PAUL AUBIN: Welcome everyone to Code Blocks not Required, Dynamo For the Rest of Us. My name is Paul F. Aubin. And I'm an author and consultant. I've written lots of Revit books and video training on lynda.com, now known as LinkedIn Learning. And I do live training and consulting services in Chicago and around the world to architectural firms.

So I have a few administrative items to take care of first. First of all, we recently started a user group in Chicago for Dynamo. We affectionately call it Chinamo. So if anyone in this room either is someone or knows someone that would like to present at Chinamo, please see me afterwards or sometime during the conference, because I'd love to talk to you about doing that. We're always looking for presenters.

Nothing to do with Dynamo, Revit, or anything to do with AU, on August 21, 2017, you want to be somewhere on that gray path. A total eclipse of the sun will go across the entire continental US. And it's a spectacle not to be missed. So I highly recommend that you find a way to get somewhere on that path. And go Cubbies.

I have posted the materials that I'm using today to my website. They're also in the app. So hopefully everybody was able to get the handout in the app. But I've also got the sample files that I'll be using as a zip file. So if you go to that URL, you're able to download all of the material that I'll be using. And there's some bonus in there as well. So feel free to do that either before or after the conference.

So what are we going to talk about today? We are going to do the obligatory hello world. So we'll start off with assuming that you know nothing about programming, which I only know a little bit more than nothing about programming. So hello world is still a good speed for me. Then we will look at some simple Revit examples, like placing individual families or placing multiple families. Then some slightly less simple examples, like processing data and working with adaptive component families. And that will take up our hour quite nicely. So that's our agenda for the afternoon.

This is the beginning level course. Show of hands, how many of you are already using Dynamo? You're in the wrong course. No, I'm kidding. I'm sure there's something that you can still pick up from it. But I am going to assume nothing. So the rest of you, nobody-- who has never used Dynamo before? You guys are in the right course. Great.

I'm focusing on Dynamo for Revit. There are other flavors of Dynamo. Dynamo is rapidly changing. So there's no way we could cover everything. And I'm going to be using Dynamo 1.2, which is the latest release and has some cool new features.

So why Code Blocks not Required? Because I get really irritated by code blocks. No seriously, most of the courses that I've been to, it doesn't take long, usually right after the introduction, and they're already pounding away in code. And I found that frustrating as a learner, because I was like, wait, what if I don't know the code? I thought the whole point was to use visual blocks and nodes and wires and everything. And so this has been a little bit of a frustration for me in my journey.

And that's where the idea for this course came from is there ought to be some way for me to do something useful in Dynamo without having to know all the programming language. Or at least that was kind of the point, I thought. So I said, we're going to do a course with no code. So there will be no code blocks in this session.

But make no mistake, this is still programming. So you do need to get your mind thinking the way a programmer thinks. So just because you won't be typing the code out yourself, if you don't think at least a little bit the way programmers think, it becomes very difficult for you to get anything useful out the other end, because you have to follow the logic streams and structure things a certain way.

So let us jump into the software then and do our first hello world example. So I'm going to start from nothing here. Use an architectural template in Revit just because. And it's just going to be empty screen, of course. And then if you're using 2017, you're going to find Dynamo on the Manage tab. If you're using a prior version, it's on the Add-ins tab.

So I'll launch that. And it's important to always launch Dynamo first. I just said that wrong. It's important to always launch Revit first, then Dynamo, because Dynamo tethers itself to the current session of Revit and the current open file, whether it's a project or a family file. So very important that you get that first.

Now I'm just working this blank file. Over here I can create a new Dynamo file, open an existing one. We call these graphs. So I'm going to create a new graph. And that is the Dynamo interface there. So you have-- they gave me this laser, but I suppose the folks in the recording won't see this. But you have menus across the top with some toolbars. You have your library over here. There's an execution mode down at the bottom and some navigation tools here. And then of course, all the action takes place in the middle of the screen.

So let me position my screen as best I can here. I'm going to snap that to one side and Revit over here to the other, because I'm only working on one screen. If you have two screens, put Dynamo on one. Put Revit on the other. And it's a little more comfortable to work.

So hello world. There's a couple of ways you can find the pre-written chunks of code. We call these nodes. And one way is to just simply browse the library. So I could go to my Core and my Input and find something called a string. So string is programmer speak for text.

Now I'm not a programmer. So I may get some of this stuff-- it may not be the official definition. But generally speaking, what we mean by a string is information that the computer will understand as plain old text. So if you put a number in the string box, it will treat it not like a one or two or three, it'll just treat it like a character, a piece of text. And of course, I could type in whatever I want here, like being quite as literal as hello world. And I'll just click away from that to accept it.

Now to prevent it from just doing something instantly, and we don't see any-- we miss it-- I'm going to change the execution mode from automatic to manual. And that gives me a Run button here. And that means it won't do anything until I tell it I'm ready. Let me just zoom out slightly there.

And the other way that you can work with your library and find these pre-baked bits of code is to search for them. So I'm going to type in the word watch. And that brings up this little thing here, which is almost like a monitor screen. So I want to watch what's coming out of some other part of my graph. And in this case, my graph is going to be painfully simple. These guys are called ports. So you have sometimes ports on both sides, like this one. You have one on the left. You have one on the right. This is an input port. This is an output port.

The string only has an output port. So you can't feed anything into the string. But you can take something out of it. So I'm going to take whatever is happening in the string. And I'm going to feed that into the watch. And you do that just by click click. And that's a wire. So thus the whole nodes and wires thing.

Now of course, the watch hasn't done anything yet, because I told it to go manual. So now when I click Run, yay. Hello world, OK, thank you very much everyone. Have a great AU. No, I'm just kidding. But it really is that simple.

Now I could select this thing and Copy it and Paste it. I just did Control-C, Control-V just like you'd expect. And maybe I want to remove world from there and remove hello from here. It's important to understand that only one wire can go into each input. So if I do what I just did, it replaces the existing wire that's there with the new wire.

But coming out of the outputs, that's not the case. You could actually have multiple outputs like so. So it's possible to go from one output to several inputs, but only in one into the input. Well so what would I do here if I really had this situation? I'm going to go back to my full library here. I just clicked the little x there to clear the search. And down here under Operators, we have a bunch of things that look like mathematical operators.

So I'm going to use one called the plus right there. It's quite literally got the plus sign there. And there's an x and a y. But those are just general variables. Now the other thing you want to get in the habit of doing when you bring in a node that you haven't used before is hovering over the input and output ports, particularly the input ports. Now I apologize for those of you in the back that that's so small. I can zoom the node itself. But I can't zoom the Tooltip.

But the Tooltip is saying that it will accept a var with some weird brackets after that. And that's just the programmer way of saying, you can put anything you want in here, which means that depending on what you feed in, the node will decide how to interpret it. So if I fed two numbers in, it would say OK, I'm going to add those together. So if I feed one into the top node and two into the bottom port, and then it would say your value is three.

But in this case, I could take hello and world. Let's reposition these things a little bit, just to make it neater, more tidy. And now when I run, it puts the two together. And of course, the one down there is just saying the one word, because we left it connected, which is, of course, another thing you can do.

So very exciting? But all of you are probably-- is everybody a Revit user? Most of you. Who's not a Revit user? I get one or two hands. That's kind of what I figured. So let's do something a little more geometry related, which will eventually lead us into something Revit related. So I'm just going to move this junk out of the way.

And what would be the equivalent of hello world for geometry? Well we've got a tree branch here. And I'll scroll down and find a point. And then you'll have several different nodes potentially that can do the same thing. So another good habit to get into is to hover over each one and look at the Tooltip, because it will explain to you a little bit more about what that note does. So in this case, the first two choices here is x, y or x, y, and z. And in this case, I'm going to keep it 2D. So I only care about x and y. So there's a point by coordinates.

Now another thing that you'll see is when you hover over the input, that one says double, which for the computer programmers in the room, double is a fancy word for number, basically. I know I'm wrong there. If I was a computer science person, I would argue with me. But we're not computer science people. We're Revit people. So double, this input is essentially looking for a number.

And then it says default value is 0, which means that I don't really have to do anything at all. Watch right here when I press Run. And you're going to see a little dot appear. You're going to see a dot appear over there in the Revit workspace as well. So the input automatically defaulted to 0 and 0. And that's fine. But if you want to change those, then you've got to come back up here to your inputs under Core and feed it some numbers.

So I'll do Copy and Paste again. And of course, if I did this, nothing would change. But now if I changed one of these numbers, that'll just simply move my point. That's exciting and all. Let's take all of this now, Copy, Paste. Set this one back to 0. Run it. Move that slightly.

Now by the way, if I want to navigate the 3D, there's a toggle up here. That temporarily grays out the graph. And now all your navigation is in the 3D window. So if I want to, I can use a combination of left and right clicks and wheel mousing to navigate the 3D view.

So I've created two points now. So of course, two points makes a line. So let's see what we've got here under Geometry. Scroll down and look for lines. And wouldn't you know, there just happens to be a note here that says creates a straight line between two input points. Well, we happen to have two input points, one right here and one right there. And when I run this, I will get a line between them.

Now at this point, I've created Dynamo geometry. It's not the same thing as Revit geometry, even though we're getting a little preview over there. I'm not going to show you this, but take my word for it. If I close Dynamo, and you refreshed the screen over there, all that would disappear. That's temporary geometry in the Revit screen.

So let's think about things that are similar to that in the Revit world, like-- oh I don't know--

maybe a wall. Isn't a wall essentially like a 3D version of a line? It's got two endpoints. And you can draw it on screen in a similar fashion. So if I scroll all the way to the bottom here, here's Revit. And what can we do in Revit? Bunch of stuff. But under Elements, there's a whole special category just for walls. And there's only two choices in there.

So you're choosing between whether you want to define the height of the wall or the levels of the wall, which is really just another way of defining the height so I'll do by curve and levels. Move this thing over here. And oops, I just threw another one of those programmer words at you, curve. Well, think back to math class back in high school. I know some of you might still be hungover from last night. And that, maybe, is painful to think about that.

But think back to math class. And a curve didn't have to be curvy right. A curve could also be a straight line. A straight line is just a curve that was constant, constant value or something. Not a math teacher, so forgive me if I'm not defining it correctly. But a straight line is a kind of curve. It's a special case, just like a circle is a special case of an ellipse mathematically speaking.

So this line right here is Dynamo geometry. And I can feed that into this C port, which is the curve port. And it will be happy. Now if I run this, I still don't have enough information to generate a wall, because I also need those other three ports filled in.

Now it's saying start level, end level, and wall type. So let me scroll up here, under Revit still, selection and we'll find everything we need right there. Here's levels. And I'm going to click it twice. You can do Copy and Paste again. I'll click it twice. That'll make two of those. And then here's wall types. It's got its own little node. That's handy. And all three of these nice little drop downs. And it's reading the current project that you have opened in Revit.

So my project, you saw me started from the architectural template. Those of you that you that have used the architectural template before know that it only has level one and level two. So that's why those are the only two choices there. Also if you're familiar with the architectural template, then this list of wall types probably looks familiar, at least for the folks in the front that can actually read it. And that list of wall types is coming directly from that project as well. I'll just use a generic eight inch.

And now drum roll please-- nothing. Well there is something, except it's over here. Did I actually click Run? It's over here. It's not over here. I didn't wire it up, dummy. Let's wire it up. I'm calling myself dummy. You guys were all looking at me. You were like, he didn't do something. OK, now let's try it.

Nothing happens here. But something happens here. Let me go to 3D on this. All my buttons are squished. There we go. Very exciting, I now have a wall. I don't have a wall here, because again, there's a difference between Dynamo geometry and Revit geometry. And one can connect to the other. But they're not exactly the same.

Now there is a node that I can use that will create geometry here to match that. It's expensive, computationally speaking. So if you don't need to see the geometry in the Dynamo graph, don't use the-- it's like element.geometry. Don't use that node, because you'll wait. It'll churn away while it's generating all this 3D geometry in there. So if you don't need to see it, you can dispense with it, which is what I've done here.

Now what we've just wired up here is actually going to be reused in one of my future examples. So except for the hello world stuff at the top there, but all of this we're going to see again. So that will be coming in just a few moments. But this is essentially what I mean by hello world, both in and out of Revit.

So just to refresh a few of these things. Nodes are pre-written blocks of code. You connect them with wires. I forgot to say this explicitly. But you probably noticed that information flows left to right. That's kind of important when you're planning out your graph to think that way. Inputs can only receive one wire. But you can have multiple wires coming out of the outputs. And so that's what we just saw right there.

So let me close this. And I'm actually going to just close the graph here. I'm not going to save it, because I have another version of it pre-baked already. And I'll close this as well. And this time, I'm going to open up a really simple file. And it is called place family start. And it has two model lines.

So we've already talked about curve being the basis for some of this stuff. So in this case, I've sketched in a few model lines. Now this one is curvy. And this one is straight. And what I want to do is place families now. So I'm going to open up a graph and load my place families starting file here.

Now right here it says, Open in Manual Execution Mode. And that's checked. I'm going to do that. It's a really good idea when you open a graph from someone else, or even from yourself that you created a long time ago, that you open it manually, just because you may not want it to run instantly the minute you open the file. So that means it gives you a chance to look at it and see what's going on.

I'm not going to have time to share with you all the ins and outs of the way Revit behaves, because this is really more focused on the Dynamo. But Dynamo can't do anything that Revit doesn't allow. So you're not going to create something in Dynamo that you couldn't create in Revit. Dynamo just talks to Revit through the API and does things that Revit can already do, sometimes way more efficiently than you and I could do individually.

So the reason I'm saying that is there's a difference between system families that we just created, the wall, and what we're about to create, a component family. And that's true in the Revit interface. And it's also true in Dynamo. So that means we're going to have to approach it a little differently and choose different nodes, because Dynamo understands that there's a difference between these things in Revit. And therefore you can't-- it's not like one size fits all kind of thing.

So what do I have here? Just to get me started, I did some of the basics. This guy is called Select Model Element. And it's got a selection button. And if I click it, that actually puts me in selection mode here in the Revit side of the screen. And then I can highlight something and click on it. And that node goes from being a yellow color, which was unhappy, to being gray and happy. And then it says element. And there's a number there. That's the element ID of that model line. So that tells you that you've got a valid selection.

So what this also tells you is you don't have to build everything in Dynamo. A moment ago, that was a pretty slow and inefficient way to build a wall. Can you imagine if you had to build all your walls that way? Place every point. Place every line. Do all that stuff. You wouldn't want to do that. So one of the whole points of Dynamo is you can work back and forth between the Revit interface and the Dynamo interface to let each tool do what it does best.

So I've drawn the curve here manually. And now I'm going to use that curve as a basis for placing families. Now this weird looking note here, if I just fed the model element directly into the next step, it would fail, because it's Revit geometry, pure Revit geometry. And what this node in the middle does is negotiates between Revit geometry and Dynamo geometry, that same idea again, so in reverse.

So it's called element.curves. And what it's doing is it's looking at the object you selected and pulling the curves out of it. In this case, it's going to look exactly the same when I run it. And you're going to see a curvy thing there that matches that curve. So far so good. We now have the curve pulled out of Revit.

This node here is called Curve Point at Parameter. Show of hands, adaptive component people in the room? A couple of you. So for you folks, you've already heard this terminology before, this notion of at parameter. But for the rest of you, this is just a way of saying, I have some distance, some domain, a straight line in this case. It's a curvy straight line, but it's a line nonetheless, a linear path. Let's say it that way. I have a linear path.

And if we say one end of the path is 0, and the other end is 1, then we can define locations along that path by just using a decimal. And that's what they mean by at parameter. So at parameter is some decimal value between 0 and 1. So this is going to take this curve, and then a parameter, which is a double, and default value is 0, and it's going to put the first point initially right at the start of the curve. I could see it there and Dynamo. Put a little point previewed there in Revit.

But nothing's happened yet, because I haven't told it what I want to put there. But it figured out what that point is. Now if I use this number node that I happen to have sitting right here, and I feed in a different value-- let's do 0.25. Move this out of the way. Now the point moves to there. Let's try, maybe, 0.75. I think you get the idea. Run it. It jumps over here. Set that back to 0. So that gives us a little bit of flexibility on where along the curve we want this point to occur.

Now we just saw wall types. Family types is a little bit more generic. It's going to show you the list of component families that are in the current file. And in this case, I'm going to stick with a really simple single click family in our PC tree.

So I want to place a tree on that path. So I've chosen that family type from the list. And now over here, I've got Family Instance By Point, which sounds like what it is. It's going to take some family and place an instance of it at a point. RCBTree only needs a single click. We've got a point. It's currently at the end of the curvy line. So let's do it. All we've got to do is wire this bad boy up and see what we get.

So there's the family type. There is the point. Let's run it. And there is a tree. But of course, again, I did not do the element.geometry, because it would be expensive to shrink wrap that tree and show it in Dynamo. And I don't really need to see it in Dynamo. But there it is right there.

Now we also have another way that we could place this parameter. Down here I have something called the Number Slider. And a number slider just makes it a little more interactive. People love sliders. Let's be honest. This is fun. Now notice though that it's going from 0 to 100, which would mean that it would exceed my quote, unquote, "domain" of the 0 to 1 one that I'm looking for. So there's this little curvy thing right here. I can click that. And I can change the max value to 1.

So now I'm saying I want the slider only go between 0 and 1 and step by-- it defaults to 0.1. I'll leave that. And now it steps like that. Now this is going to be way more exciting if I wire this instead of number and change this to automatic, because now you'll see the tree move around. Ooh. That got a reaction out of somebody. So now we're starting to get to some of the things that make Dynamo special.

OK great. Well that's great for placing one tree and placing it along this windy path. Well, what if I wanted more than one tree? So let's look at that. I mentioned that you could only put one wire into one input. But I was very specific about saying one wire. I didn't say only one value. So what I really wanted to get to in this graph is two critical concepts in Dynamo-- lists and lacing. If you don't understand lists and you don't understand lacing, your Dynamo career is already over. So you need to understand these concepts.

And it's a long learning curve, by the way. So I'm still, everyday, learning more and more about lists and lacing. So let's start with lists. I want to create more than one tree. So what I'm going to use is a really simple out of the box node called a Range. So I'm going to come up here. Go to Core. Go to List. And right here, I've got some nodes that I can choose from. And I'll choose Range.

Range starts at a value of 0, ends at a value of 9, and it steps by 1, which means that if I do nothing to it and I run-- well, it already ran. Let's go back to manual. But it's already generated a list. And what you haven't shown you yet is you get these little something bubbles. If Zack is in the room, I can never remember what kind of bubbles they are. But they're bubbles, this little bubble. And it's telling me-- he's told me 100 times what kind of bubbles these are. I just have a mental block.

But if you hover over the little bubble, there's a push pin. Let me zoom in as close as I can on this. There's your range of values. Now for the non-programmers in the room, myself included, you might notice that it starts a little weird, because it starts at 0 and counts to 9. So programmers do not have five fingers. They have four fingers-- zero, one, two, three, and four. Just get used to it, because that's the way programming works. Remember this is programming. anyway, 0 through 9 for a total of 10 right there.

So what do I want to do with that? I can take this number node that I had over here. And I'm fine with starting at 0. But where do I want to end? This guy can only go between 0 and 1. So let's end that one. And then this slider, instead of feeding it there, I'm going to put it here. And then I'll take all this stuff and move it around. And I'm going to feed that into there.

So what is that going to generate? Well right now, I'm set to 0, which is going to make it fail. Let me try 0.2. Let me run it. Right here, I'm going to get only six values now. They go from 0 to 1 and space 0.2 instead of spacing 1 like they did before. And I get 0.2, 0.4, and so on. So that means that by changing the start and ends and all the other stuff in the sequence, then I can start to get numbers that are valid.

The thing is, I just ran it. And the tree didn't budge at all. What's going on with the tree? To explain that, I'm going to go into the second concept that's really important. So we do now have a list of values feeding into the parameter port. But it clearly looks like it's ignoring that.

So next we're going to talk about lacing. And to demonstrate lacing, under Revit, under Selection, I am going to grab this one, Select Model Elements with an "s." And it Looks almost exactly the same as the other one, except for the s, because this one, when you go into Select Mode, does a multiple selection. So the first one only let you pick one thing. The second one wants two or more things that you do with a crossing or some other window selection. So you can see I have two elements now. And I'll change my selection.

So if we think about the logic, what does that mean? Well, now I have a list actually feeding into both ports, don't I? Up there I've got a list of two elements. Down here I've got a list of six numbers. Two and six, let's run it. Was anybody expecting that? So again, this is where the logic and thinking like a programmer comes in, because you have to understand what it's actually doing here.

So this is lacing that you're witnessing right now. And there are three possibilities-- longest, shortest, cross product. And we're looking at shortest. So shortest lacing means that it's going to take your two lists, list A and list B. It's going to match them up with one another. It's going to take item one off list A and match it with item one off list B. Then it'll go to the next one. Item two on list A, match it to item two on list B. And because it's shortest, the shortest list wins. You got two matches. You're done. It throws away the rest. And we get what we see there on the left in Revit.

What if I switch this to longest lacing? And by the way, where and which node and where would I do that? Well, the one that I want to switch is actually right here, where I'm creating the points. And if you right click on the node, there's your lacing, shortest, longest, and cross-product. So let me switch to longest. Run it. Was anybody expecting that outcome?

So now what's happening? Item one to item one, item two to item two. The longer list now takes precedence. But because it doesn't have a corresponding match from three to six, it just takes the last item on the shorter list and just matches it again to all the other items. Therefore the last item on the shorter list was the straight line. All the trees go to the straight line. Everybody kind of with me here?

So cross-product looks for every possible match. So with cross-product, you get that. Now, I know it takes a little time for that to gel in and sink in. But again, trust me when I say that list management and lacing are two of the most important concepts in Dynamo. And hopefully that really simple example gives you some idea of how those concepts work.

So let me just summarize a few of those things before we move on to the next example here. Whoops, 1 point component families. So remember that Dynamo can't do anything that Revit can't do. So it's just using the API. Lists are very important as we just saw, as is lacing. You can feed a list into an input where you can only still do one wire and get lots of data coming in. And I haven't done that one yet. But I will shortly. You can end up with a list of lists. So stay tuned on that one. That bullet point shouldn't be there. And then lacing determines how the list will be compared or processed to one another. So very important concepts.

So let's switch back to Revit and Dynamo here. And we're done with this graph. And that was a component family graph. And I mentioned to you before that we were going to reuse some of that code from the first graph. And so what we're going to do now is compare a very similar exercise creating multiple instances of a system family as opposed to multiple instances of a component family.

So I'm going to open up a starter file, which is really just a copy of the imperial template again. And open up this one that I called wall warehouse, opening it in manual mode. Wait a minute now. What is all that stuff? So before anybody freaks out and says, oh, he's doing it just like those guys he criticizes. He's going right-- putting the pedal to the floor now.

I just put some documentation in here. Documentation in your graphs is very important, because you'll come back to your graph later and not remember what you did. And if you share your graph with other people, they won't have any idea what you did.

So there's two ways you can document your graph. Allow me to demonstrate. One is to go to Edit here and create a note, or do Control-W. And there is a little note right there. You double click it. And you type something in. And that is all these things are. I just typed in what I wanted that note to say. And that note just tells people a little bit about this part of the graph so that they understand what I did there.

The other documentation feature that I'm taking advantage of is a group. So notice when I click on that shaded region, all of this stuff is grouped together. So I'm going to un-group that with Control-U and just do it in reverse. Select all the stuff, Control-G. Now it's grouped. You can double click here. Give it a title. You can right click, change the color. You can right click again, change the font size. Whatever you want to do to, again, say these things all do something. And they're all related to one another. And here's how. And some people even come up with color schemes.

So what I've been doing with these graphs, the rest of the ones we're going to see, is grouping them numerically, just because those are the parts we're going to talk about in that order. So let's talk about this first chunk first. And there's one other concept I want to share with you before we do that.

I'm going to select this sequence node here, the first one that gets wired into from group one to group two. Right click it and choose something called Freeze. Now that will put a dash box around it, gray it out. And notice that everything downstream froze as well. So what this means is I can now click Run and-- let me just maximize this for just half a second here so we can zoom in better. And it runs just this first group of the graph.

So let's talk about this one first. You've already seen the dropdown-type nodes. This one is slightly different than the one I showed you before, where this one is, instead of asking for individual families, this one is asking for types. And what I chose here was wall types. So instead of looking at the individual objects, I'm saying, hey Revit, what are all the types in this project?

And then the next node just says do exactly that. Build a list of all those types. So it's going into Revit, finding all the wall types, and then making a list. And the watch node is just showing you that list. And again, for those of you that are familiar with the imperial template, this looks familiar. So here is a list of all the wall types in the imperial template, happens to be 27 down there at the bottom.

I cannot use 27. That's just for my information. I can't use it later in the graph. So that's what this node right here is for, list.count. It just says how many items are on that list? 27. Why do I care? Because this time, instead of doing arrange, we're going to do a sequence. And a sequence doesn't ask where you want to start and where you want to end. It says, where do you want to start? And how many do you want? How many do I want? That many.

But by doing it that way, by counting, it's parametric. So if somebody comes in and adds new wall types to the file, we could just run the graph again. And it will get a new count. And everything else downstream will work. If you put in a hard number, you have to manually maintain that. So here I'm letting Revit decide how many, based on how many wall types there are. Everybody clear so far?

So that's all I'm doing here. And I didn't feed anything in, because I was fine with it starting at the default of zero. So it starts at zero. It ends at this number. And I'm stepping by five with a slider. Why? Well, let's run this part of the graph and see. I'm going to freeze here. So we just, again, run it piece by piece. Let's put this back over here and run it.

What did that do? Well, it looks like the 3D preview is oriented slightly differently than the one in Revit. But you get the same idea. It just made a bunch of points, here, a sequence. And the five is just saying how far apart to space those points. So if I drag this slider, it'll just compress or stretch out the number of points. See the list there? So instead of doing one, two, three, four, I'm doing zero, five, ten, 15, 20. That's all it's doing. So that was the step value in that spacing.

So let's move on to number three here. This is the part of the code we've already seen, because we did this in the first exercise. A point by coordinates, another point by coordinates, I've set them apart by 15 right here. So if I run this, I get a-- whoops I went too far. I forget to freeze downstream. So we're going to see the whole thing run. But if I run this, I'm going to get a second batch of points there that are 15 units away.

And then you can see the levels here peeking in. That was the part where it's going to create walls from them. I'm sorry. I probably have element.geometry turned on. So we're waiting. I forgot to Freeze. My apologies. I was going to say, please don't crash. So it's always nerve racking when you see the little spinning wheel.

So what it did was it made the points, made the lines. This is just the part we did before. Then I fed that list of lines into the curve. So now instead of doing one line, I fed in 27. And then same start and end levels, one and two. And then the wall types, you see how there's a wire going back, back, back. Where is it going to? It's going back to this list.

So you can see what happened in Revit. It created one of every wall type. So instead of getting generic eight for all of them 27 times, I got one of every wall. And it's basically a wall warehouse. Some of you who are CAD or BIM managers probably have made these things manually. Well there you go. With a couple clicks, you can make one automatically.

So that's a little bit different. In a way, this is actually a lot simpler graph than the one we just did with the trees, because there was no lacing or any of that other stuff. But it's because it's system families. So really all I'm trying to do there is show you that there's a different approach when you go at this from a system family.

So now let me come back over here. Let's summarize a couple of those things. And I already did this one. So system families and component families treated a little differently. You can build the list of the components for multiple elements. And then of course, don't forget using groups and notes to document your graph just so you remember what you did the last time.

And we reuse a purpose-- I didn't. How did I do that? I actually went and I opened up the other graph, grabbed the nodes. Did Control-C. Then I came over and opened up this graph, did Control Paste, Control-V, and then positioned it. So that's what I mean by reusing portions of the previous graph. I didn't rebuild all of that. So if you remember, oh, I think I did this in another graph. You can copy and paste between them. That's all I did there. I just didn't show you that part.

So those are all-- in a way, everything we just did is still kind of hello world. Because even though maybe the wall warehouse has a little practical use for some of you, the BIM managers in the room, or possibly some folks might be saying, oh, I could put trees along a curve or something, those are simple examples.

So I wanted to leave you with an example that was a little bit more complicated and a little bit more, maybe, practical, but still not so complicated that it blasts us into the stratosphere, keeping in notion of what we're trying to do here, which is just introduce you to what Dynamo can do. So here's the example. We close this graph, not going to save. Close this file, not going to save. Open up my starter file. And it is this guy. And I'll talk about that in just a second.

And then I'm going to open up my starter graph, which is this guy in manual mode. And again, it's got groups to guide us along. So first of all, my thanks to my longtime client SCB in Chicago for providing me the family that's on this curtain wall and really the idea behind this example. So I had a discussion with my contact over there. And they were talking about how when they make these custom curtain panels, it's great and all. But then when they have different things going on in the facade of the building, you end up with these irregular shape wedges that don't work with curtain panels.

So for those of you that have never worked with custom curtain panels before, let me just tab in here and show you what this is. So if I tab into that curtain wall, select this guy, go to Edit Family, what they did here, design wise, is used sweeps and extrusions to create the geometry. So they're not using mullions. They're building it right into the curtain panel. And I already see some people nodding. And I know it's a pretty common thing to do among firms.

So that's great. And this works beautifully. It's very parametric. You can adjust the three zones there, are that. Works great, as long as everything stays a rectangle. But over here, not so much. So these are, if I tab in, system panels. And what I did was I just changed them to red, just so they would stand out more for you guys in the back.

But those, when you try and apply this curtain panel across the facade, Revit says, oh sorry, can't do all of these guys. You get the error down in the bottom. And it replaces them with system panels. It doesn't ask if it's OK. It just does it.

So some of you raised your hands that you've worked with adaptive components before. What's the solution? Well what I did was created this guy. And when you selected-- I don't know if you can see there. It's really small and gray. But one, two, three, four, this is a four point adaptive component. And I'll do Create Similar. And it works like this-- one, two, three, four. And you can see that that will flex the family to some irregular shape.

So my thought was, on all those weird oddball conditions, we'll just put an adaptive component there. Now if anybody's ever worked with an adaptive component before-- this particular file has 13 of those red zones-- I might be willing to place 13 without wanting to jump off a building. But if you had 1,300 of them, not so much. And it's very tricky. They're very finicky. So when you're placing them, sometimes you get to that last point and it says, sorry, can't do it. Sorry, can't do it. So it's a very frustrating process, to say the least.

So now we know what we're trying to solve. What I'm trying to solve is can I use Dynamo to go in and select all those red things and replace them with these adaptive components? And the answer is yes you can. And let me walk you through the graph.

So what I'm going to do here is, just to make it easier to see, I'm going to right click Select All Instances of the red panels and then just go to my light bulb here and hide those elements. They're still there. I'm just hiding them. And let's start looking at the graph now.

So my selection is based on the solid red panel. So I just said, I want to select all the system panels called solid red. So it's going to get those 13 panels that I just hit. Once it gets those 13 panels, we're going to make sure that we grab-- well, sorry, once I start with the panel name, then this one's going to get the 13 panels. So this one's saying all elements of that type. I've got a watch node here just to make sure. I just like to pepper those in so often, just to say, did I get the thing I thought I did. So I threw that in, not required.

And then this node right here called Curtain Panel Boundaries, this is actually going to trace the outline of those curtain panels that are in the selection and create something called a PolyCurve. A PolyCurve is very much like a polyline for the AutoCAD folks in the room. But is that something we can make in Revit? Can you make polylines in Revit? No, you can't, Paul. So that's going to be Dynamo geometry. So we're going to have to take those PolyCurves that I'm creating in Revit and do something with them.

So I'll talk about flatten when I run it. And then I'm going to start-- well let me do the first half here. So I've already pre-frozen some of the groups. So I think to explain these, it's going to make more sense if I run it.

You can see what happened on the Revit side. Even though the panels were hidden, it traced them. Over here in the Dynamo side, there's what it traced. So it doesn't care about the field in the middle, because we're not asking it's like those. And here you've got-- one of the handy things about this watch node is these little things are actually hyperlinks. And I hid the element. So that's why it's saying that. But if I hadn't hid the element, it would zoom right in on it. So if you're not sure which panel that is, you can actually click these little green buttons. Just don't hide them first.

And then what about flatten here? Well, this is the bullet point that I said, oh, I didn't tell you about that yet. So I'm going to show you that right now. So let me move this watch out of the way. Let's move this note out of the way. Go to that little Reveal bubble there, which I can never remember the name of, on both of these. And then let's just do that and zoom in. Here, let me make this full screen and zoom in. Come on. There we go

You see the difference in the structure? So again, I told you list management is critical to using Dynamo. So what we've got is the same information just organized in two different lists. So there list one is a list containing the first PolyCurve. So it's the first curtain panel. And that curtain panel generated one PolyCurve. If somehow the curtain panel could generate more than one PolyCurve, you'd have item two, three, four on that list. But that's not the case. So in this case, I'm getting a list of a list. But both lists contain one item. Little weird, but it is.

So the first list has 13 items for the 13 panels. And each of those sub lists only contains one item, one PolyCurve. All I cared about was the PolyCurves. So flatten just removes the structure. So sometimes you want the list structure. Sometimes you don't. So if the list structure is getting in your way, get rid of it with a flatten. That's all this does.

And so now I have 13, remember zero through 12, but 13 PolyCurves. And you'll notice here that they say four, six, four, six, four, six. Well, that information is there for my benefit, my information. But I can't do anything in the graph with that information. Therefore that note up there, PolyCurve number of curves, is saying how many curves are in each PolyCurve. And why does that matter? Anybody have a guess why that matters in this case?

AUDIENCE: [INAUDIBLE].

PAUL AUBIN: I sort of heard that.

AUDIENCE: Adaptive component.

PAUL AUBIN: My adaptive component, and how many points does my adaptive component have?

AUDIENCE: Four.

PAUL AUBIN: So one of the things that I need to do here in processing the data is make sure that what output from Dynamo gives me exactly what Revit is looking for. Otherwise it will fail. So because I'm trying to place a four point adaptive component, I need a list that contains four points for each adaptive component I want to place. That's the goal of all these nodes I'm about to show you. Just wanted to make sure I said that first so that we don't get too far.

So what this telling me is that some of them are good and some of them aren't. So what I did here is an Equals node. You'll find that under Operators. It's down here with the other operators. And it's just saying, is this equal to that? And so I'm saying is the number of curves equal to four? And you get a list of trues and falses. So some of them are. Some of them aren't.

And then you get this fun little one. Filter by Bool Mask, bool short for Boolean, because I guess three more letters would have been to too many or something. But Filter by Bool Mask, so all this is saying is, I'm going to take a list of true falses-- that's your mask-- and I'm going to separate it from one list into two, The ones that are true and the ones that are false. So you see the in and out?

So what am I saying? There's my original list off screen, putting it back in, taking my equality here, which is a nice, tidy list of trues and falses, feeding that into the mask. And then it generates two lists. List zero says, here's all your four-point curves. List one says, here's all your six-point curves. So the ones that are four points are in. We branch those this way. The ones that are six points need more work. We branch those that way. And that's going to be the next group.

So if I zoom out here, this is going to be simple. I'll talk about that later, the existing four-point curves. But the six-point curves-- there was four of them-- that didn't comply, those are the ones I need to deal with next. So let me unfreeze this and talk about what we're doing here.

Now on this part of the graph, I got some help on. And if you want to get help on your Dynamo graphs, there are wonderful resources online. The dynamobim.org is the main site for Dynamo. And one of the great resources on there is the forums. And the forums have people that are just itching to help you. Really, these guys are terrific. You'll post stuff. And you post your graph. And they look at it. And they go, oh, you missed something here. You didn't do something there. But I've got to warn you, they use a lot of code blocks. So just saying.

So what are we doing here? I'm going to take that list of out items, the four six-point curves, and explode them, because right now, they're PolyCurves. And I want to get each line. So that's what that does.

From each of those lines, I'm going to ask for their start points and their end points. So I'm doing what we did in the first example in reverse now. In the first example, we made a point. We made a point. We drew a line. And then we took the line and made it into Revit geometry. Now we're taking the complex geometry, making it into lines, and then taking out the points. So sometimes you go in the exact opposite order.

So let's run this part. And you're going to see some little zigzagy stuff going on down there. I'll show you what that does in a minute. But I've got two lists of points there, start points and end points. And if you look at your little reveal bubbles there, you can see the x, y and z-coordinates.

The start points, I'm just feeding across to another line node. And this is where you might be scratching your head. Why would he explode it, pull it down to points, and then draw another line? And I can explain this a lot better in PowerPoint.

So I have an illustration here to explain what's going on. So you've got these wedge shaped parallelograms, or not parallelograms, polygons. And there's too many edges around them. So that little hand-drawn sketch there on the right, let me just animate some stuff there. We're going to start with the first two points. And we're going to draw a line between those points. And again by the way, these are the folks that helped me on the forum. So I wanted to give a shout out to those guys.

So I'm tracing between those two points. Then I'm going to take point two there and shift it one vertex to the next point and draw a new line. And then we're going to ask the question, is the original point collinear with the new line? Do they overlap one another? And if they do, I want to get rid of that point.

Let's go to the next one. So now we move on to point two. We draw a line to point three. We shift point three. We draw another line. Is the point collinear? No it's not. That point's in. And you keep repeat all the way around the shape. And you'll end up with four points. And again, I can't take credit for this logic. True hardcore real programmers helped me figure this out. But it's brilliant. And it worked really nicely. And that's why I'm exploding everything and then drawing a new set of lines with it back over here.

So the original start points stay where they are. But then we take the endpoints, and we shift them by negative one, one vertex. And then this little doodad here, for the folks in the room who have been using Dynamo for a while, this is brand new in Dynamo 1.2. It's called List At Levels. I do have a version of this graph in the dataset that uses the old fashioned list map. But this new functionality essentially eliminates the need for list maps. So I'm not even going to explain list map in here.

But what this means is you've got a structured list over here. Sorry, zoom over here. So I-- get this guy out of the way-- list zero has these points. List one has these points, and so on. If I don't tell it where I want to operate on, it will choose one level of the list. And it might shift the wrong thing.

So I've got four panels, each four panel containing six points. I want to take the list of six points and take the one at the top, move it to the end, and shift everybody up. I don't want to take the list of panels and take the one at the top, move it to the bottom, and shift everybody up. If you don't say operate at level two, that's exactly what it will do. It will operate on the list of panels not the list of points within those panels. Clear as mud? Well, it better be, because we have three minutes left.

What are we going to do with this next? So does intersect, and then that gives us another bool mask. So we get a new series of trues and falses. And this time, notice I'm ignoring the ins. I don't care about those. So I'm only taking the points that came out of the out, because that's the four points that I want.

Now we take the original panels that had four points to begin with, explode those, and just get one of their points. I did start point. You could have done end. It doesn't matter, because now I have a collection of panels with four points each. I now have a collection of process panels with four points each. And List Join is just going to make it into one list. So I just stick these together right here. Unfreeze that. And now I've got a joined list.

Now this next chunk of code Zach Kron at Autodesk helped me figure out, because it was also a mindbender-- for me, anyway. So the problem I had now was sometimes it would start there. Sometimes it would start there, sometimes there. And if you try and create an adaptive component, and you start each one at different points, they're all going to twist and turn and do weird things. And it will fail.

So Zach suggested putting some point off in the distance somewhere and measuring the distance of every point to that point. So let's measure the distance of that one. Let's measure the distance of that one. Let's measure the distance of that one, and so on. And then you're going to compare all of those and find out which one is shorter. And so theoretically that gives me the lower left hand corner of every panel, because this point stays put all the time.

So we already learned how to do that, how to make a point that's in a certain location all the time. That was just point by coordinates. And I just arbitrarily dropped that point at negative 100. Just made up a number. Measure the distance. See what the minimum values are.

Let's run that. So here's all your distances. But here's the minimum value only in those distances. Notice that those two list structures do not match. So here you have lists containing lists. But here you just have one list. So if you look at the L1 and L2 there, you can see they're different here. And I'm going to just steal what Zack had to say about this. We could explain really detailed logic for how this works and how to choose which ones. But they designed this interface on purpose to allow you to just click things until you got what you wanted.

And I'm got to admit that's exactly what I did. This is what I wanted. I just kept clicking things until it gave me zero, one, two, three. Because what's that telling me? How far to shift. So sometimes I'm starting at the right point. Sometimes I'm starting one away, two away, three away. I multiply it by a negative number. I don't remember why I did negative. But I'm shifting them by the negative value here. And then I finally have my master list of points.

And the nice thing is here, creating an adaptive component once you've done all that legwork is super simple. Give me a list of points. And which one you want to place? So let's run this guy and see what we get. And not a moment too soon, 26 seconds. Run. Tick tock, tick tock, tick tock. There it is. Look at that. Nothing? Really, no?

[APPLAUSE]

I keep dreaming of the day when I'm not going to have to prompt that applause. So there you have it. So it places those points. So the hardest thing there is actually thinking like a programmer, figuring out all that logic. But again, dynamobim.org is a great resource to help you do that. So if you know where you want to get, and you got the starting point. You're just stuck in the creamy middle. Those guys are really helpful for that. And it can help you get over the hump. And you can do some amazing stuff with Dynamo.

That is our lightning tour of Dynamo. I don't actually know if I have any time for questions. How much time do I really have, because now it's on a new timer? So I guess I'll take a few questions until they kick me out.

AUDIENCE: Does any documentation show all the lists of these nodes?

PAUL AUBIN: So the question is, is there any documentation that shows all the list of all these nodes? They're working on it. They actually just released something called the Dynamo Dictionary. And I don't remember the exact URL. But if you Google Dynamo Dictionary, you'll probably hit it.

AUDIENCE: Who's working on it?

PAUL AUBIN: Autodesk. But also the community can contribute to it too. So yeah, that's happening. It's just not completely there yet. Anything else? In the back, first, yeah.

AUDIENCE: How many years has Dynamo been around?

PAUL AUBIN: Dynamo has been around about four years, I think. And it's taken me about that long to get this far. So you're on a learning curve. Yeah, in the front here. You had one?

AUDIENCE: I wondered, if you could flash the location where datasets are.

PAUL AUBIN: Oh, yeah, yeah. I think I had that in the last slide again. Whoops, nope. Oh yeah, right there, the first one. /au, and that's just lynda.com links. I know you guys have other places to go. They probably need to flip the room. So thank you very much. And I can maybe take a few more up here. But thank you.