grid.

Shift_y

Population shift in y direction.

X_Length

Number of x grid points to stretch along.

Y_Length

Unit grid y length.

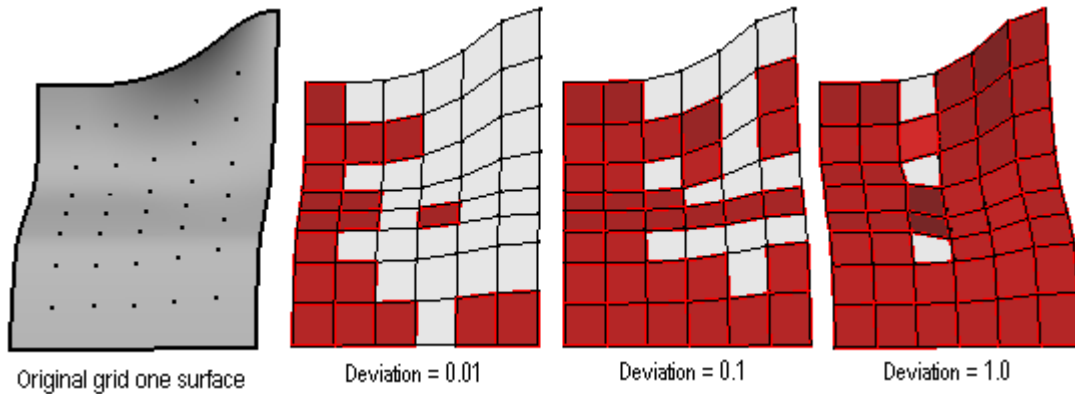
Paneling Planar Quadrangles

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

MaximumDeviation

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.

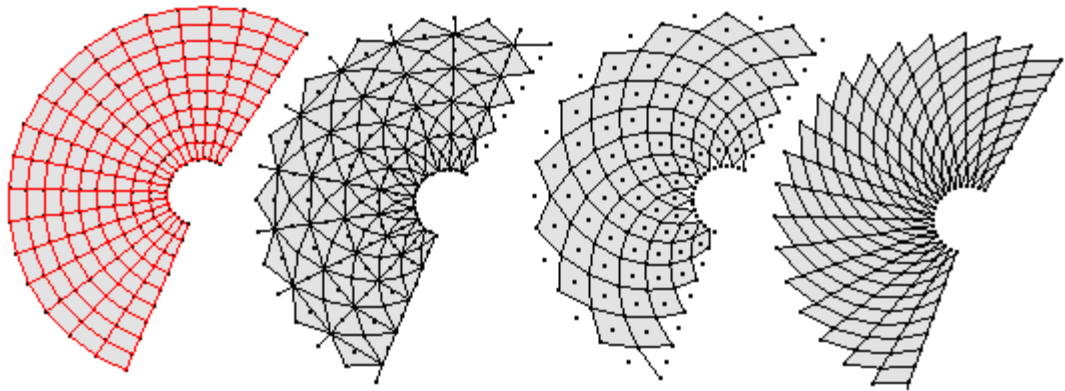
Name

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 unit grid points.

FitBasePt0

Best-fit plane through min unit grid point.

FitBasePt1

Best-fit plane through max point in the min row in a unit grid.

FitBasePt2

Best-fit plane through max point in the max row in a unit grid.

FitBasePt3

Best-fit plane through min point in the max row in a unit grid.

TangentToCenter

Best-tangent plane to surface center.

AddMesh

Add a mesh.

Group

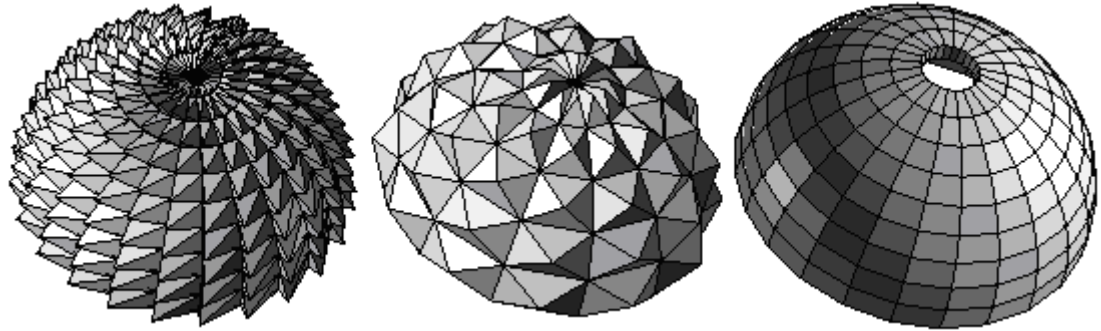
If **Yes**, group the resulting points.

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.



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.

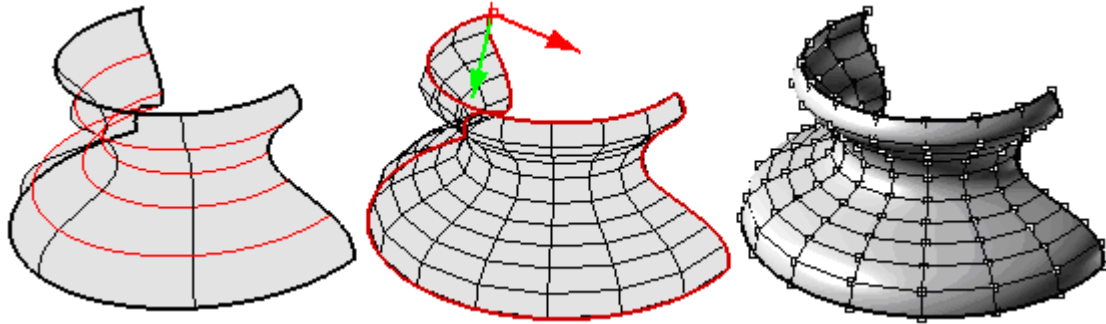
Paneling without a Grid

The **ptPanelSurface** and **ptPanelCurve** commands create a paneling grid and then panel it directly.

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.

ptPanelSurface

The **ptPanelSurface** command uses a base surface to generate the paneling grid and panels. This is the same as using the **ptGridSurfaceUV** command and then the **ptPanelGrid** command.



Command flow

- 1 Start the **ptPanelSurface** command.
- 2 Select a base surface.
- 3 Press **Enter** to accept grid options.
- 4 Press **Enter** to accept paneling options.

Surface divide options

DivideMethod

Surface division methods.

BySurfaceUV

U_Method

U-direction 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_RoundMethod

Up
Down

V_Method

V-direction 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_RoundMethod

Up

Down

BySurfaceDistance**U_Length**

Distance in the u-direction.

V_Length

Distance in the v-direction.

Extend

If **Yes**, extend surface to get better coverage.

Paneling options**Pattern**

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

PanelShape**Straight**

Line 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 unit grid points.

FitBasePt0

Best-fit plane through min unit grid point.

FitBasePt1

Best-fit plane through max point in the min row in a unit grid.

FitBasePt2

Best-fit plane through max point in the max row in a unit grid.

FitBasePt3

Best-fit plane through min point in the max row in a unit grid.

TangentToCenter

Best-tangent plane to surface center.

AddMesh

Add a mesh.

Group

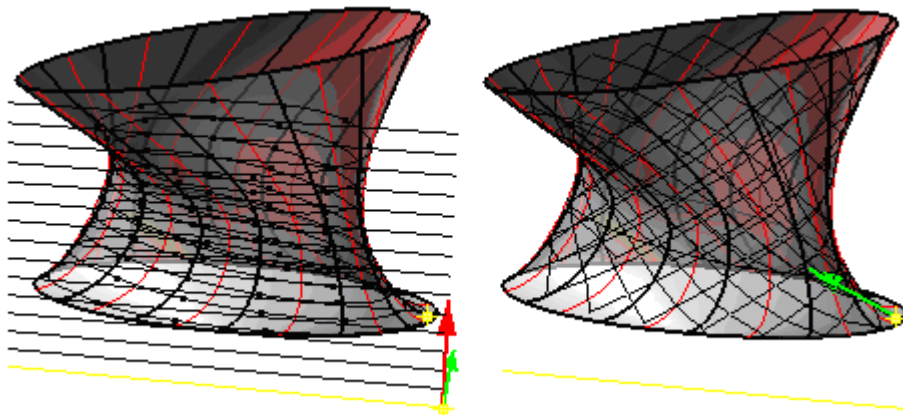
If **Yes**, group the resulting points.

Name

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

ptPanelCurve

The **ptPanelCurve** command generates panels for a surface or polysurface using direction curve. The direction curve can be open or closed. This is the same as using the **ptGridCurve** command and then **ptPanelGrid** command.



Command flow:

- 1 Start the **ptPanelCurve** command.
- 2 Select base surface or polysurface.
- 3 Select base curve or draw a line on the fly.
- 4 Press **Enter** to accept grid creation options.
- 5 Press **Enter** to accept paneling options.

Grid Creation Options

CurveOptions**NumberOfCuts**

Number of curves to be projected to the object.

Spacing/Angle

Distance 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**

Length on base curve or straight-line distance.

Round

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

RoundingMethod**Up****Down****ChordLength****Distance**

Length along the base curve or straight-line distance.

AddEndPoint

If **Yes**, add a point to the end.

Paneling Options

Pattern

Box, BoxX, Triangular, TriBasic Dense, Diamond, AngleBox, Wave, Brick, 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 the 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 unit grid points.

FitBasePt0

Best fit plane through min unit grid point.

FitBasePt1

Best-fit plane through max point in the min row in a unit grid.

FitBasePt2

Best-fit plane through max point in the max row in a unit grid.

FitBasePt3

Best-fit plane through min point in the max row in a unit grid.

TangentToCenter

Best-tangent plane to surface center.

AddMesh

Add a mesh.

Group

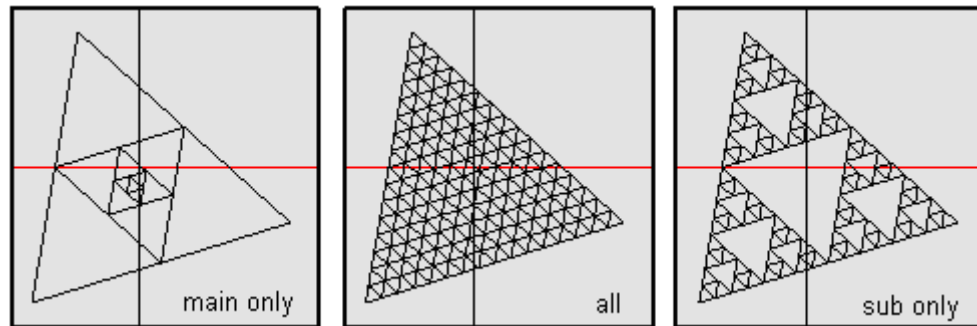
If **Yes**, group the resulting points.

Name

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

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

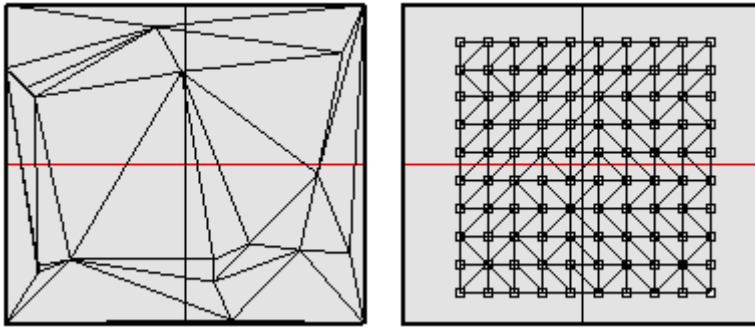
Subdivide main curve only.

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

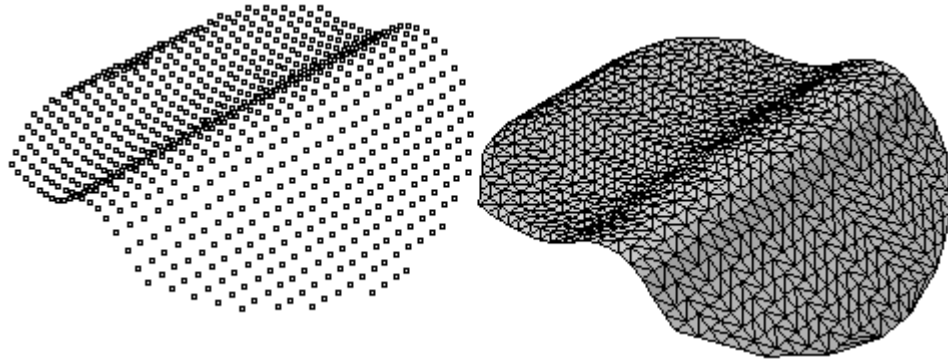
Number of points to be generated.

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.

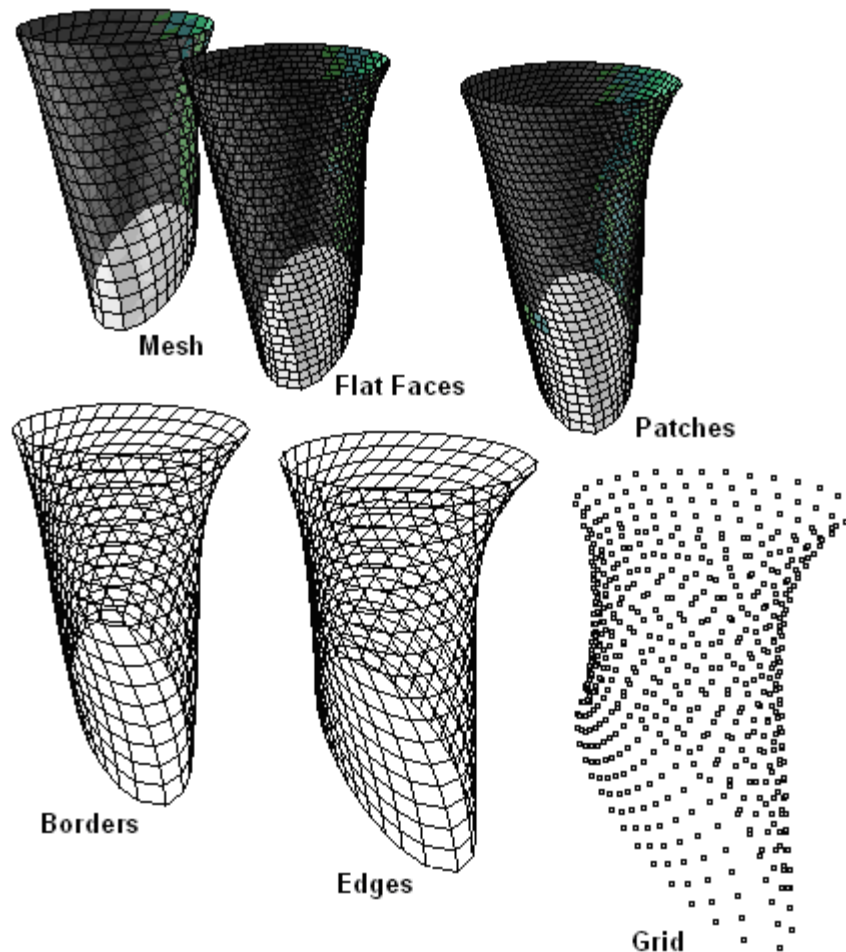
5 Paneling Output

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

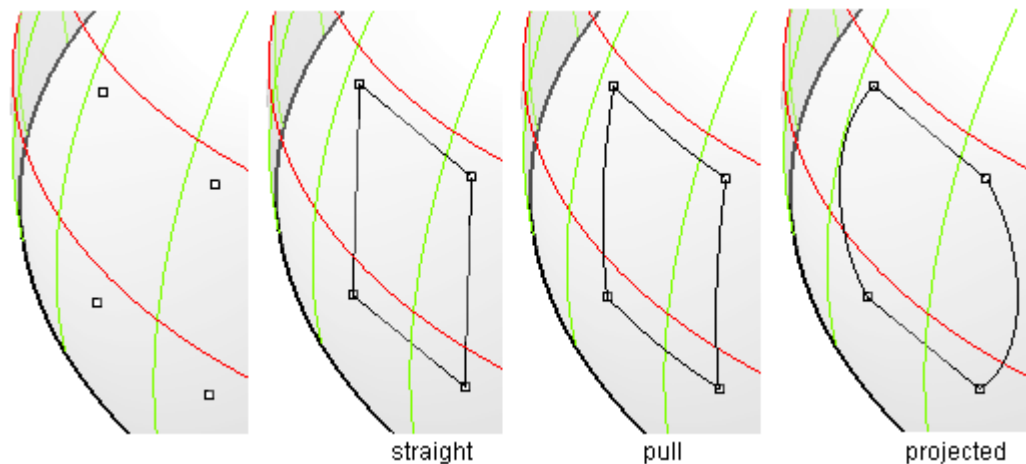
ON_Mesh

Paneling shape

The paneling shape can only be straight when there is no reference surface. 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.

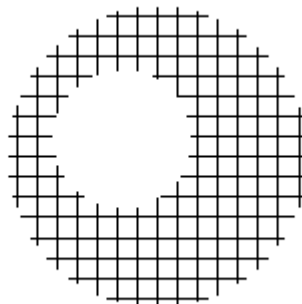
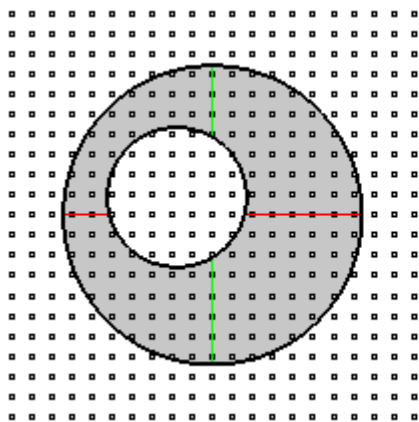
Projected

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

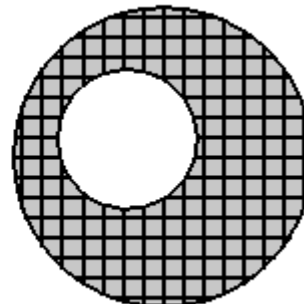
Trimmed surfaces

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 and the new second point. In some cases this process fails to find useful result and few panels might need to be trimmed manually.



Trimmed edge panels



Trimmed face panels

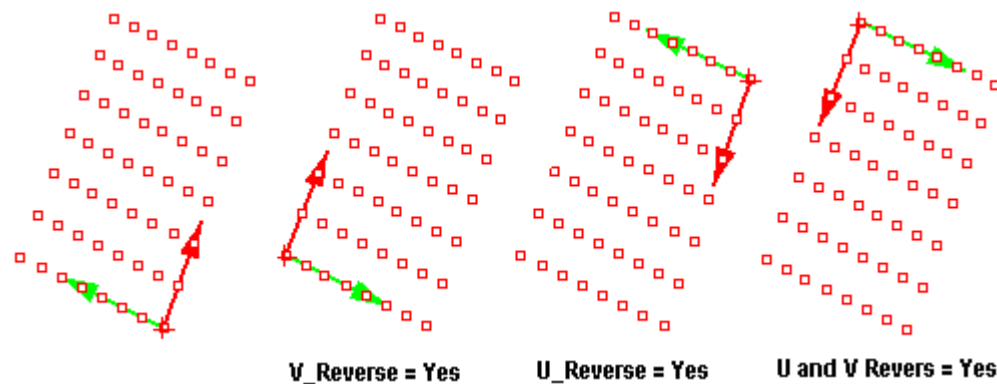
6 Utility Functions

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.



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

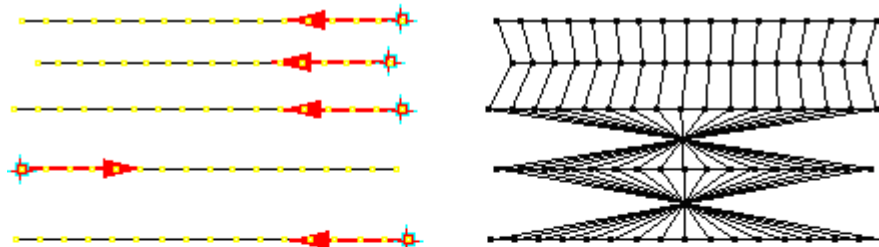
If **Yes**, group the resulting points.

NameOfGrid

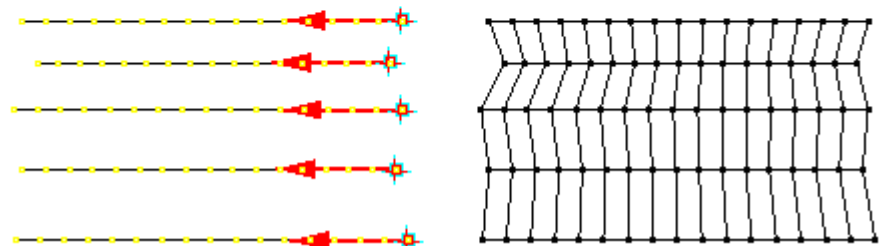
Name of paneling grid. Default to selected grid point.

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

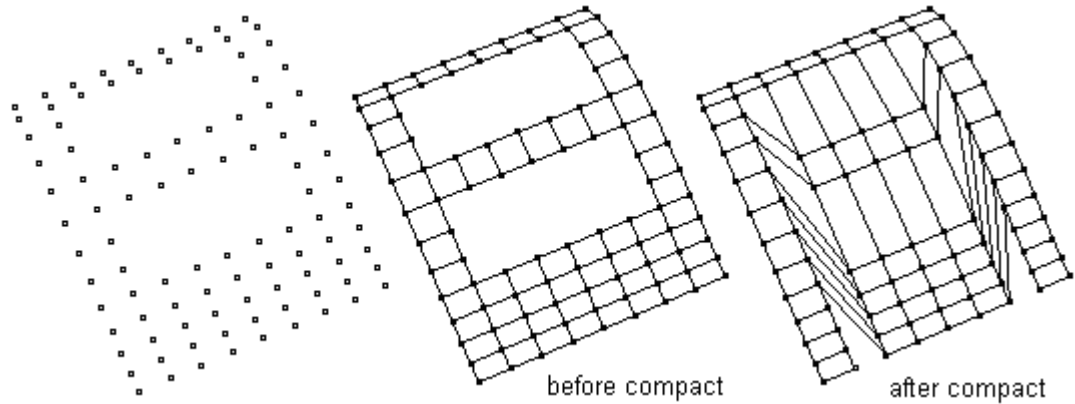
If **Yes**, group the resulting points.

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.

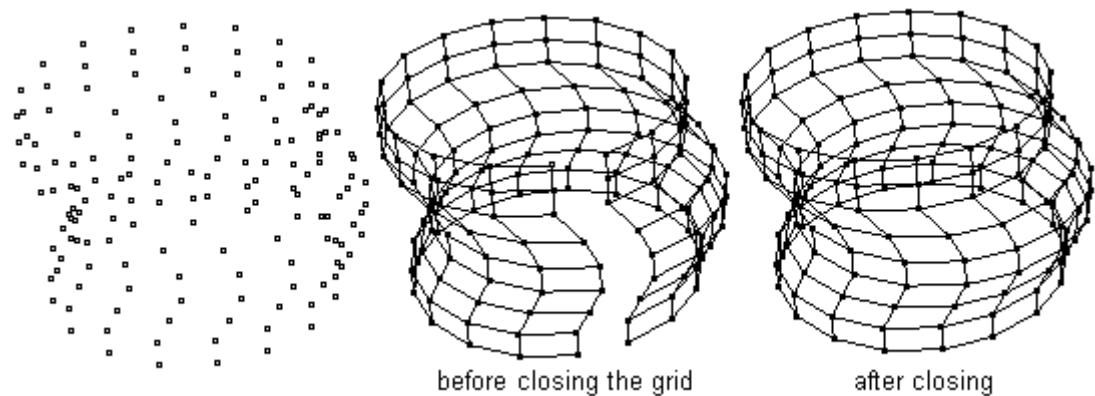


Command flow

- 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.

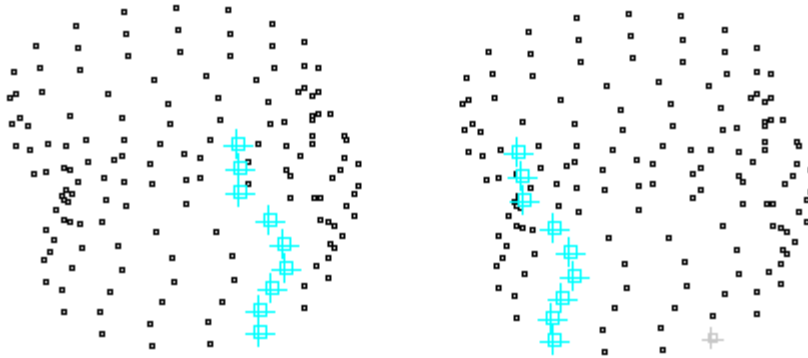


Command flow:

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

ptGridSeam

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



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

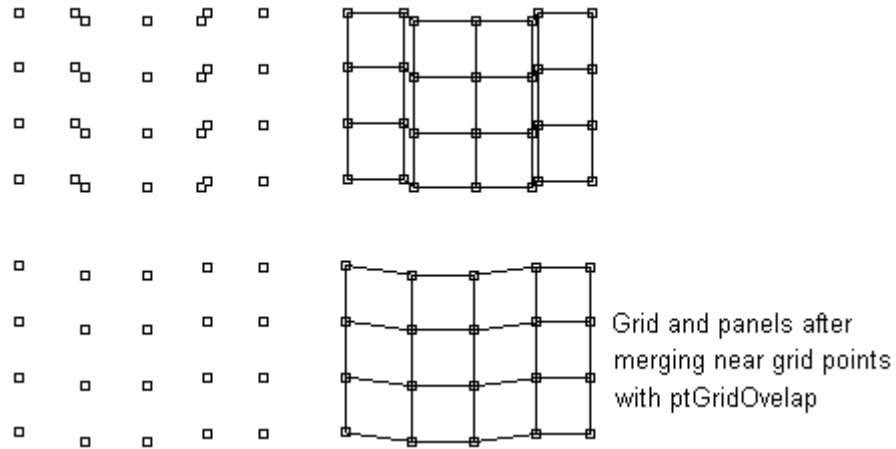
If **Yes**, group the resulting points.

NameOfGrid

Name of paneling grid. Defaults to selected grid point.

ptCleanOverlap

The **ptCleanOverlap** command removes overlapped points in a 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.



Command flow:

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

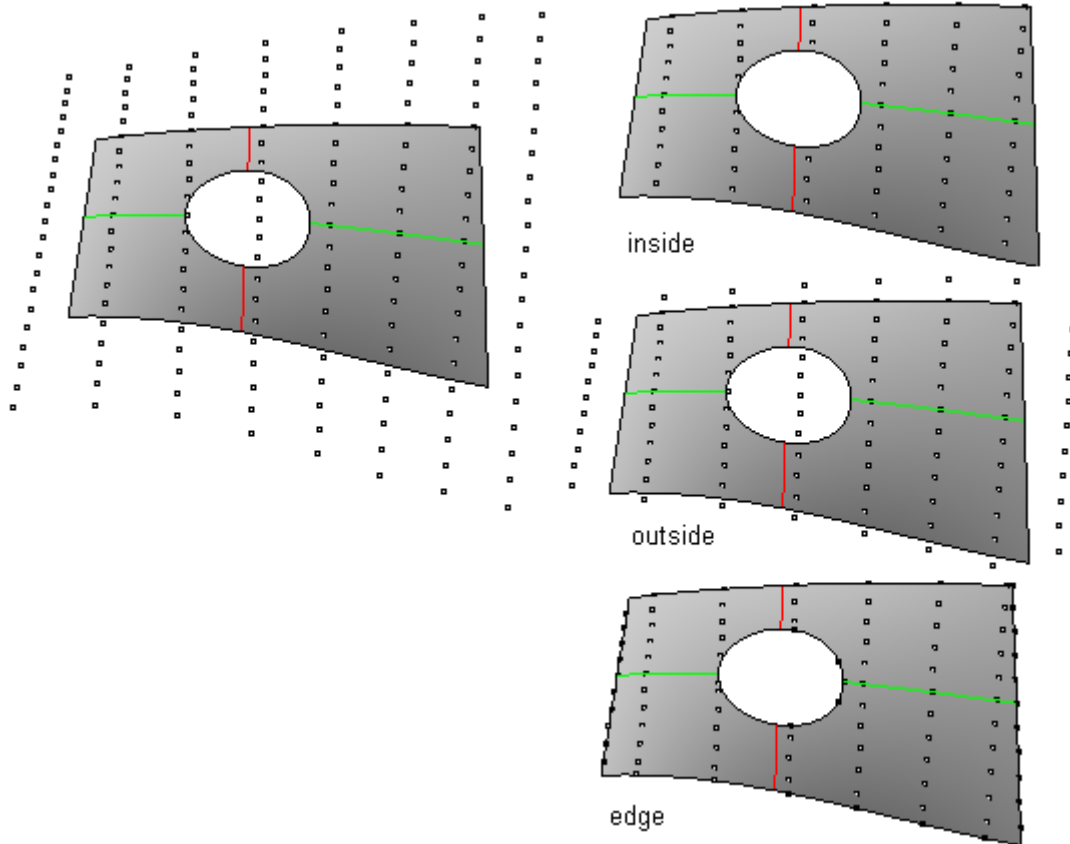
Options

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

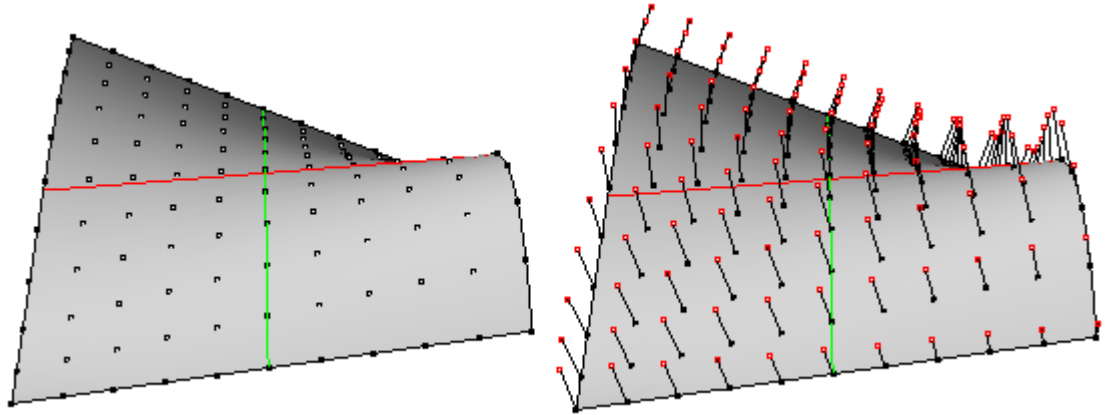
If **Yes**, group the resulting points.

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.

**Command flow**

- 1 Start the **ptOffsetPoints** command.
- 2 Select a grid and pick options.
- 3 Select base surface or polysurface.

Options

OffsetAmount

Offset distance.

Group

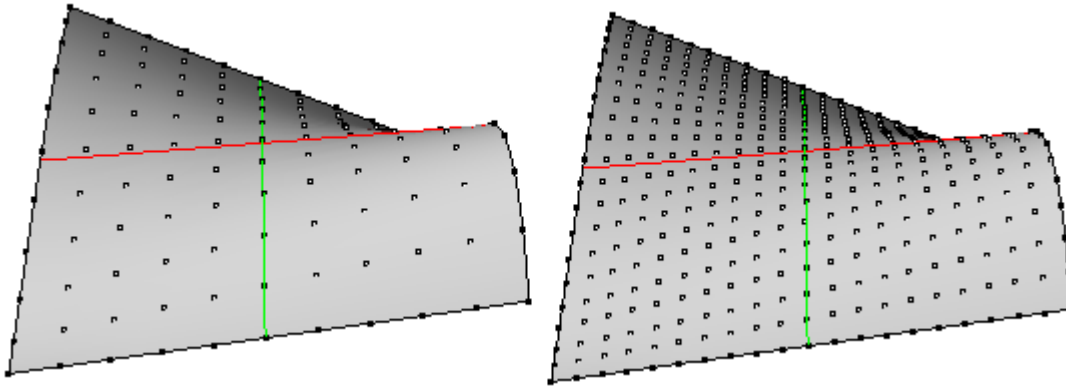
If **Yes**, group the resulting points.

Connect

Draw lines between points and their corresponding offset points.

ptChangeGridDensity

The **ptChangeGridDensity** command increases or decreases the 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

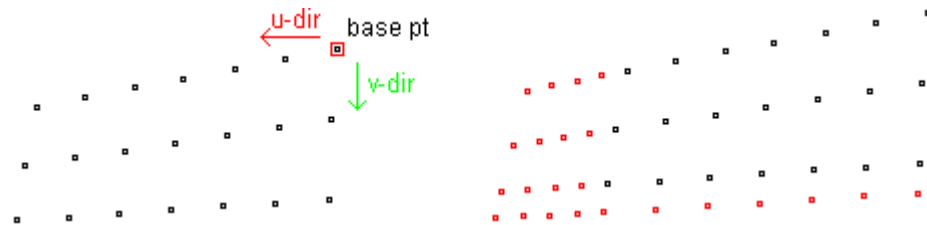
If **Yes**, group the resulting points.

NameOfGrid

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

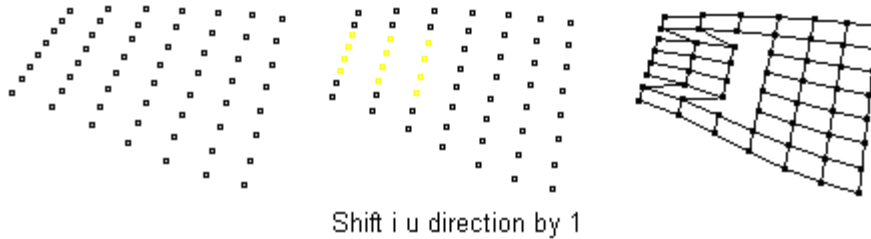
If **Yes**, group the resulting points.

NameOfGrid

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.



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

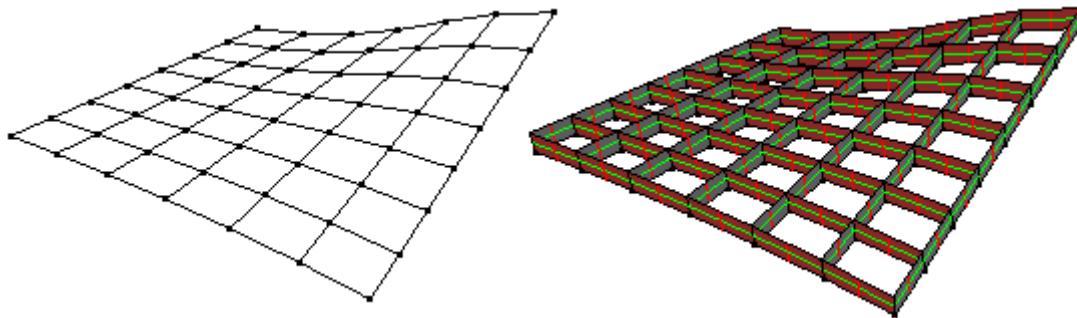
Number of steps shifted in the v-direction.

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

Height

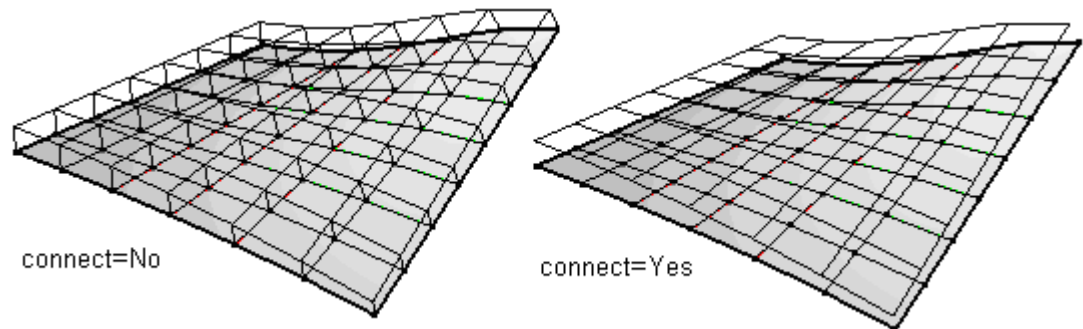
Extrusion distance.

NameEnding

Prefix 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.

**Command flow**

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

Options

Distance

Offset distance.

Connect

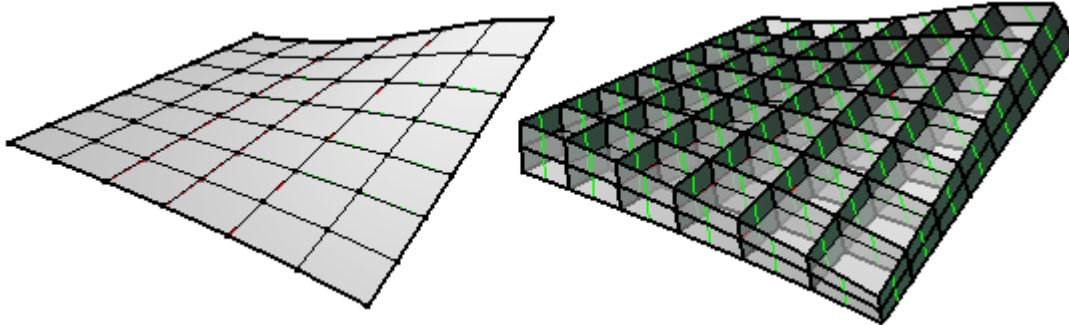
If **Yes**, connect offset edges with input edges.

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

Distance

Offset distance.

NameEnding

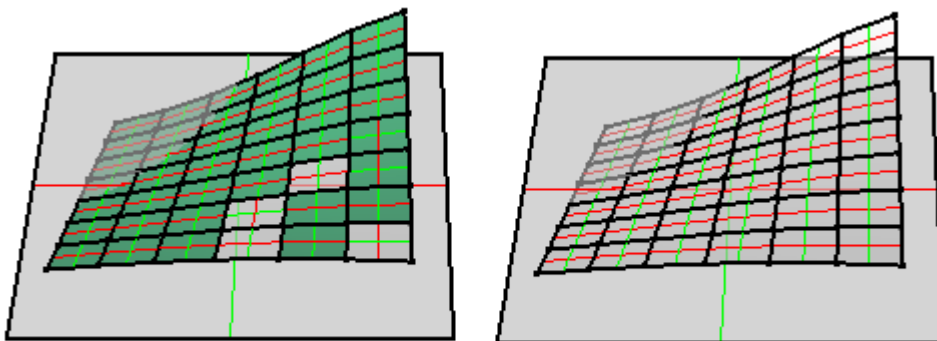
Suffix added to edge name to serialize extruded parts.

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

Command flow

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

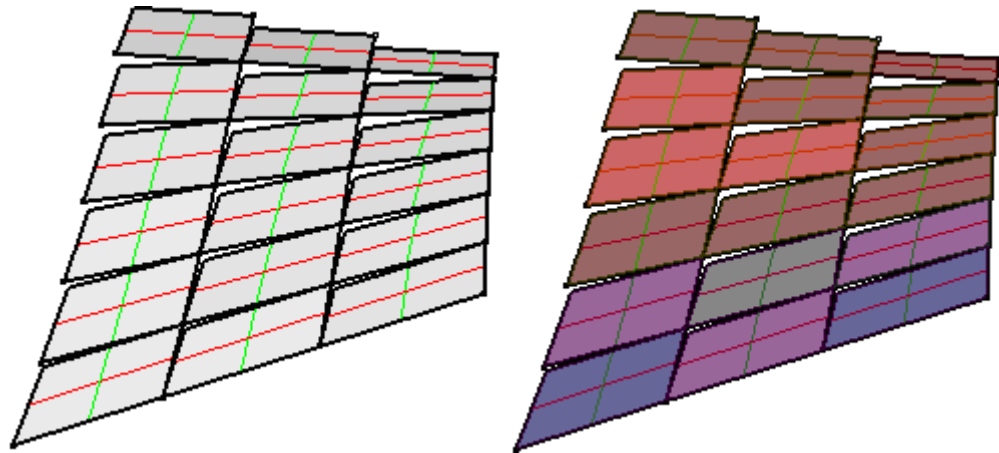
Options

UnifyUV

Match reference surface u- and v-directions.

ptAnalyzeFlatFaces

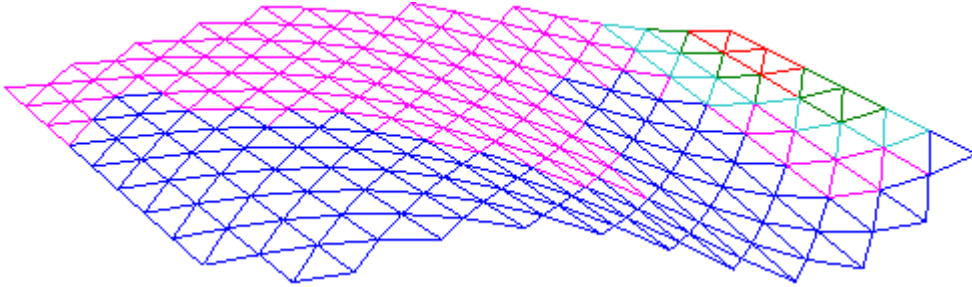
The **ptAnalyzeFlatFaces** command creates an analysis mesh to show amount of deviation of flat surfaces from their base surface. When creating flat surfaces with the **ptPanelGrid** command, the deviation amount is saved as user data on each surface. This information is used to create the analysis mesh.

**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

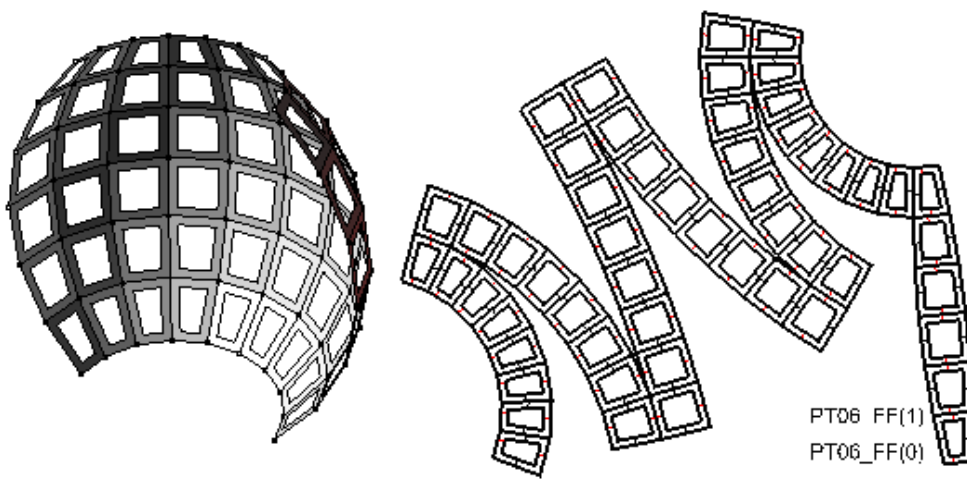
Difference in edge length allowed for panels to be considered similar.

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.



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

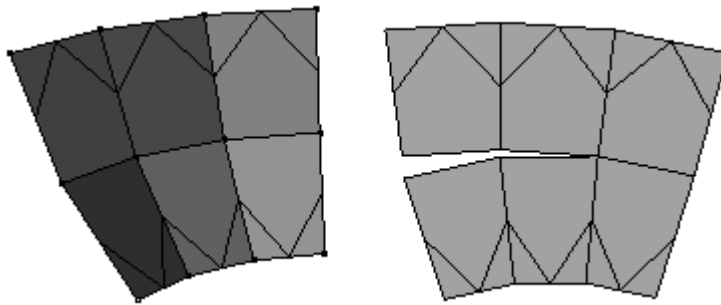
Add unrolled faces to a new sub layer.

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 faces to current layer.

NewSubLayer

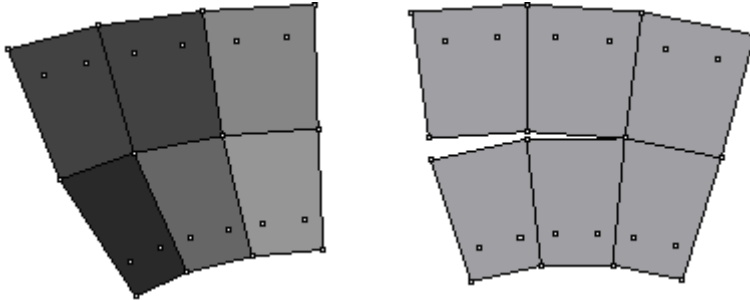
Add unrolled faces to a new sub layer.

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 faces to current layer.

NewSubLayer

Add unrolled faces to a new sub layer.

SubLayerName

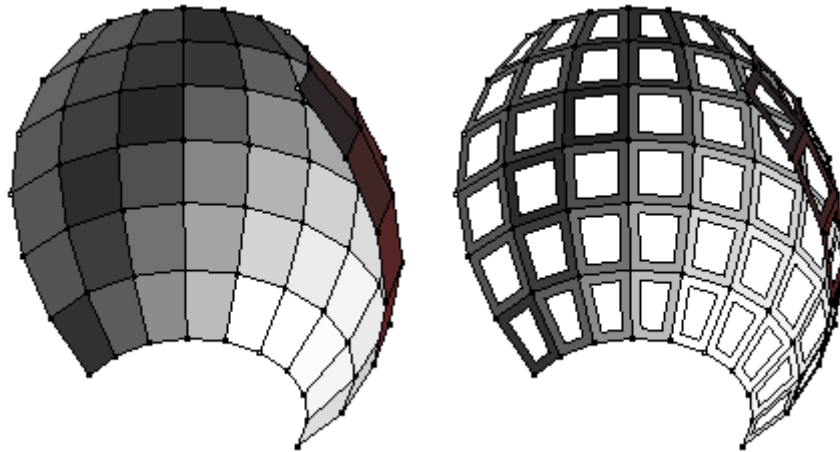
Name of the sub layer.

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

Distance

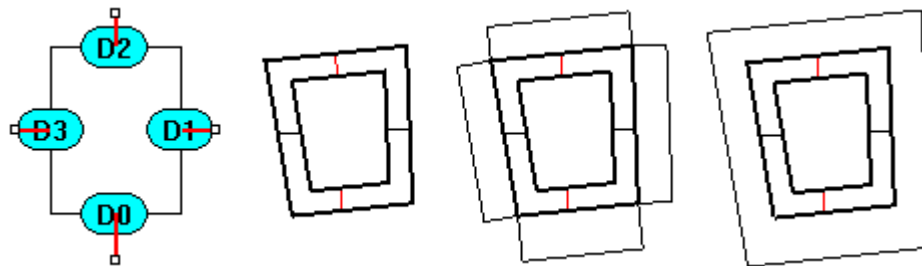
Offset distance on surface.

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

Output format.

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

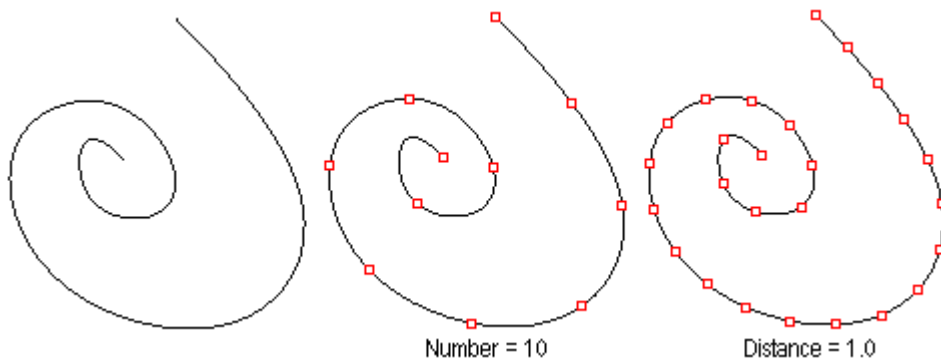
Offset distance. If **TypeOfDistance**=*Variable*, each tagged edge will have a separate distance value to set (D1, D2, etc.)

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

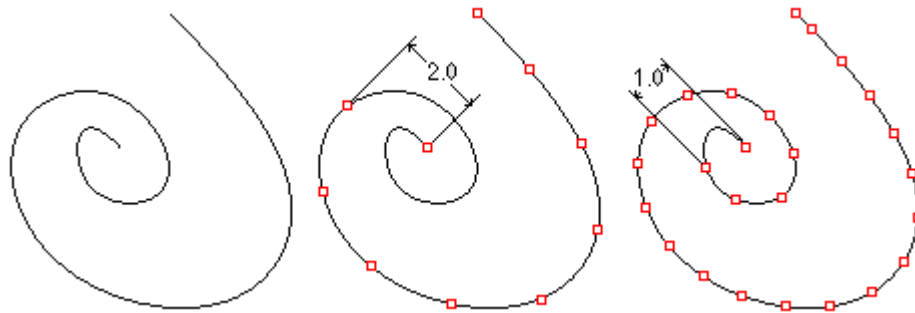
Down

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

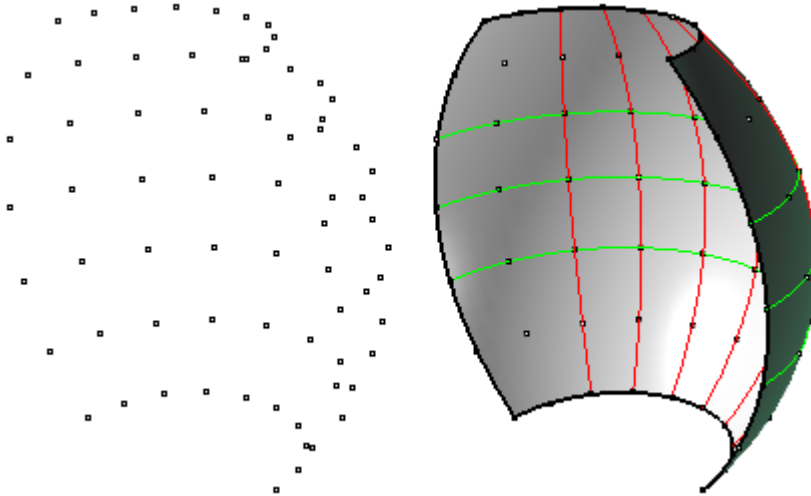
If **Yes**, add a point to the end of the curve.

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.

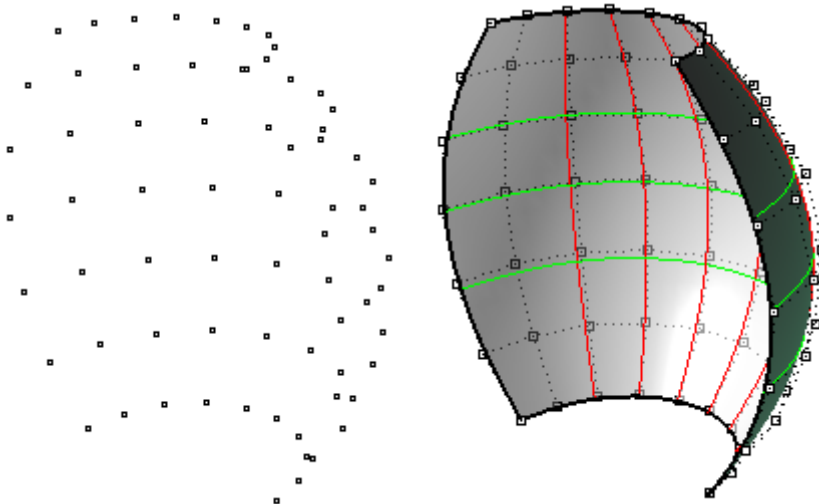


Command flow

- 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.

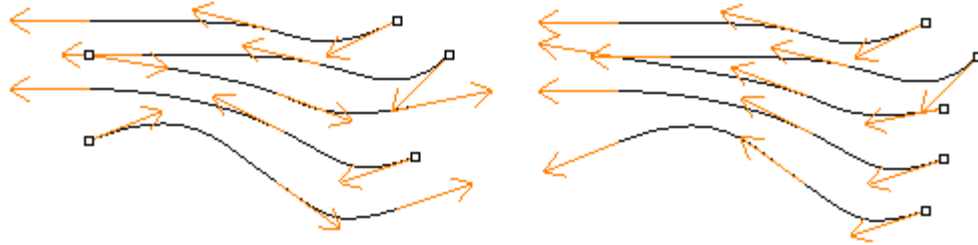


Command flow

- 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.



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

Using the above four methods.

OrderOfSelection

Selection order.

Coordinates

World coordinates.

Direction

User-defined direction.

Surface

Reference surface.

Prefix

Name prefixed to serial number.

StartIndex

Starting number.

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.