Twice Baked: Creating Your Own Adaptive Components and Panels with Autodesk® Revit®

Robert Manna – Stantec
Zachary Kron - Autodesk

AB4391-L (Session 1 & 2) This hands-on lab builds on the topics covered in last year’s “Au Bon Panel: Baking Your Own Adaptive Components and Panels with Autodesk Revit Architecture.” Revit 2012 has introduced the ability to place Adaptive Components directly into projects, which opens up even more possibilities of what you can do with this powerful family modeling tool. We will address more complex topics, including using Hosted Points and Shape Handles, and will provide tips on how to make use of nested components. Our examples will include both the practical and perhaps not-so-practical, including how to use Adaptive Components as “design aids” (“computational” design on the fly). You will learn how to build “stuff” that behaves how you want it to and not how Revit thinks it should. So, come prepared, review the courses from 2010, and get ready for a whirlwind lab!

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

- Understand the concept of “computational design on the fly”
- Control the orientation and alignment of components
- Understand how to use adaptive components and panels to create well-quantified designs, not just sculptural elements
- Use nesting families to create complex adaptive components and panels
About the Speakers

Robert works for Stantec in the Boston office. He has been a key team member on multiple projects, including one of the first Autodesk® Revit® pilot projects in 2006. He has also been an integral part of the team that strategized and planned Burt Hill's (now part of Stantec) transition from a 2D workflow to a BIM process. He has taught internally and helped develop the curriculum for training new users, as well as developing long term BIM goals and plans for the entire firm. His day-to-day activities include working on projects, helping other teams, and managing the development and implementation of new tools. He was a speaker at the RTC North America, has been a guest lecturer at the Boston Architectural College, and has presented at BIM events hosted by the AIA, ACEC, Autodesk and resellers. He has written two articles about Revit for the AUGI® AEC Edge magazine, and has written assessment questions for KnowledgeSmart. He maintains a personal blog dedicated to Revit and BIM.

http://Dorevit.blogspot.com
robert.manna@stantec.com

Zach Kron is an architectural designer and a quality assurance analyst for Autodesk. For the last five years, he has worked closely with Autodesk® Revit® Architecture product designers and developers creating and testing the rendering, modeling, and analysis tools. In addition to internal teaching and curriculum development at Autodesk, Zach has helped create and run workshops at MIT & ACADIA, and maintains a personal blog. Before joining Autodesk, Zach worked as a designer in several Boston-area architectural offices on projects ranging from furniture to bridges. He has more than fifteen years of professional experience in form-making and visualization.

http://buildz.blogspot.com
Introduction

What is this lab about?
Hopefully it is about opening up your eyes to new things that you can do with Adaptive Components, hopefully it is about getting you more comfortable with working with Adaptive Components, and hopefully it is about learning how to do some really interesting stuff in a part of Revit that just does not get much attention from many people. When we sat down to write this lab, we had all sorts of ideas, in the end to develop a 90 minute AU lab you need lots of structure, and a very confined set of goals. Therefore similar to last year much of the lab depends on “pre-baked” families to make it easier to move through the exercises and for people to catch up if something goes awry during the process.

This year’s course has three exercises, each one builds on the next and all follow the same pattern and structure. Each exercise increases the complexity both of what is being created in Revit and the theoretical approach to how the actions are being carried out as we dig deeper into the families we’re working with.

The key to this year’s lab is that we are using nested families to build a component in Revit that is “contextually” aware such that each panel instance in a divided surface can react individually to the context, without having to use the API. If you learn and understand the approach it opens up a variety of applications to using divided surfaces and panels. Remember that while our specific example (reacting to direction or where the sun comes from) is fairly simple, the reality is that the panels could be anything, and could be reacting to anything, it may just take some ingenuity thought and consideration to “leverage” the functionality. Our exercises take you up through a family and massing environment workflow, but the intelligence that is embedded in the families can be further exploited using standard project level tool such as schedules, shared parameters, and filters. Lastly, if you do throw in some simple scripting or API tools, then it would be very easy to make these “tools” even more powerful.

While the exercises will dwell on tracking the orientation of panel geometry, please keep in mind that these techniques can be used to embed all sorts of intelligence in your components. Weight, planar deformation, thermal conductivity, sightlines, and any number of other pieces of information can be parametrically defined in your families. Orientation is a generally useful piece of information, but is not the only one available.
Exercise One – Direction;  
*jumping in with both feet.*

The intent of this exercise is to understand how manipulating the adaptive points allows the family to understand a direction relative to the form it is placed on. The ability to understand direction will further allow us to develop intelligent geometry that responds to the direction it is facing.

1. Open the file *Forms.rfa* if it is not already open on your computer

2. In the Project Browser select the Curtain Panel family *Compass Panel* and right click and choose “Edit” to open the family (this is a Curtain Panel by Points based family).

3. Select the Adaptive Point 3, and chance its Orientation property to “Orthogonal on Placement”
   
   a. Click and drag the point up vertically.

   *This change is critical to allow the panel to understand what direction it is facing relative to the cardinal directions in the project.*

   By using “orthogonal on placement” this point will orient itself to match the coordinate system of the project, regardless of the direction the panel faces. Therefore allowing us to create a “compass point” that we can reference to further manipulate geometry in the family.
4. Click on the “Set” Work Plane button in the Work Plane Panel in the Modify Tab
   a. Click on the Front/Back (YZ) vertical plane of the third adaptive point to set it as the Work Plane.
   b. Click on the “Point” tool:
   c. Click on the third placement point to place it on top of the placement point. Escape out of the command
   d. With the Modify tool select the new point and change its Offset property to 20'-0" (you may need to use the Tab command to select the proper point).
   e. Check the property “Visible”; so that the point will show up when the panel is loaded into the Forms file.

5. Click on the “Set” Work Plane button in the Work Plane Panel.
   a. Click on the Top/Bottom (XY) horizontal plane of the third adaptive point to set it as the Work Plane.
   b. Click on the “Point” tool:
   c. Click on the third placement point to place it on top of the placement point. Escape out of the command
d. With the Modify tool select the new point and change its Offset property to -12'-0" (you may need to use the Tab command to select the proper point).

6. Select the Model Line tool, make sure that 3D Snapping is checked in the options bar and draw a triangle using the 3rd placement point and the two new points that were just created

   a. Select the model lines and click on “Create Form”.
   
   b. Choose to create a flat plane.
   
   c. Select the new form and in its properties change its Subcategory to “Compass”.

7. Click on the “Load Into Project” button and load the family into the Forms file.

   a. Choose to overwrite the existing family.

8. Select the divided surface(s) visible in the project and using the Type Selector change it to **Compass Panel**.

9. Select the elliptical form with the applied pattern family and rotate it with the rotate tool and note the changes to the line in the family.

---

1 This is a subcategory we added to the file for you in advance; a custom material is also assigned to the subcategory.
Exercise Two – Simple Geometry; exploiting relationships

In our first exercise we examined how each panel can understand something about direction in the project relative to the direction the panel faces in the project. Now we will take the process one step further and have geometry react or change based on that understanding of direction. In this case a family with a circular opening will have the opening size increase or decrease based upon the amount of angular change.

1. In the Project Browser, find the Generic Model family Exercise 2, right click and choose “Edit” to open the family.

   This family is an example of what we’re going to call a “Measuring” panel. Its job is to measure the amount of angular change, so that we can cause some geometry to change. In this lab we’re going to hook up the geometry, so we can see what will happen, in the exercise that follow’s we’ll take a look at how the measuring panel is actually built from the ground up.

2. In the family open the Project Browser to Generic Models
   a. Find the family named “Oculus Panel Simple” (this is an Adaptive Component).
   b. Click and drag the family into the active view.
   c. Starting with the front left point (Point 1) click to start placing.
   d. Move counter-clockwise to Point 2, then for the third point click on Point 5 and for the last point click on Point 6.
3. Select the family that was just placed.
   
   a. In the properties palette find the property “Modulation” and click on the button to link it to a parameter, link it to the existing parameter “Modulation”
   
   b. Click and drag the corner point of the pattern that is elevated from the pattern grid. Note how the diameter of the opening changes as the angle between the two Reference Lines changes.

4. Click on Load into Project and load the family into the file “Compass Panel” (make sure to un-check “Forms.rfa”)
5. Drag and drop the family “**Exercise 2**” from the project browser to place it.
   
   a. Start at the front left corner (Point 1) of the Panel to place the first point.
   
   b. Move counter-clockwise to the front right corner (Point 2)
   
   c. Then to the second point created in Exercise 1 (bottom of triangle)
   
   d. Next click on the first point created in Exercise 1 (tip of triangle)
   
   e. After that move to Point 3 (back right) and last Point 4 (back left)

6. Click on Load into Project and load the family into the “Forms” file.
   
   a. Choose to over-write the existing.
   
   b. Note how oculus adjusts itself according to the direction the panel is facing.
Reporting Parameters, Host elements, and Sequential Regeneration: or “what the hell does this mean?!?!?”

When putting together the Compass panel, we started with a relatively simple geometry:

Why is it you cannot simply put a reporting parameter on this angle to drive changes into a panel? The answer, as the error dialog opaquely indicates, is that only certain elements can be used to drive formulas. Host elements are almost always those undeletable elements that come with a template file when you first open it. These elements are the first thing that Revit looks at when placing and regenerating a family.

Regeneration in Revit is key to understanding how Reporting Parameters function. When you move a wall in Revit, all the other walls that are connected to it are affected. Joins need to be checked, some alignments might conflict with dimensional constraints, hosted elements are lost, etc. For each of these changes, the conflicts or multiple possible outcomes need to be arbitrated, and the only way that they can be decided is by priority: who moved first or who is the strongest constraint. Priority/Strength is determined by where the conflict happens in the hierarchical chain. It’s not who moved first, so much as who gets to move first.

These changes need to be propagated sequentially, NOT in parallel. Remember how frustrated you were the first time that you saw that Revit wasn’t multi-threaded? This is why most (not all) Revit operations only take over one processor, to keep different regenerations from bashing into one another by only happening one after another rather than all at once.

In the wall example, if you grab a point and move it, the point that is being moved seems like a sensible place to start regeneration from.
What about a placed family though, where does Revit start with that? Families have regeneration too and therefore they also have a starting point. All families have some number of built-in elements in their templates. These are the things that cannot be deleted and are generally referred to as "host" elements. In a column family, for instance, the most conspicuous hosts are Upper Ref Level and Lower Ref Level. After all, what is a column without a top and a bottom? The column family MUST have these elements and all regeneration starts with them.

Back to reporting parameters and the compass panel; imagine for a moment that there are two parameters set up as such in the adjacent diagram.

Where Height is driven by a value of parameter Angle. Angle is measuring between a Host reference line and a line by points that is hosted on an edge and a vertex of the extrusion.

It is impossible to set this relationship, as Revit would respond with:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height (default)</td>
<td>3' 6 57/128&quot;</td>
<td>= (Angle / 10°) * 1'</td>
</tr>
<tr>
<td>Angle (default)</td>
<td>35.371°</td>
<td>=</td>
</tr>
</tbody>
</table>

![Autodesk Project Vaari](image)
In this case, Revit can detect that Angle is driving Height, and that the result would create a conflict with Angle, and so the whole relationship is blocked. There are some early warning signs that you are creating a cyclical relationship that will never be finished if set in motion.

A 50 degree angle would drive the Extrusion to 5’.

The 5’ extrusion drives the angle down to 37 degrees.

And the cycle repeats until your form model breaks or your CPU fries.
Your propagation graph would look like this:

If this was executed, you would never stop regenerating the model and the final elements in the chain would not be reached. This is a clear problem and needs to be blocked.

With Reporting Parameters this conflicting regeneration becomes more difficult to detect and the possible problems a bit more subtle. Imagine there is a panel with a reporting parameter on the angle that drives the height of the surface.

Now the panel is loaded and placed in a divided surface. The “Host” reference line between points 3 and 4 is regenerated first, which may then proceed to regenerate the surface. Then (and this is a theoretical sequence) the reporting parameter is regenerated and drives a change through a formula and affects the surface.

The problem here is that the extrusion is regenerating twice, once for the “host” elements and then later for some other random regeneration that comes later on. Revit does not like this, and it saves you and the extrusion the hassle and bloodshed of finding this out later on by saying this:

A reporting parameter can be used in a formula only if its dimension references are all to host elements in the family.
The propagation graph for this attempted regeneration would look like this:

While this is not an endless loop, like the first instance, it does result in revisiting elements that have already been regenerated, which while conceptually possible, is against the rules of sequential regeneration. Revit creates a broad barrier to stop this from happening. You could argue that in MOST cases this would not create a problem, but Revit is conservative and tries to err on the side of safety (and not corrupting your models).

So the trick is to get the reporting parameter to be guaranteed an earlier spot in the chain of regeneration to absolutely avoid the possibility of looping. The only way to do this is have the reporting parameter directly attached to “host” geometry which will always be regenerated first.

It is possible to just put a reporting parameter on the Compass panel, and there is PLENTY of information that can be created using schedules, filters, and shared parameters. For now though, it is important to understand that to create and manipulate geometry and changes to formulas, then the reporting parameter needs to be at the front of the regeneration path to avoid any of Revit’s roadblocks (self-protection).

Exercise Three – Measuring; establishing relationships

This is the last (and most complicated) exercise. Here we will build a measuring panel from the ground up, now that we understand what the measuring panel “does”. This measuring panel will be able to measure change both in the compass directions and angular change vertically. Once the measuring panel is working, we will add some geometry and finally put the measuring panel into a compass panel and load it into our test forms.

1. In the “Forms” file, find the curtain panel family named “Exercise 3” and choose “Edit Family” to open the file.

   a. This is a slightly modified version of the Compass Panel from the previous exercises. There are two points for measuring angle, one points directly “south” to measure from North, South, East and West, the other points up to measure any vertical change. Points 2 & 4 have both been set to be “Orthogonal on Placement”.

   b. In the project browser of the Exercise Three family under Generic Families, find the family “EX3_Measure” right click and choose edit family.
c. This family is a new variant of our previous measuring panel, it uses the “Arrow” pattern so that there is a “box” for measuring each angle, similar to the one box in the Half Step that was used to measure one angle.

2. To start, choose the reference line tool and in the options bar check “3D Snapping”.

   a. Draw a line from the eighth adaptive point to the third adaptive point, and then create another from the third point to the sixth point.

   b. Now select the reference lines that define the rectangle that makes up the bottom of the “V” in the pattern and choose “Create Form”, choose to make a cube.

   c. Select the form and assign it to the subcategory “Not Visible”²

   d. In the properties palette uncheck the “Visible” property.

   e. Set the Negative Offset to 8'-0" and set the Positive Offset to 0'-0"

   The form just created will act as a “Rig” to define a Work Plane for creating dimensions in the next steps. Without a consistent Work Plane for the dimension to be created on, the

² This is a subcategory we added to the file for you in advance; a custom material is also assigned to the subcategory.
dimension would break and the family would fail. The Work Plane has to be able to flex with the family as it changes.

Making the form extruded in the negative direction makes it easier to work with the family, and does not affect how the dimensions work.

Making the form visible is also a good way to diagnose regeneration problems when panels break. By making elements visible (forms, reference lines, points, etc) you can see what Revit is trying to do, which might help determine where the problem is.

3. Click on the angular dimension tool; click on the “Set” button in the Work Plane Panel then select the back left face of the extrusion. Create the dimension between the two Reference Lines that make the outside perimeter (make sure to Tab to the Reference Lines and do not dimension to the extrusion).

   a. Select the dimension and in the options bar set the label to “rAngle”

   b. Repeat the same procedure for the back right face of the extrusion and assign the dimension to the “rAngle_UpDn” parameter.

   c. Click and drag points one and five up so that they are no longer on the pattern grid, and so that the new parameter values change.

4. Open the family types dialog box:

   For the parameter “Modulation” in the formula column enter the following formula (note a similar formula is already pre-populated for the ModulationUpDn):

   $$(\sin(rAngle) + 1) / 2$$
This formula is important as it takes the value of the angle and generates a numerical value using the Sine function, the plus one and divide by two normalizes the function between 0 and 1, rather than a straight Sine function which would result in a positive and negative values, which would be more difficult to use to drive geometry.

**Periodic Functions in Formulas or:** “thinking about this makes my head hurt, is it really important???”

If we’ve learned nothing else from Barbie, it is this: Math is hard. Hard it may be, but oh so useful and possibly even enjoyable for those who allow it to be.

Trigonometry is particularly useful in basic architecture (figuring out roof slopes for instance, check out Daryl Gregoire at RevitRocks\(^3\)), and more generally for finding angles and controlling things for which you only have partial information\(^4\).

But we want to take a closer look at the basic geometry behind these periodic functions, that is, Trigonometry describes cyclic changes; transformations that return to the same place **PERIODICALLY**\(^5\).

Examples of forms driven by simple periodic functions:

---


\(^3\) For instance, the sun is periodic….
Back to the Compass and Measuring panels: What is the point of the formula?

<table>
<thead>
<tr>
<th>Modulation (default)</th>
<th>0.277892</th>
</tr>
</thead>
</table>

The goal is to have an output for every position of the Compass family that is a useful and generalizable number. You could just use the raw output from the Measure family of 0° to 360°, but then there would be a gradient.

<table>
<thead>
<tr>
<th>Height (default)</th>
<th>14' 11 13/36&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (report)</td>
<td>148.843°</td>
</tr>
</tbody>
</table>

Using constant change through 360 degrees the result is not so useful as a sunshade in the Northern Hemisphere generally wants to be deep on the South, nothing in the North, and something in between on both the East and West ends. That is, the shape wants to Oscillate depending on the number of degrees off of a fixed compass point.

Another way to put this is to say we desire a relative output that runs from; zero (0) in the North to one half (.5) in the East, one (1) in the South, and back to one half (.5) in the West. The Sine function can mathematically return that exact result as it oscillates.

The catch though is that a basic sine wave oscillates from -1 to 1 over 0° to 360°. The problem is that multiplying a negative result (0 to -1) by length will result in some odd (or even broken) results. Therefore the Sine wave needs to be adjusted (offset up) to avoid negative results.

If two is added to the Sine formula the range is from (0 to 2) in lieu of (-1 to 1). The result is useful, but now any further math must be done in the context of a range of values from (0 to 2) if the number value is translated to percent that...
means 0% to 200%. Therefore the offset can be further refined to bring the returned values down to (0 to 1) by dividing by two; now the values can be directly equated to 0% through 100% when using the values to manipulate geometry.

Now, when the Measuring panel is loaded into the Compass panel:

Now you can design any new panel to take this generic input: South=1, North=0, East/West=.5. This value is then used to drive RELATIVE change in any panel set to accept the Modulation parameter.
5. In the project browser, under generic families, find the Adaptive Component “Frame”
   a. Click and drag on the family into the canvas (this family has four adaptive points).
   b. Snap the four points to the middle square (in order, counter-clockwise):
      i. Point 3
      ii. Point 6
      iii. Point 7
      iv. Point 8
   c. Select the family, in the properties palette link the two Modulation Parameters to their respective parameters in the family:
      i. Modulation -> Modulation
      ii. Modulation_UpDn -> Modulation_UpDn
6. Click and drag either or both points one and five to see how the Geometry changes based on the change in the angle(s).
   
   a. **Make sure** to leave the point “up” (above the pattern grid) when you are done flexing the family.

7. Click on “Load Into Project” and load the family into “**Exercise 3**”
   
   a. Choose to Overwrite Existing

8. In the Project Browser find “**EX3_Measure**” and click and drag it into the canvas.
   
   1) Starting at the **Horizontal Measuring Point** click to start placing the component.
   
   2) **Move to Point 4**
   
   3) Then move down to **Point 1**
4) Next go to **Point 2**

5) Then go up to the **Vertical Measure Point**.

6) Go back down to **Point 2**

7) Next go to **Point 3**

8) Finish on **Point 4**

9. Click on Load into Project and Load the Family into “Forms”
   
   a. Choose to Overwrite Existing
   
   b. Change the panel type on the Forms to “**Exercise 3**”
Conclusion

In these exercises we have explored three primary concepts:

- how to use built in functionality to create more intelligent - in this case “contextually aware” - components
- how to drive this intelligence back into the component to create geometric changes
- how to control and articulate that information into useful and standardized outputs.

While all of these concepts become tools available without any API or scripting, there is a need to understand both the underlying logic of how Revit works (in terms of regeneration and the operation of reporting parameters) and to make use of some basic mathematical operations to create meaningful outputs.

The next steps involve taking the geometry into a real project environment with an eye to analyzing costs, determining solar or view impacts, constructability, etc. Essential considerations for this are Shared Parameters, Filters, Schedules, and Shared versus Unshared Families.

Shared Parameters - If all of the parameters that we have been using are added as shared parameters to a project environment, they become available at the project level for other uses, such as; view filter/controls, schedules and calculated values. This is the first (and necessary) step to bringing the intelligence embedded in the families up to the project level.

Shared versus Unshared families - Depending on the specific application, the embedded nested components need to be shared or unshared so that Shared Parameters are available at the project level. For instance, if you are only going to use a single kind of orientation sensitive panel, you might want to make your individual Adaptive Component families Unshared so that they become subsumed by the Compass Panel for scheduling purposes. If you are going to use several kinds of Adaptive Component geometries inside of a Compass Panel, or you want to schedule orientation separately from geometry, you might want to have Shared Families to allow them to bubble up to be apparent in the project. There are a number of ways to express your information in the project and this property can be helpful!

Schedules - Once the intelligence is embedded in your panels, you can mine your model for useful information, such as what percentage of your glazing is facing North, or how many square feet of the really expensive South facing glass do you have.

Filters - Even if you only use reporting parameters that do not need to create any geometric or formulaic change in a family, you can extract visually explicit, readable information from your model with filters.
In this image, a Compass panel with a Shared Parameter for Up/Down and Left/Right orientation and no nested families is loaded into a divided surface in the project environment. View Filters identify panels that lie within a defined range and override the graphics in that view. Schedules create a tabular version of the same data.

Finally, orientation is not the only intelligence that can be embedded in family geometry. Using the same techniques, we have created families that know how big each of their pieces are, how planar or non-planar they are, what their thermal conductivity is, doors that know how thick their wall is, windows that know their sightlines, and so on. Once the information is in a parameter, it is up to you what you want to do with it. The key to creating these families is to break your problem down into pieces, find functionality that addresses those pieces, and re-assemble them into usable components.