

Lab2 – L 形柱の作成

March 2010 by M. Harada
Updated by DevTech AEC WG
Last modified: 6/9/2017

<VB.NET>VB.NET バージョン</VB.NET>

目的:この実習では、ファミリ API の基本を学習します。学習する項目は次のとおりです。

- 参照面を追加する
- パラメータを追加する
- 寸法を追加する

タスク: Lab1 で定義したコマンドを拡張し、L 字形のプロファイルを持つ柱を作成します。前の実習では、事前に定義された参照面、パラメータおよび寸法を使用しました。L-形のプロファイルを持つ柱を定義するために、独自の参照とパラメータ、寸法を加える必要があります。

- (0) Lab1 で定義したコマンド クラスを利用します。これは、この実習の開始点になります。ファミリ エディタと「柱(メートル単位).rft」テンプレートを使用し続けます
- (1) L 字形の「厚み」を計測する 2 つの参照面を追加する (例、「OffsetH」と「OffsetV」)
- (2) L 字形の「厚み」を定義する 2 つのパラメータを追加する(例、「Tw」と「Td」)
- (3) L 字形の「厚み」を計測する寸法をつかして、新しいパラメータでそれらにラベルを付ける(例、「Tw」と「Td」)

図 1 は、この実習に定義する予定の L-形の柱のイメージを示します。

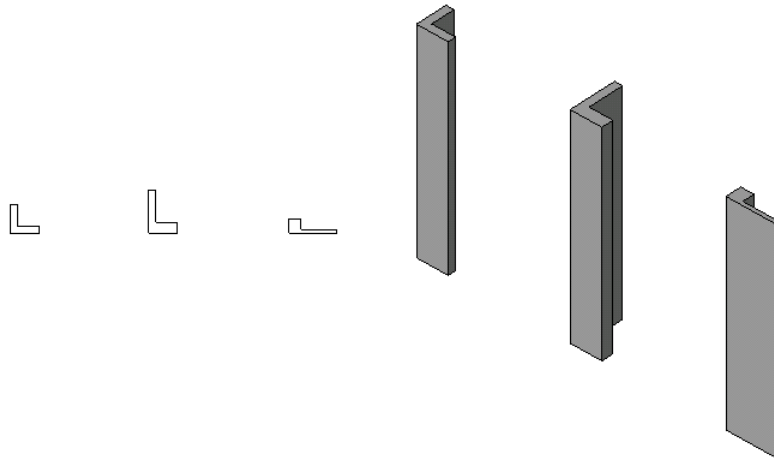
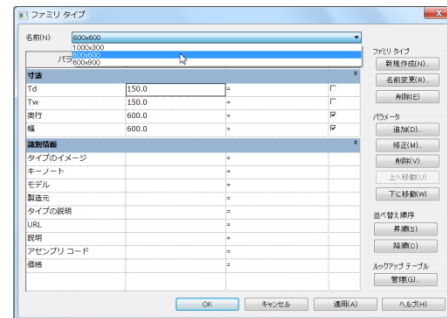
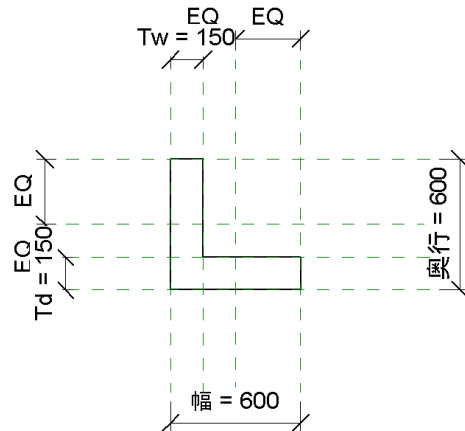


図 1.Lab2 で作成する L 字形プロファイルを持った柱ファミリ

この実習の実装と確認の手順は、下記のとおりです:

1. 別の外部コマンドを定義する
2. 参照面を追加する
3. L-形プロファイルで押し出しソリッドを作成する
4. `addAlignments()` を更新する
5. パラメータを追加する
6. 寸法を追加する
7. `addTypes()` を更新する
8. 作成した柱をテストする

1. 別の外部コマンドを定義する

Lab1 で定義したコマンドを拡張します。新しいクラスを定義するために Lab1 をコピーするか、Lab1 自体を拡張することができます(後者の場合、Lab1 の完了状態のプロジェクトをバックアップすることをお勧めします)。

1.1 Lab1 からコマンド クラスをコピーし、Lab2 で作業するための新しいクラスを定義してください。ファイル名とクラス名は、下記のようにしてください:

- ファイル名: **2_ColumnLShape.vb**
- コマンド クラス名: **RvtCmd_FamilyCreateColumnLShape**

(繰り返しになりますが、ここで希望する名前を使用しても構いません。ただし、その場合、プロジェクト名など、このドキュメント内では記述されている名称は、自分でつけた名称で代替して参照してください)

```
<VB.NET>
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
<Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.Manual)> _
Public Class RvtCmd_FamilyCreateColumnLShape

    ...

End Class
</VB.NET>
```

2. 参照面を追加する

単純な L 字形のプロファイルを定義します。このために、2 つの参照面を追加していきます。

- 参照面「OffsetH」－ 水平、「正面」参照面の 150mm 上に
- 参照面「OffsetV」－ 垂直、「左」参照面から 150mm 右に

2.1 次の関数をクラスに付け加えてください:

```
<VB.NET>
''' =====
''' (1.1) add reference planes
''' =====
```

```

Sub addReferencePlanes()

    ''
    '' we are defining a simple L-shaped profile like the following:
    ''
    '' 5 tw 4
    '' +-+
    '' | | 3          h = height
    '' d | +---+ 2
    '' +-----+ td
    '' 0         1
    '' 6 w
    ''
    ''
    '' we want to add ref planes along (1) 2-3 and (2)3-4.
    '' Name them "OffsetH" and "OffsetV" respectively. (H for horizontal,
V for vertical).
    ''
    Dim tw As Double = mmToFeet(150) ' thickness added for Lab2. Hard-
coding for simplicity.
    Dim td As Double = mmToFeet(150)

    ''
    '' (1) add a horizontal ref plane 2-3.
    ''
    '' get a plan view
    Dim pViewPlan As View = findElement(GetType(ViewPlan), "下参照レベル")

    '' we have predefined ref plane: Left/Right/Front/Back
    '' get the ref plane at Front, which is aligned to line 2-3
    Dim refFront As ReferencePlane = findElement(GetType(ReferencePlane),
"正面")

    '' get the bubble and free ends from front ref plane and offset by
td.
    ''
    Dim p1 As XYZ = refFront.BubbleEnd
    Dim p2 As XYZ = refFront.FreeEnd
    Dim pBubbleEnd As New XYZ(p1.X, p1.Y + td, p1.Z)
    Dim pFreeEnd As New XYZ(p2.X, p2.Y + td, p2.Z)

    '' create the new one
    ''
    Dim refPlane As ReferencePlane =
_rvtDoc.FamilyCreate.NewReferencePlane(pBubbleEnd, pFreeEnd, XYZ.BasisZ,
pViewPlan)
    refPlane.Name = "OffsetH"

    ''
    '' (2) do the same to add a vertical ref plane.
    ''

    '' find the ref plane at left, which is aligned to line 3-4
    Dim refLeft As ReferencePlane = findElement(GetType(ReferencePlane),
"左")

```

```

        ' ' get the bubble and free ends from front ref plane and offset by
td.
        ' '
        p1 = refLeft.BubbleEnd
        p2 = refLeft.FreeEnd
        pBubbleEnd = New XYZ(p1.X + tw, p1.Y, p1.Z)
        pFreeEnd = New XYZ(p2.X + tw, p2.Y, p2.Z)

        ' ' create the new one
        ' '
        refPlane = _rvtDoc.FamilyCreate.NewReferencePlane(pBubbleEnd,
pFreeEnd, XYZ.BasisZ, pViewPlan)
        refPlane.Name = "OffsetV"

    End Sub
</VB.NET>

```

ここで、2つの参照面を作成しています。最初の参照面を見てみましょう。平面図を見ると、テンプレート内で「Lower Ref. Level」と名付けられていることがわかります(日本語テンプレートでは「下参照レベル」)。

新しい参照面を作成する主要なメソッドは、次のとおりです:

```
_rvtDoc.FamilyCreate.NewReferencePlane(pBubbleEnd, pFreeEnd, XYZ.BasisZ, pViewPlan)
```

BubbleEndとFreeEndが平面上の2点を決定します。BubbleEndは、平面上の原点と見なされます。3番目の引数は、切断ベクトルを示し、平面に沿って始点(BubbleEnd)と終点(FreeEnd)で定義される線分に直行するベクトルです。(注意: NewReferencePlane2と呼ばれる同様のメソッドも存在します。これは、平面上で3点をとりまします)。

このケースでは、既存の参照面のオフセットを作成して、既存の参照面から2つの端点座標をコピーするアプローチをとり、Y方向にオフセット値を加えています。

2.2 メインコマンドの Execute() 関数の IsRightTemplate() 関数を呼び出す後で、かつ、createSolid() 呼び出しの前の位置で、addReferencePlanes() 関数を呼び出してください。

```

<VB.NET>
    If Not isRightTemplate(BuiltInCategory.OST_Columns) Then
        MsgBox("Please open 柱(メートル単位).rft")
        Return Result.Failed
    End If

    ' ' (1.1) add reference planes
    addReferencePlanes()

    ' ' (1.2) create a simple extrusion. This time we create a L-shape.
    Dim pSolid As Extrusion = createSolid()
</VB.NET>

```

2.3 この時点で、コードをビルドし、実行して確認することができます。

3. L 字形プロファイルで押し出しを作成します

前の実習では、長方形のプロファイルを押し出してソリッドを定義しました。この実習では、L 字形のプロファイルを使用します。前回との違いは、プロファイルの頂点を定義する部分だけです。このため、押し出し用コードは同じままで結構です。

3.1 次の関数をクラスに追加してください。このコードは、L 字形のプロファイルを定義します:

```
<VB.NET>
''' =====
''' (1.2a) create a simple L-shaped profile
''' =====
Function createProfileLShape() As CurveArray
    '''
    ''' define a simple L-shaped profile
    '''
    ''' 5 tw 4
    ''' +-+
    ''' | | 3          h = height
    ''' d | +---+ 2
    ''' +-----+ td
    ''' 0          1
    ''' 6 w
    '''

    ''' sizes (hard coded for simplicity)
    ''' note: these need to match reference plane. otherwise, alignment
won't work.
    ''' as an exercise, try changing those values and see how it behaves.
    '''
    Dim w As Double = mmToFeet(600) ' those are hard coded for
simplicity here. in practice, you may want to find out from the references)
    Dim d As Double = mmToFeet(600)
    Dim tw As Double = mmToFeet(150) ' thickness added for Lab2
    Dim td As Double = mmToFeet(150)

    ''' define vertices
    '''
    Const nVerts As Integer = 6 ' the number of vertices
    Dim pts() As XYZ = {New XYZ(-w / 2, -d / 2, 0), New XYZ(w / 2, -d / 2,
0), New XYZ(w / 2, -d / 2 + td, 0), _
                        New XYZ(-w / 2 + tw, -d / 2 + td, 0), New XYZ(-w
/ 2 + tw, d / 2, 0), New XYZ(-w / 2, d / 2, 0), _
                        New XYZ(-w / 2, -d / 2, 0)} ' the last one is to
make the loop simple

    ''' define a loop. define individual edges and put them in a
curveArray
    '''
    Dim pLoop As CurveArray = rvtApp.Create.NewCurveArray
    Dim lines(nVerts - 1) As Line
    For i As Integer = 0 To nVerts - 1
        lines(i) = Line.CreateBound(pts(i), pts(i + 1))
    Next i
End Function
```

```

        pLoop.Append(lines(i))
    Next

    ' then, put the loop in the curveArrArray as a profile
    '
    Dim pProfile As CurveArrArray = _rvtApp.Create.NewCurveArrArray
    pProfile.Append(pLoop)
    ' if we come here, we have a profile now.

    Return pProfile

End Function
</VB.NET>

```

ここでは、L 字形状のサイズと頂点と同様に、コードを単純化するためにオフセット値をハードコーディングしています。この段階でそれらの固定値を定義する主な目的は、参照で位置合わせを設定するためです。これらの値にパラメータを割り当てれば、それらを再定義することができます。

3.2 定義したプロファイルを使用して、押し出しソリッドを作成します。createSolid() 関数を参照して、createProfileRectangle() 関数の呼び出しを createProfileLShape() 関数呼び出しに置き換えてください:

```

<VB.NET>
' ' =====
' ' (1.2) create a simple solid by extrusion with L-shape profile
' ' =====
Public Function createSolid() As Extrusion

    ' '
    ' ' (1) define a simple L-shape profile
    ' '
    'Dim pProfile As CurveArrArray = createProfileRectangle()
    Dim pProfile As CurveArrArray = createProfileLShape() ' Lab2

    ...
</VB.NET>

```

3.3 この時点で、コードをビルドし、実行して確認することができます。

4. addAlignment()を更新する

位置合わせを行う関数を修正する必要があります。長方形のプロファイルでは、6つの対応参照面への6つの面を位置合わせさせました。先程定義したL字形プロファイルでは、2つの面が追加されています。位置合わせの基本概念は同じままです。ただし、findFace() ヘルパー関数は、面をより正確に識別するために3つ目のパラメータとして参照面を利用するように拡張する必要があります。

4.1 addAlignments() 関数を見つけて、次のコードで関数を更新してください:

```
<VB.NET>
''' =====
''' (2.1) add alignments
''' =====
Sub addAlignments(ByVal pBox As Extrusion)

    '''
    ''' (1) we want to constrain the upper face of the column to the
    "Upper Ref Level"
    '''

    ''' which direction are we looking at?
    '''

    Dim pView As View = findElement(GetType(View), "正面")

    ''' find the upper ref level
    ''' findElement() is a helper function. see below.
    '''

    Dim upperLevel As Level = findElement(GetType(Level), "上参照レベル")
    Dim ref1 As Reference = upperLevel.GetPlaneReference()

    ''' find the face of the box
    ''' findFace() is a helper function. see below.
    '''

    Dim upperFace As PlanarFace = findFace(pBox, New XYZ(0, 0, 1)) ' find
a face whose normal is z-up.
    Dim ref2 As Reference = upperFace.Reference

    ''' create alignments
    '''

    _rvtDoc.FamilyCreate.NewAlignment(pView, ref1, ref2)

    '''
    ''' (2) do the same for the lower level
    '''

    ''' find the lower ref level
    ''' findElement() is a helper function. see below.
    '''

    Dim lowerLevel As Level = findElement(GetType(Level), "下参照レベル")
    Dim ref3 As Reference = lowerLevel.GetPlaneReference()

    ''' find the face of the box
    ''' findFace() is a helper function. see below.
    '''

    Dim lowerFace As PlanarFace = findFace(pBox, New XYZ(0, 0, -1)) '
find a face whose normal is z-down.
    Dim ref4 As Reference = lowerFace.Reference

    ''' create alignments
    '''

    _rvtDoc.FamilyCreate.NewAlignment(pView, ref3, ref4)
```



```

''
'' (3) same idea for the width and depth.
''
'' get the plan view
'' note: same name maybe used for different view types. either one
should work.
Dim pViewPlan As View = findElement(GetType(ViewPlan), "下参照レベル")

'' find reference planes
''
Dim refRight As ReferencePlane = findElement(GetType(ReferencePlane),
"右")
Dim refLeft As ReferencePlane = findElement(GetType(ReferencePlane),
"左")
Dim refFront As ReferencePlane = findElement(GetType(ReferencePlane),
"正面")
Dim refBack As ReferencePlane = findElement(GetType(ReferencePlane),
"背面")
Dim refOffsetV As ReferencePlane =
findElement(GetType(ReferencePlane), "OffsetV") ' added for L-shape
Dim refOffsetH As ReferencePlane =
findElement(GetType(ReferencePlane), "OffsetH") ' added for L-shape

'' find the face of the box
'' note: findFace needs to be enhanced for this as face normal is
not enough to determine the face.
''
Dim faceRight As PlanarFace = findFace(pBox, New XYZ(1, 0, 0),
refRight) ' modified for L-shape
Dim faceLeft As PlanarFace = findFace(pBox, New XYZ(-1, 0, 0))
Dim faceFront As PlanarFace = findFace(pBox, New XYZ(0, -1, 0))
Dim faceBack As PlanarFace = findFace(pBox, New XYZ(0, 1, 0),
refBack) ' modified for L-shape
Dim faceOffsetV As PlanarFace = findFace(pBox, New XYZ(1, 0, 0),
refOffsetV) ' added for L-shape
Dim faceOffsetH As PlanarFace = findFace(pBox, New XYZ(0, 1, 0),
refOffsetH) ' added for L-shape

'' create alignments
''
_rvtDoc.FamilyCreate.NewAlignment(pViewPlan, refRight.GetReference(),
faceRight.Reference)
_rvtDoc.FamilyCreate.NewAlignment(pViewPlan, refLeft.GetReference(),
faceLeft.Reference)
_rvtDoc.FamilyCreate.NewAlignment(pViewPlan, refFront.GetReference(),
faceFront.Reference)
_rvtDoc.FamilyCreate.NewAlignment(pViewPlan, refBack.GetReference(),
faceBack.Reference)
_rvtDoc.FamilyCreate.NewAlignment(pViewPlan,
refOffsetV.GetReference(), faceOffsetV.Reference)
_rvtDoc.FamilyCreate.NewAlignment(pViewPlan,
refOffsetH.GetReference(), faceOffsetH.Reference)
End Sub
</VB.NET>

```

ここまでで、ヘルパー関数 findFace() には 2 つのバージョンが出来ました:

- `PlanarFace findFace(Extrusion pBox, XYZ normal)`
- `PlanarFace findFace(Extrusion pBox, XYZ normal, ReferencePlane refPlane)`

最初の関数は以前のものと同じです。2 つめの関数は、3 番目の引数として、位置合わせのために必要とする参照面を利用します。この関数には、与えられた法線で、面が与えられた参照面上にあるかチェックする機能が加えてあります。今回は、面を識別するために、2 つめのバージョンの関数を使用する必要があります。法線として「右」、「背面」、「OffsetH」、「OffsetV」の参照面に沿う面は、面を決定するには不十分です。完全なコードは、このドキュメントの終わりに記載されています(付録 A)。このクラスの終わりに コピー&ペースト してください。

面を見つけるルーチン以外は、ロジックの残りは同じままです。参照面「OffsetH」と「OffsetV」で追加の位置合わせを持つことができました。

4.2 ここで、コードをビルドし、実行して確認することができます。

5. パラメータを追加する

次に、2 つのパラメータ「Tw」および「Td」を追加する必要があります。その後で、L 字形プロファイルの厚み寸法として、これらのパラメータを関連づけます。

5.1 次の関数をクラスに加えてください:

```
<VB.NET>
'' =====
'' (3.1) add parameters
'' =====
Sub addParameters()

    '' parameter group for Dimension is PG_GEOMETRY in API
    ''
    Dim paramTw As FamilyParameter =
_rvtDoc.FamilyManager.AddParameter("Tw", BuiltInParameterGroup.PG_GEOMETRY,
ParameterType.Length, False)
    Dim paramTd As FamilyParameter =
_rvtDoc.FamilyManager.AddParameter("Td", BuiltInParameterGroup.PG_GEOMETRY,
ParameterType.Length, False)

    '' give initial values
    ''
    Dim tw As Double = mmToFeet(150.0) ' hard coded for simplicity
    Dim td As Double = mmToFeet(150.0)
    _rvtDoc.FamilyManager.Set(paramTw, tw)
    _rvtDoc.FamilyManager.Set(paramTd, td)

End Sub
</VB.NET>
```

パラメータを加えるために、私たちは、Family Manager クラスの addParameter() メソッドを使用します:

```
_rvtDoc.FamilyManager.AddParameter( "Tw",  
    BuiltInParameterGroup.PG_GEOMETRY, ParameterType.Length, False)
```

最初の引数はパラメータの名前で、第2引数は、タイプ ダイアログでパラメータがどこに表示されるかを決定するパラメータ グループです。今回のケースでは、PG_Geometry を指定し、「幅」と「奥行」と同じように、パラメータは「寸法」の下に表示するようにします。第3引数は、パラメータのタイプです。ここでは、「Tw」を「長さパラメータ」として設定しています。最後の引数は、パラメータがインスタンス・パラメータかファミリ・パラメータかを示すフラグです。

設定値は、ここまでの実習で用いたものと同じです:

```
_rvtDoc.FamilyManager.Set(paramTw, tw)
```

5.2 メインの コマンド関数から addParameters() を呼び出してください:

```
<VB.NET>  
Public Function Execute(ByVal commandData As ExternalCommandData, ByRef  
  
    ...  
  
    '' (2.1) add alignment  
    addAlignments(pSolid)  
  
    '' (3.1) add parameters  
    addParameters()  
  
    ...  
  
End Function  
</VB.NET>
```

5.3 この時点で、コードをビルドし、実行して確認することができます。

6. 寸法を追加する

下記のような定義で、参照面間に 2 つの寸法を追加して、各パラメータのラベルを作成します:

- 「左」と「OffsetV」の間の寸法 — パラメータ「Tw」
- 「正面」と「OffsetH」の間の寸法 — パラメータ「Td」

6.1 次の関数をコマンド クラスに追加します:

```
<VB.NET>
'' =====
'' (3.2) add dimensions
'' =====
Sub addDimensions()

    '' find the plan view
    ''
    Dim pViewPlan As View = findElement(GetType(ViewPlan), "下参照レベル")

    '' find reference planes
    ''
    Dim refLeft As ReferencePlane = findElement(GetType(ReferencePlane),
"左")
    Dim refFront As ReferencePlane = findElement(GetType(ReferencePlane),
"正面")
    Dim refOffsetV As ReferencePlane =
findElement(GetType(ReferencePlane), "OffsetV") ' added for L-shape
    Dim refOffsetH As ReferencePlane =
findElement(GetType(ReferencePlane), "OffsetH") ' added for L-shape

    ''
    '' (1) add dimension between the reference planes 'Left' and
'OffsetV', and label it as 'Tw
    ''

    '' define a dimension line
    ''
    Dim p0 As XYZ = refLeft.FreeEnd
    Dim p1 As XYZ = refOffsetV.FreeEnd
    Dim pLine As Line = Line.CreateBound(p0, p1)

    '' define references
    ''
    Dim pRefArray As New ReferenceArray
    pRefArray.Append(refLeft.GetReference())
    pRefArray.Append(refOffsetV.GetReference())

    '' create a dimension
    ''
    Dim pDimTw As Dimension = _rvtDoc.FamilyCreate.NewDimension(pViewPlan,
pLine, pRefArray)

    '' add label to the dimension
    ''
    Dim paramTw As FamilyParameter =
_rvtDoc.FamilyManager.Parameter("Tw")
    pDimTw.FamilyLabel = paramTw

    ''
    '' (2) do the same for dimension between 'Front' and 'OffsetH', and
lable it as 'Td
    ''
```

```

    ' define a dimension line
    '
    p0 = refFront.FreeEnd
    p1 = refOffsetH.FreeEnd
    pLine = Line.CreateBound(p0, p1)

    ' define references
    '
    pRefArray = New ReferenceArray
    pRefArray.Append(refFront.GetReference())
    pRefArray.Append(refOffsetH.GetReference())

    ' create a dimension
    '
    Dim pDimTd As Dimension = _rvtDoc.FamilyCreate.NewDimension(pViewPlan,
pLine, pRefArray)

    ' add label to the dimension
    '
    Dim paramTd As FamilyParameter =
_rvtDoc.FamilyManager.Parameter("Td")
    pDimTd.FamilyLabel = paramTd

End Sub
</VB.NET>

```

水平寸法と垂直寸法の2つの寸法を追加しています。ここでは、水平寸法について解説しますが、基本的に垂直寸法にも同じロジックを当てはめることができます。

寸法を作成する主なメソッドには次のメソッドを使用します:

```
_rvtDoc.FamilyCreate.NewDimension(pViewPlan, pLine, pRefArray)
```

最初の引数はビューです。今回の場合、平面図を指定します。第2引数は、寸法の初期位置です。ここで、2つの参照面から1つを使用します。3番目は、次のコードが示すような、「左」と「OffsetV」参照を含む、参照の配列です:

```

Dim pRefArray As New ReferenceArray
pRefArray.Append(refLeft.GetReference())
pRefArray.Append(refOffsetV.GetReference())

```

下記では、前のセクションで定義したパラメータ「Tw」でラベルを追加します:

```

Dim paramTw As FamilyParameter = _rvtDoc.FamilyManager.Parameter("Tw")
pDimTw.Label = paramTw

```

6.2 メインのコマンド関数から addDimensions() を呼び出してください:

```
<VB.NET>
Public Function Execute(ByVal commandData As ExternalCommandData, ByRef
    ...

    '' (3.1) add parameters
    addParameters()

    '' (3.2) add dimensions
    addDimensions()

    ...

End Function
</VB.NET>
```

6.3 ここで、コードをビルドし、実行して確認することができます。

7. addTypes() の更新

タイプを定義する際に考慮しなければならない、さらに 2 つのパラメータがあります。addType() 関数を更新していきましょう。今回、L 字形の厚みを定義する 2 つの追加引数 “Tw” と “Td” を用意します。「幅」、「深さ」、「Tw」および「Td」に対応する寸法名で、次の値を持つ 2 つのタイプを加えましょう:

- “600 x 900” - 600 x 900 x 150 x 225
- “1000 x 300” - 1000 x 300 x 250 x 75
- “600 x 600” - 600 x 600 x 150 x 150

7.1 下記の関数をクラスに加えてください:

```
<VB.NET>
'' add one type (version 2)
''
Sub addType(ByVal name As String, ByVal w As Double, ByVal d As Double,
ByVal tw As Double, ByVal td As Double)

    '' get the family manager from the current doc
    Dim pFamilyMgr As FamilyManager = _rvtDoc.FamilyManager

    '' add new types with the given name
    ''
    Dim type1 As FamilyType = pFamilyMgr.NewType(name)

    '' look for 'Width' and 'Depth' parameters and set them to the given
value
    ''
    Dim paramW As FamilyParameter = pFamilyMgr.Parameter("幅")
    Dim valW As Double = mmToFeet(w)
```

```

    If paramW IsNot Nothing Then
        pFamilyMgr.Set(paramW, valW)
    End If

    Dim paramD As FamilyParameter = pFamilyMgr.Parameter("奥行")
    Dim valD As Double = mmToFeet(d)
    If paramD IsNot Nothing Then
        pFamilyMgr.Set(paramD, valD)
    End If

    '' let's set 'Tw' and 'Td
    ''
    Dim paramTw As FamilyParameter = pFamilyMgr.Parameter("Tw")
    Dim valTw As Double = mmToFeet(tw)
    If paramTw IsNot Nothing Then
        pFamilyMgr.Set(paramTw, valTw)
    End If

    Dim paramTd As FamilyParameter = pFamilyMgr.Parameter("Td")
    Dim valTd As Double = mmToFeet(td)
    If paramTd IsNot Nothing Then
        pFamilyMgr.Set(paramTd, valTd)
    End If

End Sub
</VB.NET>

```

これら2つのパラメータの設定以外に、変更箇所はありません。同じロジックは「Tw」および「Td」の設定にも当てはまります。

7.2 下記でaddType()を更新してください:

```

<VB.NET>
'' =====
'' (3.3) add types
'' =====
Public Sub addTypes()

    '' addType(name, Width, Depth, Tw, Td)
    ''
    addType("600x900", 600.0, 900.0, 150, 225)
    addType("1000x300", 1000.0, 300.0, 250, 75)
    addType("600x600", 600.0, 600.0, 150, 150)

End Sub
</VB.NET>

```

作成した柱をテストする

ビルドしてテスト用に実行する準備が整いました。

テスト用に次のような行を Revit のアドイン マニフェスト ファイルに付け加えることができます (新しいコマンドを追加するか、Lab1から置き換えることができます)。もちろん、お使いの環境と一致するよう、必要な調節を行ってください。

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>

  <AddIn Type="Command">
    <Assembly>FamilyVb.dll</Assembly>
    <AddInId>FC5E150A-967B-4cc9-A7B0-3AA29C5DA9D9</AddInId>
    <FullClassName>FamilyVb.RvtCmd_FamilyCreateColumnLShape</FullClassName>
    <Text>Family Labs Create L-Shape Column</Text>
    <Description>Family API lab 2 to create L-shaped column</Description>
    <VisibilityMode>NotVisibleInProject</VisibilityMode>
    <AccessibilityClassName>Revit.Samples.SampleAccessibilityCheck </Accessib
ilityClassName>
    <VendorId>ADNP</VendorId>
    <VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
  </AddIn>

</RevitAddIns>
```

「柱(メートル単位).rft」テンプレートを使用して、ファミリ エディタを起動してください。

コマンドを実行すると、柱のプロファイルがL字形を示すことができます。柱をテストしてみてください:

- 平面図に2つの寸法が追加されたことを確認できますか?
- 寸法は正しくラベル付されていますか?
- ファミリ タイプ ダイアログで「寸法」の下の2つの追加パラメータ「Tw」と「Td」が表示されますか?
- 3つのタイプが正しく作成されていますか?
- 異なるタイプを選択して適用すると、それに沿って柱のサイズが変更されますか?
- プロジェクトに作成したファミリをしてください。柱は期待した「振る舞い」を持っていますか?

次の実習では、ここで作成した柱ファミリ上に、計算式とマテリアルを追加していきます。

付録 A. Lab2 で使われるヘルパー関数

Lab2 では、もう 1 つのヘルパー関数を追加しました。必要に応じて下記のコードをコピー&ペーストで作成中のコードに貼り付けてください。

- findFace () – バージョン 2 です。与えられた押し出しソリッドで、与えられた法線を持ち、与えられた参照面上に沿う平面を見つけます。

```
<VB.NET>
''' =====
''' helper function: given a solid, find a planar face with
''' the given normal (version 2)
''' this is a slightly enhanced version which checks if the
```



```

'' face is on the given reference plane.
'' =====
Function findFace( _
    ByVal pBox As Extrusion, _
    ByVal normal As XYZ, _
    ByVal refPlane As ReferencePlane _
) As PlanarFace

    '' get the geometry object of the given element
    ''
    Dim op As New Options
    op.ComputeReferences = True
    Dim geomObjs As GeometryElement = pBox.Geometry(op)

    '' loop through the array and find a face with the given normal
    ''
    For Each geomObj As GeometryObject In geomObjs

        If TypeOf geomObj Is Solid Then '' solid is what we are
interested in.

            Dim pSolid As Solid = geomObj
            Dim faces As FaceArray = pSolid.Faces

            For Each pFace As Face In faces
                Dim pPlanarFace As PlanarFace = pFace
                If Not (pPlanarFace Is Nothing) Then
                    '' check to see if they have same normal
                    If pPlanarFace.FaceNormal.IsAlmostEqualTo(normal)
Then
                        '' additionally, we want to check if the face is
on the reference plane
                        ''
                        Dim p0 As XYZ = refPlane.BubbleEnd
                        Dim p1 As XYZ = refPlane.FreeEnd
                        Dim pCurve As Line = Line.CreateBound(p0, p1)
                        Dim res As SetComparisonResult =
pPlanarFace.Intersect(pCurve)
                        If res = SetComparisonResult.Subset Then
                            Return (pPlanarFace) '' we found the face
                        End If
                    End If
                End If
            Next

        ElseIf TypeOf geomObj Is GeometryInstance Then

            '' will come back later as needed.

        ElseIf TypeOf geomObj Is Curve Then

            '' will come back later as needed.

        ElseIf TypeOf geomObj Is Mesh Then

```

```
        '' will come back later as needed.  
  
    Else  
        '' what else do we have?  
  
    End If  
Next  
  
    '' if we come here, we did not find any.  
Return Nothing  
  
End Function  
</VB.NET>
```

Autodesk Developer Network