Revit API UI Labs

Lab 2 – Selection

May, 2011 by A. Nagy

Last updated Date : March 03, 2018

<C#>C# Version**</C#>**

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

1. Create a new External Command

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

1.1 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

```
<C#>
  public class UISelection : IExternalCommand
  {
    UIApplication _uiApp;
    UIDocument _uiDoc;

    public Result Execute(
      ExternalCommandData commandData,
      ref string message,
      ElementSet elements )
    {
      _uiApp = commandData.Application;
      _uiDoc = _uiApp.ActiveUIDocument;

      // …
</C#>
```

2. Show current selection

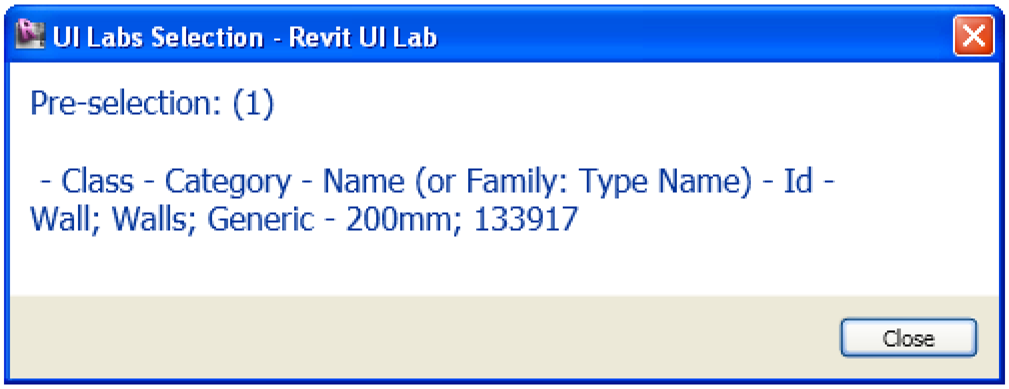
In order to show information about the current selection let's create some helper functions. These will also be used by the other functions as well.

```
<C#>
    public void ShowElementList(IEnumerable elemIds, string header)
    {
      string s = "\n\n - Class - Category - Name (or Family: Type Name) - Id
- " + "\r\n";

      int count = 0;
      foreach (ElementId eId in elemIds)
      {
        count++;
        Element e = _uiDoc.Document.GetElement(eId);
        s += ElementToString(e);
      }

      s = header + "(" + count + ")" + s;

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

    public string ElementToString(Element e)
    {
      if( e == null )
      {
        return "none";
      }

      string name = "";

      if( e is ElementType )
      {
        Parameter param = e.get_Parameter(
          BuiltInParameter.SYMBOL_FAMILY_AND_TYPE_NAMES_PARAM );
        if( param != null )
        {
          name = param.AsString();
        }
      }
      else
      {
        name = e.Name;
      }


      return e.GetType().Name + "; " + e.Category.Name + "; " + name + "; "
        + e.Id.IntegerValue.ToString() + "\r\n";
    }


    public static string PointToString(XYZ pt)
    {
      if( pt == null )
      {
        return "";
      }

      return "(" + pt.X.ToString( "F2" ) + ", " + pt.Y.ToString( "F2" ) + ",
        " + pt.Z.ToString( "F2" ) + ")";
    }
</C#>
```

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

```
<C#>
  ICollection<ElementId> selectedElementIds = _uiDoc.Selection.GetElementIds();
  ShowElementList(selectedElementIds, "Pre-selection: " );
</C#>
```

3. Pick an object


UI Labs Selection - Revit UI Lab

Pre-selection: (1)

 - Class - Category - Name (or Family: Type Name) - Id -
Wall; Walls; Generic - 200mm; 133917

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.

<C#>
```
    public void PickMethod_PickObject()
    {
      Reference r = _uiDoc.Selection.PickObject(
        ObjectType.Element, "Select one element" );

      Element e = _uiDoc.Document.GetElement( r );

      ShowBasicElementInfo( e );// Defined in DBElelemt Lab in the Intro Labs
    }
```
</C#>

4. 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.

<C#>
```
    public void PickMethod_PickObjects()
    {
      IList<Reference> refs = _uiDoc.Selection.PickObjects(
        ObjectType.Element, "Select multiple elemens" );

      // Put it in a List form.

      IList<ElementId> elemIds = new List<ElementId>();
      foreach( Reference r in refs )
      {
        elemIds.Add(r.ElementId);
      }

      ShowElementList( elemIds, "Pick Objects: " );
    }
```
</C#>

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

```
<C#>
    public void PickMethod_PickElementByRectangle()
    {
      // Note: PickElementByRectangle returns the list of element.
      // not reference.
      IList<Element> elems = _uiDoc.Selection.PickElementsByRectangle(
        "Select by rectangle" );
      IList<ElementId> eids=new List<ElementId>();
      foreach( Element e in elems )
      {
        eids.Add(e.Id);
      }

      // Show it.

      ShowElementList( eids, "Pick By Rectangle: " );
    }
</C#>
```

5. Pick point

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

```
<C#>
    public void PickMethod_PickPoint()
    {
      XYZ pt = _uiDoc.Selection.PickPoint( "Pick a point" );

      // Show it.

      string msg = "Pick Point: ";
      msg += PointToString( pt );

      TaskDialog.Show( "PickPoint", msg );
    }
</C#>
```

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

```
<C#>
    public void PickPointOnElement()
    {
      Reference r = _uiDoc.Selection.PickObject(
        ObjectType.PointOnElement,
        "Select a point on element" );

      Element e = _uiDoc.Document.GetElement( r );
      XYZ pt = r.GlobalPoint;
      string msg = "";
      if( pt != null )
      {
        msg = "You picked the point " + PointToString( pt )
          + " on an element " + e.Id.ToString() + "\r\n";
      }
      else
      {
        msg = "no Point picked \n";
      }

      TaskDialog.Show( "PickPointOnElement", msg );
    }
</C#>
```

6. 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.

```
<C#>
    public void PickFace()
    {
      Reference r = _uiDoc.Selection.PickObject( ObjectType.Face,
        "Select a face" );
      Element e = _uiDoc.Document.GetElement( r );

      Face oFace = e.GetGeometryObjectFromReference( r ) as Face;

      string msg = "";
      if( oFace != null )
      {
        msg = "You picked the face of element " + e.Id.ToString() + "\r\n";
      }
      else
      {
        msg = "no Face picked \n";
      }

      TaskDialog.Show( "PickFace", msg );
    }
</C#>
```

Try implementing a similar function to select edges.

7. 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.

```
<C#>
  class SelectionFilterWall : ISelectionFilter
  {
    public bool AllowElement( Element e )
    {
      return e is Wall;
    }

    public bool AllowReference( Reference reference, XYZ position )
    {
      return true;
    }
  }
</C#>
```

Now we can use this to filter the selection:

```
<C#>
    public void PickWall()
    {
      SelectionFilterWall selFilterWall = new SelectionFilterWall();
      Reference r = _uiDoc.Selection.PickObject( ObjectType.Element,
        selFilterWall, "Select a wall" );

      // Show it
      Element e = _uiDoc.Document.GetElement( r );

      ShowBasicElementInfo( e );// Defined in DBElelemt Lab in the Intro Labs

    }
</C#>
```

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

8. 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

9. 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