**Revit API UI Labs**

**Lab 2 – Selection**

May, 2011 by A. Nagy

Last updated, Date : March 10, 2015

**<VB.NET>**VB.NET Version**</VB.NET>**

**Objective:** In this lab, we will learn how to use the various selection functions. We’ll learn how to:

* Select an object/objects/point
* Select element geometry
* Restrict the selection

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

1. Create a new External Command
2. Show current selection
3. Pick an object
4. Pick multiple objects
5. Pick point
6. Pick face/edge
7. Selection filter
8. Create house interactively
9. Summary
10. **Create a new External Command**

We’ll add another external command to the current project.

* 1. Add a new file and define another external command to your project. Let’s name them as follows:
* File name: **2\_Selection.vb (or .cs)**
* Command class name: **UISelection**

(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 following the instruction in this document.)

**Required Namespaces:**

Namespaces needed for this lab are:

* Autodesk.Revit.DB
* Autodesk.Revit.UI
* Autodesk.Revit.ApplicationServices
* Autodesk.Revit.Attributes
* Autodesk.Revit.UI.Selection (this is for selection)

Note (VB.NET only): if you are writing in VB.NET and you import namespaces at the project level, (i.e., in the project properties, there is no need to explicitly import in each file.

Let’s declare some variables in the class that will reference the UIApplication and the active UIDocument

**<VB.NET>**  
Public Class UISelection

Implements IExternalCommand

' Member variables

Dim \_uiApp As UIApplication

Dim \_uiDoc As UIDocument

Public Function Execute(ByVal commandData As ExternalCommandData, \_

ByRef message As String, \_

ByVal elements As ElementSet) \_

As Result \_

Implements IExternalCommand.Execute

' Get the access to the top most objects.

‘ (we may not use them all in this specific lab.)

\_uiApp = commandData.Application

\_uiDoc = \_uiApp.ActiveUIDocument

' …

**</VB.NET>**

1. **Show current selection**

In order to show information about the current selection let’s create some helper functions. These will also be used by the other functions as well.

**<VB.NET>**  
 Public Sub ShowElementList(elemIds As IEnumerable, header As String)

Dim s As String = vbLf & vbLf & " - Class - Category - Name (or Family: Type Name) - Id - " & vbCrLf

Dim count As Integer = 0

For Each eId As ElementId In elemIds

count += 1

Dim e As Element = Me.\_uiDoc.Document.GetElement(eId)

s += Me.ElementToString(e)

Next

s = header + "(" + count.ToString() + ")" + s

TaskDialog.Show("Revit UI Lab", s)

End Sub

Function ElementToString(ByVal e As Element) As String

If e Is Nothing Then

Return "none"

End If

Dim name As String = ""

If TypeOf e Is ElementType Then

Dim param As Parameter = \_

e.Parameter(BuiltInParameter.SYMBOL\_FAMILY\_AND\_TYPE\_NAMES\_PARAM)

If param IsNot Nothing Then

name = param.AsString

End If

Else

name = e.Name

End If

Return e.GetType.Name + "; " + e.Category.Name + "; " \_

+ name + "; " + e.Id.IntegerValue.ToString + vbCr

End Function

Public Shared Function PointToString(ByVal pt As XYZ) As String

If pt Is Nothing Then

Return ""

End If

Return "(" + pt.X.ToString("F2") + ", " + pt.Y.ToString("F2") + ", " \_

+ pt.Z.ToString("F2") + ")"

End Function

**</VB.NET>**

We can get back the list of currently selected elements from UIDocument.Selection

**<VB.NET>**   
 Dim selSet As ICollection(Of ElementId) =

\_uiDoc.Selection.GetElementIds()

ShowElementList(selSet, "Pre-selection: ")

**</VB.NET>**

1. **Pick an object**



All the various pick functions can be found under UIDocument.Selection. First let’s have a look at PickObject() which prompts the user to pick a single object. This function has many overrides. You can specify the type of object you allow to be selected, in our case it will be Element this time, and provide the prompt for the user. You could also specify a selection filter class – see later.

**<VB.NET>**   
 Sub PickMethod\_PickObject()

Dim r As Reference = \_uiDoc.Selection.PickObject(ObjectType.Element, \_

"Select one element")

Dim e As Element = \_uiDoc.Document.GetElement(r)

ShowBasicElementInfo(e)'Defined in DBElelemt Lab in the Intro Labs

End Sub

**</VB.NET>**

1. **Pick multiple objects**

If you want to prompt the user to select multiple objects then call PickObjects(), which will return a list of Reference’s. In order to use our existing utility function to show information about the selected elements, we need to turn the list of references into a list of elements.

**<VB.NET>**   
 Sub PickMethod\_PickObjects()

Dim refs As IList(Of Reference) = \_

\_uiDoc.Selection.PickObjects(ObjectType.Element, \_

"Select multiple elemens")

' put it in a List form.

Dim elems As IList(Of Element) = New List(Of Element)

For Each r As Reference In refs

elems.Add(\_uiDoc.Document.GetElement(r))

Next

ShowElementList(elems, "Pick Objects: ")

End Sub

**</VB.NET>**

You could also ask the user to select elements that fall inside a rectangle

**<VB.NET>**

Sub PickMethod\_PickElementByRectangle()

' Note: PickElementByRectangle returns the list of element. not reference.

Dim elems As IList(Of Element) = \_

\_uiDoc.Selection.PickElementsByRectangle("Select by rectangle")

' Show it

ShowElementList(elems, "Pick By Rectangle: ")

End Sub

**</VB.NET>**

1. **Pick point**

If you simply want to pick a point in space then you can use PickPoint() for that

**<VB.NET>**

Sub PickMethod\_PickPoint()

Dim pt As XYZ = \_uiDoc.Selection.PickPoint("Pick a point")

Dim msg As String = "Pick Point: "

msg += PointToString(pt)

TaskDialog.Show("PickPoint", msg)

End Sub

**</VB.NET>**

However, if you want to pick a point on an element, then you need to call pick object with ObjectType.PointOnElement

**<VB.NET>**

Sub PickPointOnElement()

Dim r As Reference = \_

\_uiDoc.Selection.PickObject( \_

ObjectType.PointOnElement, \_

"Select a point on element")

Dim e As Element = \_uiDoc.Document.GetElement(r)

Dim pt As XYZ = r.GlobalPoint

Dim msg As String

If pt Is Nothing Then

msg = "No point picked."

Else

msg = "You picked the point " + PointToString(pt) \_

+ " on element " + e.Id.ToString

End If

TaskDialog.Show("PickPointOnElement", msg)

End Sub

**</VB.NET>**

1. **Pick face/edge**

We can also use the PickObject() function to pick a face or edge of an element, we just need to pass in ObjectType.Face or ObjectType.Edge respectively. Once we got back the reference we can use Element. GetGeometryObjectFromReference() to get back the Face or Edge of the element that was selected.

**<VB.NET>**

Sub PickFace()

Dim r As Reference = \_

\_uiDoc.Selection.PickObject(ObjectType.Face, "Select a face")

Dim e As Element = \_uiDoc.Document.GetElement(r)

Dim oFace As Face = e.GetGeometryObjectFromReference(r)

' show a message to the user.

Dim msg As String = ""

If oFace IsNot Nothing Then

msg = "You picked the face of element " + e.Id.ToString + vbCr

Else

msg = "no Face picked" + vbCr

End If

TaskDialog.Show("PickFace", msg)

End Sub

**</VB.NET>**

Try implementing a similar function to select edges.

1. **Selection filter**

When selecting objects using PickObject()/PickObjects() then we can also specify an instance of a class that implements ISelectionFilter to filter the selection in the way we want. As an example let’s filter the selection so that only walls can be selected by the user.

First we create the class that implements the ISelectionFilter interface. This interface has two methods that can be implemented: AllowReference() and AllowElement(). If you pass ObjectType.Element to PickObject() then only the AllowElement() will be called. If you pass ObjectType.Face/Edge/PointOnElement then AllowReference() will be called.

**<VB.NET>**

Class SelectionFilterWall

Implements ISelectionFilter

Public Function AllowElement(ByVal e As Element) \_

As Boolean Implements ISelectionFilter.AllowElement

If e.Category Is Nothing Then Return False

If e.Category.Id.IntegerValue.Equals(BuiltInCategory.OST\_Walls) Then \_

Return True

Return False

End Function

Public Function AllowReference( \_

ByVal reference As Reference, \_

ByVal position As XYZ) \_

As Boolean Implements ISelectionFilter.AllowReference

Return True

End Function

End Class

**</VB.NET>**

Now we can use this to filter the selection:

**<VB.NET>**

Sub PickWall()

Dim selFilterWall As New SelectionFilterWall

Dim r As Reference = \_

\_uiDoc.Selection.PickObject(ObjectType.Element, selFilterWall, \_

"Select a wall")

' Show it

Dim e As Element = \_uiDoc.Document.GetElement(r)

ShowBasicElementInfo(e)'Defined in DBElelemt Lab in the Intro Labs

End Sub

**</VB.NET>**

Try filtering the selection to PlanarFace when selecting a Face object.

1. **Create house interactively**

The Intro Labs contains a part which exposes house creation functions. Open the Intro Labs, compile it, then add a reference to the created Add-In dll from the current project.

Under the ModelCreationExport class of the referenced Intro Labs dll you’ll find the various functions that will help you create the house interactively. The functions are:

* CreateWalls()
* AddDoor()
* AddWindow()
* AddRoof()

Create a program that does the following:

* Add a new external command to the 2\_Selection.vb (or cs) file
* Inside the command prompt the user to pick the two corner points of the house, and create the 4 walls based on those points
* Prompt the user to select one of the walls, then add a door to it, and then add a window to each of the other 3 walls
* Create a roof for the house

1. **Summary**

In this lab, we learned how to use the various selection functions. We’ve learned how to:

* Select an object/objects/point
* Select element geometry
* Restrict the selection

Autodesk Developer Network