**Revit Family API Labs**

**Lab4 – Add Visibility Control**

March 2010 by M. Harada

Last updated, Date : March 19, 2014

C# version

**Objective:** In this lab, we will learn how to:

* add visibility control

**Tasks:** We’ll extend the command we have defined in the previous labs and add visibility control. We’ll add a line presentation of the column for the coarse view.

1. Take the command class which we have defined in the Lab3. This will be the starting point of this lab. We’ll continue using the Family Editor and "Metric Column.rft" template.
2. Define line representations for a column:
   * Single lined symbolic-line representation in a plan view
   * Single lined model-line representation in a model view
3. Set the visibility control so that the single line representation is used in coarse view.

Figure 1 shows the image of the graphical representation in Medium/Fine and Coarse detail level.



Figure1 Representation of a column: a column is represented with   
lines in Coarse detail level and a solid in Medium and Fine.

The following is the breakdown of step by step instructions in this lab:

1. [Define Another External Command](#defineAnotherExternalCommand)
2. [Add Line Objects for Coarse Level](#addLineObjectsForCoarseLevel)
3. [Change the Visibility of the Solid](#chanegVisibilityOfSolid)
4. [Test Your Column](#testYourColumn)
5. **Define Another External Command**

We’ll be extending the command we have defined in Lab3. You can either copy it to define a new class or continue extending the existing one on top of it. (Just make sure to back up in case you need it to start again.)

* 1. Copy the command class from Lab3 and define a new one to work on in Lab3. Let’s name them as:
* File name: 4**\_ColumnVisibility.cs (or .vb)**
* Command class name: **RvtCmd\_FamilyCreateColumnVisibility**

(Once again, you may choose to use any names you want here. When you do so, just remember what you are calling your own project, and substitute these names as needed while reading the instruction in this document.)

[Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Automatic)] [Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.Automatic)]  
 class RvtCmd\_FamilyCreateColumnVisibility : IExternalCommand

{

...

}

**2.** **Add Line Objects for Coarse Level**

We are going to add simple line representations for a column. We then set their visibilities to coarse level :

* two symbolic lines in a L-shape. They will be visible in the top view only.
* a vertical model line. This will be visible for a 3D view.

2.1 Add the following function to the class:

// =====================================================================

// (5.1.1) create simple line objects to be displayed in coarse level

// =====================================================================

public void addLineObjects()

{

//

// define a simple L-shape detail line object

//

// 0

// + h = height

// | (we also want to draw a vertical line here at point 1)

// d |

// +-----+

// 1 2

// w

//

// sizes

double w = mmToFeet( 600.0 ); // modified to match reference plane. otherwise, alignment won't work.

double d = mmToFeet( 600.0 );

double h = mmToFeet( 4000.0 ); // distance between Lower and Upper Ref Level.

double t = mmToFeet( 50.0 ); // slight offset for visbility

// define vertices

//

XYZ[] pts = new XYZ[] { new XYZ( ( -w / 2.0 ) + t, d / 2.0, 0.0 ), new XYZ( ( -w / 2.0 ) + t, ( -d / 2.0 ) + t, 0.0 ), new XYZ( w / 2.0, ( -d / 2.0 ) + t, 0.0 ) };

XYZ ptH = new XYZ( ( -w / 2.0 ) + t, ( -d / 2.0 ) + t, h ); // this is for vertical line.

//

// (2) create a sketch plane

//

// we need to know the template. If you look at the template (Metric Column.rft) and "Front" view,

// you will see "Reference Plane" at "Lower Ref. Level". We are going to create lines there.

// findElement() is a helper function that find an element of the given type and name. see below.

// Note: we did the same in creating a profile.

//

ReferencePlane pRefPlane = findElement( typeof( ReferencePlane ), "Reference Plane" ) as ReferencePlane;

SketchPlane pSketchPlane = SketchPlane.Create(\_rvtDoc,pRefPlane.Plane);

// for vertical line, we draw a straight vertical line at the point[1]

XYZ normal = new XYZ(1.0, 0.0, 0.0);

Plane pGeomPlaneH = \_rvtApp.Create.NewPlane(normal, pts[1]);

SketchPlane pSketchPlaneH = SketchPlane.Create(\_rvtDoc,pGeomPlaneH);

// (3) create line objects: two symbolic curves on a plan and one model curve representing a column like a vertical stick.

//

Line geomLine1 = Line.CreateBound ( pts[0], pts[1);

Line geomLine2 = Line.CreateBound ( pts[1], pts[2);

Line geomLineH = Line.CreateBound ( pts[1], ptH);

SymbolicCurve pLine1 = \_rvtDoc.FamilyCreate.NewSymbolicCurve( geomLine1, pSketchPlane );

SymbolicCurve pLine2 = \_rvtDoc.FamilyCreate.NewSymbolicCurve( geomLine2, pSketchPlane );

ModelCurve pLineH = \_rvtDoc.FamilyCreate.NewModelCurve( geomLineH, pSketchPlaneH ); // this is vertical line

// set the visibilities of two lines to coarse only

//

FamilyElementVisibility pVis = new FamilyElementVisibility( FamilyElementVisibilityType.ViewSpecific );

pVis.IsShownInFine = false;

pVis.IsShownInMedium = false;

pLine1.SetVisibility( pVis );

pLine2.SetVisibility( pVis );

FamilyElementVisibility pVisH = new FamilyElementVisibility( FamilyElementVisibilityType.Model );

pVisH.IsShownInFine = false;

pVisH.IsShownInMedium = false;

pLineH.SetVisibility( pVisH );

}

We start by defining the initial value of vertices used to draw lines. As always, we are hard-coding them for simplicity.

You can use Document.FamilyCreate to draw symbolic or model lines:

* \_rvtDoc.FamilyCreate.NewSymbolicCurve(geomLine1, pSketchPlane)
* \_rvtDoc.FamilyCreate.NewModelCurve(geomLineH, pSketchPlaneH)

Here is our key piece to set the visibility:

FamilyElementVisibility pVis = new FamilyElementVisibility( FamilyElementVisibilityType.ViewSpecific );

pVis.IsShownInFine = false;

pVis.IsShownInMedium = false;

pLine1.SetVisibility( pVis );  
  
FamilyElementVisibility is a helper class that keeps track of the information about visibility of a specific element. It takes EamilyElementVisitlityType in the constructor, which are a type of either ViewSpecific or Model:

* ViewSpecific - The element will be displayed only in the view where it is created. This applies to items like detail components, annotations, and view-specific imports.
* Model - The element will be displayed in 3D views and model other model views.

By default, visibilities are all set as True. You can turn off by setting them False for each viewing conditions. We then set it to each element.

2.2 Call addLineObjects() function from your main command function Execute():

...

// (4.2) add materials

addMaterials(pSolid);

// (5.1) add visibilities

addLineObjects();

...

2.3 Your code should build and run.

**3.** **Change the Visibility of the Solid**

One last thing we need to do is turn off the visibility of the extrusion we have for L-shape solid in a coarse view.

3.1 Add the follow function to the class:

// =================================================================

// (5.1.2) set the visibility of the solid not to show in coarse

// =================================================================

public void changeVisibility( Extrusion pSolid )

{

// set the visibility of the model not to shown in coarse.

//

FamilyElementVisibility pVis = new FamilyElementVisibility( FamilyElementVisibilityType.Model );

pVis.IsShownInCoarse = false;

pSolid.SetVisibility( pVis );

}

The exact same idea as we have done for line objects earlier. Create a FamilyElementVisibility class, this time, using FamilyElementVisibilityType.Model. Set its IsShownInCoarse property as False, and set it to the solid.

**4.** **Test Your Column**

Your code is ready to build and run for testing.

You can add lines like the following to your Revit .addin manifest file to test this. (You can either add a new command or replace with one from Lab 3). Make necessary adjustment to match with your environment, of course.

<?xml version="1.0" encoding="utf-16" standalone="no"?>

<RevitAddIns>

<AddIn Type="Command">

<Assembly>C:\Revit SDK 2013\Family Labs\FamilyLabsCS\bin\Debug\FamilyLabsCS.dll</Assembly>

<AddInId>943A6BDC-2D64-4033-A3B0-E18BF3598006</AddInId>

<FullClassName>FamilyLabsCS.RvtCmd\_FamilyCreateColumnVisibility</FullClassName>

<Text>Family API 4 CS - Define Visibility</Text>

<Description>Family API lab 4 to create L-shaped column with visibility settings</Description>

<VisibilityMode>NotVisibleInProject</VisibilityMode>

<AccessibilityClassName>Revit.Samples.SampleAccessibilityCheck </AccessibilityClassName>

<VendorId>ADNP</VendorId>

<VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>

</AddIn>

</RevitAddIns>

Remember to start with Family Editor and use "Metric Column.rft" template.

After running a command, examine the following:

* Do you see the symbolic line objects in the plan view? A model line in the 3D view?
* Switch visibility levels. You should see some lines grayed out in a certain view.
* Do line change when you change types? Or a level or height?
* Start a new project and load your family to it. Place your columns. Try changing view detail levels. Compare various viewing level and directions.

Congratulations!! You have completed your Family API Labs.

Autodesk Developer Network