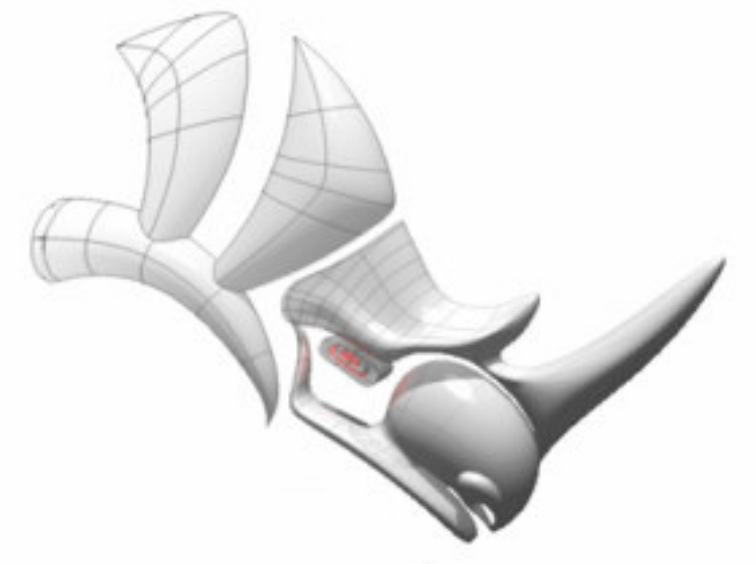
Using Rhinoceros

ArchCut Scripting



Overview

ArchCut scripting for Rhino4 aims to make ArchCut methods available for RhinoScript developers. More importantly, it allows users to define custom patterns for paneling.

This document includes full description and examples of ArchCut scripting methods. It also illustrates how to define custom patterns.

Accessing ArchCut methods: how to get ArchCut plug-in object:

First step is to get hold of ArchCut plugin object. Make sure ArchCut.rhp plugin is loaded when you start Rhino (use PluginManager command to load ArchCut.rhp).

The plugin object is accessed using **"GetPluginObject("*Plugin Name*")"** as shown in the following.

Syntax

Rhino.GetPluginObject (strPlugIn)

Parameters

strPlugIn Required. String. The name of a registered plug-in that supports scripting. If the plug-in is registered but not loaded, it will be loaded.

Returns

Object A scriptable object if successful.

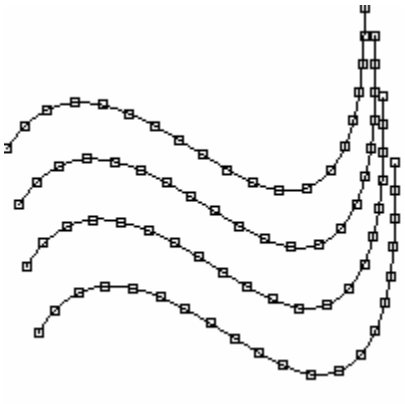
Null If not successful, or on error.

Example

```
Call Main()
Sub Main()
    Dim ArchCut
    On Error Resume Next
    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If
    'Use ArchCut Object Method
    MsgBox ArchCut.About

    'Reset ArchCut Plugin Object
    Set ArchCut = Nothing
End Sub
```

Curve dividing points by distance

Finds dividing points of a curve by distance. There are rounding options available.



ArchCut.DivideByDistance

Syntax

ArchCut. DivideByDistance (*strObject, doubleDis, boolRound, boolRoundMethod, boolAdd)*

Parameters

strObject	String. Curve object to be divided.
doubleDis	Double. Distance between dividing points
bRound	Bool. Rounding to fit curve length
bRoundMethod	Bool. true = round down. false = round up
bAdd	Bool. Add dividing points to context

Returns

Array	Array of dividing points
Null	If not successful, or on error.

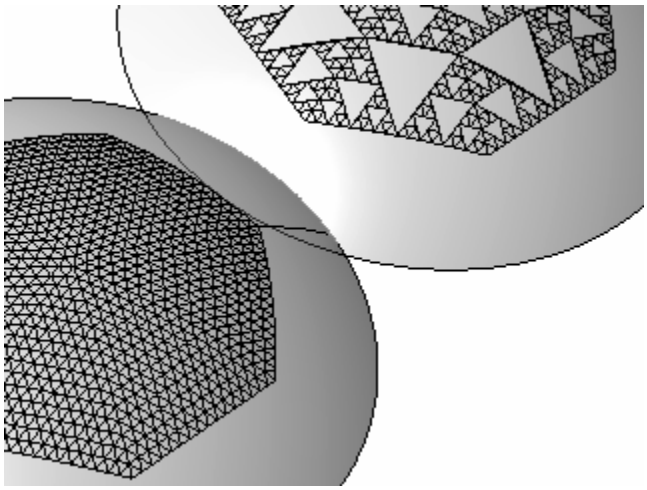
Example

```
Call Main()
Sub Main()

    Dim strObjects, strObjects , arrPoints, ArchCut
    Const doubleDis = 1.5          'Dividing Distance
    Const bRound = true            'Rounding
    Const bRoundMethod = false     'True=RoundDown  False=RoundUp
    Const bAdd = true              'Add points to document
```

```
    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select curves to divide
    strObjects = Rhino.GetObject("Select curves",4)

    If IsArray(strObjects) Then
        For Each strObject in strObjects
            If Rhino.IsCurve(strObject) Then

                'Call Curve Dividing Method
                arrPoints = ArchCut.DivideByDistance( strObject,
doubleDis , bRound, bRoundMethod, bAdd )

            End If
        Next
    End If

    Set ArchCut = Nothing
End Sub
```

Sub paneling

Paneling surface using seed polylines. Polylines are sub divided and pulled back to surfaces to generate panels.



ArchCut.SunPaneling

Syntax

ArchCut. SubPaneling(***strSrfObject, arrCrvObject, intMethod, intDegree, bPull)***

Parameters

strSrfObject	String. Surface object to be sub-paneled
arrCrvObject	Array of Strings. Polyline objects to sub-panel surface with
intMethod	Integer. 0=all, 1=subs only, 2=mains only
intDegree	Integer. Levels of sub paneling
bPull	Bool. Pull panels to surface or keep straight

Returns

Array	Array of string of panel curves objects
Null	If not successful, or on error.

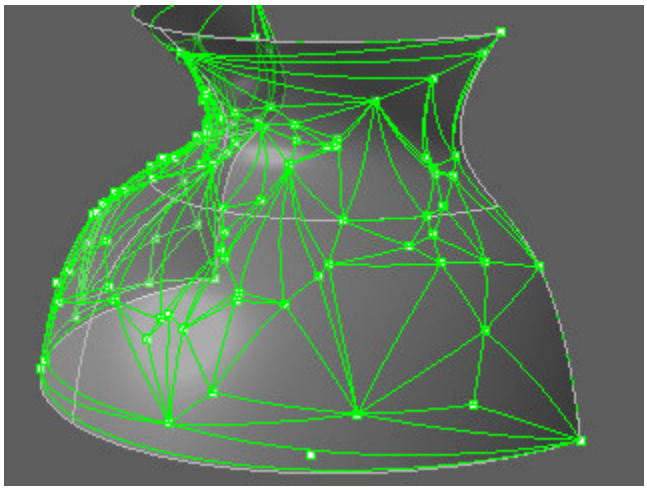
Example

```
Call Main()
Sub Main()

    Dim arrCrvObject, strSrfObject, arrPanels, ArchCut
    Const intMethod = 0 'Method = 0=all, 1=subs only, 2=mains only
    Const intDegree = 4 'Degree = Levels of sub paneling
    Const bPull = true  'Pull panels to surface

    On Error Resume Next
    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a surface
    strSrfObject = Rhino.GetObject("Select a surface",8)

    If Rhino.IsSurface(strSrfObject) Then

        'Get any number of polylines to Sub Panel surface with
        arrCrvObject = Rhino.GetObjects("Select polylines",4)

        If IsArray(arrCrvObject) Then

            'Call Sub Paneling Method
            arrPanels = ArchCut.SubPaneling( strSrfObject,
arrCrvObject, intMethod, intDegree, bPull )
```

```
        End If
    End If

    Set ArchCut = Nothing
End Sub
```

Random paneling

Triangular paneling using random set of points on surface.



ArchCut.RandomPaneling

Syntax

ArchCut. **RandomPaneling(** ***strSrfObject, intNumOfPoints, bPull)***

Parameters

strSrfObject	String. Surface object to be paneled randomly
intNumOfPoints	Integer. Number of points generated randomly on surface
bPull	Bool. Pull panels to surface or keep straight

Returns

Array	Array of string of panel curves objects
Null	If not successful, or on error.

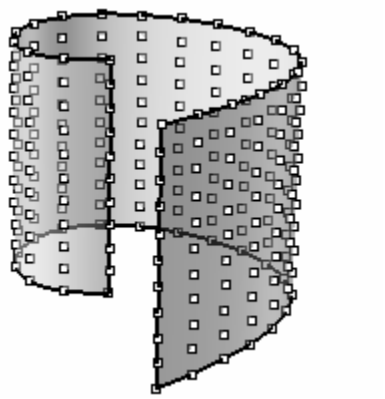
Example

```
Call Main()
Sub Main()
```

```
    Dim arrCrvObject, strSrfObject, arrPanels, ArchCut
    Const intNumOfPoints = 30 'Number of Points generated randomly
    Const bPull = false       'Pull panels to surface

    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a surface
    strSrfObject = Rhino.GetObject("Select a surface", 8)

    If Rhino.IsSurface(strSrfObject) Then

        'Call Random Paneling Method
        arrPanels = ArchCut.RandomPaneling( strSrfObject,
intNumOfPoints , bPull )
    End If

    Set ArchCut = Nothing
End Sub
```

Generate UV points based on number

Generates UV grid points of given surface by number of points in U & V directions. This method has optional pattern definition parameters (see custom patterns section).



ArchCut.GenerateUVPointsNUM

Syntax

ArchCut.GenerateUVPointsNUM(***strBrepObject, intUNum, intVNum*** **)**

Parameters

strBrepObject	String. Object to generate point grid for
intUNum	Integer. Number of points in first (U) direction
intVNum	Integer. Number of points in second (V) direction

Returns

ArrayOfArray	Array of point-objects array (2 dimensional array of points)
Null	If not successful, or on error.

Example A

```
Call Main()
Sub Main()

    Dim strBrepObject, arrPoints, ArchCut
    Const intUNum = 20  'Number of points in U direction
    Const intVNum = 10  'Number of points in V direction

    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a polysurface
    strBrepObject = Rhino.GetObject("Select object to generate UV based
grid points by number", 8 +16)

    If (strBrepObject <> vbNull) Then

        'Call UV Paneling Method
        arrPoints = ArchCut.GenerateUVPointsNUM( strBrepObject,
intUNum, intVNum)
    End If

    Set ArchCut = Nothing
End Sub
```

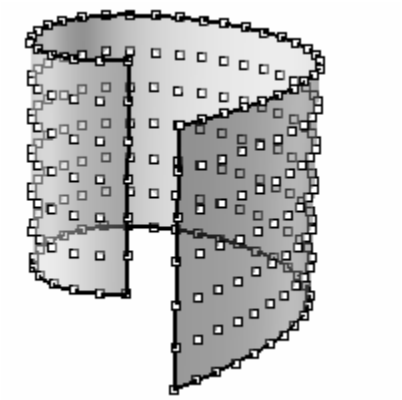
Example B

ArchCut.**GenerateUVPointsNUM()** returns an array of points arrays (two dimensional array of point objects). Following is ExampleA with test part to show how to access points.

```
Call Main()
Sub Main()

    Dim strBrepObject, arrPoints, ArchCut, Points, Pt
    Const intUNum = 20  'Number of points in U direction
    Const intVNum = 10  'Number of points in V direction

    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a polysurface
    strBrepObject = Rhino.GetObject("Select object to generate UV based
grid points by number", 8 +16)

    If (strBrepObject <> vbNull) Then

        'Call UV Paneling Method
        arrPoints = ArchCut.GenerateUVPointsNUM( strBrepObject,
intUNum, intVNum)
    End If

    ' TEST POINTS
    Rhino.Print "Point Grid Objects:"
      If IsArray(arrPoints) Then
        For Each Points In arrPoints
          If IsArray(Points) Then
            For Each  Pt In Points
              Rhino.Print Pt
            Next
          End If
        Next
      End If

    Set ArchCut = Nothing
End Sub
```

Generate UV points based on distance

Generates UV grid points of given surface by distance between points in U & V directions. This method has optional pattern definition parameters (see custom patterns section).



ArchCut.GenerateUVPointsDIS

Syntax

ArchCut.GenerateUVPointsDIS(***strBrepObject, doubleUDis, doubleVDis, bRound, bRoundMethod***)

Parameters

strBrepObject	String. Object to generate point grid for
doubleUDis	Double. Distance bewteen points in first (U) direction
doubleVDis	Double. Distance bewteen points in second (V) direction
bRound	Bool. Rounding to fit surface U,V length
bRoundMethod	Bool. true = round down. false = round up

Returns

ArrayOfArray	Array of point-objects array (2 dimensional array of points)
Null	If not successful, or on error.

Example

```
Call Main()
Sub Main()

    Dim strBrepObject, arrPoints, ArchCut
```

```
    Const doubleUDis = 1.1     'Number of points in U direction
    Const doubleVDis = 1.6     'Number of points in V direction
    Const bRound = true        'Rounding
    Const bRoundMethod = true  'Rounding Method

    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a polysurface
    strBrepObject = Rhino.GetObject("Select object to generate UV based
grid points by distance", 8 +16)

    If (strBrepObject <> vbNull) Then

        'Call UV Paneling Method
        arrPoints = ArchCut.GenerateUVPointsDIS( strBrepObject,
doubleUDis, doubleVDis, bRound, bRoundMethod)
    End If

    Set ArchCut = Nothing
End Sub
```

Generate direction-based point grid

It is possible to generate points following a user-defined direction using "ArchCut.GenerateDirPointsNUM" and "ArchCut.GenerateDirPointsDIS" as in the following two examples:

Syntax

ArchCut.GenerateDirPointsNUM(*strBrepObject, arrDir, doubleDis, intNum*)

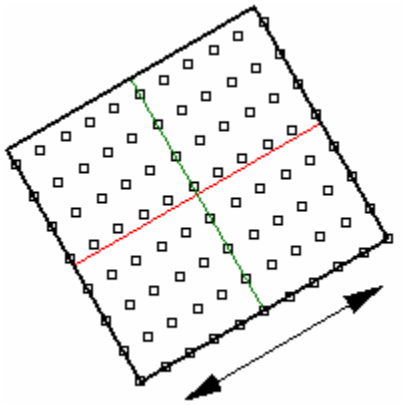
Parameters

strBrepObject	String. Object to generate point grid for
arrDir	Array of 2 points to define cutting direction
doubleDis	Double. Distance between cuts
intNum	Integer. Number of points per cut

Returns

ArrayOfArray	Array of point-objects array (2 dimensional array of points)
Null	If not successful, or on error.

ExampleA



ArchCut.GenerateDirPointsNUM

```
Call Main()
Sub Main()

      Dim strBrepObject, ArchCut, arrPoints

      Const doubleDis = 2.0   'Distance between cuts
      Const intNum = 10     'Number of panels per cut

      'Direction = CPlaneY
      Dim dir_p0 : dir_p0 = Array(0,0,0)
      Dim dir_p1 : dir_p1 = Array(0,1,0)
      Dim arrDir : arrDir = Array(dir_p0, dir_p1)

      On Error Resume Next
      Set ArchCut = Rhino.GetPluginObject("ArchCut")
      If Err Then
            MsgBox Err.Description
            Exit Sub
      End If

      strBrepObject = Rhino.GetObject("Select object to generate UV
based grid points by number", 8 +16)

       If (strBrepObject <> vbNull) Then
            arrPoints = ArchCut.GenerateDirPointsNUM( strBrepObject,
arrDir, doubleDis, intNum )

      End If
      Set ArchCut = Nothing
End Sub
```

Syntax

ArchCut.GenerateDirPointsDis(*strBrepObject, arrDir, doubleDis, doubleUDis, bRound, bRoundMethod)*

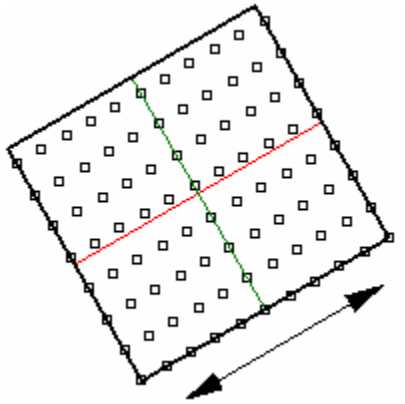
Parameters

strBrepObject	String. Object to generate point grid for
doubleDir	Array of 2 points to define cutting direction
doubleDis	Double. Distance between cuts
doubleUDis	Double. Distance between points per curve
bRound	Bool. Rounding to fit surface U,V length
bRoundMethod	Bool. true = round down. false = round up

Returns

ArrayOfArray	Array of point-objects array (2 dimensional array of points)
Null	If not successful, or on error.

ExampleB



ArchCut.GenerateDirPointsDIS

```
! -_Runscript
(
Call Main()
Sub Main()

      Dim strBrepObject, ArchCut, arrPoints

                Const doubleDis = 5.0    'Distance between cuts
                Const doubleUDis = 1.0   'Distance of panels per cut
                Const bRound = true      'Rounding
```

```
                Const bRoundMethod = true 'Rounding Method

                'Direction = CPlaneX
                Dim dir_p0 : dir_p0 = Array(0,0,0)
                Dim dir_p1 : dir_p1 = Array(1,0,0)
                Dim arrDir : arrDir = Array(dir_p0, dir_p1)

      On Error Resume Next
      Set ArchCut = Rhino.GetPluginObject("ArchCut")
      If Err Then
            MsgBox Err.Description
            Exit Sub
      End If

      strBrepObject = Rhino.GetObject("Select object to generate UV
based grid points by number", 8 +16)

      If (strBrepObject <> vbNull) Then
            arrPoints = ArchCut.GenerateDirPointsDIS( strBrepObject,
arrDir, doubleDis, doubleUDis, bRound, bRoundMethod )

      End If
      Set ArchCut = Nothing
End Sub
)
```

Paneling and patterns

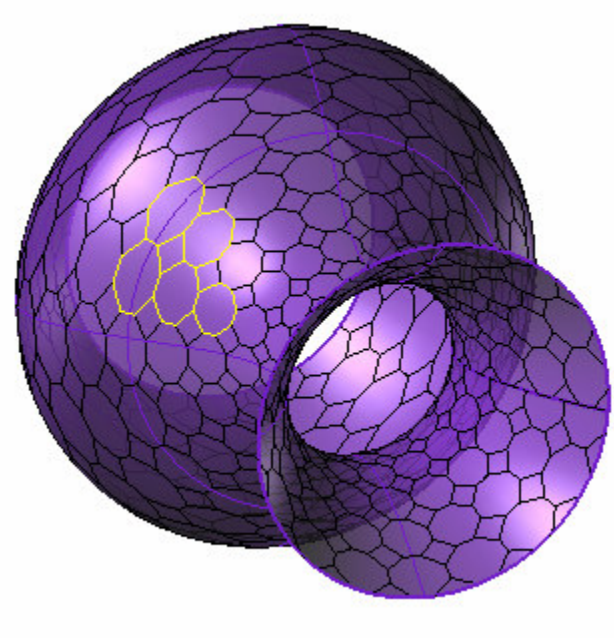
Once point grid is generated, patterns are defined with three parameters: base, shift and pattern curve points as follows:
Base: 2d point (u,v) that defines the start of the paneling pattern relative to point grid.

Shift: another 2d point (u,v) that defines spacing among repeated pattern units.

Pattern curve(s) points: this defines the polyline points of the unit panel. It can be closed or open polyline. Even if the base point is not (0,0), polyline is defined relative to a (0,0) base point. In effect it is the shift in point grid relative to the base point.

Generate points grid and panels in one step

GenerateUVPanelsNUM method creates surface point grid and use it to panel with some user-defined pattern. Later we will discuss another method where point grid generation and paneling is done in two separate steps. The latter is useful when the user like to edit the point grid such as changing points locations. In addition, if there is more than one pattern to be added to the point grid, then it is faster to creates points first then use those to create panels (instead of recalculating points multiple times).



ArchCut.GenerateUVPanelsNUM

Syntax

ArchCut.GenerateUVPanelsNUM(***strBrepObject, intUNum, intVNum, bAdd, bPull, arrBase, arrShift, arrPattern*** **)**

Parameters

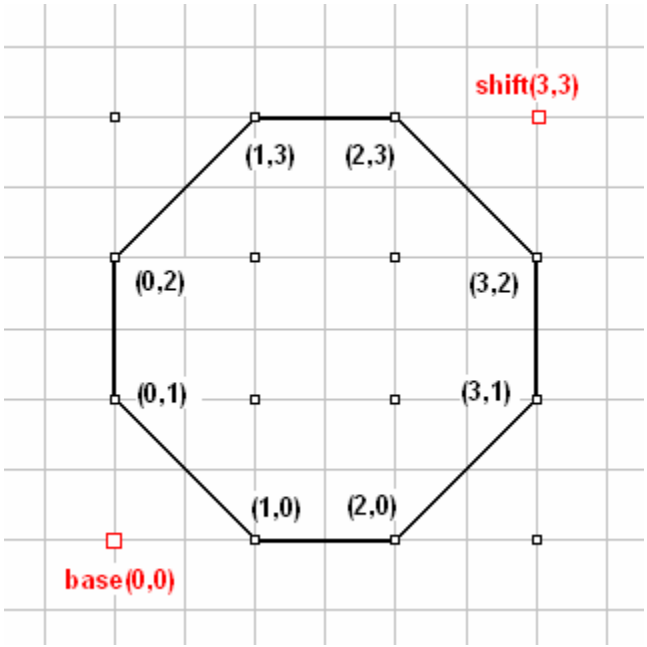
strBrepObject	String. Object to generate point grid for
intUNum	Integer. Number of points in first (U) direction
intVNum	Integer. Number of points in second (V) direction
bAdd	Bool. Add point grid to document
bPull	Bool. Pull panels to surface or make them straight if set to false
arrBase	Array of two integer numbers defining base relative to point grid
arrShift	Array of two integer numbers defining u and v shift in pattern units
arrPattern	Array of even number of integer numbers defining pattern unit polyline

Returns

Array	Array of curve-objects (Panels).
Null	If not successful, or on error.

Example

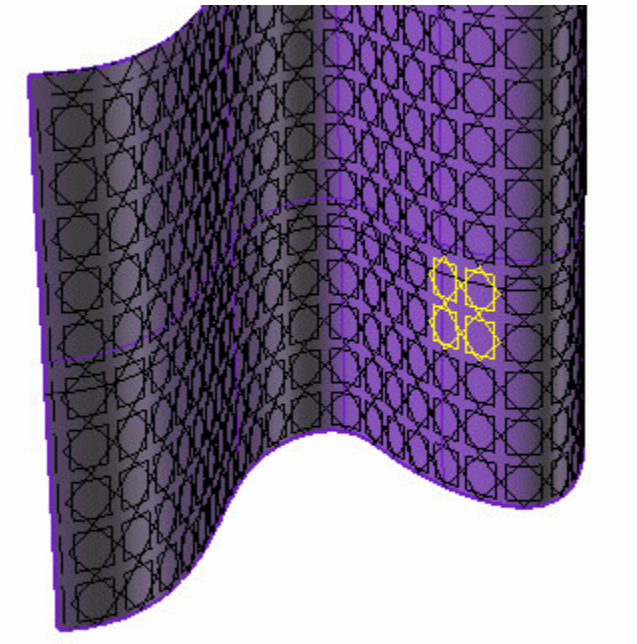
User need to define point grid through specifying number of points in each of the two surface directions. Drawing a pattern diagram helps define pattern parameters as illustrated in the following:



Pattern definition

```
Call Main()
Sub Main()

    Dim strBrepObject, arrPoints, ArchCut, arrBase(1), arrShift(1),
arrPattern(17)
    Const intUNum = 50 'Number of points in U direction
    Const intVNum = 50 'Number of points in V direction
    Const bAdd = false 'Add points to context
    Const bPull = true 'Pulled to surface. false=straight panels

    'PATTERN DEFINITION
    'Base u,v of pattern unit
    arrBase(0) = 0 : arrBase(1) = 0

    'Shift amount in u,v of unit for pattern repetition
    arrShift(0) = 3 : arrShift(1) = 3

    '(u,v) Panel points shift (relative to (0,0) base point)
    arrPattern(0) = 1 : arrPattern(1) = 0    ' (1,0)
    arrPattern(2) = 2 : arrPattern(3) = 0    ' (2,0)
    arrPattern(4) = 3 : arrPattern(5) = 1    ' (3,1)
    arrPattern(6) = 3 : arrPattern(7) = 2    ' (3,2)
    arrPattern(8) = 2 : arrPattern(9) = 3    ' (2,3)
    arrPattern(10) = 1 : arrPattern(11) = 3 ' (1,3)
    arrPattern(12) = 0 : arrPattern(13) = 2 ' (0,2)
    arrPattern(14) = 0 : arrPattern(15) = 1 ' (0,1)
    arrPattern(16) = 1 : arrPattern(17) = 0 ' (1,0)

    'END OF PATTERN DEFINITION
    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a polysurface
    strBrepObject = Rhino.GetObject("Select object to generate UV based
grid points by number", 8 +16)

    If (strBrepObject <> vbNull) Then

        'Call UV Paneling Method with pattern definition
        arrPoints = ArchCut.GenerateUVPanelsNUM( strBrepObject,
intUNum, intVNum, bAdd, bPull, arrBase, arrShift, arrPattern )

    End If

    Set ArchCut = Nothing
End Sub
```

GenerateUVPanelsDIS method also generates paneling using an object as an input. Difference from previous function is that point grid is defined by distance instead of number. In the following example, pattern consists of two polylines. The function need to be called twice (once for each pattern).



ArchCut.GenerateUVPanelsNUM

Syntax

ArchCut.GenerateUVPanelsDIS(***strBrepObject, doubleUDis, doubleVDis, bRound, bRoundMethod, bAdd, bPull, arrBase, arrShift, arrPattern*** **)**

Parameters

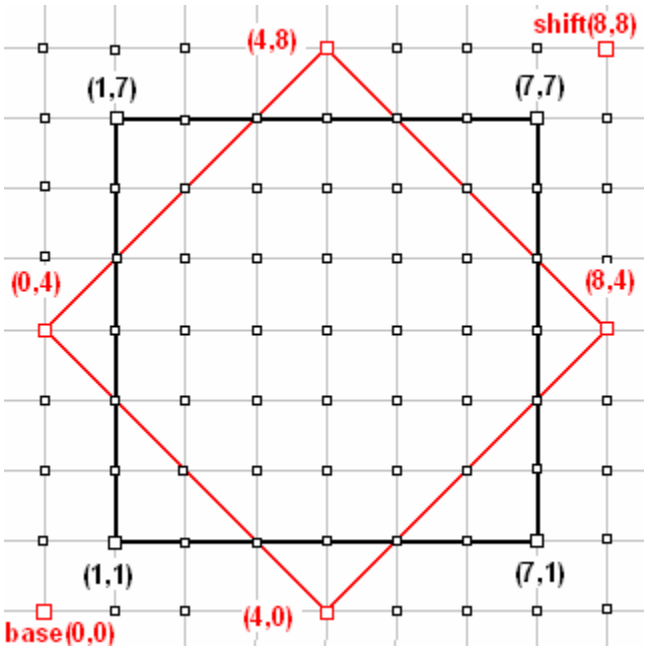
strBrepObject	String. Object to generate point grid for
doubleUDis	Double. Distance between points in first (U) direction
doubleVDis	Double. Distance between points in second (V) direction
bRound	Bool. Rounding to fit surface U,V length
bRoundMethod	Bool. true = round down. false = round up
bAdd	Bool. Add point grid to document
bPull	Bool. Pull panels to surface or make them straight if set to false

arrBase	Array of two integer numbers defining base relative to point grid
arrShift	Array of two integer numbers defining u and v shift in pattern units
arrPattern	Array of even number of integer numbers defining pattern unit polyline

Returns

Array	Array of curve-objects (Panels).
Null	If not successful, or on error.

Example



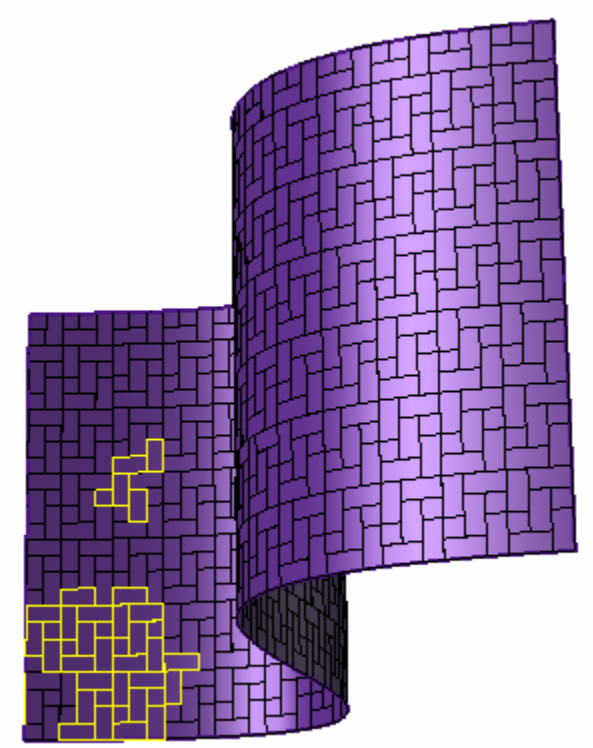
Pattern definition

```
Call Main()
Sub Main()

    Dim strBrepObject, arrPoints, ArchCut, arrBase(1), arrShift(1),
arrPatternA(9), arrPatternB(9)
    Const doubleUDis =  0.2 'Distance between points in U dir
    Const doubleVDis = 0.2  'Distance in V dir
    Const bRound = true 'Round distance to fit total length
    Const bRoundMethod = true 'Rounding method Down/Up
    Const bAdd = false 'Add points to context
    Const bPull = true 'Pulled to surface. false=straight panels

    'PATTERN DEFINITION
    'Base u,v of pattern unit
    arrBase(0) = 0 : arrBase(1) = 0
```

```
    'Shift amount in u,v of unit for pattern repetition
    arrShift(0) = 8 : arrShift(1) = 8

   '(u,v) Panel points shift (relative to (0,0) base point)
    arrPatternA(0) = 1 : arrPatternA(1) = 1 ' (1,1)
    arrPatternA(2) = 7 : arrPatternA(3) = 1 ' (7,1)
    arrPatternA(4) = 7 : arrPatternA(5) = 7 ' (7,7)
    arrPatternA(6) = 1 : arrPatternA(7) = 7 ' (1,7)
    arrPatternA(8) = 1 : arrPatternA(9) = 1 ' (1,1)

   '(u,v) Panel points shift (relative to (0,0) base point)
    arrPatternB(0) = 0 : arrPatternB(1) = 4 ' (0,4)
    arrPatternB(2) = 4 : arrPatternB(3) = 0 ' (4,0)
    arrPatternB(4) = 8 : arrPatternB(5) = 4 ' (8,4)
    arrPatternB(6) = 4 : arrPatternB(7) = 8 ' (4,8)
    arrPatternB(8) = 0 : arrPatternB(9) = 4 ' (0,4)

    On Error Resume Next

    'Get ArchCut Object
    Set ArchCut = Rhino.GetPluginObject("ArchCut")

    If Err Then
        MsgBox Err.Description
        Exit Sub
    End If

    'Select a polysurface
    strBrepObject = Rhino.GetObject("Select object to generate UV based
grid points by number", 8 +16)

    If (strBrepObject <> vbNull) Then

        'Call UV Paneling Method for each pattern
        arrPoints = ArchCut.GenerateUVPanelsDIS( strBrepObject,
doubleUDis, doubleVDis, bRound, bRoundMethod, bAdd, bPull, arrBase,
arrShift, arrPatternA )

        arrPoints = ArchCut.GenerateUVPanelsDIS( strBrepObject,
doubleUDis, doubleVDis, bRound, bRoundMethod, bAdd, bPull, arrBase,
arrShift, arrPatternB )

    End If

    Set ArchCut = Nothing
End Sub
```

Generate panels using a pre-defined point grid

This is particularly useful if the user have custom point grid. Also makes it much more efficient to generate points using any of the UV or directional methods and then feed the point gid to this function. This is especially true with complex patterns. Note that point grid has to be an array of points array.



ArchCut.GeneratePanels()

Syntax

ArchCut.GeneratePanels (*strBrepObject, arrPoints, bPull, arrBase, arrShift, arrPattern*)

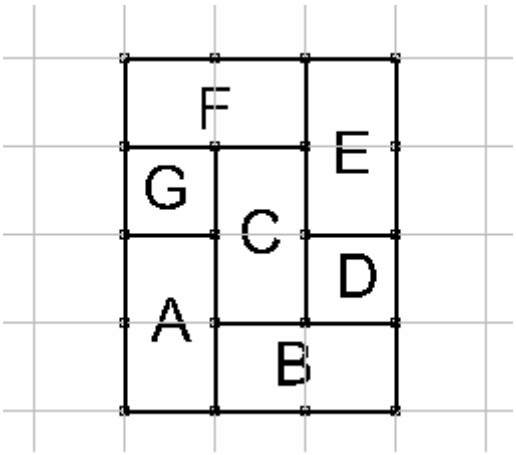
Parameters

strBrepObject	String. Object to generate point grid for
arrPoints	Array of Points Array. Paneling two dimensional point grid
bPull	Bool. Pull panels to surface or make them straight if set to false
arrBase	Array of two integer numbers defining base relative to point grid
arrShift	Array of two integer numbers defining u and v shift in pattern units
arrPattern	Array of even number of integer numbers defining pattern unit polyline

Returns

Array	Array of curve-objects (Panels).
Null	If not successful, or on error.

Example



Pattern definition

```
! -_Runscript
(
Call Main()
Sub Main()

      Dim strBrepObject, arrPoints, arrPanelsA, arrPanelsB, arrPanelsC,
arrPanelsD, arrPanelsE, arrPanelsF, arrPanelsG, ArchCut, arrBase(1),
arrShift(1), arrPatternA(9), arrPatternB(9), arrPatternC(9),
arrPatternD(9), arrPatternE(9), arrPatternF(9), arrPatternG(9), Panels,
Panel


      Const doubleUDis=  1
      Const doubleVDis = 1
      Const bRound = true
      Const bRoundMethod = true
      Const bPull = true


      'PATTERN DEFINITION
      arrBase(0) = 0 : arrBase(1) = 0
      arrShift(0) = 3 : arrShift(1) = 4

      arrPatternA(0) = 0 : arrPatternA(1) = 0
      arrPatternA(2) = 1 : arrPatternA(3) = 0
      arrPatternA(4) = 1 : arrPatternA(5) = 2
      arrPatternA(6) = 0 : arrPatternA(7) = 2
      arrPatternA(8) = 0 : arrPatternA(9) = 0

      arrPatternB(0) = 1 : arrPatternB(1) = 0
      arrPatternB(2) = 3 : arrPatternB(3) = 0
      arrPatternB(4) = 3 : arrPatternB(5) = 1
      arrPatternB(6) = 1 : arrPatternB(7) = 1
      arrPatternB(8) = 1 : arrPatternB(9) = 0
```

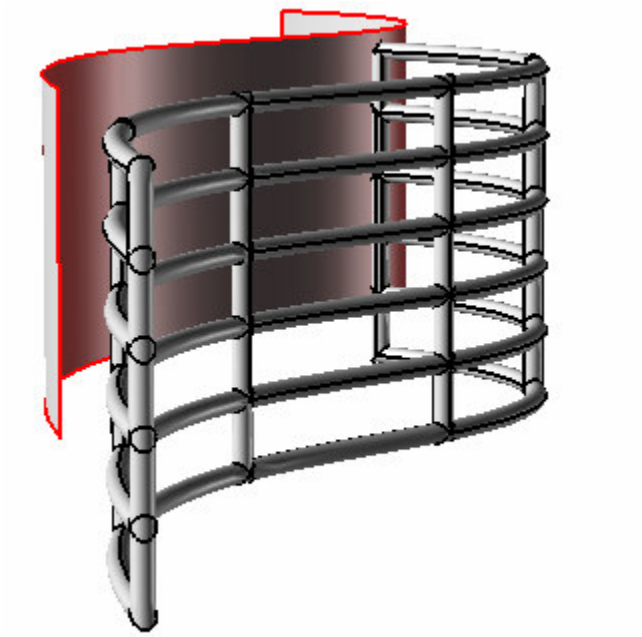
```
	arrPatternC(0) = 1 : arrPatternC(1) = 1
	arrPatternC(2) = 2 : arrPatternC(3) = 1
	arrPatternC(4) = 2 : arrPatternC(5) = 3
	arrPatternC(6) = 1 : arrPatternC(7) = 3
	arrPatternC(8) = 1 : arrPatternC(9) = 1

	arrPatternD(0) = 2 : arrPatternD(1) = 1
	arrPatternD(2) = 3 : arrPatternD(3) = 1
	arrPatternD(4) = 3 : arrPatternD(5) = 2
	arrPatternD(6) = 2 : arrPatternD(7) = 2
	arrPatternD(8) = 2 : arrPatternD(9) = 1

	arrPatternE(0) = 2 : arrPatternE(1) = 2
	arrPatternE(2) = 3 : arrPatternE(3) = 2
	arrPatternE(4) = 3 : arrPatternE(5) = 4
	arrPatternE(6) = 2 : arrPatternE(7) = 4
	arrPatternE(8) = 2 : arrPatternE(9) = 2

	arrPatternF(0) = 0 : arrPatternF(1) = 3
	arrPatternF(2) = 2 : arrPatternF(3) = 3
	arrPatternF(4) = 2 : arrPatternF(5) = 4
	arrPatternF(6) = 0 : arrPatternF(7) = 4
	arrPatternF(8) = 0 : arrPatternF(9) = 3

	arrPatternG(0) = 0 : arrPatternG(1) = 2
	arrPatternG(2) = 1 : arrPatternG(3) = 2
	arrPatternG(4) = 1 : arrPatternG(5) = 3
	arrPatternG(6) = 0 : arrPatternG(7) = 3
	arrPatternG(8) = 0 : arrPatternG(9) = 2

	On Error Resume Next
	Set ArchCut = Rhino.GetPluginObject("ArchCut")
	If Err Then
		MsgBox Err.Description
		Exit Sub
	End If

	strBrepObject = Rhino.GetObject("Select object to generate UV
based grid points by number", 8 +16)
		If (strBrepObject <> vbNull) Then

		'Get point grid
		arrPoints = ArchCut.GenerateUVPointsDIS( strBrepObject,
doubleUDis, doubleVDis, bRound, bRoundMethod)

		arrPanelsA = ArchCut.GeneratePanels( strBrepObject,
arrPoints, bPull, arrBase, arrShift, arrPatternA )
		arrPanelsB = ArchCut.GeneratePanels( strBrepObject,
arrPoints, bPull, arrBase, arrShift, arrPatternB )
		arrPanelsC = ArchCut.GeneratePanels( strBrepObject,
arrPoints,  bPull, arrBase, arrShift, arrPatternC )
		arrPanelsD = ArchCut.GeneratePanels( strBrepObject,
arrPoints, bPull, arrBase, arrShift, arrPatternD )
		arrPanelsE = ArchCut.GeneratePanels( strBrepObject,
arrPoints,  bPull, arrBase, arrShift, arrPatternE )
```

```
            arrPanelsF = ArchCut.GeneratePanels( strBrepObject,
arrPoints, bPull, arrBase, arrShift, arrPatternF )
            arrPanelsG = ArchCut.GeneratePanels( strBrepObject,
arrPoints, bPull, arrBase, arrShift, arrPatternG )


  'TEST PANELS
  Rhino.Print "Panel Objects:"
  If IsArray(arrPanelsA) Then
    For Each Panels In arrPanels
          Rhino.Print Panel
    Next
  End If

      End If
      Set ArchCut = Nothing
End Sub
)
```

Scripting examples to customize ArchCut commands

Pipe panels:

This example shows how to pipe paneling curves.



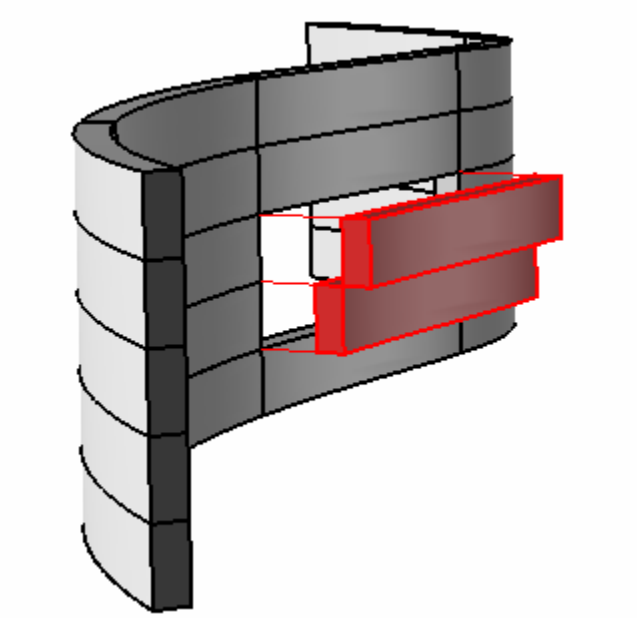
Piping example

```
! -_Runscript
(

'Piping of box paneling pattern

Sub Pipe(strPanel, pipeRadius)
      Dim strCmd
      'Pipe the curve using rhino's command string format
      strCmd = "! _Pipe _SelID " & strPanel & " " & pipeRadius & "
_Enter _Enter"
      Rhino.Command strCmd
End Sub



Call Main()
Sub Main()

      Dim strBrepObject, arrPanels, ArchCut, arrPtsNames, strPanel,
arrBase(1), arrShift(1), arrPattern(9)
      Const intUNum = 5
```

```
	Const intVNum = 5
	Const bAdd = false

	'PATTERN DEFINITION
	Const bPull = true
	arrBase(0) = 0
	arrBase(1) = 0
	arrShift(0) = 1
	arrShift(1) = 1

	arrPattern(0) = 0
	arrPattern(1) = 0
	arrPattern(2) = 1
	arrPattern(3) = 0
	arrPattern(4) = 1
	arrPattern(5) = 1
	arrPattern(6) = 0
	arrPattern(7) = 1
	arrPattern(8) = 0
	arrPattern(9) = 0


	On Error Resume Next
	Set ArchCut = Rhino.GetPluginObject("ArchCut")
	If Err Then
		MsgBox Err.Description
		Exit Sub
	End If

	strBrepObject = Rhino.GetObject("Select object to generate UV
based grid points by number", 8 +16)
		If (strBrepObject <> vbNull) Then

		arrPtsNames = ArchCut.GenerateUVPointsNUM(
strBrepObject, intUNum, intVNum )
		arrPanels = ArchCut.GeneratePanels( strBrepObject,
arrPtsNames, bPull, arrBase, arrShift, arrPattern )

		'PIPE
		Dim pipeRadius : pipeRadius = 0.2
		If IsArray(arrPanels) Then
		    For Each strPanel In arrPanels
			Call Pipe(strPanel, pipeRadius)
		    Next
		End If

	End If
	Set ArchCut = Nothing

End Sub
)
```

Offset panels:

In this example, 2D paneling is offset as 3D solid .



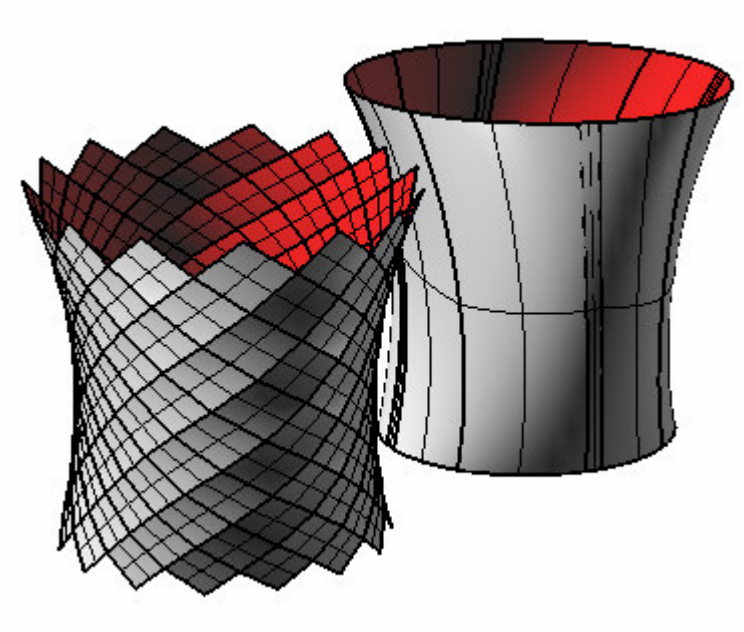
Offset solid example

```
! -_Runscript
(
'Offset solid with paneling pattern
Sub OffsetNormal(strBrepObject, offsetDis)
      Dim strCmd
      'Offset surface solid using rhino's command string format
      strCmd = "! _OffsetSrf _SelID " & strBrepObject & " _Enter _Solid
" & offsetDis & " _Enter"
      Rhino.Command strCmd
End Sub

'Split with paneling pattern
Sub SplitFace(strBrepObject, arrPanels)
      Dim strCmd, name
      'Split the surface using rhino's command string format
      strCmd = "! _SplitFace _SelID " & strBrepObject & " _Enter
_Curves "
      'Select Panels
      If IsArray(arrPanels) Then
            For Each name In arrPanels
                  strCmd = strCmd & " _SelID " & name
            Next
      End If

      strCmd = strCmd & " _Enter "
      Rhino.Command strCmd
```

```
End Sub

Call Main()
Sub Main()

      Dim strBrepObject, arrPanels, ArchCut, arrPtsNames, names, name,
arrBase(1), arrShift(1), arrPattern(9)
      Const intUNum = 5
      Const intVNum = 5
      Const bAdd = False

      'PATTERN DEFINITION
      Const bPull = True
      arrBase(0) = 0 : arrBase(1) = 0
      arrShift(0) = 1 : arrShift(1) = 1

      arrPattern(0) = 0
      arrPattern(1) = 0
      arrPattern(2) = 1
      arrPattern(3) = 0
      arrPattern(4) = 1
      arrPattern(5) = 1
      arrPattern(6) = 0
      arrPattern(7) = 1
      arrPattern(8) = 0
      arrPattern(9) = 0

      On Error Resume Next
      Set ArchCut = Rhino.GetPluginObject("ArchCut")
      If Err Then
            MsgBox Err.Description
            Exit Sub
      End If

      strBrepObject = Rhino.GetObject("Select object to generate UV
based grid points by number", 8 +16)
      If (strBrepObject <> vbNull) Then
            arrPtsNames = ArchCut.GenerateUVPointsNUM( strBrepObject,
intUNum, intVNum )
            arrPanels = ArchCut.GeneratePanels( strBrepObject,
arrPtsNames, bPull, arrBase, arrShift, arrPattern )

            Rhino.UnselectAllObjects()
            'Split Face
            Call SplitFace(strBrepObject, arrPanels)
            Rhino.UnselectAllObjects()
            'Offset Face
            Dim offsetDis : offsetDis = 0.5
            Call OffsetNormal(strBrepObject, offsetDis)

      End If
      Set ArchCut = Nothing
End Sub
)
```

Panels surfaces:

Following script uses EdgeSrf command to turn all panel faces outline into surfaces .



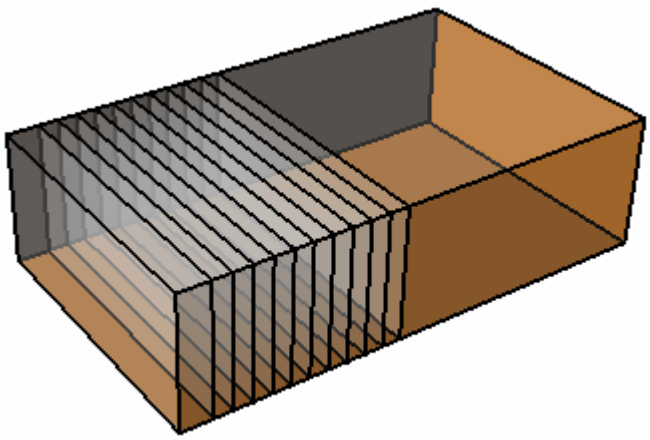
Group EdgeSrf example

```
! -_Runscript
(
Option Explicit
'Script written by Rajaa Issa
'Script copyrighted by Robert McNeel & Associates
'Script version Thursday, February 01, 2007 2:41:25 PM
'Create panels surfaces - works only with three and four sided panels

Sub EdgeSrfAll(arrCrvSegments)
      Dim strCmd, strCrv
      Rhino.UnselectAllObjects()

      'Select Panels
      If IsArray(arrCrvSegments) Then
            For Each strCrv In arrCrvSegments
                  Rhino.SelectObject( strCrv )
            Next
      End If

      'EdgeSrf panel
      strCmd = "NoEcho ! _EdgeSrf _Enter"
      Rhino.Command strCmd
End Sub

Call Main()
Sub Main()
```

```
	Dim arrCrvObject, strCrv, arrCrvSegments

	arrCrvObject = Rhino.GetObjects("Select Panels",4)
	If IsArray(arrCrvObject) Then
		For Each strCrv In arrCrvObject
			'Explode
			arrCrvSegments = Rhino.ExplodeCurves( strCrv )
			'EdgeSrf
			Call EdgeSrfAll(arrCrvSegments)
		Next
	End If

End Sub
)
```

Multiple sections with fixed and variable distances:

Following script enables creating specified number of sections with user defined spacing. The user defines base point relative to active CPlane (construction plane). The script sections through all visible objects.

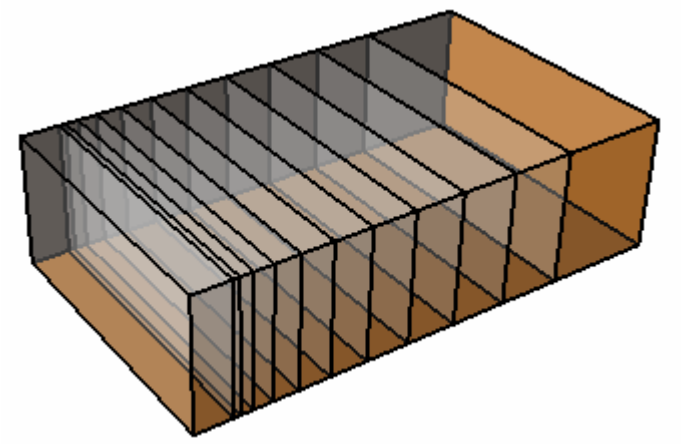


Array of sections

```
! -_Runscript
(
Call Main()
Sub Main()

 Dim y, arrPt, i

 Const x = 0
 Const z = 0
 Const Num = 10          'Number of Sections
 Const Spacing = 1.5     'Spacing between sections
```

```
 For i=0 To Num
  y = y + Spacing
  arrPt = Array(x,y,z)
  str = "ArchCut_Create _Enter " & Rhino.Pt2Str(arrPt) & " _Enter"

  'Call command line
  Rhino.Command str

  Rhino.UnselectAllObjects()
 Next
End Sub
)
```

Here is same example but with variable distances.



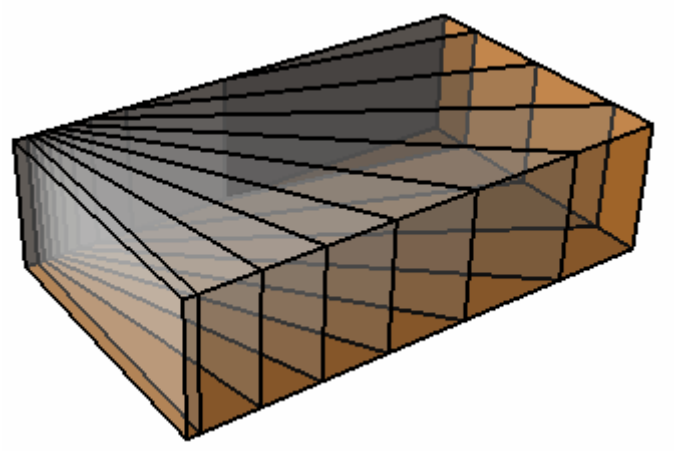
Array of sections with variable distance

```
! -_Runscript
(
Call Main()
Sub Main()

 Dim y, arrPt, i
 'Base point (x,y,z)
 Const x = 0
 y = 2
 Const z = 0

 Const Num = 10          'Number of Sections
 Const Spacing = 0.5     'Spacing between sections

 For i=0 To Num

  y = y + i*Spacing
  arrPt = Array(x,y,z)
  str = "ArchCut_Create _Enter " & Rhino.Pt2Str(arrPt) & " _Enter"
```

```
   'Call command line
   Rhino.Command str

   Rhino.UnselectAllObjects()

 Next
End Sub
)
```

Multiple sections with fixed and variable angles:

Now, if you like to rotate sectioning direction, following example script can help.

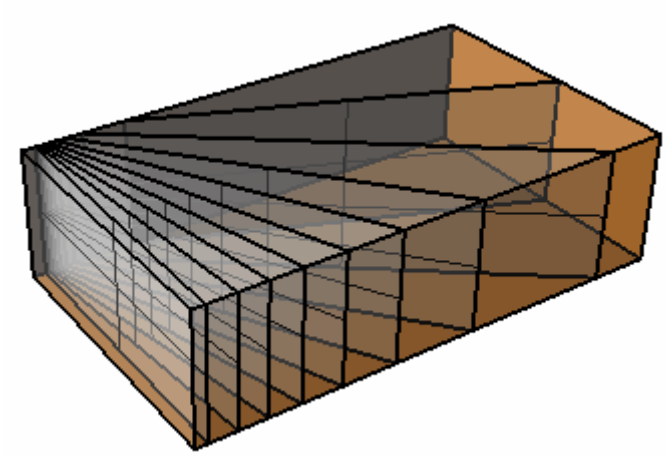


Polar array of sections

```
! -_Runscript
(
Call Main()
Sub Main()

 Dim fromPt, toPt, str, i, arrMatrix(3,3), Angle

 'Define cut direction relative to active CPlane
 fromPt = Array(0,0,0)   'Base point
 toPt = Array(10,0,0)    'To Point

 Const Num = 10                    'Number of sections
 Const DA = 5                      'Angle in Degrees
 Angle = Rhino.ToRadians( DA )     'Angle in Radians

 'Define Rotation Materix
 arrMatrix(0,0) = Cos(Angle) : arrMatrix(0,1) = -Sin(Angle) :
arrMatrix(0,2) = 0 : arrMatrix(0,3) = 0
 arrMatrix(1,0) = Sin(Angle) : arrMatrix(1,1) = Cos(Angle) :
arrMatrix(1,2) = 0 : arrMatrix(1,3) = 0
```

```
 arrMatrix(2,0) = 0 : arrMatrix(2,1) = 0 : arrMatrix(2,2) = 1 :
arrMatrix(2,3) = 0
 arrMatrix(3,0) = 0 : arrMatrix(3,1) = 0 : arrMatrix(3,2) = 0 :
arrMatrix(3,3) = 1


 For i=0 To Num

  'Call ArchCut commands and set direction
  str = "ArchCut_Create Dir Pick " & Rhino.Pt2Str(fromPt) & " " &
Rhino.Pt2Str(toPt) & " _Enter " & Rhino.Pt2Str(fromPt) & " _Enter"
  Rhino.Command str

  'Rotate the "to" point by Angle using rotation matrix
  toPt = Rhino.PointTransform( toPt, arrMatrix )
  Rhino.UnselectAllObjects()

 Next

End Sub
)
```

For variable angles, use the following.



Polar array of sections with variable angle

```
! -_Runscript
(
Call Main()
Sub Main()

      Dim fromPt, toPt, str, i, arrMatrix(3,3), Angle, DA
      fromPt = Array(0,0,0)
      toPt = Array(10,0,0)
```

```
      Const Num = 10    'Number of sections
      DA = 5            'Angle in Degrees
      Angle = Rhino.ToRadians( DA ) ' Angle in Radians


      For i=0 To Num
            'Calculate Rotation Materix
            arrMatrix(0,0) = Cos(Angle) : arrMatrix(0,1) = -Sin(Angle)
: arrMatrix(0,2) = 0 : arrMatrix(0,3) = 0
            arrMatrix(1,0) = Sin(Angle) : arrMatrix(1,1) = Cos(Angle) :
arrMatrix(1,2) = 0 : arrMatrix(1,3) = 0
            arrMatrix(2,0) = 0 : arrMatrix(2,1) = 0 : arrMatrix(2,2) =
1 : arrMatrix(2,3) = 0
            arrMatrix(3,0) = 0 : arrMatrix(3,1) = 0 : arrMatrix(3,2) =
0 : arrMatrix(3,3) = 1

            'Call ArchCut commands and set direction
            str = "ArchCut_Create Dir Pick " & Rhino.Pt2Str(fromPt) & "
" & Rhino.Pt2Str(toPt) & " _Enter " & Rhino.Pt2Str(fromPt) & " _Enter"
            Rhino.Command str

            'Rotate the "to" point by Angle using matrix
            toPt = Rhino.PointTransform( toPt, arrMatrix )

            'Set new angle
            DA = DA+ (i/2)
            Angle = Rhino.ToRadians( DA )

            Rhino.UnselectAllObjects()

      Next

End Sub
)
```

