Automation Prototyping: Dynamo & the Revit API

Nuri Miller

Technical Manager at Gehry Technologies
@nurimiller

Class summary

The class will provide a walk-through of a sample project that is automated in the Dynamo visual programming language and also using the Revit software API . This comparison will help you to draw your own conclusions about how Dynamo and/or the Revit API fit into your current practices.

Key learning objectives

By the end of this class, you will:

- Discover how Dynamo’s graphical programming interface compares to the Revit API.
- Understand Revit’s API functionality in relation to family placement and manipulation.
- Learn how the mix of both Dynamo and the Revit API may be most beneficial to your current project and practice.
- Recognize Dynamo as a way for prototyping automation solutions and for sharing existing ones by dispersing their logic for other’s experimentation.

Nuri Miller

Technical Manager @ Gehry Technologies

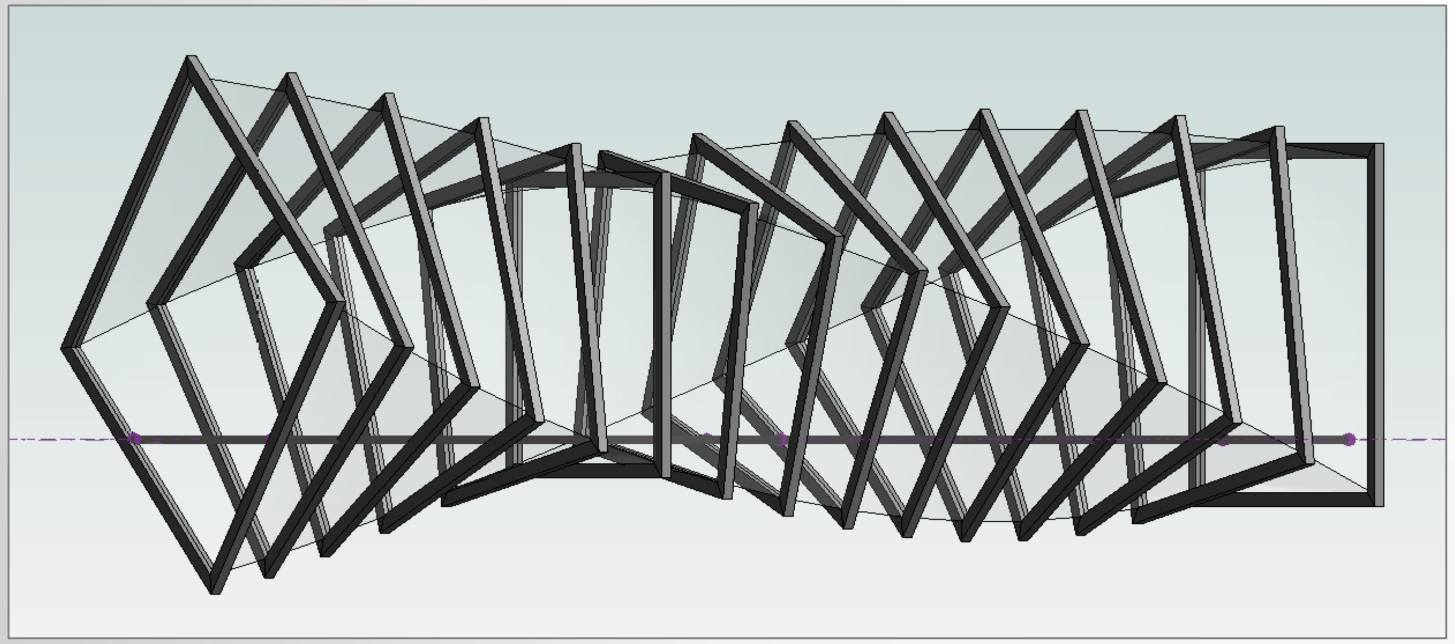
@nurimiller

I help designers, builders and owners save time and become better informed by integrating their VDC technology ecosystem through custom software solutions.

I believe wholeheartedly that AECO can be elevated through technology that takes on the burden of repetitive tasks, bridges the divide between technology platforms, and supports data exploration in building projects.

Floral Street Bridge Example

The example for this tutorial was originally developed as an introduction to parametric modeling in Digital Project. It's loosely based on a footbridge designed by Wilkinson Eyre Architects in London. The project's twisting form provides a useful vehicle for introducing parametric relationships and how a constraint solver deals with them.

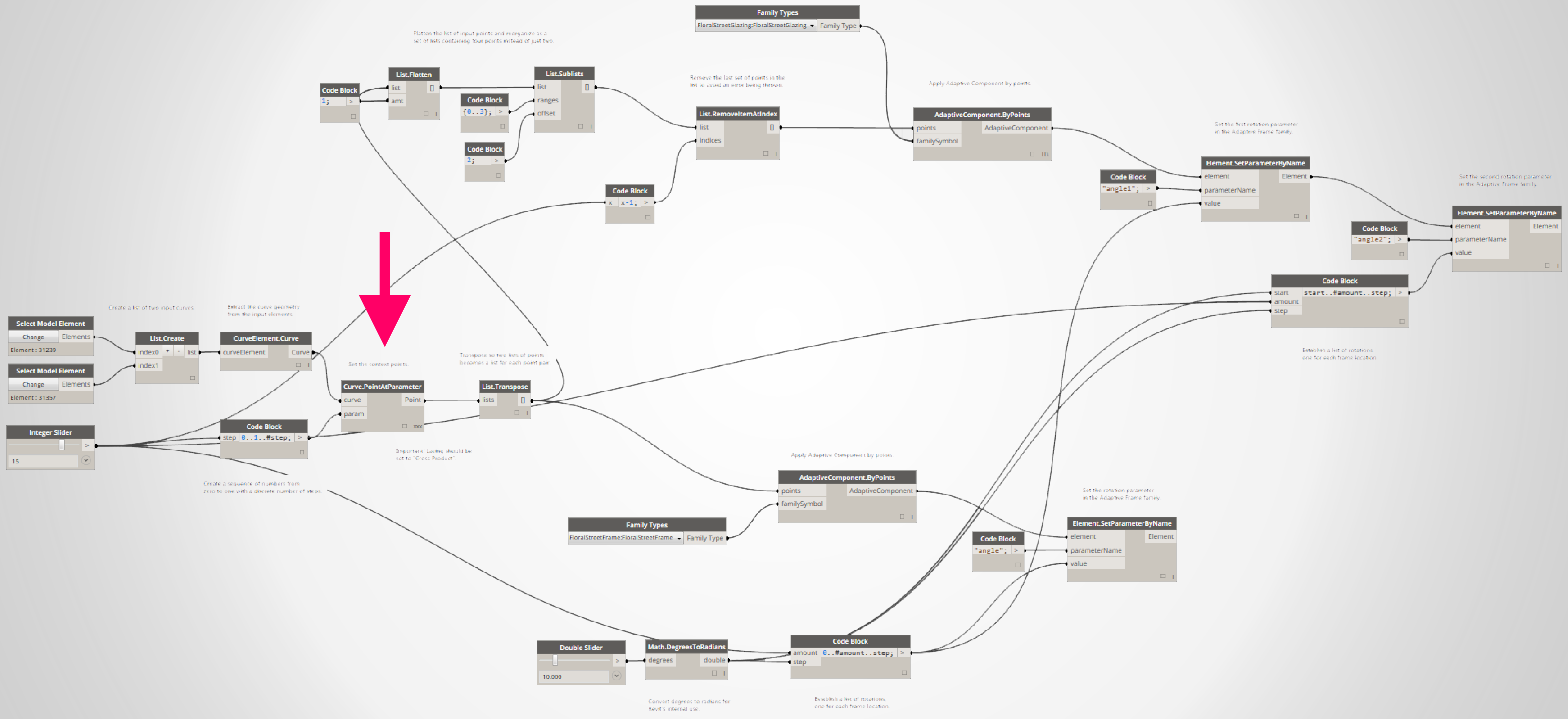


Automated Instantiation With Dynamo

A Few Helpful Hints for Dynamo

- The search tool is key.
- Watch your output as you build.
- Double click to create code.
- Sweep everything under the rug with custom nodes.

The Dynamo Definition – An Overview



Create a list of two input curves.

Extract the curve geometry from the input elements.

Set the context points.

Transpose so two lists of points becomes a list for each point pair.

Important! Lacing should be set to "Cross Product".

Create a sequence of numbers from zero to one with a discrete number of steps.

Flatten the list of input points and reorganize as a set of lists containing four points instead of just two.

Remove the last set of points in the list to avoid an error being thrown.

Apply Adaptive Component by points.

Set the first rotation parameter in the Adaptive Frame family.

Set the second rotation parameter in the Adaptive Frame family.

Establish a list of rotations, one for each frame location.

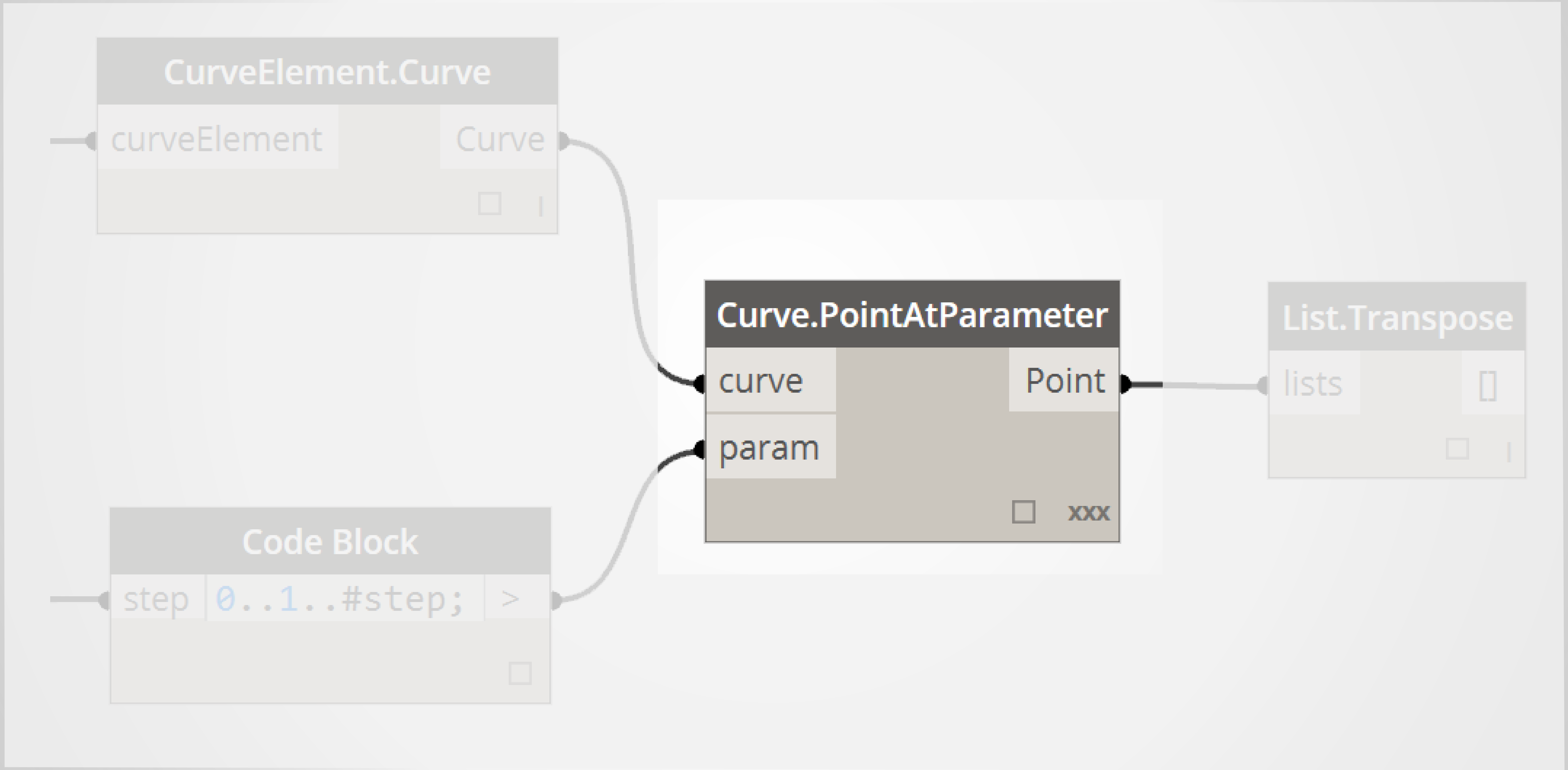
Apply Adaptive Component by points.

Set the rotation parameter in the Adaptive Frame family.

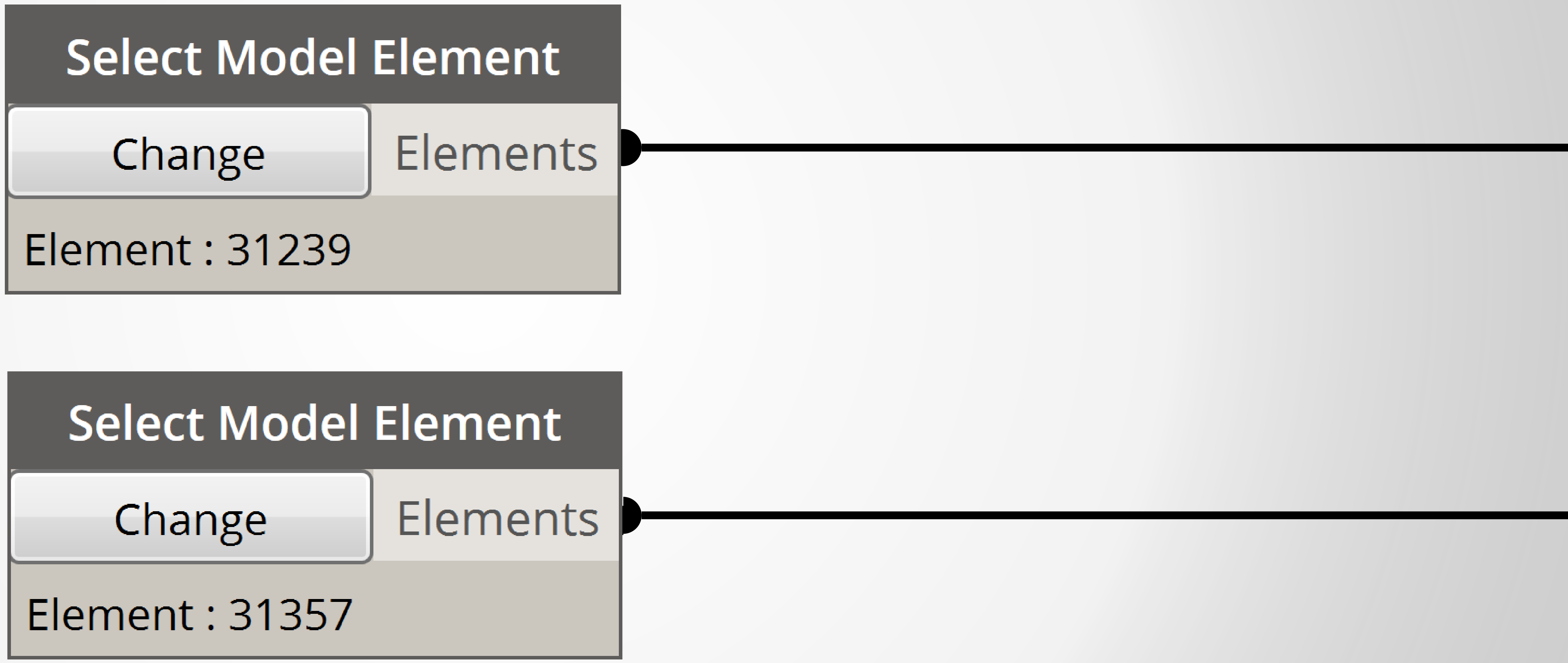
Convert degrees to radians for Revit's internal use.

Establish a list of rotations, one for each frame location.

Creating Context Points



Creating Context Points



Select Model Element

Change Elements

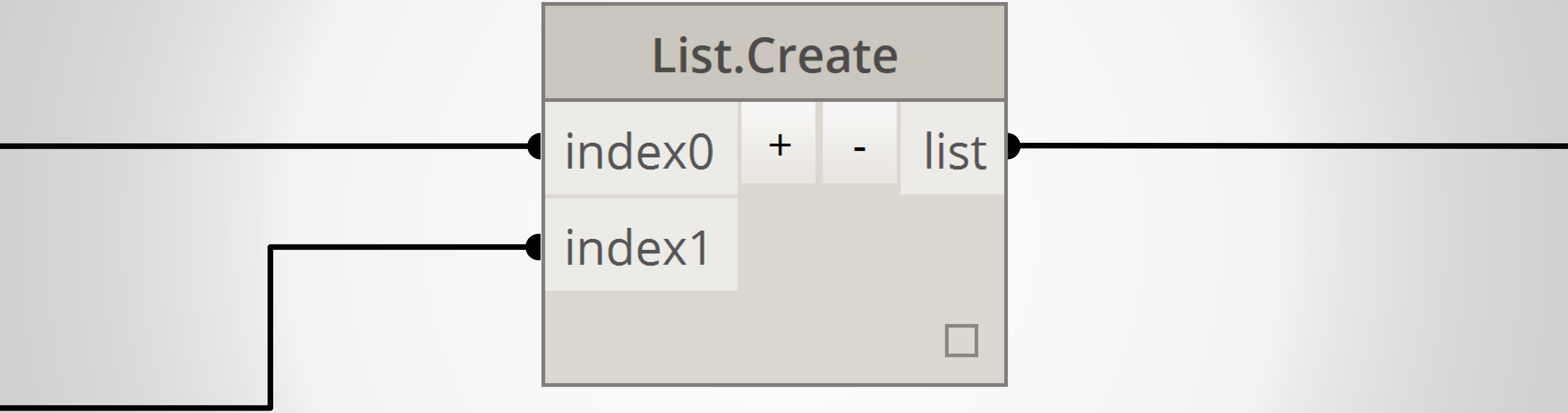
Element : 31239

Select Model Element

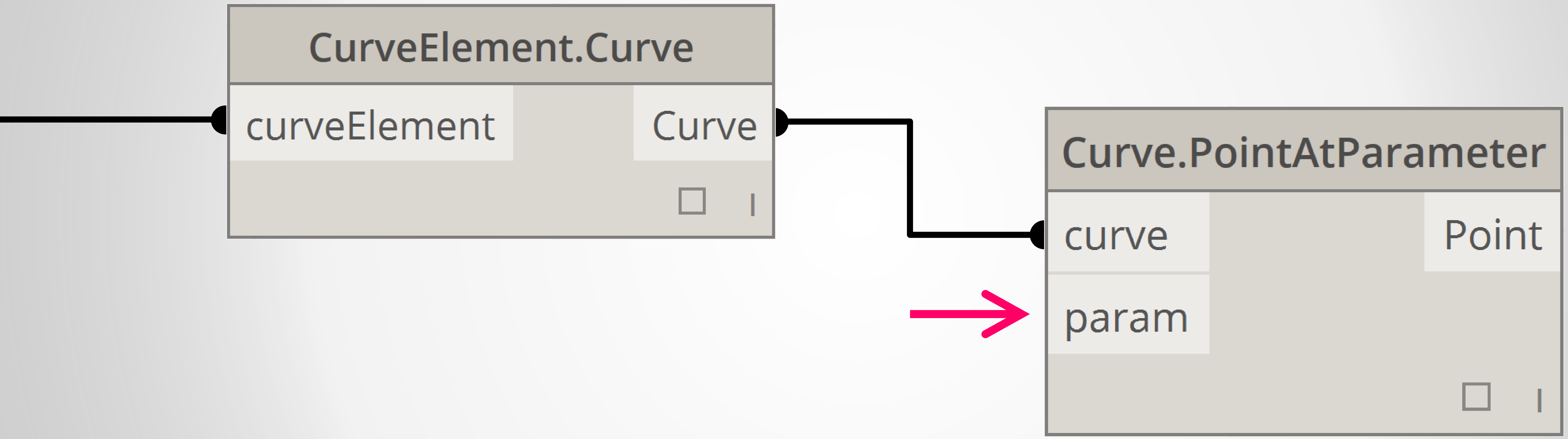
Change Elements

Element : 31357

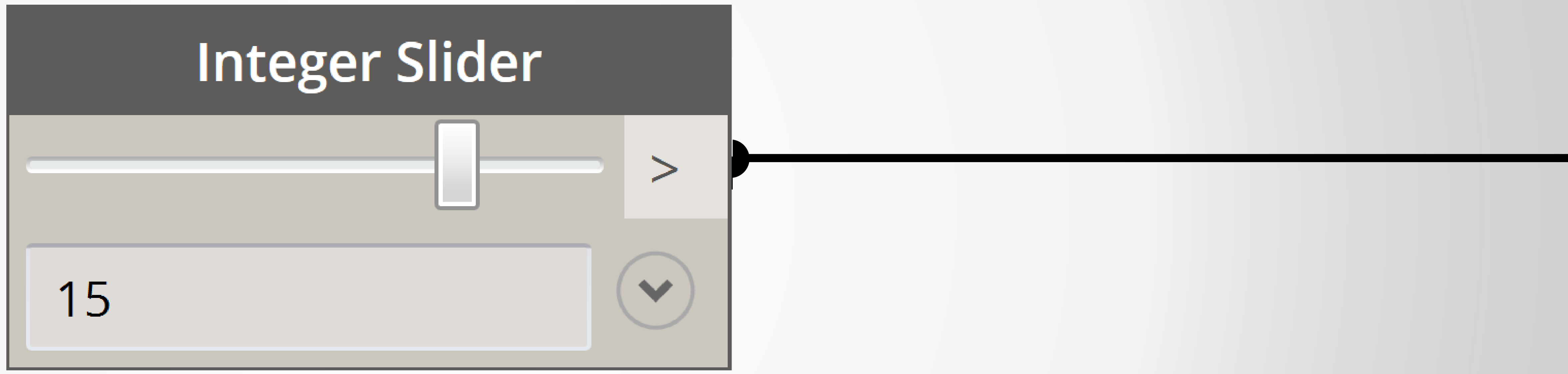
Creating Context Points



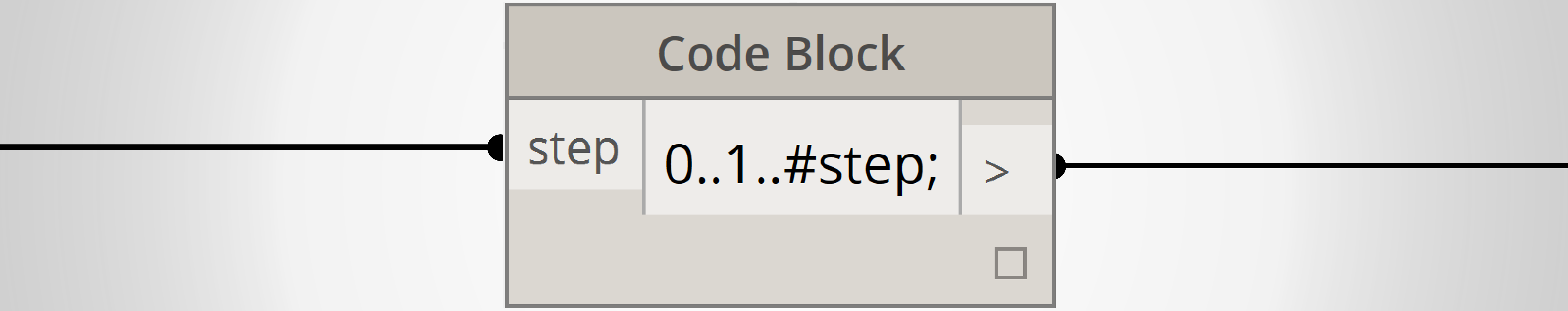
Creating Context Points



Creating Context Points

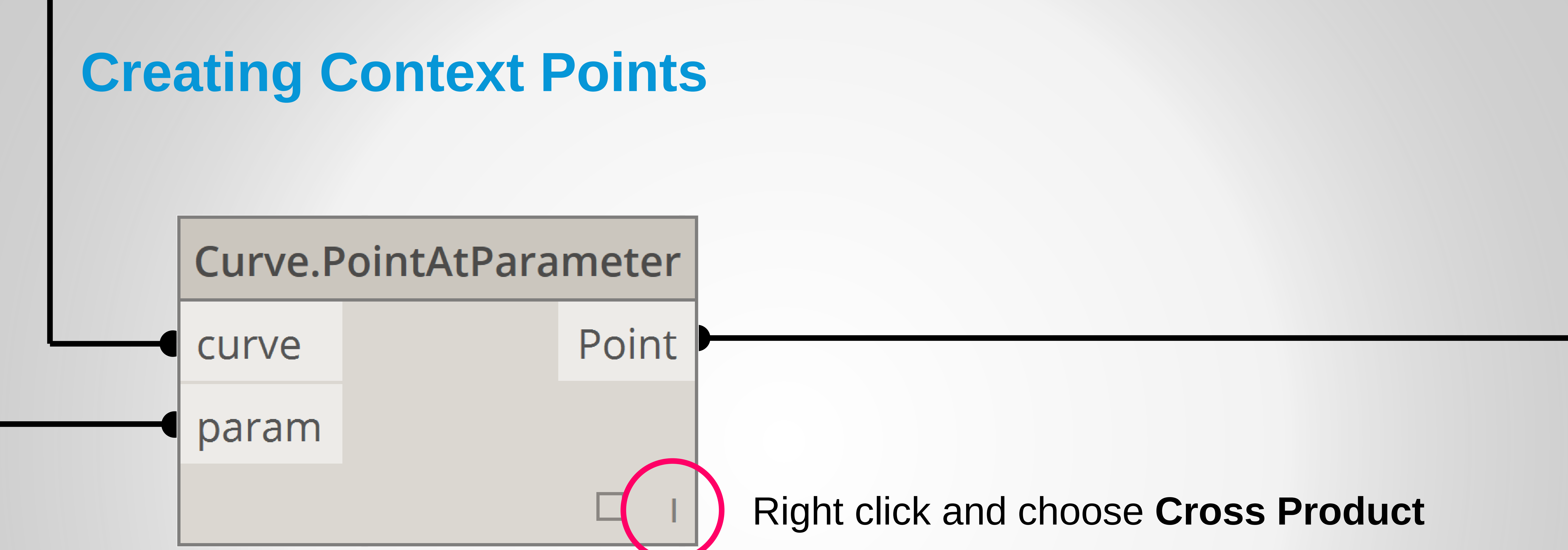


Creating Context Points

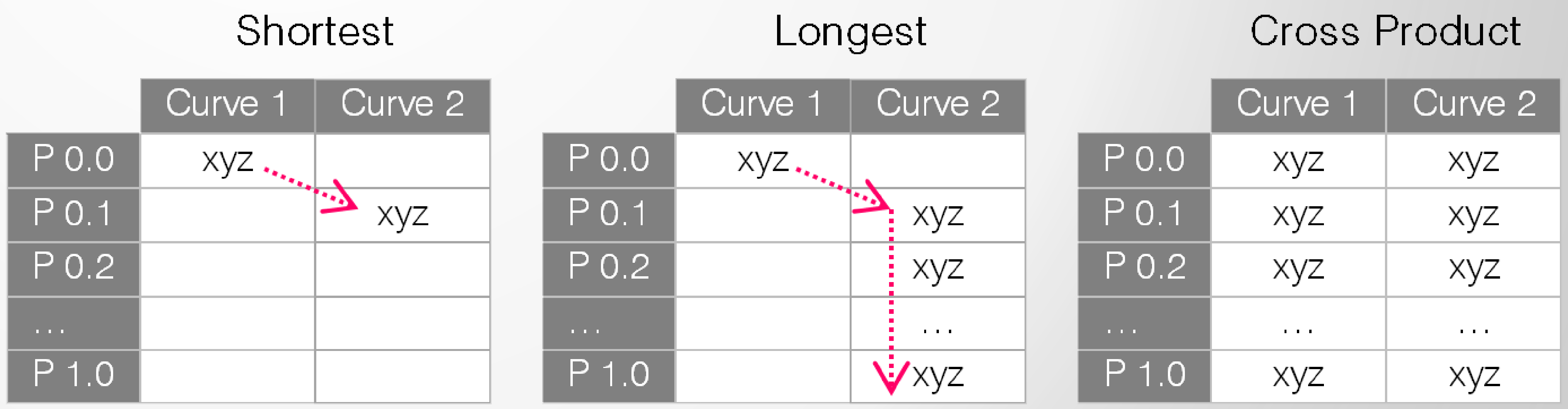


```
0..1..#step;
```

Creating Context Points



Right click and choose **Cross Product**



Shortest

	Curve 1	Curve 2
P 0.0	xyz	
P 0.1		xyz
P 0.2		
...		
P 1.0		

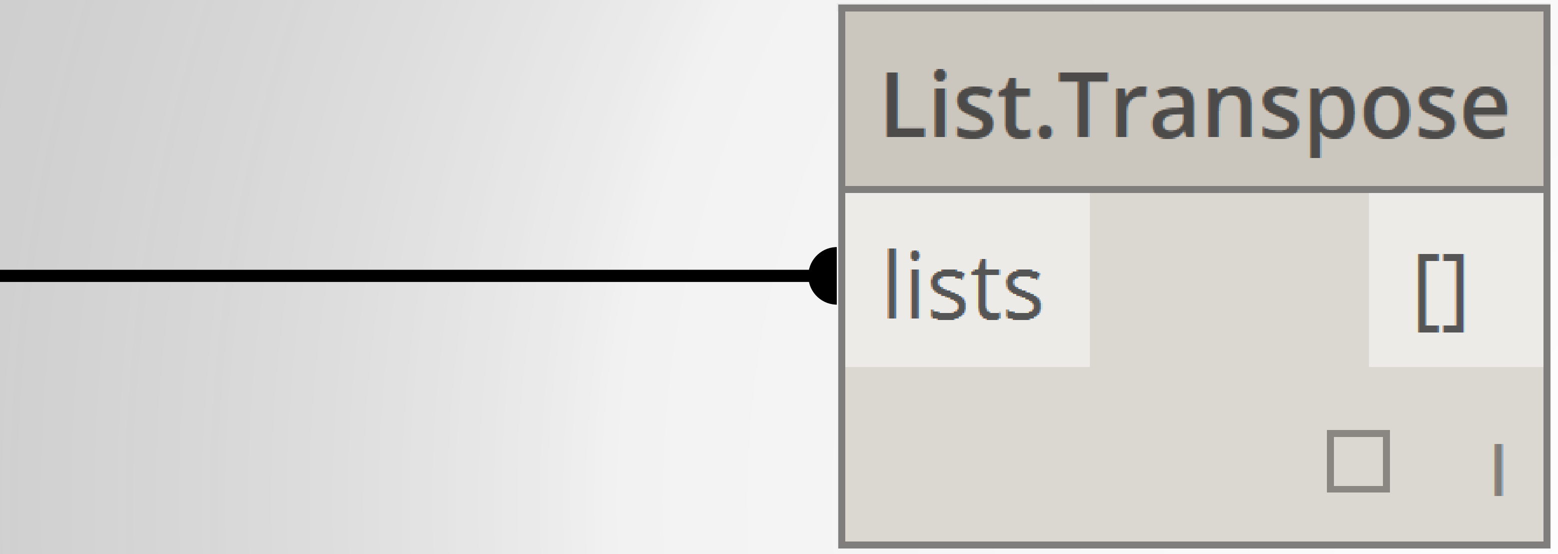
Longest

	Curve 1	Curve 2
P 0.0	xyz	
P 0.1		xyz
P 0.2		xyz
...		...
P 1.0		xyz

Cross Product

	Curve 1	Curve 2
P 0.0	xyz	xyz
P 0.1	xyz	xyz
P 0.2	xyz	xyz
...	...	...
P 1.0	xyz	xyz

Creating Context Points

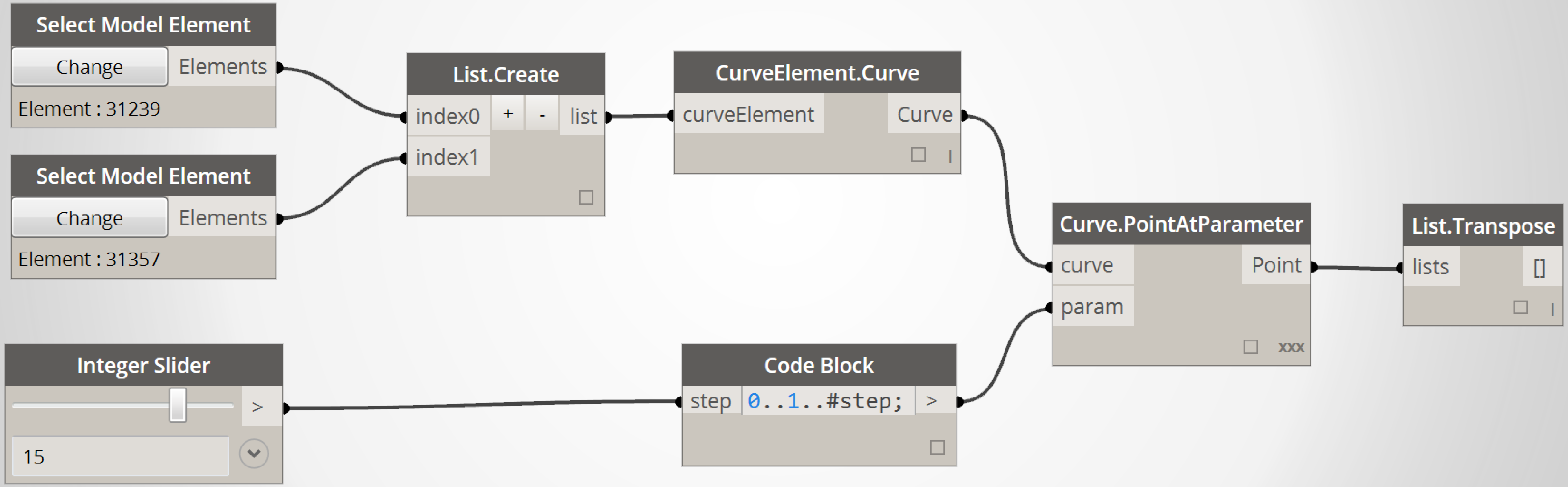


	Parameter 0.1	Parameter 0.2	Parameter 0.3	…	Parameter 1.0
Curve 1	xyz	xyz	xyz	…	xyz
Curve 2	xyz	xyz	xyz	…	xyz

…becomes…

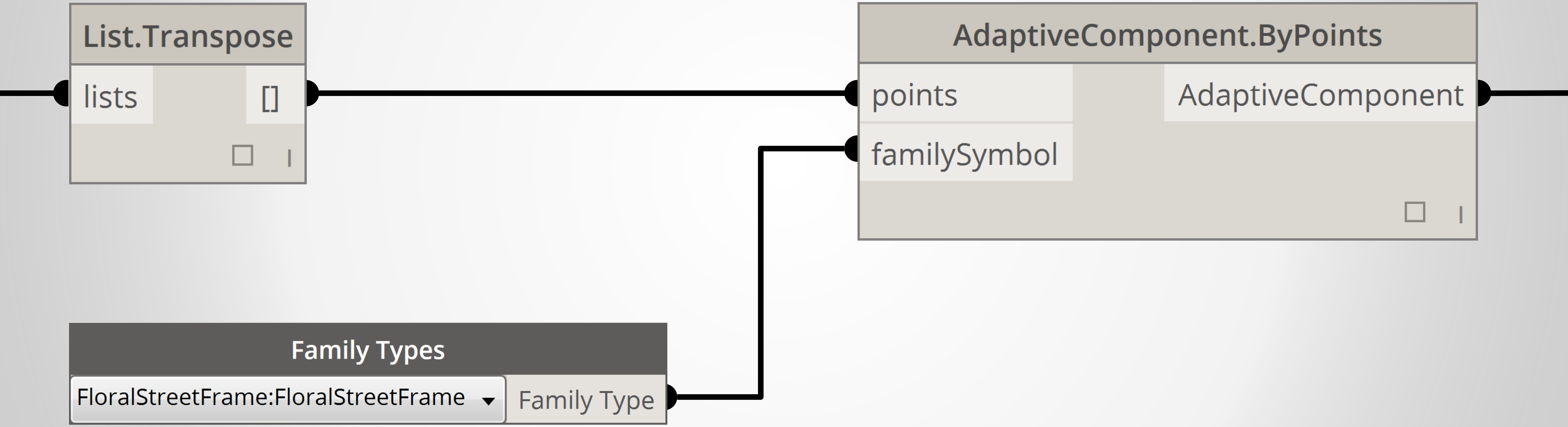
	Curve 1	Curve 2
Parameter 0.0	xyz	xyz
Parameter 0.1	xyz	xyz
Parameter 0.2	xyz	xyz
…	…	…
Parameter 1.0	xyz	xyz

Creating Context Points

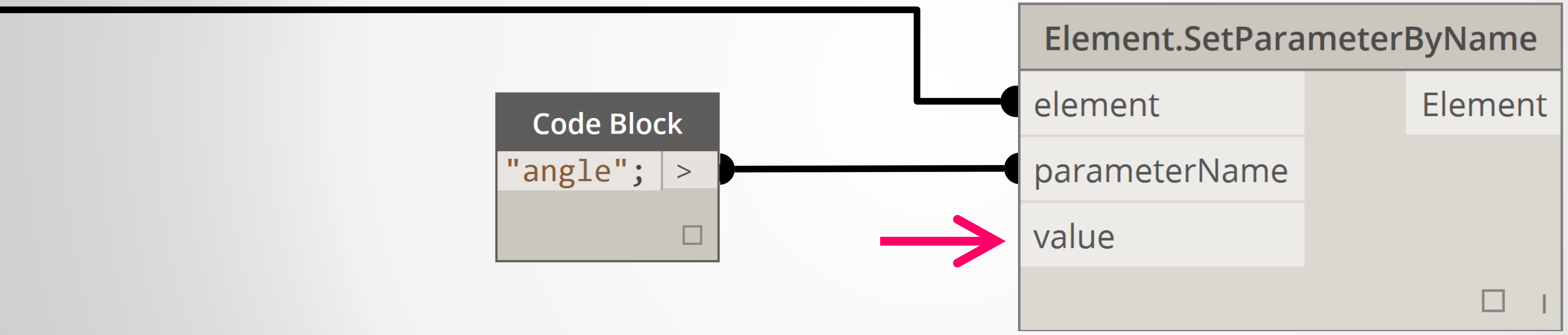


```
0..1..#step;
```

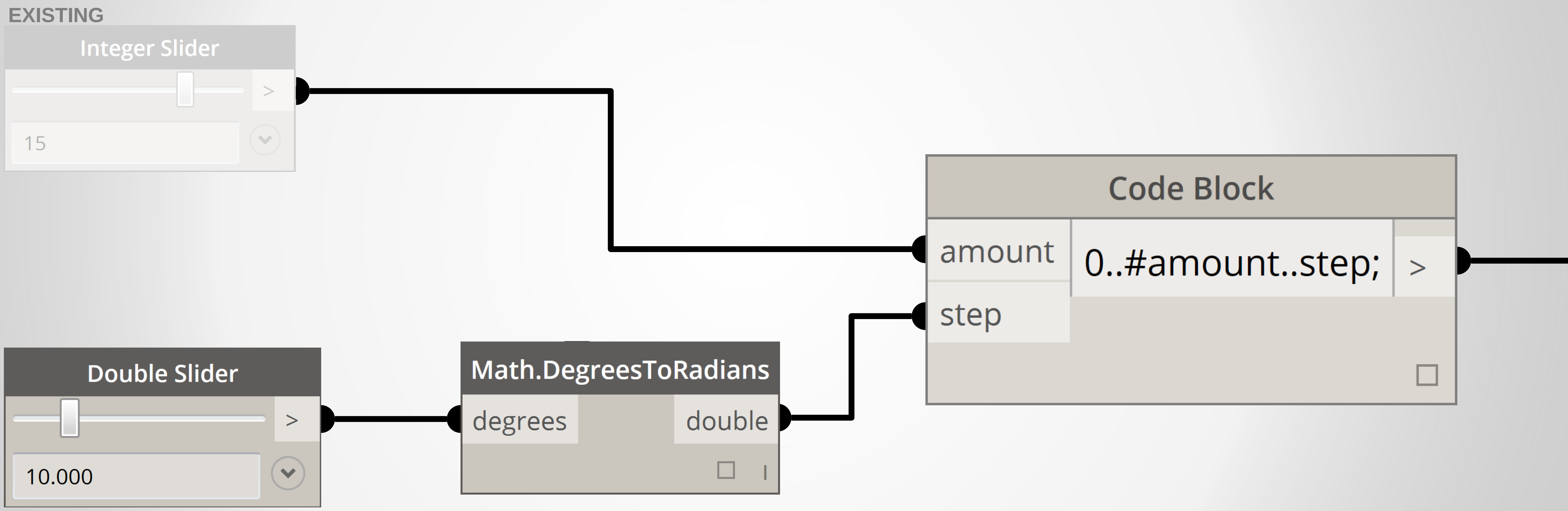
Adaptive Frame Family



Adaptive Frame Family

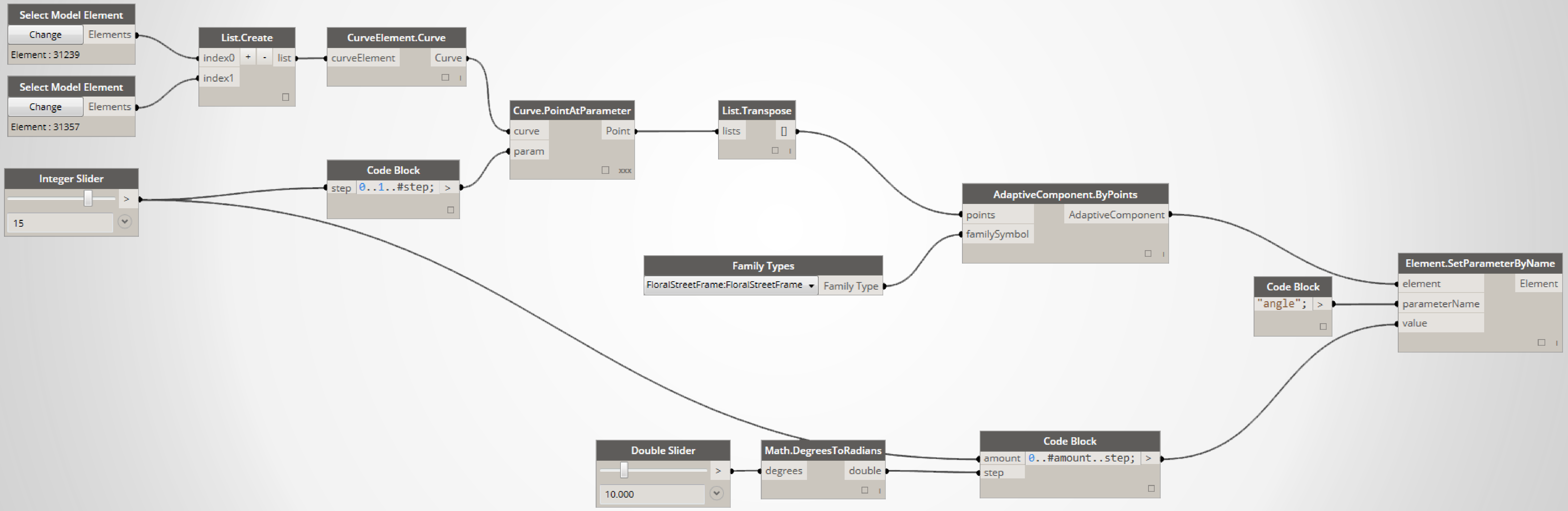


Adaptive Frame Family

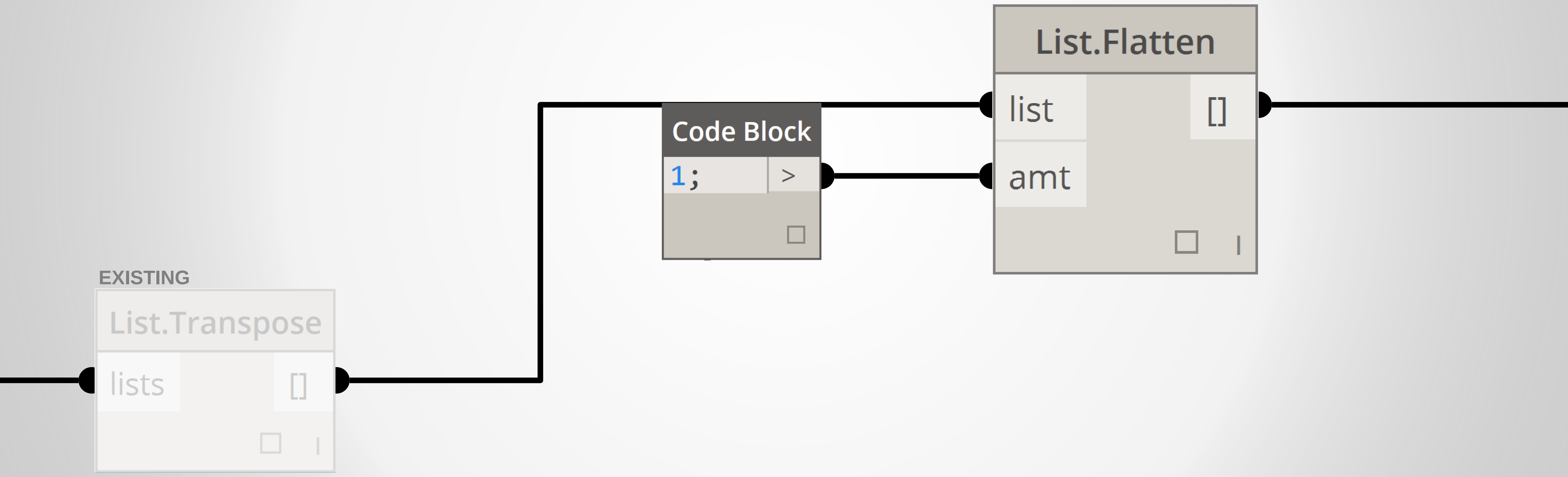


```
0..#amount..step;
```

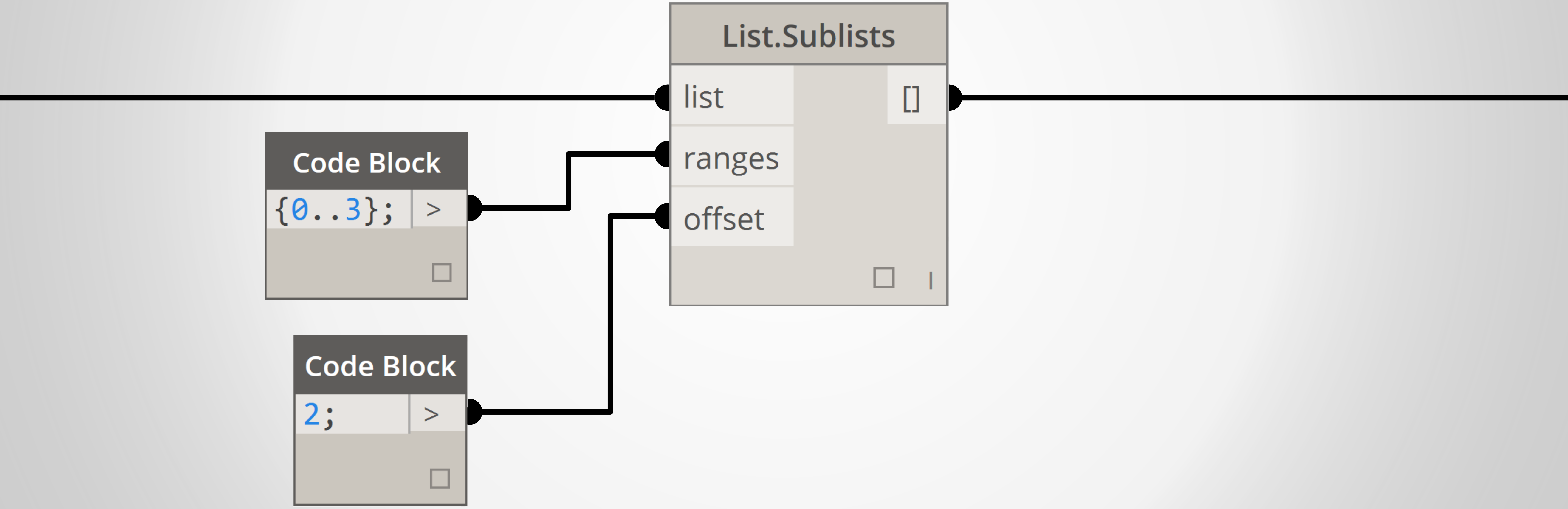
Adaptive Frame Family



Adaptive Glazing Family



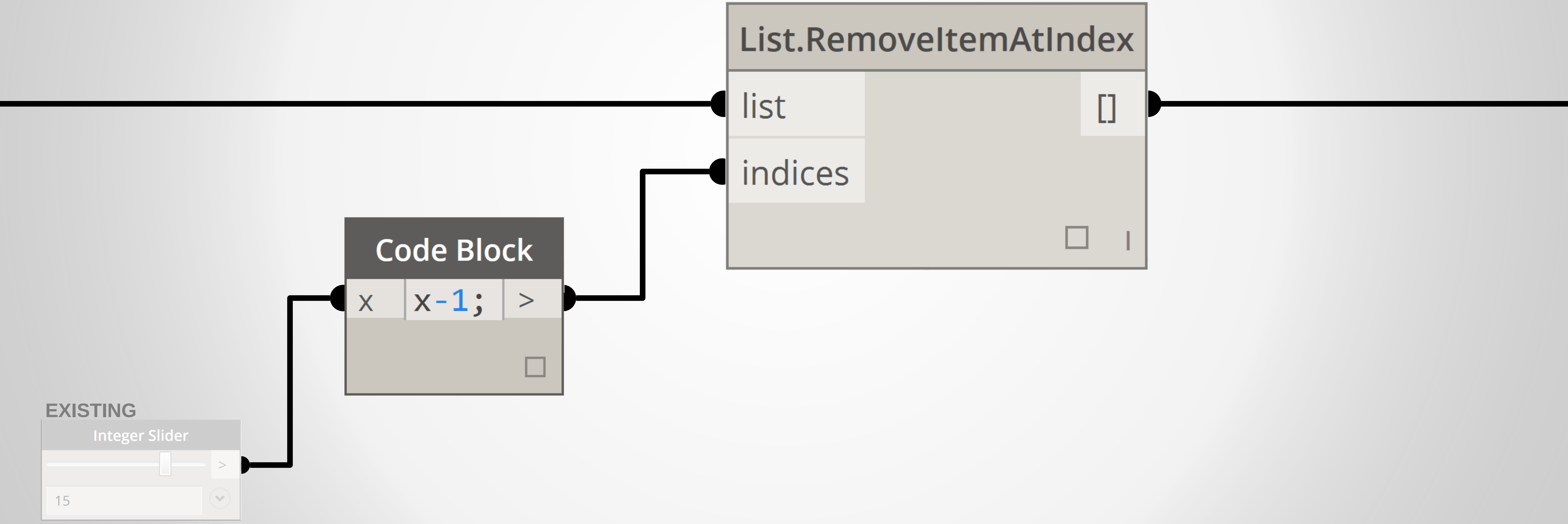
Adaptive Glazing Family



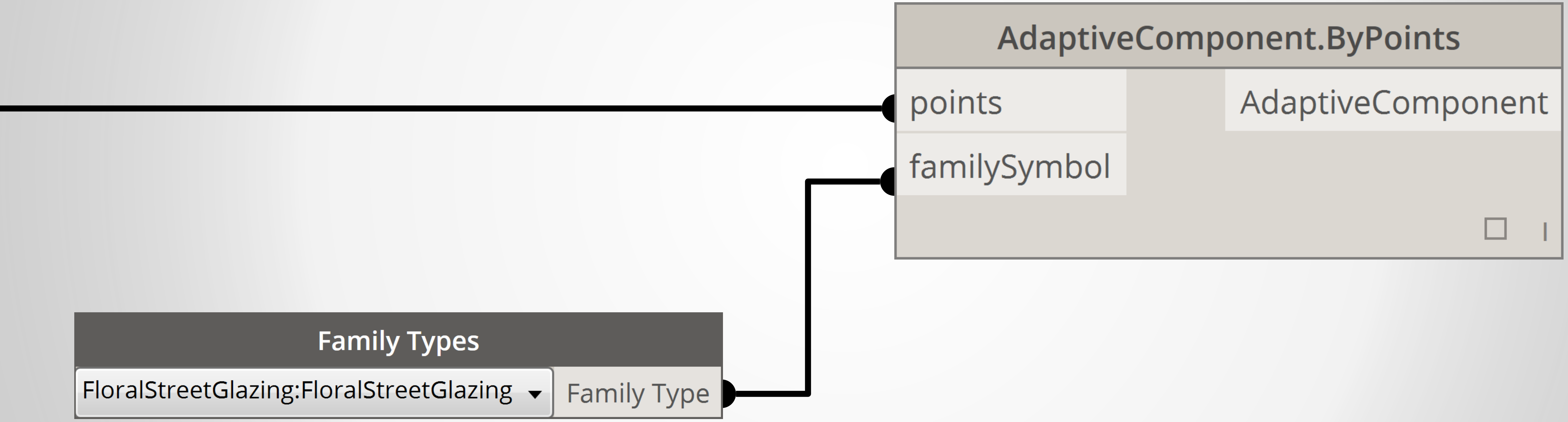
```
{0..3};
```

```
2;
```

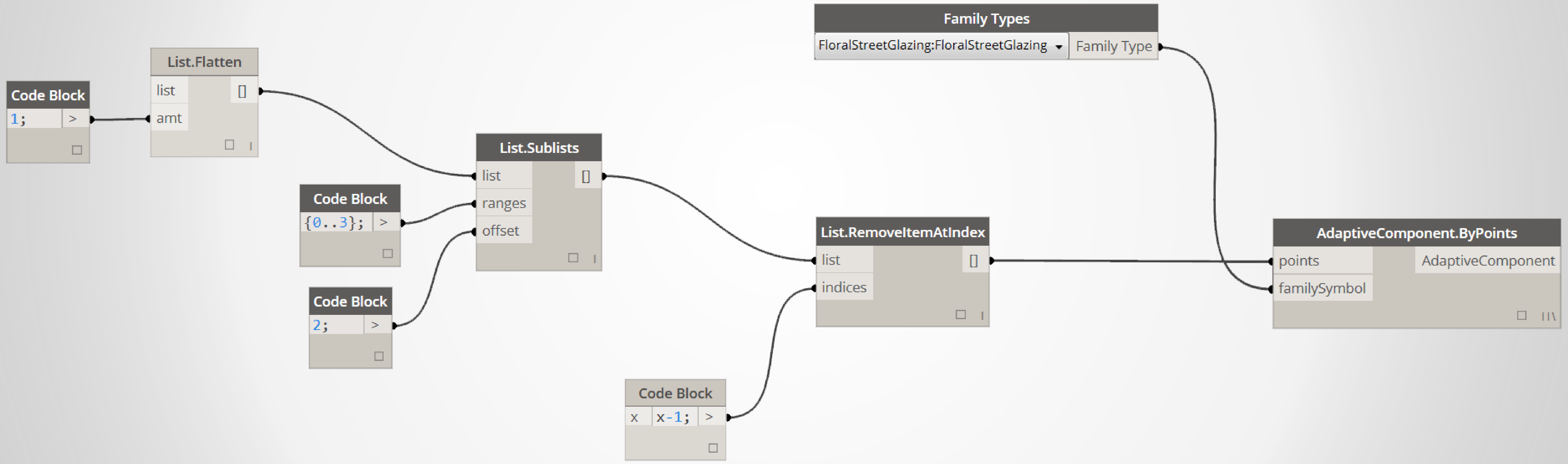
Adaptive Glazing Family



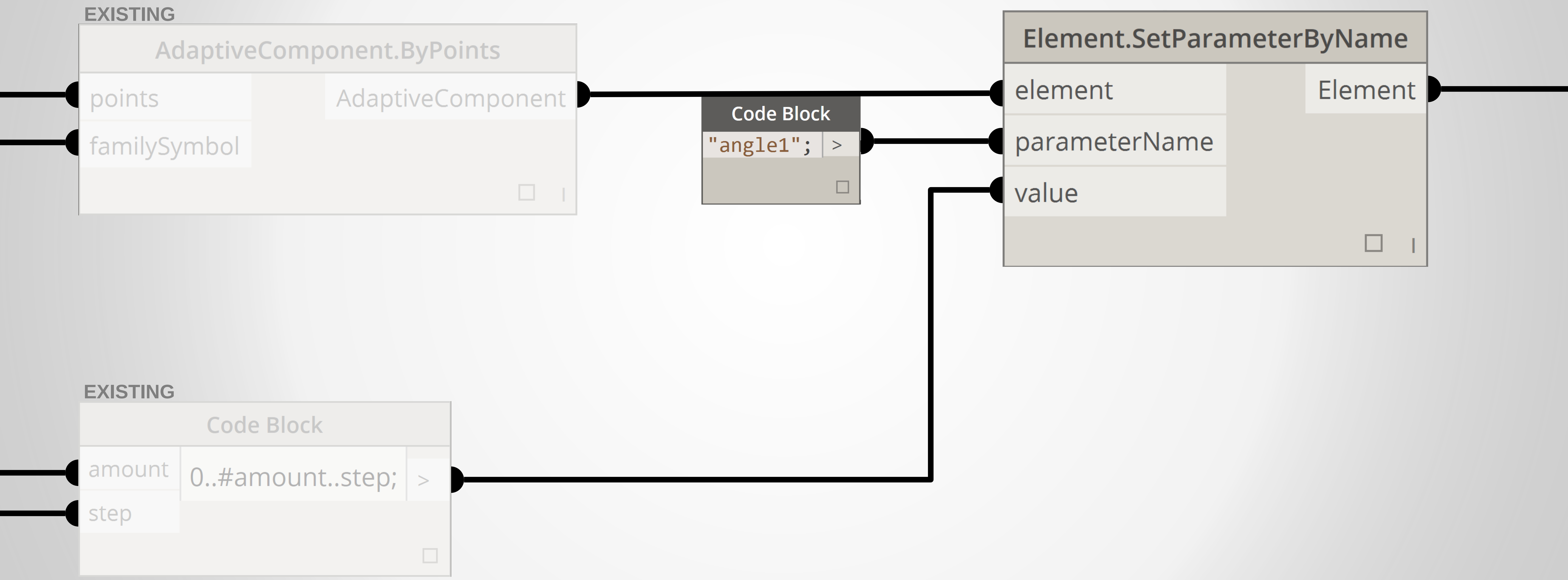
Adaptive Glazing Family


AdaptiveComponent.ByPoints
points
familySymbol
AdaptiveComponent
Family Types
FloralStreetGlazing:FloralStreetGlazing
Family Type

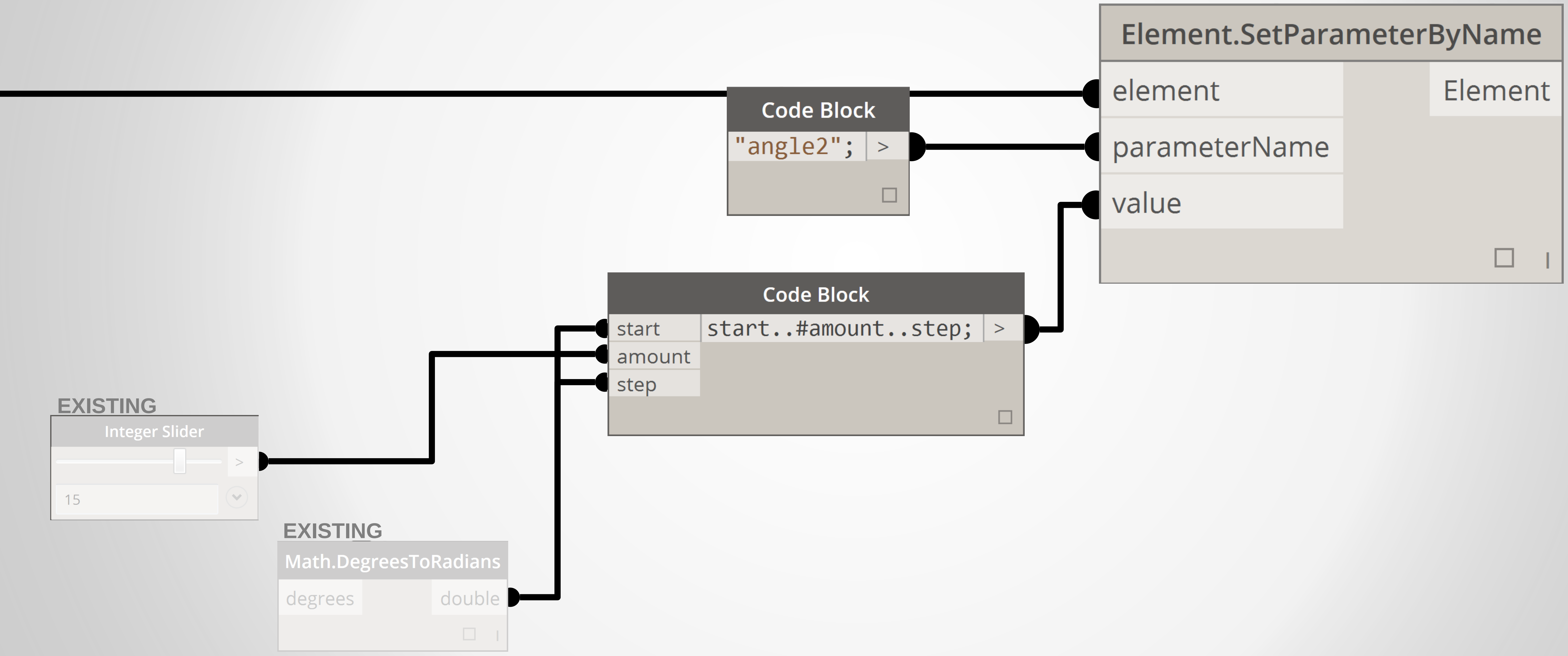
Adaptive Glazing Family



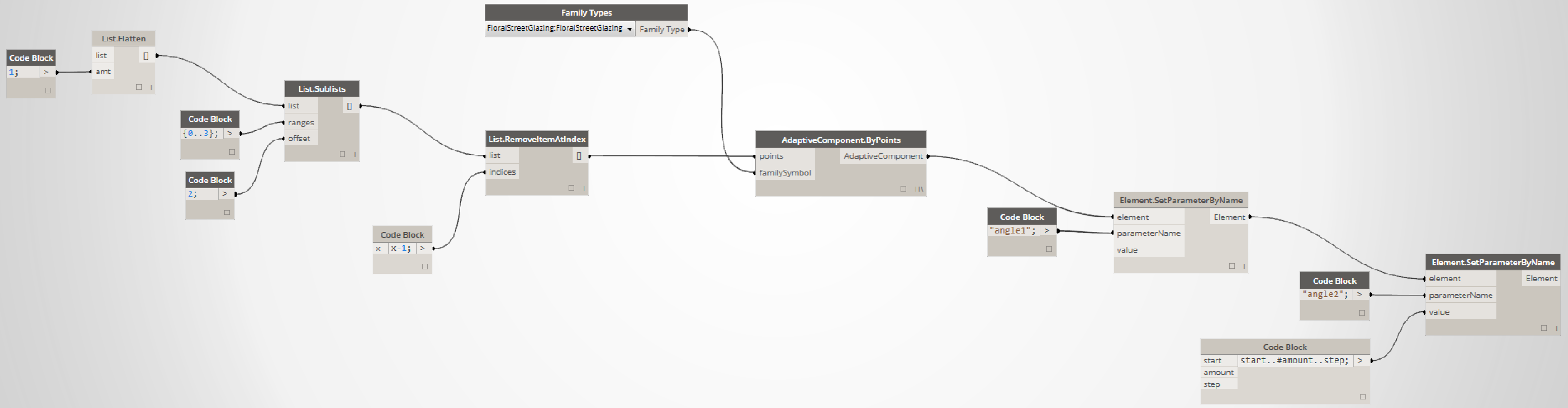
Adaptive Glazing Family



Adaptive Glazing Family

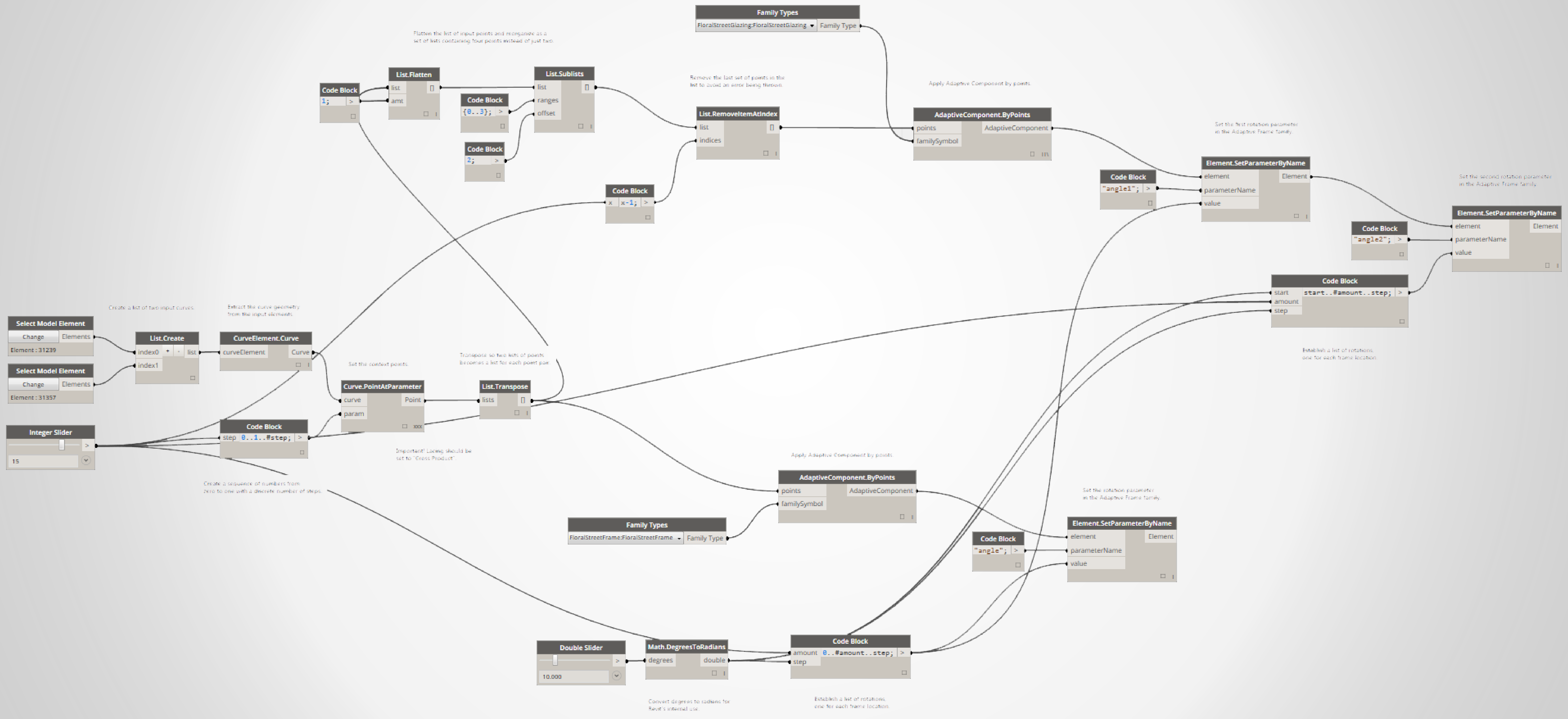


Adaptive Glazing Family



```
1;
{0..3};
2;
x-1;
FloralStreetGlazing:FloralStreetGlazing
"angle1";
"angle2";
start..#amount..step;
```

Final Dynamo Script



Flatten the list of input points and reorganize as a set of lists containing four points instead of just two.

Remove the last set of points in the list to avoid an error being thrown.

Apply Adaptive Component by points.

Set the first rotation parameter in the Adaptive Frame family.

Set the second rotation parameter in the Adaptive Frame family.

Create a list of two input curves.

Extract the curve geometry from the input elements.

Set the context points.

Transpose so two lists of points becomes a list for each point pair.

Establish a list of rotations, one for each frame location.

Important! Lacing should be set to "Cross Product".

Create a sequence of numbers from zero to one with a discrete number of steps.

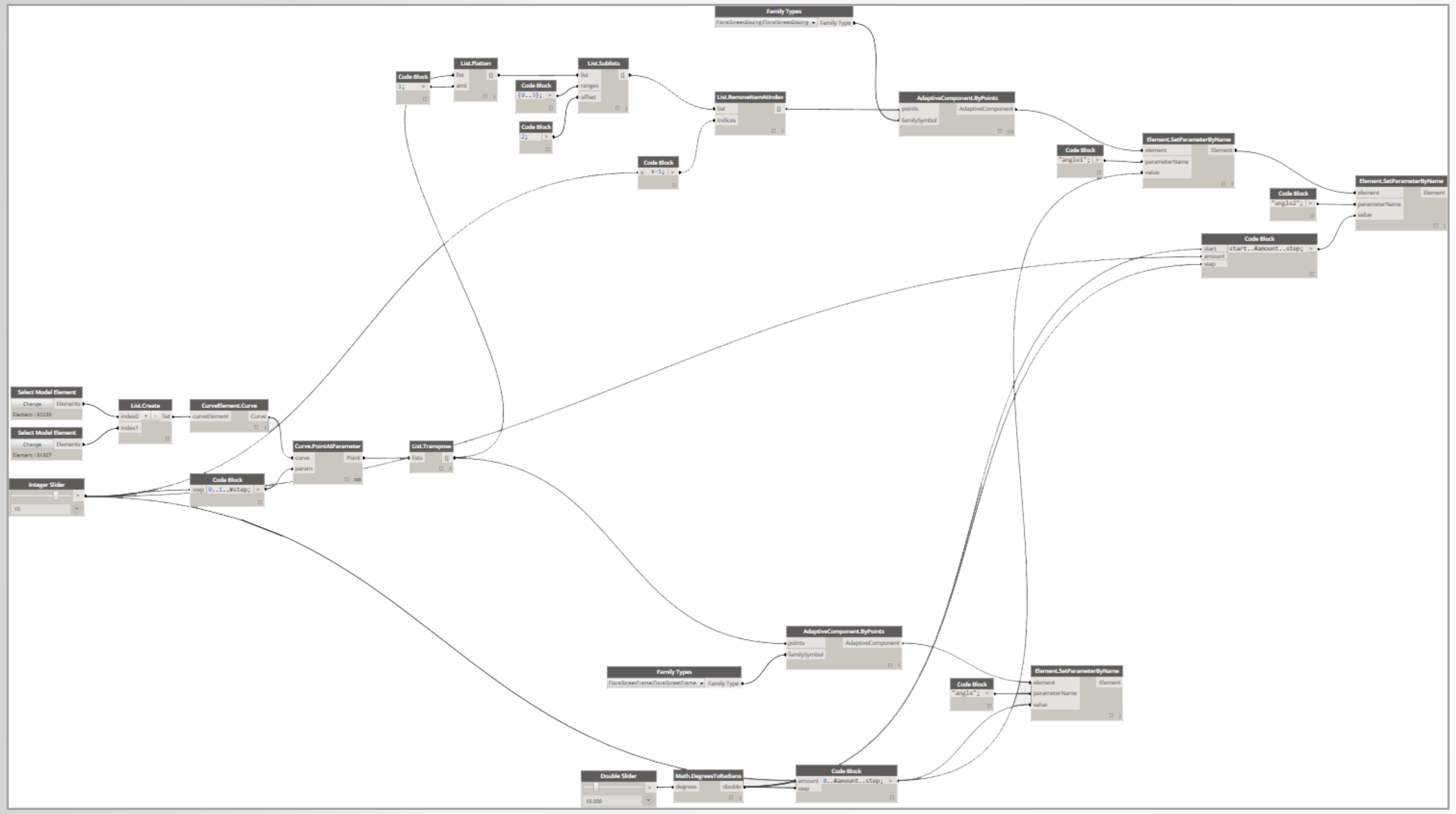
Apply Adaptive Component by points.

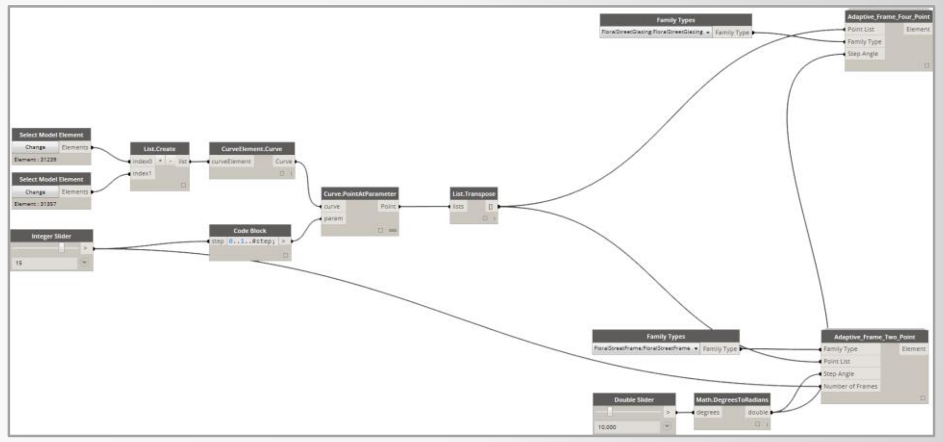
Set the rotation parameter in the Adaptive Frame family.

Convert degrees to radians for Revit's internal use.

Establish a list of rotations, one for each frame location.

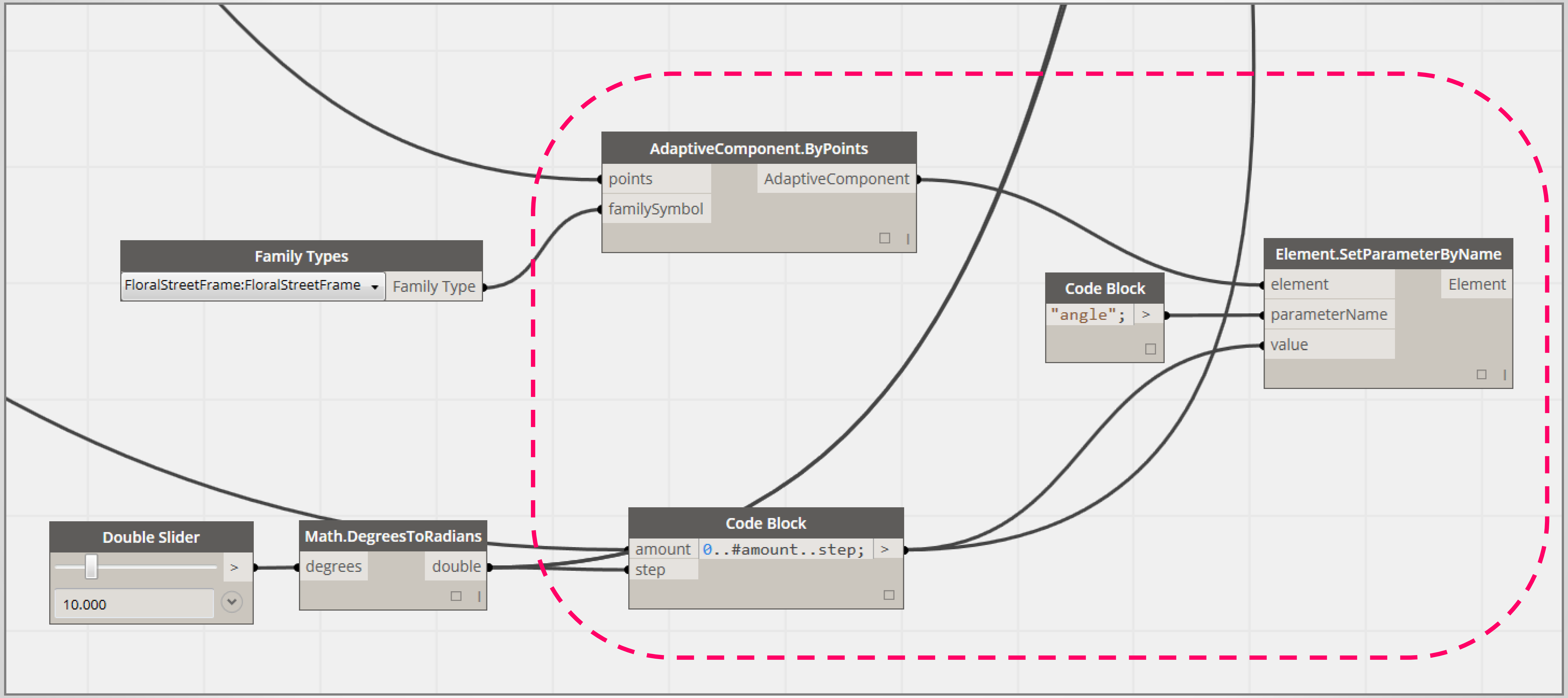
Custom Nodes in Dynamo



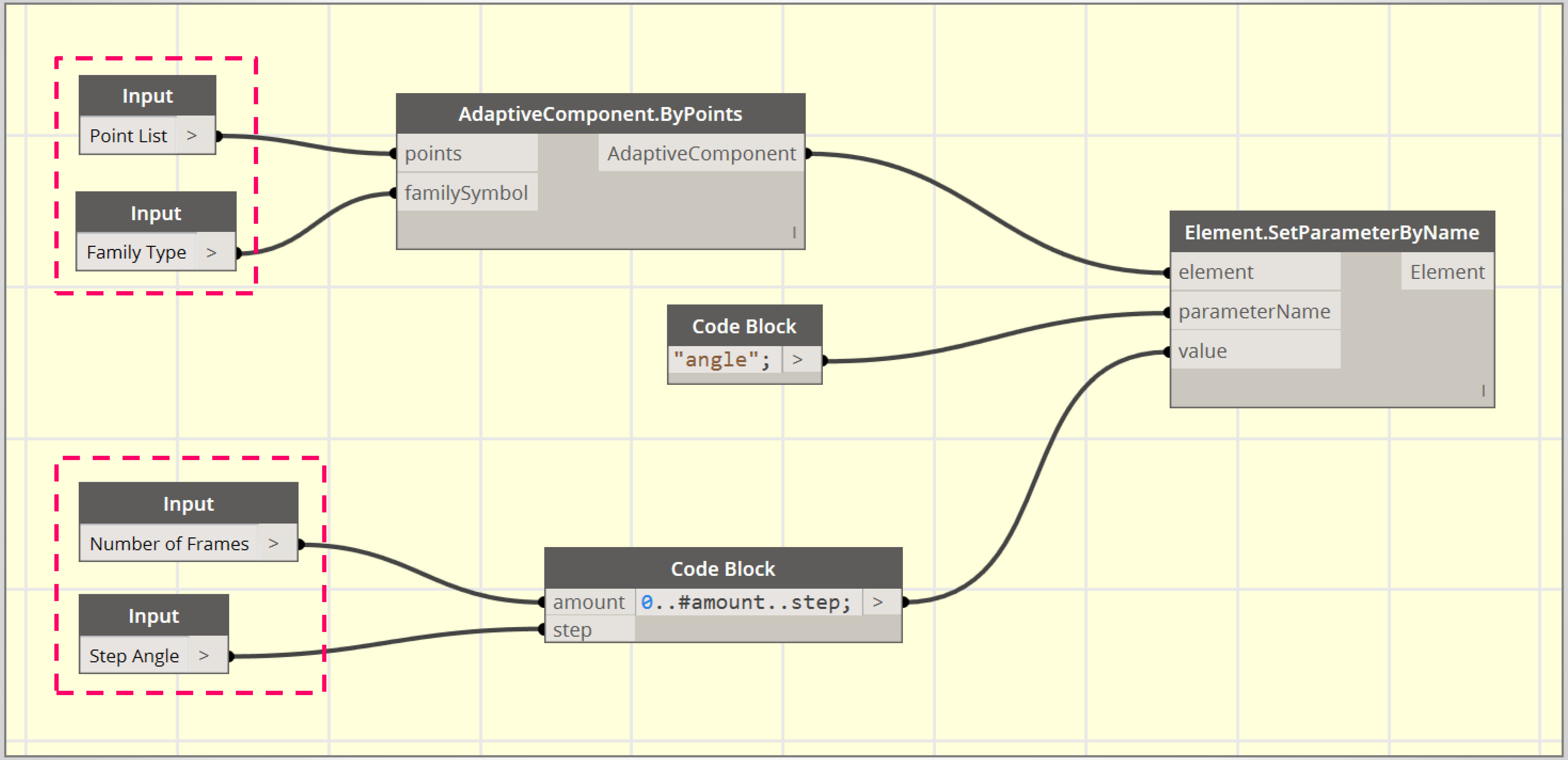


29 interconnected nodes …becomes… 14

Custom Nodes in Dynamo

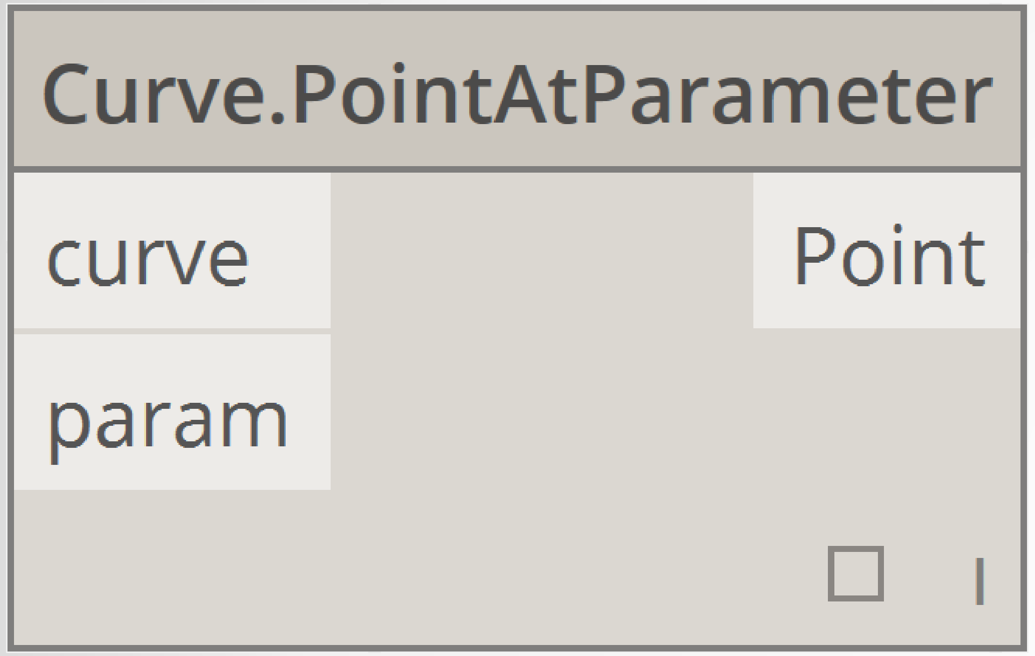


Custom Nodes in Dynamo



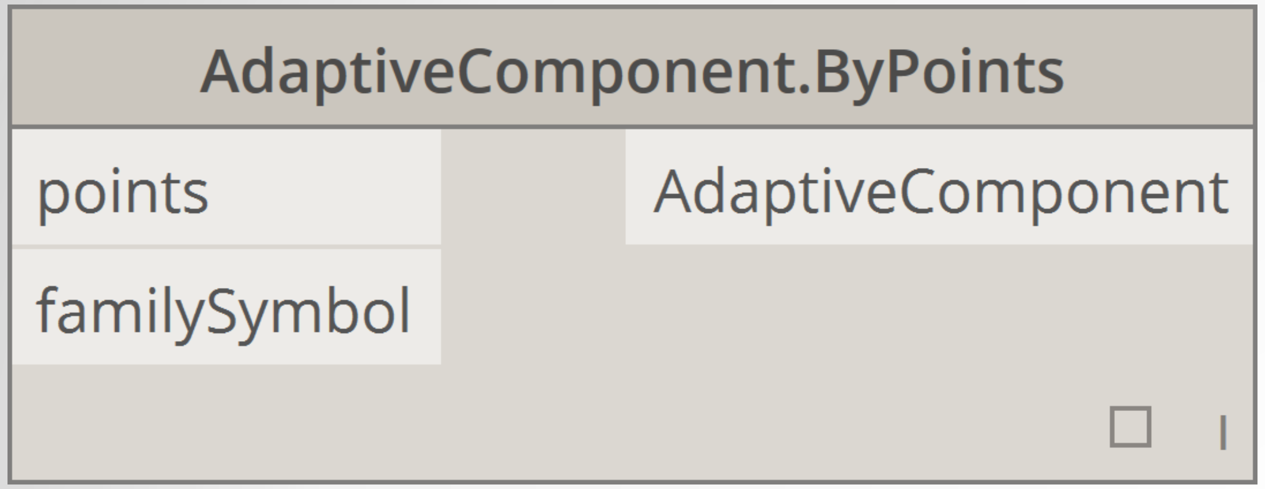
Automated Instantiation with Revit's .Net API

Core Functionality Compared



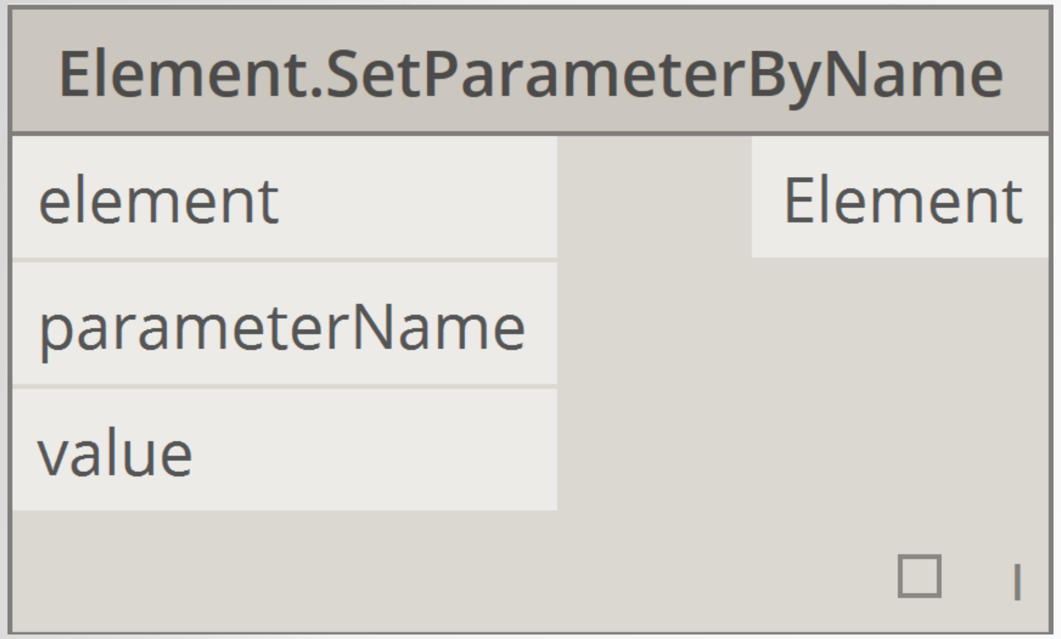
```
double curveParm = (double)i/numInstances;

pt.Position = curve.GeometryCurve
    .Evaluate(curveParm, true);
```

Core Functionality Compared



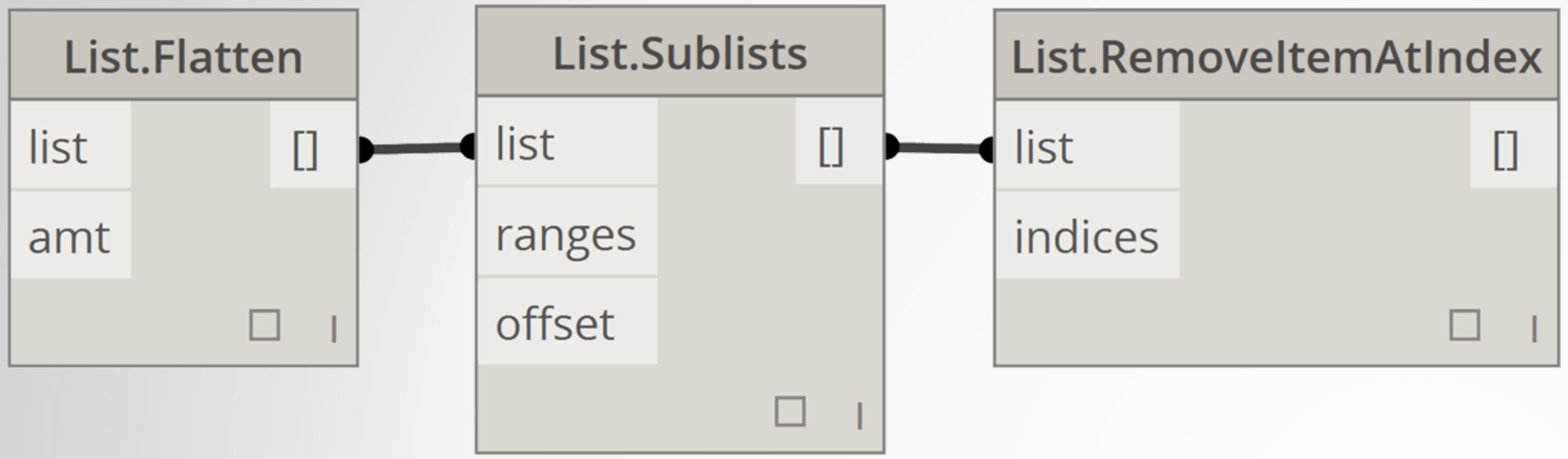
```
FamilyInstance instance = AdaptiveComponentInstanceUtils
    .CreateAdaptiveComponentInstance(doc, symbol);

ReferencePoint pt = doc
    .GetElement(placePointIds[j]) as ReferencePoint;

pt.Position = curve.GeometryCurve
    .Evaluate(curveParm, true);
```

Core Functionality Compared



```
ParameterMapIterator it =  paramMap.ForwardIterator();

Parameter angleParm = it.Current as Parameter;

angleParm.Set(((angleDegStep*i)+angleDegStart)*degToRad);
```

Core Functionality Compared



```
if (j < 2)
  curve = doc.GetElement(curves[j].ElementId) as CurveElement;

else {
  curve = doc.GetElement(curves[j-2].ElementId) as CurveElement;
  curveParm = (double)(i+1)/numInstances;
}
```

All Functions (Public and Private)

```
public void Main() {…}


private FamilySymbol FindAdaptiveComponent(Document doc, string name)
{…}


private void DeleteAdaptiveComponent(Document doc, FamilySymbol famSym)
{…}


private void InstantiateComponents
    (Document doc, FamilySymbol symbol, IList<Reference> curves) {…}
```

Main Function Workflow

```
public void Main()
{
```

1

```
// Get the active document.
Document doc = this.ActiveUIDocument.Document;
UIDocument uidoc = new UIDocument(doc);
```

Acquire the active document to start pushing and prodding through the API. Dynamo does the same. When starting up it indicates the active document in the console.

Main Function Workflow

2

```
// Locate Adaptive Components in file.
FamilySymbol frame = FindAdaptiveComponent(doc, "FloralStreetFrame");
FamilySymbol glazing = FindAdaptiveComponent(doc, "FloralStreetGlazing");

// Check to see if family symbols were found and throw error message if not.
if ((frame == null) || (glazing == null)) {
TaskDialog.Show("Error","Could not locate adaptive components in this file.");
return;
}
```

Call a function to locate the adaptive components for this example. Then throw an error if none are found.

Main Function Workflow

2

```
FilteredElementCollector col = new FilteredElementCollector(doc);
           col.OfCategory(BuiltInCategory.OST_GenericModel)
               .OfClass(typeof(FamilySymbol));
fs.Family.FamilyPlacementType == FamilyPlacementType.Adaptive
fs.Name == name
```

A filtered element collector allows for quick searching of the entire model without loading everything into memory.

Main Function Workflow

3

```
// Delete existing instances of Adaptive Components.
DeleteAdaptiveComponent(doc, frame);
DeleteAdaptiveComponent(doc, glazing);
```

```
FamilyInstanceFilter filter = new FamilyInstanceFilter(doc, famSym.Id);
ICollection<ElementId> familyInstances = collector
     .WherePasses(filter).ToElementIds();
doc.Delete(familyInstances);
```

Once the existing components are collected they need to be cleared out to make way for the new instances. This feature is built into Dynamo.

Main Function Workflow

4

```
// Prompt user to select curves & test that two have been selected.
IList<Reference> curves = uidoc.Selection.PickObjects
    (ObjectType.Element, new CurveElementSelectionFilter(),
    "Select two curves");
if (curves.Count != 2) {
    TaskDialog.Show("Error","Please select exactly two curves.");
    return;
}
```

Error handling is an important part of writing any code. In this case we throw an error if the user doesn’t pick exactly two curves.

Main Function Workflow

4

```
    public class CurveElementSelectionFilter : ISelectionFilter
    {
        public bool AllowElement( Element e )
        {
          return e is CurveElement;
        }

        public bool AllowReference( Reference r, XYZ p )
        {
          return true;
        }
    }
```

We could either allow the user to pick any two objects and then check to see if they are curves after the fact or create a filter to only allow curves to be chosen.

Main Function Workflow

5

```
// Instantiate the adaptive components.
Transaction t = new Transaction
    (doc, "Place Adaptive Component");
t.Start();
InstantiateComponents(doc, frame, curves);
InstantiateComponents(doc, glazing, curves);
t.Commit();
```

Any time a change is made to the Revit model through the API it has to start and end with a transaction.

Main Function Workflow

5

```
For each instance {
    Create a new adaptive component in the center of the
    model space;
    Set the adaptive component placement points based
    upon the calculated curve parameter position;
    Access the “angle” parameter and update;
}
```

The pseudocode (human readable code) above summarizes the steps taken to complete the instantiation of the adaptive component.

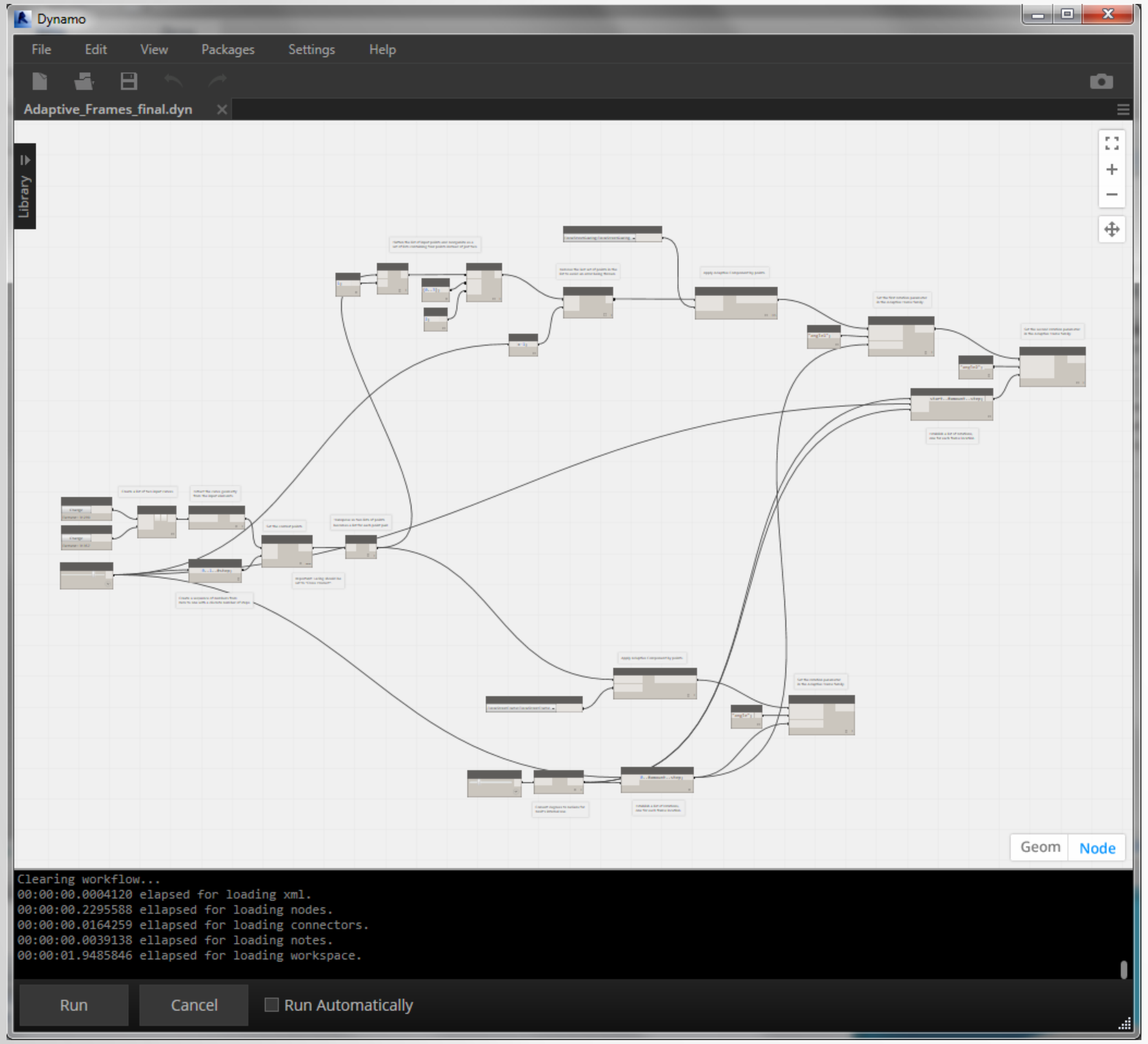
So what does this mean for you?

Which will you choose?

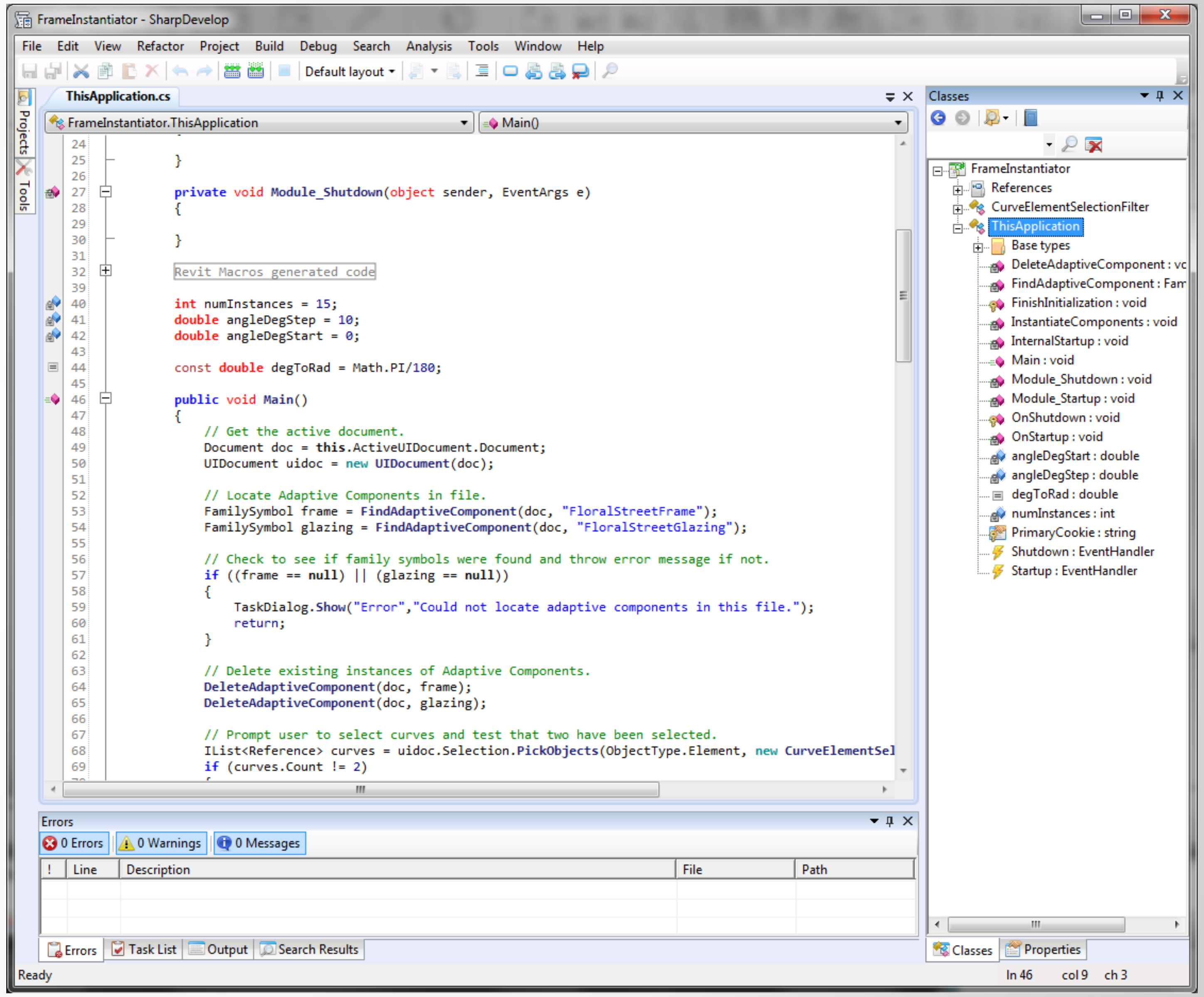
Dynamo ~~or~~ and REVIT API

It’s all about finding the right mix.

Automation Prototyping



```
Clearing workflow...
00:00:00.0004120 elapsed for loading xml.
00:00:00.2295588 ellapsed for loading nodes.
00:00:00.0164259 ellapsed for loading connectors.
00:00:00.0039138 ellapsed for loading notes.
00:00:01.9485846 ellapsed for loading workspace.
```

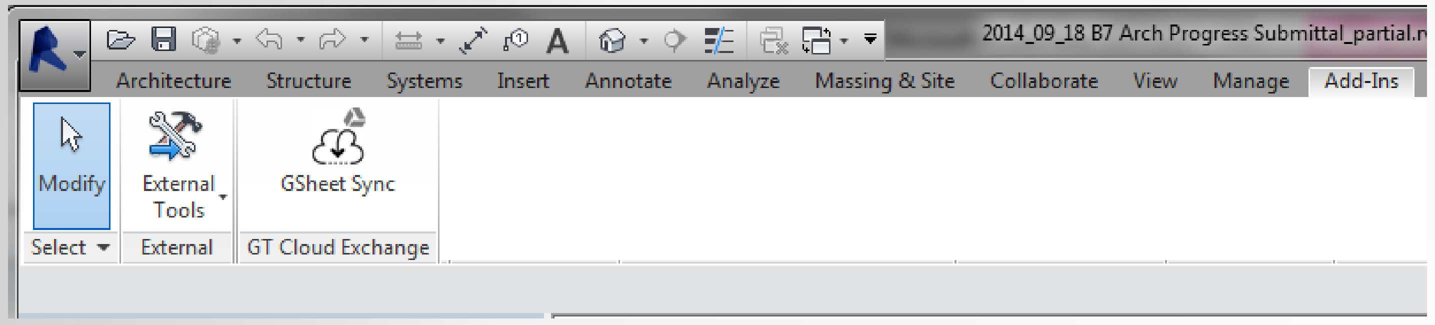


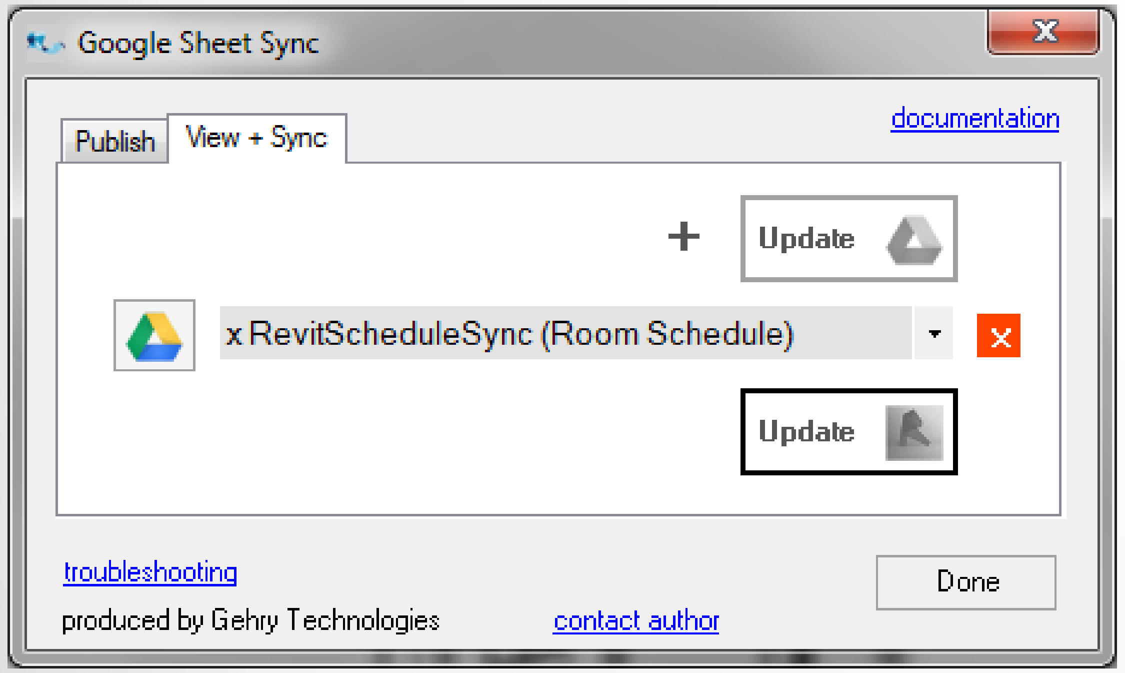
```
        }

        private void Module_Shutdown(object sender, EventArgs e)
        {

        }

        Revit Macros generated code

        int numInstances = 15;
        double angleDegStep = 10;
        double angleDegStart = 0;

        const double degToRad = Math.PI/180;

        public void Main()
        {
            // Get the active document.
            Document doc = this.ActiveUIDocument.Document;
            UIDocument uidoc = new UIDocument(doc);

            // Locate Adaptive Components in file.
            FamilySymbol frame = FindAdaptiveComponent(doc, "FloralStreetFrame");
            FamilySymbol glazing = FindAdaptiveComponent(doc, "FloralStreetGlazing");

            // Check to see if family symbols were found and throw error message if not.
            if ((frame == null) || (glazing == null))
            {
                TaskDialog.Show("Error","Could not locate adaptive components in this file.");
                return;
            }

            // Delete existing instances of Adaptive Components.
            DeleteAdaptiveComponent(doc, frame);
            DeleteAdaptiveComponent(doc, glazing);

            // Prompt user to select curves and test that two have been selected.
            IList<Reference> curves = uidoc.Selection.PickObjects(ObjectType.Element, new CurveElementSel
            if (curves.Count != 2)
```

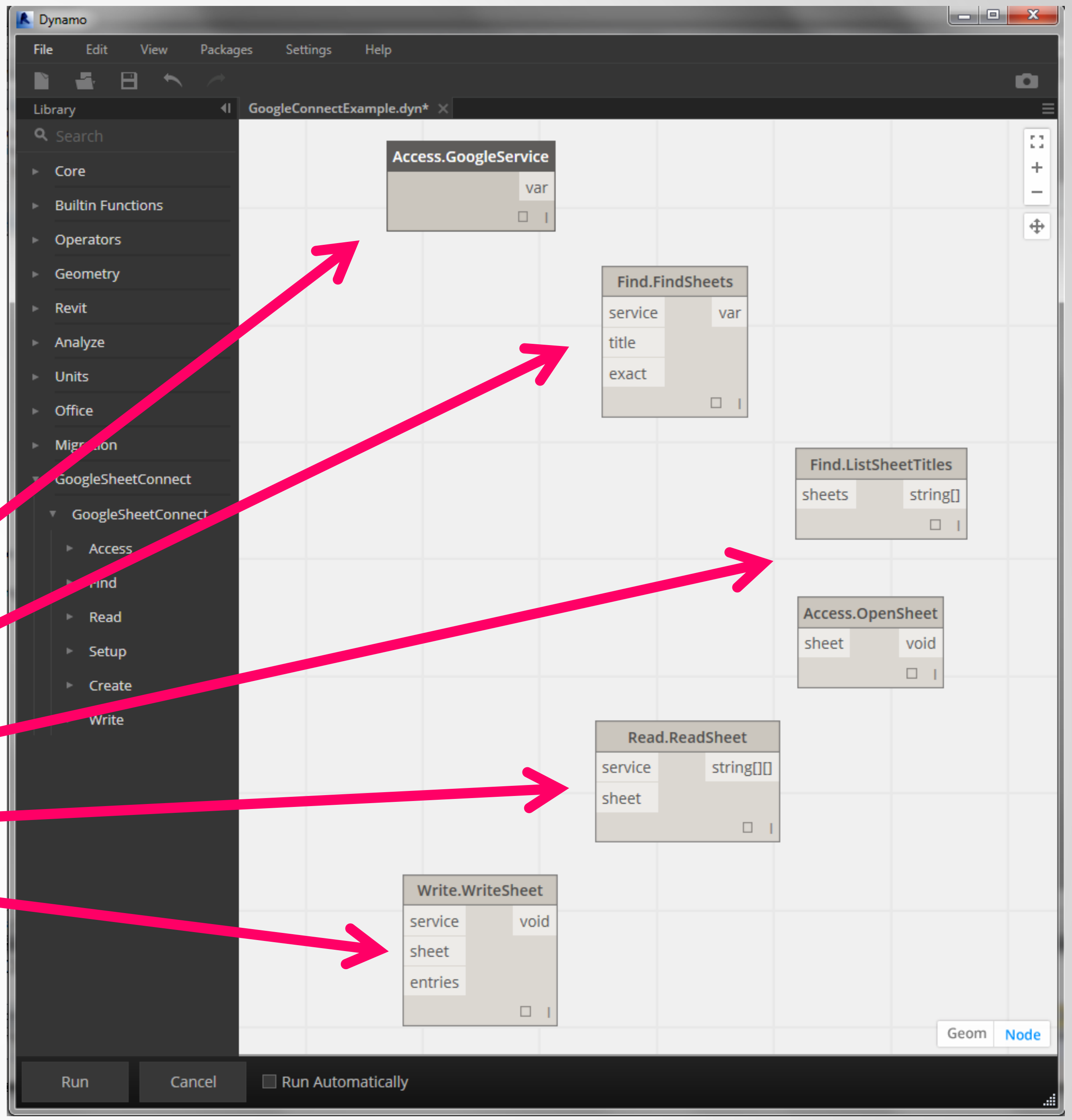
```
FrameInstantiator
  References
  CurveElementSelectionFilter
  ThisApplication
    Base types
    DeleteAdaptiveComponent : vc
    FindAdaptiveComponent : Fam
    FinishInitialization : void
    InstantiateComponents : void
    InternalStartup : void
    Main : void
    Module_Shutdown : void
    Module_Startup : void
    OnShutdown : void
    OnStartup : void
    angleDegStart : double
    angleDegStep : double
    degToRad : double
    numInstances : int
    PrimaryCookie : string
    Shutdown : EventHandler
    Startup : EventHandler
```

Dynamo can be a place to test ideas quickly and make mistakes with little time invested. Then go build a product from the ground up with more knowledge.

Logic Dispersal







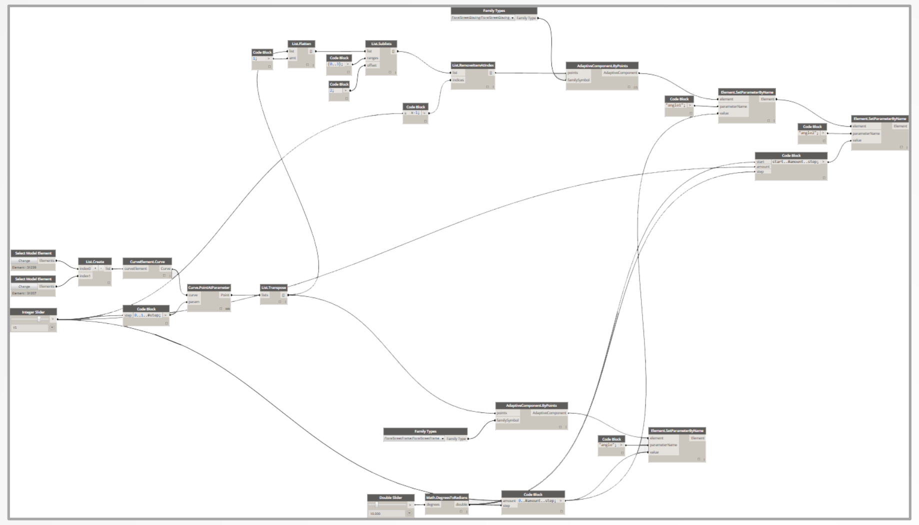
Dynamo can also be a platform for sharing functionality developed for larger proprietary solutions. The base logic for a Revit plugin for connecting to Google Spreadsheets can be reorganized as a set of Dynamo nodes.

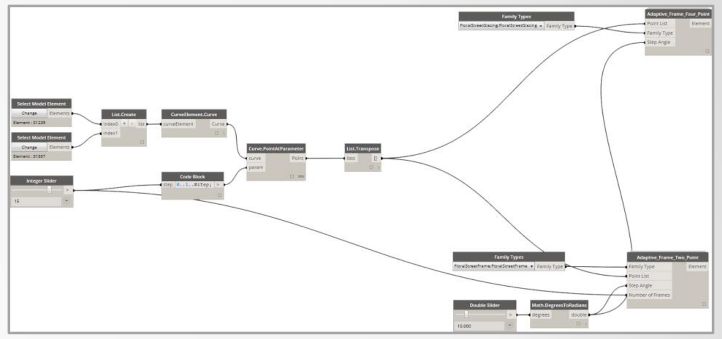
Challenges that May Affect the Mix

- Deprecation
- Debugging
- Logic Organization & Management
- Packaging & Sharing
- User Interface/Experience

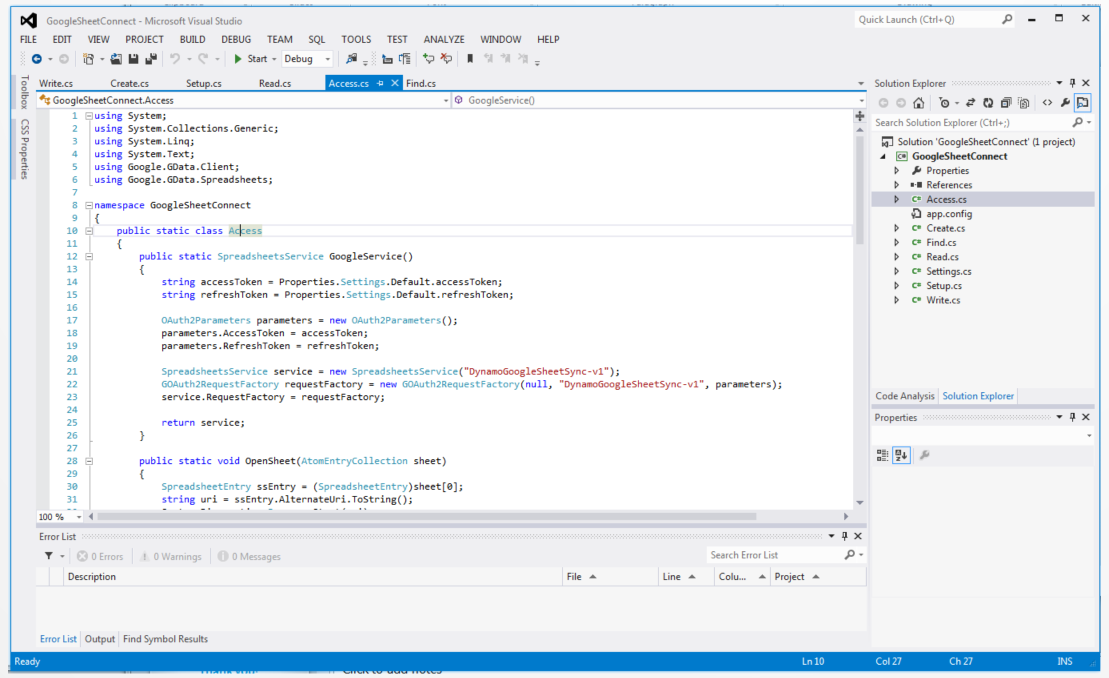
Tools to Organize your Logic

Custom Nodes

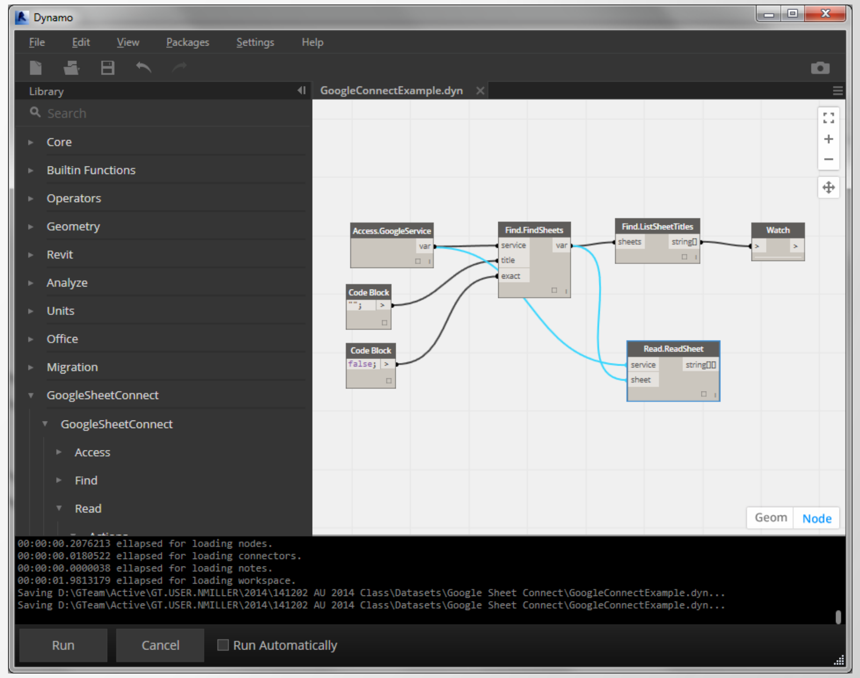




Zero Touch

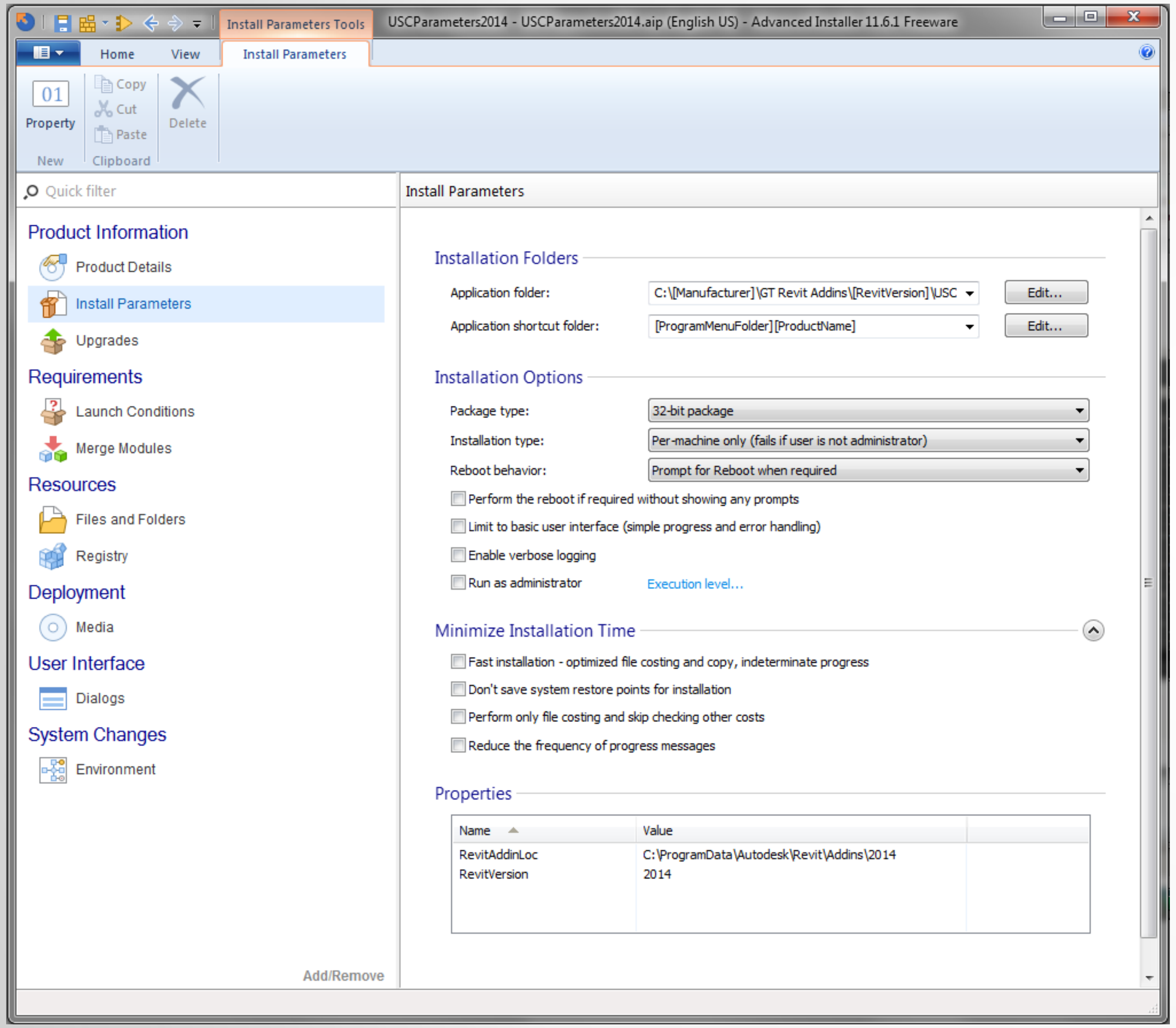


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Google.GData.Client;
using Google.GData.Spreadsheets;

namespace GoogleSheetConnect
{
    public static class Access
    {
        public static SpreadsheetsService GoogleService()
        {
            string accessToken = Properties.Settings.Default.accessToken;
            string refreshToken = Properties.Settings.Default.refreshToken;

            OAuth2Parameters parameters = new OAuth2Parameters();
            parameters.AccessToken = accessToken;
            parameters.RefreshToken = refreshToken;

            SpreadsheetsService service = new SpreadsheetsService("DynamoGoogleSheetSync-v1");
            GOAuth2RequestFactory requestFactory = new GOAuth2RequestFactory(null, "DynamoGoogleSheetSync-v1", parameters);
            service.RequestFactory = requestFactory;

            return service;
        }

        public static void OpenSheet(AtomEntryCollection sheet)
        {
            SpreadsheetEntry ssEntry = (SpreadsheetEntry)sheet[0];
            string uri = ssEntry.AlternateUri.ToString();
```



```
00:00:00.2076213 ellapsed for loading nodes.
00:00:00.0180522 ellapsed for loading connectors.
00:00:00.0000038 ellapsed for loading notes.
00:00:01.9813179 ellapsed for loading workspace.
Saving D:\GTeam\Active\GT.USER.NMILLER\2014\141202 AU 2014 Class\Datasets\Google Sheet Connect\GoogleConnectExample.dyn...
Saving D:\GTeam\Active\GT.USER.NMILLER\2014\141202 AU 2014 Class\Datasets\Google Sheet Connect\GoogleConnectExample.dyn...
```

Packaging & Sharing



Install Parameters

Installation Folders

Application folder: C:\[Manufacturer]\GT Revit Addins\[RevitVersion]\USC — Edit...
Application shortcut folder: [ProgramMenuFolder][ProductName] — Edit...

Installation Options

Package type: 32-bit package
Installation type: Per-machine only (fails if user is not administrator)
Reboot behavior: Prompt for Reboot when required

- [ ] Perform the reboot if required without showing any prompts
- [ ] Limit to basic user interface (simple progress and error handling)
- [ ] Enable verbose logging
- [ ] Run as administrator — Execution level...

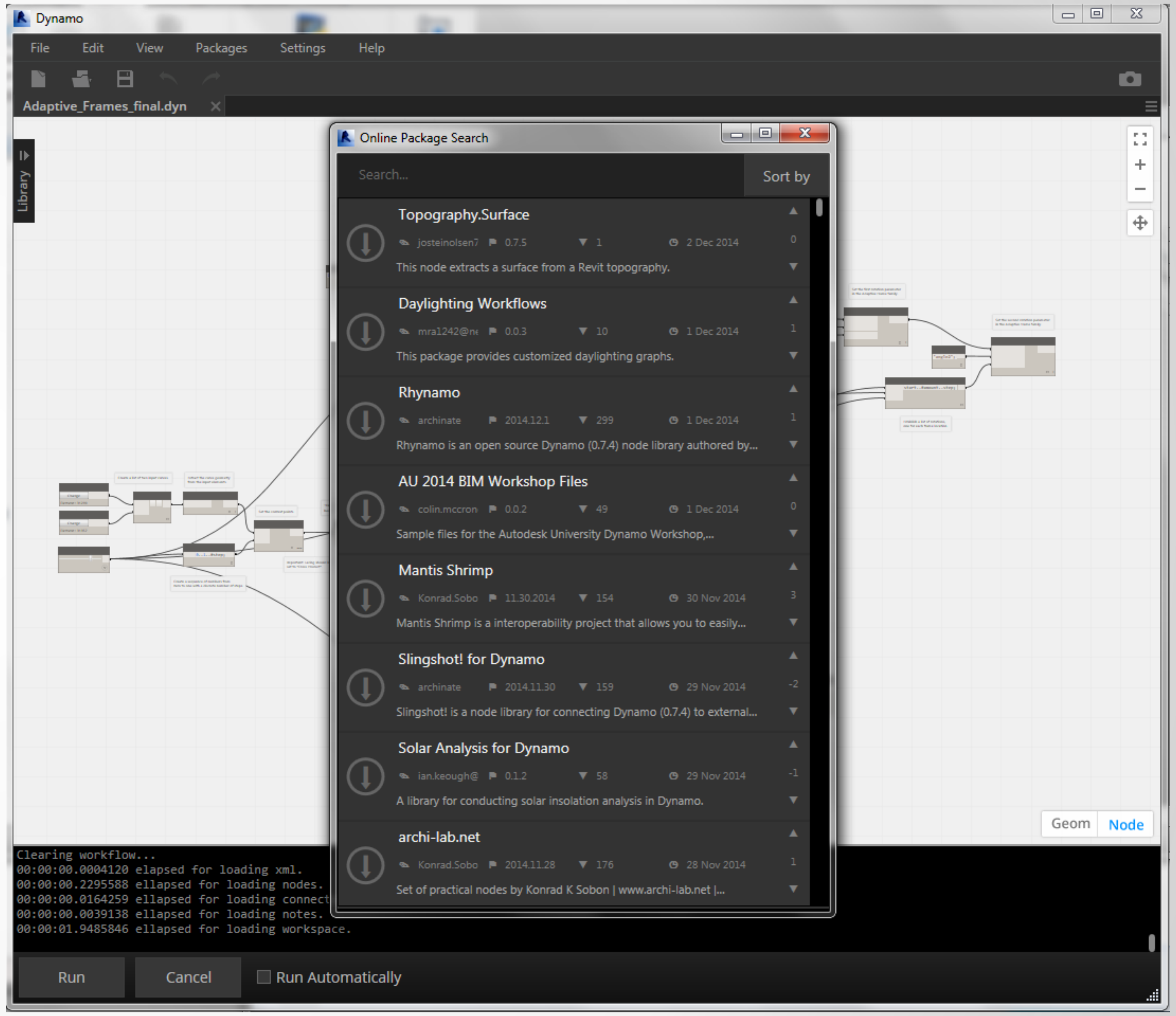
Minimize Installation Time

- [ ] Fast installation - optimized file costing and copy, indeterminate progress
- [ ] Don't save system restore points for installation
- [ ] Perform only file costing and skip checking other costs
- [ ] Reduce the frequency of progress messages

Properties

Name	Value
RevitAddinLoc	C:\ProgramData\Autodesk\Revit\Addins\2014
RevitVersion	2014

Windows Installer Setup
(Cumbersome and tedious)

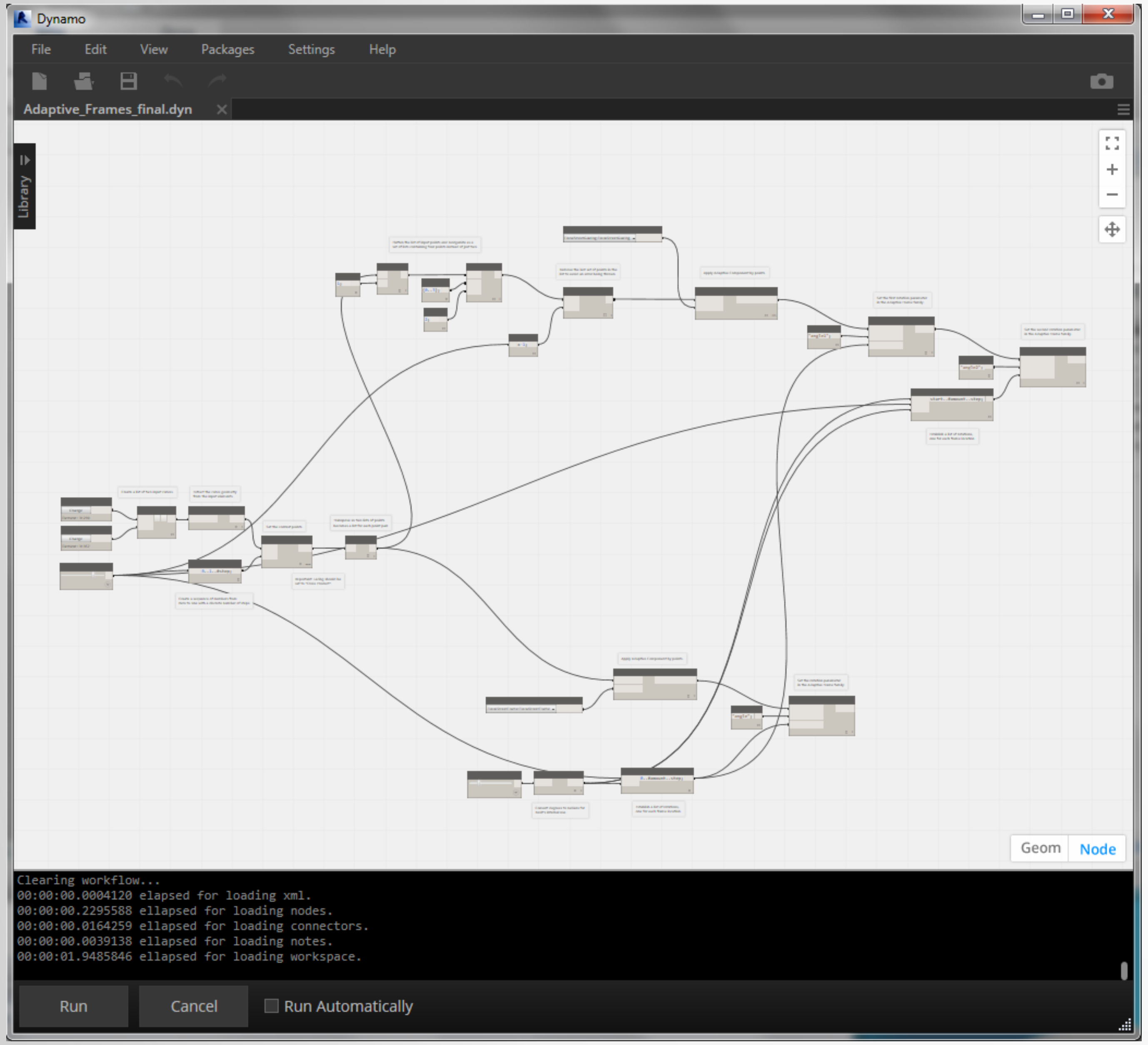


Package	Author	Version	Downloads	Date	Votes	Description
Topography.Surface	josteinolsen7	0.7.5	1	2 Dec 2014	0	This node extracts a surface from a Revit topography.
Daylighting Workflows	mra1242@n	0.0.3	10	1 Dec 2014	1	This package provides customized daylighting graphs.
Rhynamo	archinate	2014.12.1	299	1 Dec 2014	1	Rhynamo is an open source Dynamo (0.7.4) node library authored by...
AU 2014 BIM Workshop Files	colin.mccron	0.0.2	49	1 Dec 2014	0	Sample files for the Autodesk University Dynamo Workshop,...
Mantis Shrimp	Konrad.Sobo	11.30.2014	154	30 Nov 2014	3	Mantis Shrimp is a interoperability project that allows you to easily...
Slingshot! for Dynamo	archinate	2014.11.30	159	29 Nov 2014	-2	Slingshot! is a node library for connecting Dynamo (0.7.4) to external...
Solar Analysis for Dynamo	ian.keough@	0.1.2	58	29 Nov 2014	-1	A library for conducting solar insolation analysis in Dynamo.
archi-lab.net	Konrad.Sobo	2014.11.28	176	28 Nov 2014	1	Set of practical nodes by Konrad K Sobon | www.archi-lab.net |...

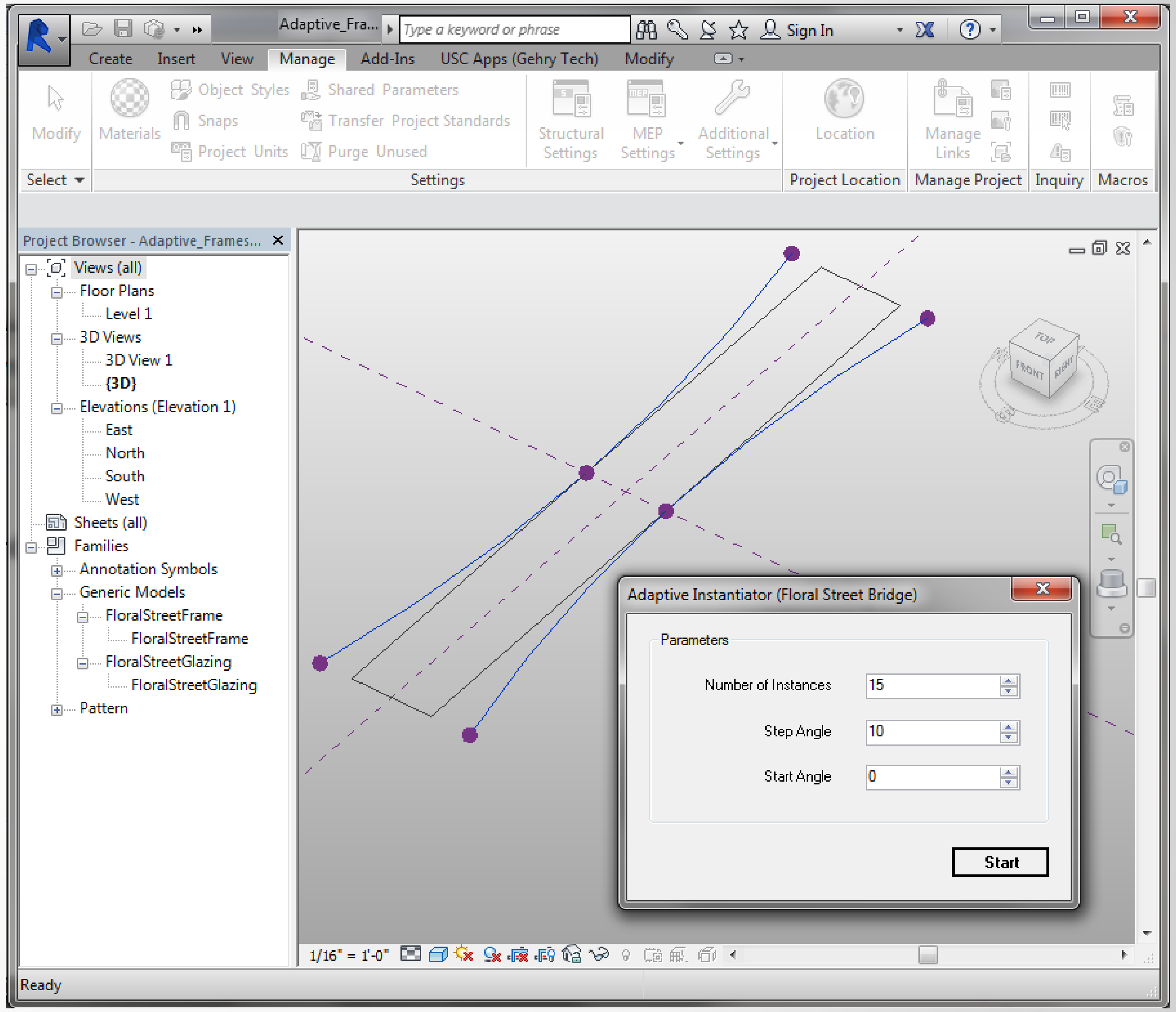
```
Clearing workflow...
00:00:00.0004120 elapsed for loading xml.
00:00:00.2295588 ellapsed for loading nodes.
00:00:00.0164259 ellapsed for loading connect
00:00:00.0039138 ellapsed for loading notes.
00:00:01.9485846 ellapsed for loading workspace.
```

Dynamo Package Manager
(Simple, quick and available anywhere)

Know Your Audience (User Experience)



```
Clearing workflow...
00:00:00.0004120 elapsed for loading xml.
00:00:00.2295588 ellapsed for loading nodes.
00:00:00.0164259 ellapsed for loading connectors.
00:00:00.0039138 ellapsed for loading notes.
00:00:01.9485846 ellapsed for loading workspace.
```



Adaptive Instantiator (Floral Street Bridge)

Parameters

Number of Instances	15
Step Angle	10
Start Angle	0

Start

Visual scripts for quick experimentation.
(For those who are more technically adventurous.)

A designed user interface with the primary inputs.
(For those who say, “Just give me a button to push.”)

Thank you!

All the materials are available on the AU website…

and

www.nodelete.org/posts/automation-prototyping

(Please leave any feedback or questions in the comments.)

Session Feedback

- Via the Survey Stations, email or mobile device
- AU 2014 passes given out each day!
- Best to do it right after the session
- Instructors see results in real-time

AUTODESK®

Students, educators, and *schools* now have

FREE access to Autodesk design software & apps.

Download at www.autodesk.com/education