CP5229: Seeing Data and More: The Analysis Visualization Framework in the Autodesk® Revit API

Matt Mason – IMAGINiT Technologies

CP5229

The analysis visualization framework (AVF) is a fantastic new part of the Revit API. Its primary purpose is to be able to visualize analysis data inside of a Revit project, but you can use it for so much more than that! You do not need to be an analysis vendor to have this be useful—you just need to want to be able to show information graphically in Revit. For this class, it is helpful to have basic knowledge of the Revit API.

Learning Objectives

At the end of this class, you will be able to:

- Understand what the AVF is and what it can do
- Understand the kinds of visualizations you can make in Revit
- Create temporary geometric elements to help with visualizing
- Create a variety of data visualizations
- Understand the limitations of AVF

About the Speaker

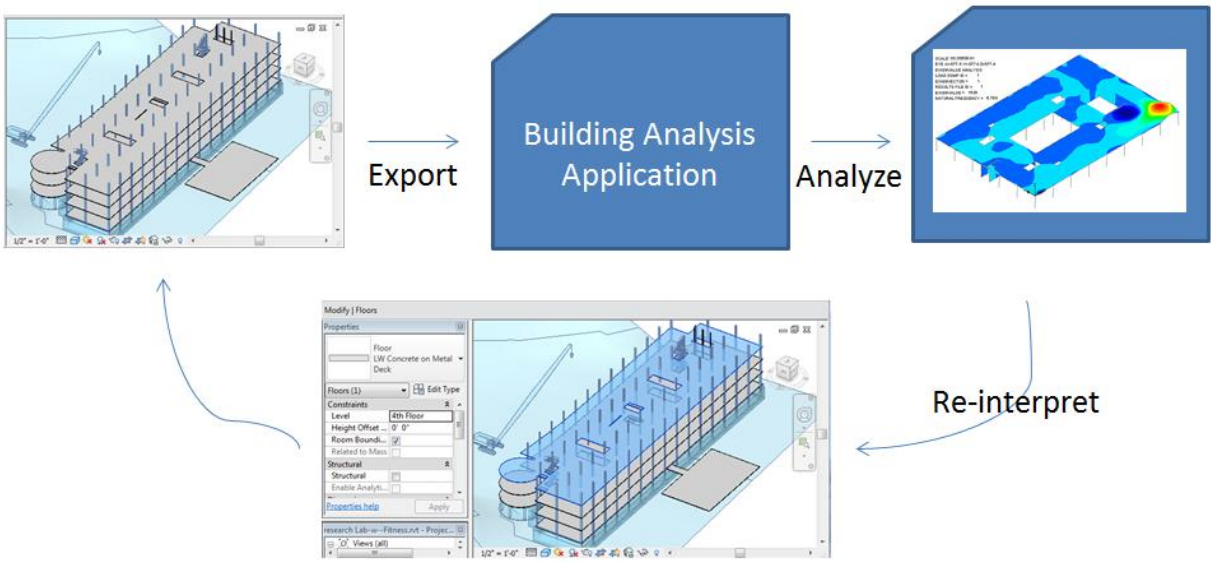
Matt is the director of software development at IMAGINiT Technologies. He has been working on software development in the CAD and PDM world for 20 years. He has worked on both packaged software and company-specific software in manufacturing, facilities management, and architecture firms. Matt holds a BSE in aerospace engineering from University of Michigan.

E-mail: mmason@rand.com

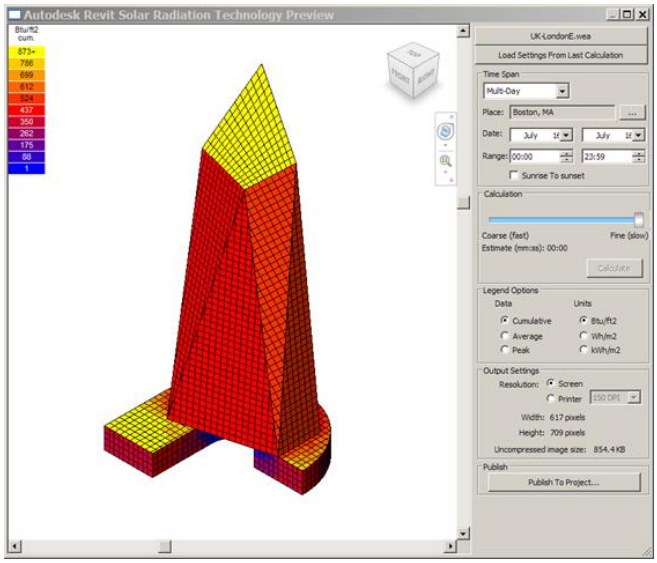
What is the Analysis Visualization Framework (AVF)?

The Analysis Visualization Framework is a part of the Revit API devoted to being able to show the results of technical analysis inside of the Revit environment. It was introduced in Revit 2011 and enhanced in 2012. The framework came about because the traditional workflow of looking at analysis results (and perhaps making changes based on them) was not very good.

Traditional Workflow:



New Workflow:



But what is the AVF, practically?

Practically, the AVF is a series of APIs which enable you to:

- Draw color graphics on the screen in a Revit Project
- Graphics are of different styles, depending on what you choose and the data you're trying to show.
- Graphics are NOT residents of the database – so they don't add to the size of the Revit model (and they don't persist – if you want them again later you need to re-make them).

What types of graphics can be created?

There are four types of analysis display styles that can be used:

- Colored Surface
- Markers (w/ optional text values)
- Diagram (w/ optional text values) *
- Vectors (w/ optional text values) *

* only available in Revit 2012 products

What are normal uses?

Normal uses of the AVF include visualizing data like:

- Lighting
- Energy
- Structural analysis
- Airflow
- Temperature

What about unusual uses?

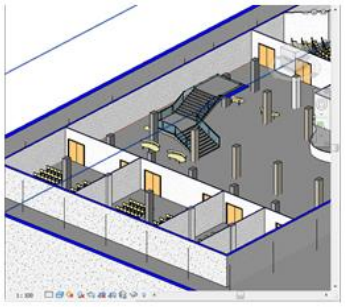
Much of this presentation will revolve around unusual uses for the AVF (since most of us do not work for analysis software companies). Ideas for this include:

- Point Cloud rendering
- Point Cloud deviation from model
- Visualizing Geometry
- Temporary Vectors (curvature, grids, etc.)
- And more…

Technical Overview (Inside Revit)

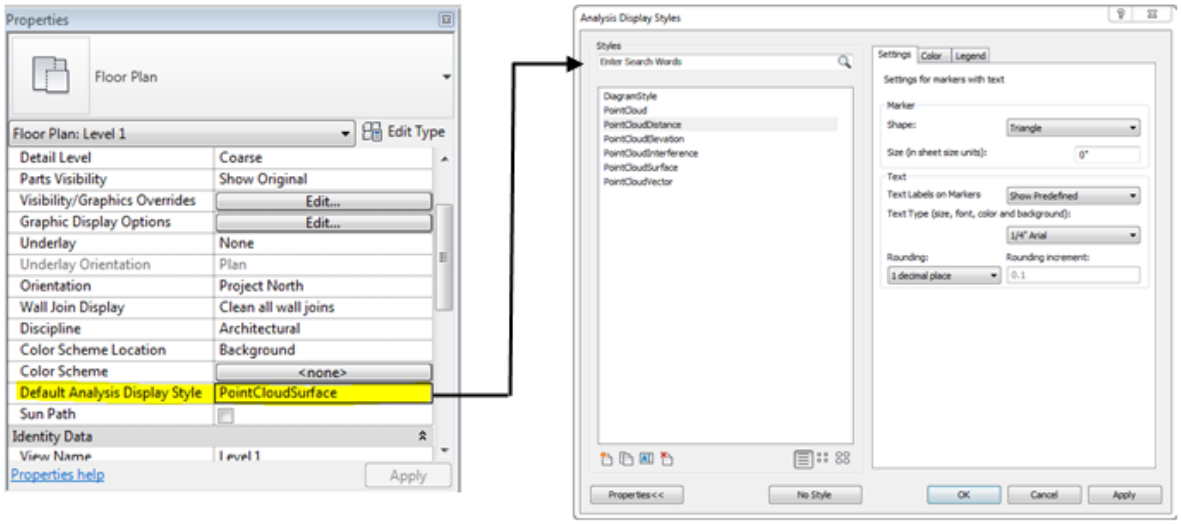
Inside Revit, there are a variety of ways that you interact with AVF as a user.

Revit Model



Specific View

Each view has a property for the default analysis display style. This controls how the data is displayed within the view. The Analysis Display Styles are elements within the model that can be applied to any view.



Each display style contains a number of different types of data, including:

- Settings (different for each type of style)
- Color-value mapping
- Legend settings

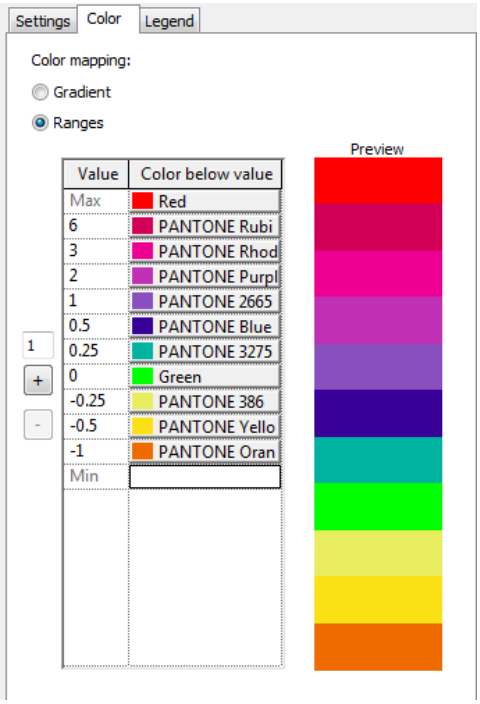
Notes on Settings

There are a few items to note about the Settings tab, depending on the type:

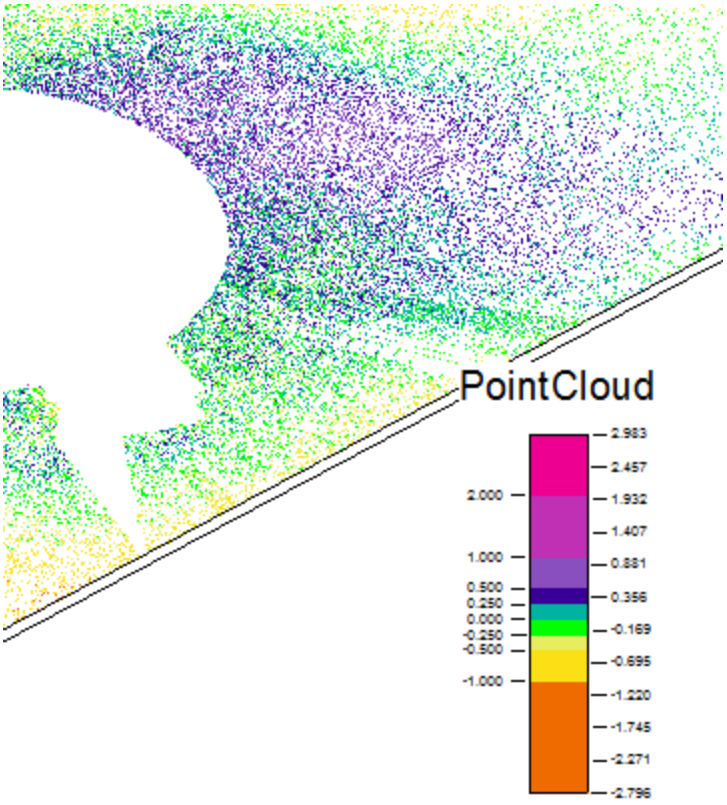
- For Marker type:
  - Choose an appropriate shape for the marker (circle, triangle, square)- depending on how many points this can be an important choice.
  - The marker size is in Sheet Size units – HOWEVER, you can specify 0 for the size – and it should reduce to just a pixel.
- For types with text:
  - The text is optional
  - The text uses one of your standard text sizes (makes your view scale important in 3D)

Setting the Color

The color tab describes how the VALUES from your data will be changed into colors on the screen.



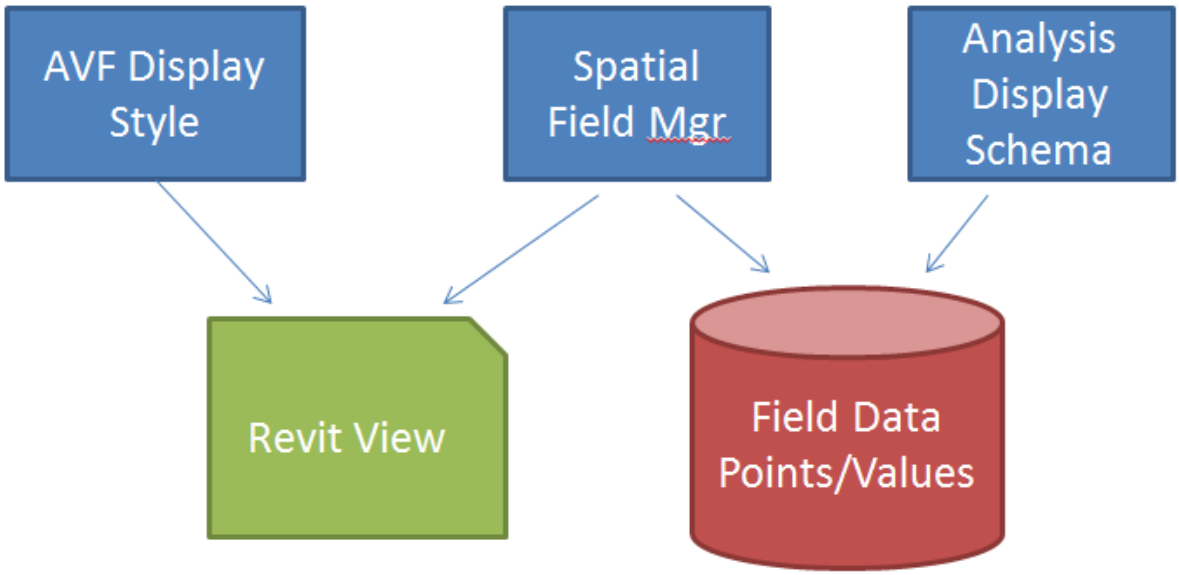
For example, on this style, we wanted to show different values depending on how the point cloud deviated from the model. We want points that are very close to zero to be green, with colors that progress more towards red/orange as they deviate further.


PointCloud
2.983
2.457
2.000
1.932
1.407
1.000
0.881
0.500
0.356
0.250
0.000
-0.169
-0.250
-0.500
-0.695
-1.000
-1.220
-1.745
-2.271
-2.796

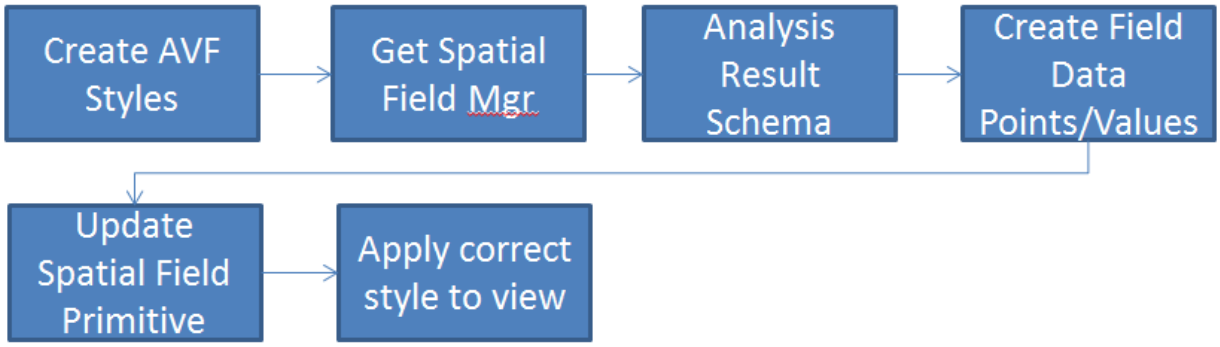
Technical Overview (API)

The following should be enough to get you started with programming the AVF in the Revit API:

First, you will need to understand the fundamental building blocks of AVF:



Then you will need to understand the typical workflow of using the AVF:



Code Fragment #1: Getting Started

// Get the Spatial Field Manager from the current view, or create if it needs it.
SpatialFieldManager sfm = SpatialFieldManager.GetSpatialFieldManager(doc.ActiveView);
 	
// create it on the active view, with one-dimension data.

```
if (sfm == null) sfm = SpatialFieldManager.CreateSpatialFieldManager(doc.ActiveView, 1);

// deal with looking up the results schema, if it exists – and create if not.
if ( _schemaId != -1)
{
        IList<int> results = sfm.GetRegisteredResults();

        if ( results.Contains(_schemaId) == false ) _schemaId = -1;

        if (_schemaId == -1)
        {
           AnalysisResultSchema resultSchema1 =
                new AnalysisResultSchema("ShowPoints", "Data Points");

           _schemaId = sfm.RegisterResult(resultSchema1);
        }
}

return sfm;
```

Code Fragment #2: Simple Point/Values

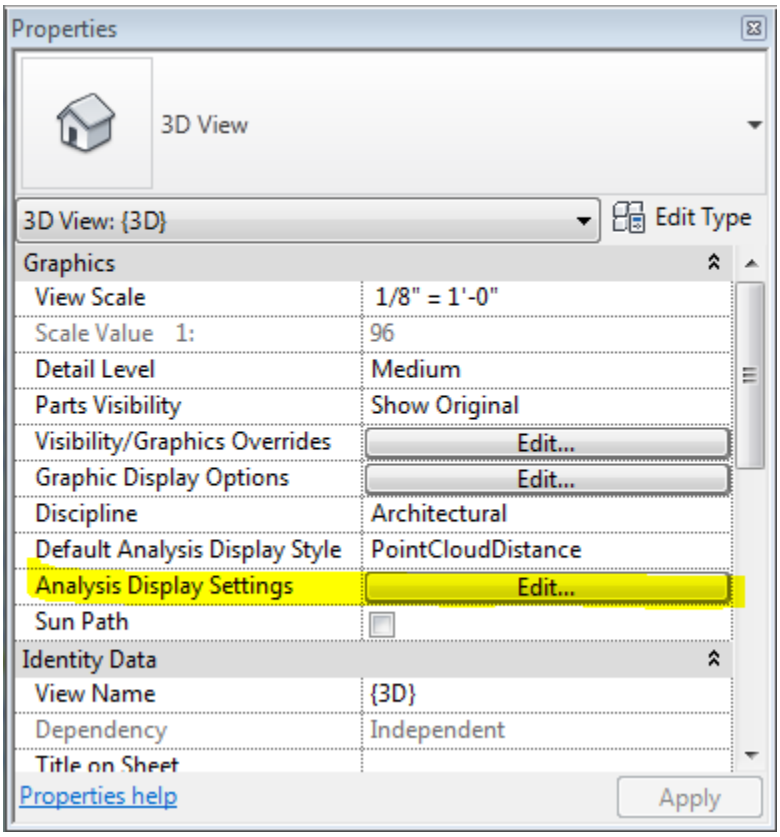
```
public void ShowValues(Document doc, List<XYZ> points, List<double> values)
{

        // take care of Spatial Field Manager / AnalysisResultSchema, etc.
        SpatialFieldManager sfm = getSpatialFieldManager(doc);

        // now we need to populate the FieldDomainPointsByXYZ and values

        FieldDomainPointsByXYZ xyzs = new FieldDomainPointsByXYZ(points);
        IList<ValueAtPoint> fieldValsAtPoint = new List<ValueAtPoint>();

        // values could be multi-dimensional, so it's a little complex to build
        foreach( double val in values )
        {
           List<double> valueList = new List<double>();
           valueList.Add(val);
           fieldValsAtPoint.Add( new ValueAtPoint( valueList ) );
        }

        FieldValues fieldVals = new FieldValues(fieldValsAtPoint);

        // add and update the spatial field primitive

        int idx = sfm.AddSpatialFieldPrimitive();
        sfm.UpdateSpatialFieldPrimitive(idx, xyzs, fieldVals, _schemaId);

    }
```

Code Fragment #3: Updating the Default View Style

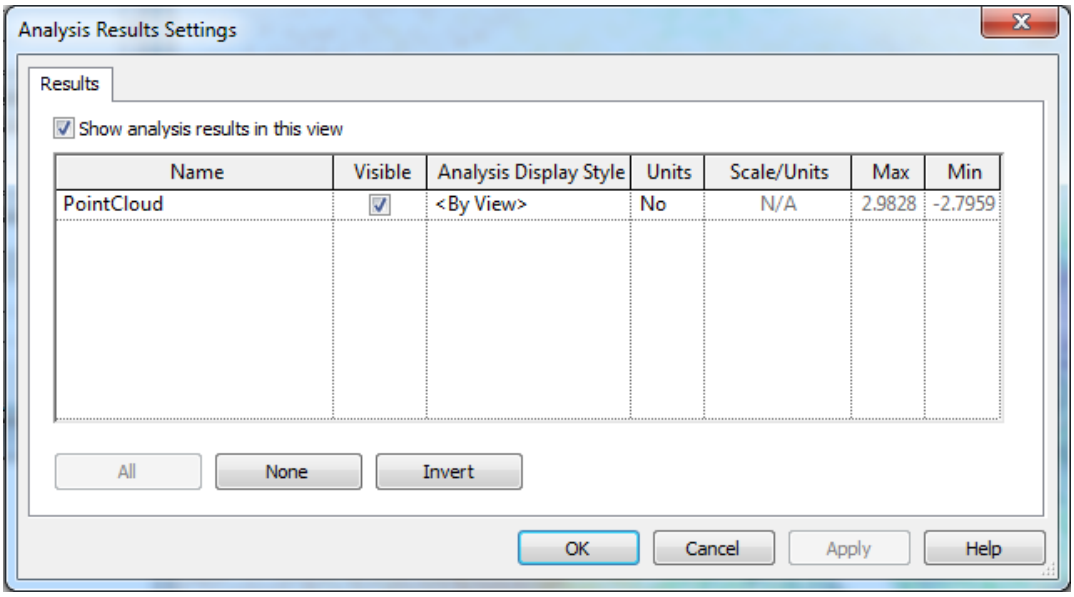
```
private void updateViewAVFStyle(View v, string stylename)
{
  //does the current view have an analysis style?
  Parameter avf = v.get_Parameter(BuiltInParameter.VIEW_ANALYSIS_DISPLAY_STYLE);
  Document doc = v.Document;

  if (avf != null)
  {

     ElementId pc = AnalysisDisplayStyle.FindByName(doc, stylename);
```

```
    if (pc.IntegerValue > 0)
    {
        bool success = avf.Set(pc);
    }
  }
}
```

After data has been created in a specific view, you will see the Analysis Display Settings property in the dialog.



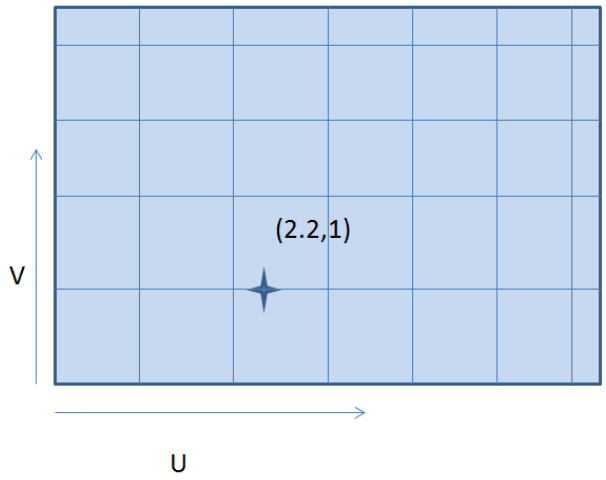
This enables you to see the one or more schemas of data in the view.



Working with surface data

When your data is “surface-oriented”, there are two important differences:

- You need to associate it to a particular surface
- You put it in surface U,V coordinates instead of X,Y,Z



The following sample shows an example of showing different values at different U,V locations on a surface.

Code Fragment #4: Simple Surface Point/Values

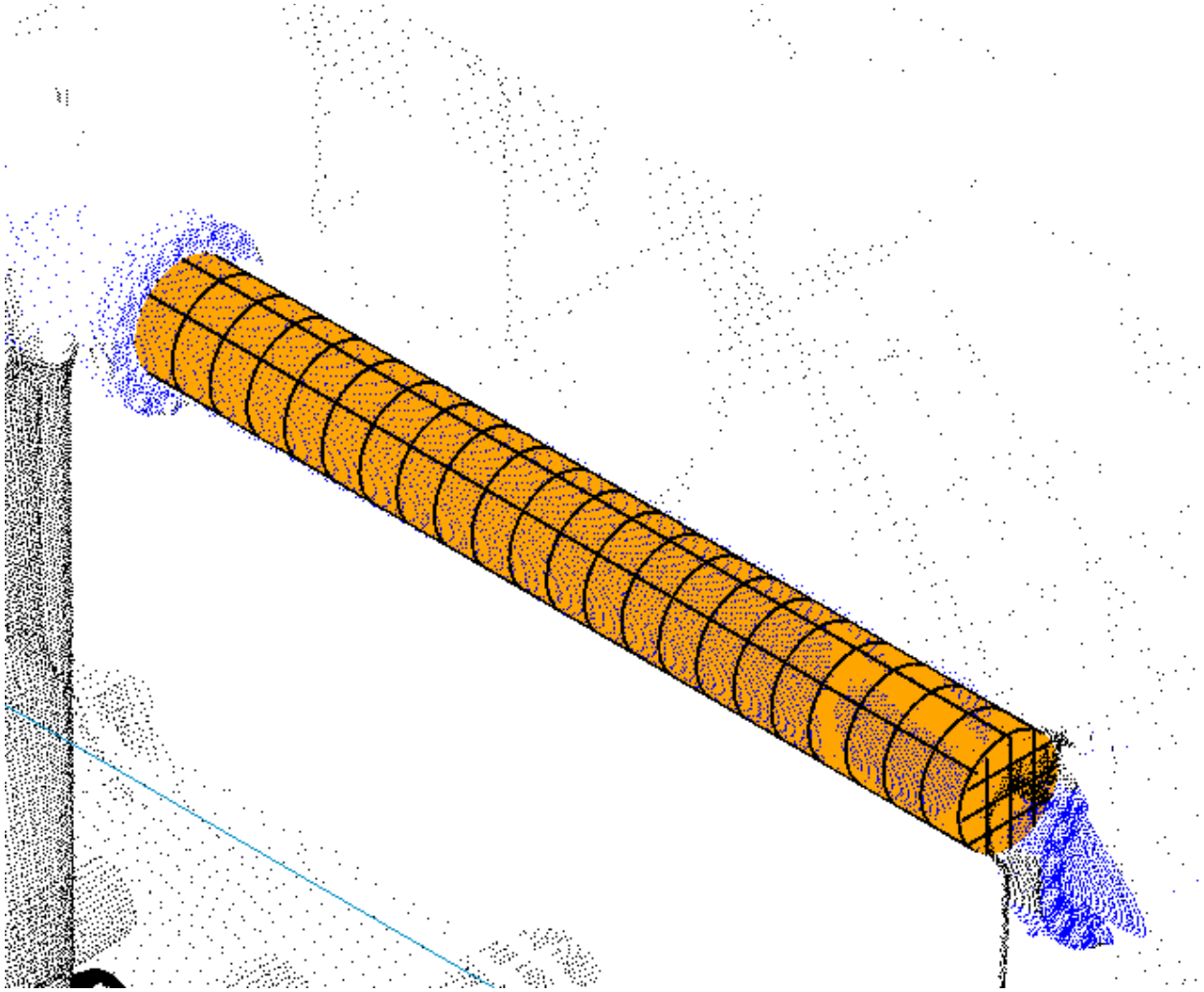
```
public void ShowValues(Document doc, Face f, List<UV> points, List<double> values)
{

        // take care of Spatial Field Manager / AnalysisResultSchema, etc.
        SpatialFieldManager sfm = getSpatialFieldManager(doc);

        // now we need to populate the FieldDomainPointsByUV and values

        FieldDomainPointsByUV pnts = new FieldDomainPointsByUV(points);
        IList<ValueAtPoint> fieldValsAtPoint = new List<ValueAtPoint>();

        // values could be multi-dimensional, so it's a little complex to build
        foreach( double val in values )
        {
           List<double> valueList = new List<double>();
           valueList.Add(val);
           fieldValsAtPoint.Add( new ValueAtPoint( valueList ) );
        }
```

```
        FieldValues fieldVals = new FieldValues(fieldValsAtPoint);

        // add and update the spatial field primitive

        int idx = sfm.AddSpatialFieldPrimitive(face, Transform.Identity);

        sfm.UpdateSpatialFieldPrimitive(idx, xyzs, fieldVals, _schemaId);

    }
```

What if you don't have a surface? The GeometryCreationUtilities

There are times when you want to show AVF data in a model – but you don't actually have a piece of Revit Geometry to reference with AVF. This is where Transient/Imaginary geometry comes into play. This is an area of the API implemented by the "GeometryCreationUtilities" class, where you can create solid geometry which is not stored in the Revit project – it is just temporary, in memory, geometry.

The kinds of solids that can be created are:	Solids created in this way can be used for:
- Blend - Extrusion - Revolved - Sweep - Swept Blend	- AVF Faces - Find Elements intersecting a solid - Boolean operations - Geometry operations (i.e. Face.Project(), Face.Intersect(), etc).



The picture above, taken from IMAGINiT Scan To BIM, shows a cylinder created to represent the geometry which was recognized from a point cloud.

Code Fragment #5: Creation of a cylinder

```
public Solid CreateCylinder(Application app, XYZ origin, XYZ vector, double radius, double
height)

{

   Plane p = app.Create.NewPlane(vector, origin);

   // need to create this as two arcs rather than one!
  Curve circle1 = app.Create.NewArc(p, radius, 0, Math.PI );
  Curve circle2 = app.Create.NewArc(p, radius, Math.PI, Math.PI * 2.0);
```

```
  CurveLoop profile = CurveLoop.Create(new List<Curve>(new Curve[2] { circle1, circle2 }));
  List<CurveLoop> loops = new List<CurveLoop>(new CurveLoop[1] { profile });
  Solid cyl = GeometryCreationUtilities.CreateExtrusionGeometry(loops, vector, height);

  return cyl;
}
```

Once you have the created solid, you can access the faces of the solid to use for AVF. Common use cases include:

- Highlighting of a single surface (find the nearest face to the location you want)
- Highlight all of the surfaces of the solid

Code Fragment #6: Simple Face highlight (one color)

```
public void ShowValues(Document doc, Face f, double value)
{

        // take care of Spatial Field Manager / AnalysisResultSchema, etc.
        SpatialFieldManager sfm = getSpatialFieldManager(doc);

        // now we need to populate the FieldDomainPointsByUV and value
        // we will use just a single minimum bounding box point

        List<UV> points = new List<UV>();
        BoundingBoxUV box = f.GetBoundingBox();
        points.Add( box.Min );
        FieldDomainPointsByUV pnts = new FieldDomainPointsByUV(points);
        IList<ValueAtPoint> fieldValsAtPoint = new List<ValueAtPoint>();
        List<double> valueList = new List<double>();

        valueList.Add( value );
        fieldValsAtPoint.Add( new ValueAtPoint( valueList ) );

        FieldValues fieldVals = new FieldValues(fieldValsAtPoint);

        // add and update the spatial field primitive

        int idx = sfm.AddSpatialFieldPrimitive(face, Transform.Identity);

        sfm.UpdateSpatialFieldPrimitive(idx, xyzs, fieldVals, _schemaId);

    }
```

Advanced Topics

There are several advanced topics that may be of interest with the AVF.

Multiple Values in AVF

AVF includes the capability to have multiple values at each point. In the case of analysis, this might be used to calculate the daylight value for each of the 365 days of the year. Values can represent different values for each scenario.

Point	Value 0	Value 1	Value 2	Value 3	Value 4
(0,0)	1.1	1.2	1.3	1.2	1.1
(1,1)	1.4	1.5	1.6	1.5	1.5
(1,2)	2.1	2.0	1.9	1.8	1.6
(2,2)	1.9	2.1	2.3	2.25	2.15

Setting multiple values for a single point is as simple as adding each value to the Value List (as seen in the samples above).

Setting Units for your Values

Units can be an important part of presenting data via AVF. In addition, there are cases when you may want to present the same data in multiple units (such as reporting in Inches, CM, and MM). These options are defined in the schema.

Code Fragment #7: Simple Set Units

```
AnalysisResultSchema resultSchema1 =
                    new AnalysisResultSchema("Deviation", "Deviation of data from geometry");

List<string> unitNames = new List<string>(new string[1]{“Inches”});
List<double> unitFactors = new List<double>(new double[1]{ 1.0 } );
resultSchema1.SetUnits( unitNames, unitFactors );
```

In the case where units are a simple multiple (for example, 2.54 centimeters to the inch) you can have multiple units associated with the schema.

Code Fragment #8: Multiple Set Units

```
AnalysisResultSchema resultSchema1 =
                    new AnalysisResultSchema("Deviation", "Deviation of data from geometry");

// make it display values in either Inches or CM

List<string> unitNames = new List<string>(new string[2]{“Inches”,”CM”});

List<double> unitFactors = new List<double>(new double[2]{ 1.0, 2.54 } );
resultSchema1.SetUnits( unitNames, unitFactors );
```

Limitations with AVF

There are a variety of limitations that are worth understanding with AVF:

Number of points

The maximum number of points that can be used in a single view is in the neighborhood of 800,000. This will be fine for many uses, but it is no longer suitable for point clouds. In 2010 it was more like 3-4 million, but the addition of the schemas and other complexities has made it “heavier” and less able to support very large numbers.

Graphic load per point

You are now a graphics programmer! In an early version of Scan To BIM, we were using circles for point markers to represent the point cloud. One of the developers noted that there is no such thing as circles in computer graphics – each circle point is represented by some number of triangles:

By changing to triangular points, we were able to get 10x as many points without graphic performance issues.

Side Note: If you set the marker size to “0”, Revit will actually give you a 1-pixel size point at any scale… giving you the maximum number of points.

Where can you use them?

Unfortunately, you cannot use Analysis Display Styles in the family environment.

Printing AVF Data

Since you've got this nice AVF data, it's only natural that you might want to print it… unfortunately, that's not really possible. It either will not print or will not print nicely with the Revit print function. However, you CAN:

- PrintScreen
- Right click the view and export to file
- Right click the view and Save to Project As Rendering Image
  (You can even insert the rendered image into a sheet, to make it formal looking!)

Conclusions

I hope that you have found this session informative. The AVF is a powerful tool – and I have tried to make it clear that it is NOT just a tool for makers of analysis software. It is a powerful tool for any advanced Revit API user.