HOMER ANAVE: So this is it. So good morning everyone, and welcome to day three, or is it day four, of Autodesk University 2015, and welcome to my session, which is Revit++, More with Your Revit, More for Your Workflow! I hope you were not intimidated with the title, especially this plus, plus thing. It might be scary at first, but things can be wild in the middle though.

Anyway, I just want to know some numbers. Who are architects here? Raise your hands. How about engineers? Contractors for owners? Thank you very much.

To start with I would like to introduce myself. My name is Homer Anave, and I came from Japan, but I am a Filipino, just like Manny Pacquiao. Do you know Manny Pacquiao?

[ONE PERSON APPLAUDS]

One fellow Filipino here, you know. [LAUGHS] Unfortunately, he was just beaten by Floyd Mayweather.

I'm currently working at Taisei Corporation for over 10 years now. And since entering the company, I belong to the BIM Solutions Team, which is an exclusive team of our technical design department, under the design division.

Here I am very challenged to do difficult but exciting work, especially when it comes to programming and development. So I never stop exploring in the possibilities that my work offers. And I can also do many things whenever possible. But when I am out of work, I do cycling on cool and warm seasons, skiing on winter, and boating all year round. I also do some video editing and music composition. So that's all about myself.

Here's our class summary for our session. So we already know the basics of Revit, so we have to bring our skills to the next level, as Revit has much more to offer. So we can build and explore more of its functionalities there. So templates, families, on-the-fly tool development, and utilization are some of those that can make it happen.

Here are the learning objectives for this session, but I won't write everything there is today. But I would like to put these objectives in few words.

We will do here a designer's template. We will also having here programmatic workflows, with some focus on visualization and viewing tools with Dynamo and Revit API. We will also having

interference checking using the interference support files, and also our plan and section detailing. All of this entails you to more customization inside Revit.

Speaking about customization, here you will see the customization hierarchy in Revit. With the lowest in the hierarchy being the easily customizable families, templates, objects files, et cetera and et cetera. Which in our session, we'll be focusing on our designer's template. So during the middle part, which is visual programming using Dynamo, and the upper part being the Revit API programming, the highest level of Revit customization.

Let's begin our designer's template. What is exactly this one, and how we came with it?

Unlike in Americas, in Europe and other BIM developed countries, Japan is lagging behind when it comes to BIM development. Although architectural designers have acquired knowledge in BIM, especially those young architectural designers, they are still on 2D workflows, and very few of them are dedicated into BIM.

Say for example, we assume our designer here, especially architectural designers in Japan, have received training in Revit. But due to the complex operation of Revit, he becomes frustrated and gives up the modeling work. Eventually passing the work to the operator. And then gives operators instructions.

But if you will try to think about it, the designer doesn't have the first hand on the model that the projects they are involved. What we think should happen is that the designer must have the first hand on the models, and must be able to operate himself.

Having this in mind, we had come up to a solution by which we work on the designers openly. And this is to create a new template. Because templates are the first objects that the user communicates with Revit.

We started thinking how the template must benefit the users, especially for architectural designers. On the designer's perspective, they must be able to produce models and drawings with the easiest possible way. They must be also able to produce presentable 2D and 3D views, and they like the easiest way. And most of all, they must be able to modify the model whenever, wherever, without having the hassles. And in connection with this, make them plan, design, and visualize nonstop while they model.

So to summarize up, we have formulated the characteristics of what we will call here the designer's template. So, it must give the designer the capability to use Revit as if they are

drawing, and be able to edit and modify them as quickly as you can. It must be also simple and lightweight. As much as possible, we must not give them the burden with all the features that the template has.

And lastly, it must present organized view and family browsing. You must help them locate the plans, elevation sections, families, et cetera, easily with a well organized project browser.

So now that given these characteristics at hand, what to expect in a designer's template. So this template may not be necessarily fully standardized. And it must have a very few everything. So that is, line and fill patterns, view templates, family types et cetera and et cetera.

But having only this will not really define the designers template, at least for our own perspective. We have to introduce this to our template.

Intuitive families. In our experience of creation and utilization of these families, we can say that they are really cool, and they make the template so cool. And of course, they must they must be preloaded into our template.

But what are these intuitive families anyway? Let us watch again parts of the videos I showed you in the introduction, but without the BGM of course. For sure, you can already own all the ideas of this, our intuitive family.

We have here a window. And once selected-- when they select that, notice easy to edit those properties here. And then of course, we can select what window set we want to use.

So when we are able to select our own windows, we'll just select the panels that we want to be here as transom, and then of course those panels that will be in our main window, actually. And then of course, we can change any time for our own liking, of course.

So that will be for our windows. For our doors actually, it has a different approach with the windows. In our doors, actually when we select our doors here, there are check-boxes here that we can select which part of the panel we have to load. Or we need in our doors there.

For example, if you set the louver glass here, so we have the glass here, including a slit here. And of course we can also remove the slit, so that our glass here would be half of our panel. And of course when we select this full frame, our panel here will be full frame.

And of course we can also edit our read here directly. And then, when we change our family here, it will automatically adopt those values for the families that we selected here. So of course we can also change those values and later than.

Then, not only windows and doors actually. We have also some plumbing fixtures. For example, he just mirrored those pictures from the women's to the men's, and then we'll just edit those quantities here. And then we'll just copy these, and then change those features into other families. And then we'll just stretch the linings we had here. And then we'll just copy there. And then we'll just change the families there.

So you can see that the quantities of our families, they're kept while we change our family. That's the power of the intuitive values that we have. Yes, it's really intuitive.

So for example, in our 3D model you can change them up. The materials here for example, it was previously pink, we'll just start that B-L-U-E, which is a material for our blue there. It's just very easy. We'll just upgrade those materials that they can be easily remembered, so that it will be automatically affected there. So that will be for our plumbing fixtures.

Next, you can also edit this intuitive families using our schedules here. I'm sorry, it must be Japanese characters here, but you will already see those changes that are already reflecting once we check those check-boxes here. For example we have the glass in the upper part, and of course we have these full frame, and some [INAUDIBLE] there. That's the preview of our intuitive families.

Let us discuss the characteristics of this intuitive family. By experience of creation and utilization of these families, we have come up with the fact that they must be flexible and elastic, stretchable and shrinkable in many cases. It must be easy to edit and modify.

The child families in it must be interchangeable too. And it must contain basically only one family type. That's the distinction there. The main reason for this is as much as possible, we don't want the designer looped into plenty of types these families may contain. And if there are too many of them, we will make them search for one that they need, eventually defeating the purpose of an intuitive family in creating our designers template.

What drives these families to work intuitively? Well? No awesome answer? OK. Those are driving our-- The answer's pretty obvious. It's the shared instance parameters. You create them using instance parameters, but there is an exception however. That we maintain to use

type parameters, or if family parameters, to some families, that we think they're using wiser. For example, if we need those structural columns in a fixed size, they must have type parameters in it. Applying instance parameters is pretty straightforward. You just have to assign those dimensions to shared parameters, as well as service other information, such as opening number, opening symbol, for example for our doors and windows. And you also use instance parameters to nested families.

But there's one thing I would like to discuss here in particular. Our one best application of these instance parameters is when we assign them, for example, to a door that has many nested family panels in it.

So let's say that in this door we have eight different panels. And these panels will have their own designated yes/no parameter. So these are the yes/no parameters, actually. It's right in here. It has their own unique names.

Well, what's the use of these parameters, and why he has no parameters? The use of these parameters is to set the visibility of their corresponding panels. These parameters also contain equality formulas, holding an integer that represents one or more properties of the door panels. Which are flush, with glass, with louvers, with slits, full frame, or inspection door. So we assign each of these properties their corresponding number, which is a power of two. But it is a power of two, so we will be having some mathematics here.

It is any number that is a product or quotient of one or several number of twos. To put it into perspective, those are the numbers like 1, 2, 4, 8, 16, 32, 64, and et cetera. So why use power of twos here? We have to remember that any integer is a sum of one or more combinations of numbers which are powers of two, without any duplication in either of them. Only these numbers can do that. This is the best way when assigning numbers, as the sum of these combinations are truly unique and no overlaps.

Looking at the table here, we list here the properties of the door panel, which is here. And we assign each of them a number which is a power of two. So here's our power of twos here. to the zero, to the one, to the second power, to the third power, and then there bit values here.

So having these number assignments in mind, let us have some tests here. Particularly with these three properties we have. So let's have a first of these three examples. Let us also have a parameter called PK-- PK is there. --that handles the sum of the power of two combinations. As of now, a flush panel is assigned by default to a number, which is one, currently set in the

panel does not contain any glass. If you put the glass into our panel like this-- so here's our parameters. Powers of two in it. Here are the check boxes that indicates our yes/no parameters there.

So when we take that glass, the number and the sum will change, making it to assign to our parameter PK here.

In case you want the louver and the glassware panel, we check both of them. So the values of the louver becomes four, and the glass becomes two, making a total of six, that will be our new value of PK. So this value of PK equals six is assigned to our panel, P2G. And finally, say that we want to add all of them. So adding slit, louver, and glass to our panel, then our result will be 14. Which will be the value of our panel here, P3G.

So taking this into account, we are now ready to assign it into our family. As you can see here, there are the yes/no parameters here. That's a visibility-- not visibility parameters, just our yes/no parameters. We also need to create integer parameters, such as glass, louver, which is over here. That pulls the value of the corresponding power of two integers. These parameters also contain if/then/else formulas. So those are the if/then/else formulas. And as of now, the values of these integer parameters has one, zero, and zero, respectively. But when you check these three check-boxes here, the value of our integer parameters change, resulting to the number 2, 4, and 8 which are their number assignments. This information will be passed to our parameter PK, which is the formula that handles the results of the yes/no parameters.

So the values here would be passed through here, and then in this case we have the values of glass, louvers, and slit chained, so this portion of PK formula takes effect. So these portions will take effect. Eventually the final value of 14 is there. So let's take this value 14, and look up another set of yes/no parameters, but this time in our visibility group. Okay, here.

By looking at the list, we will notice here that the yes/no parameter P3G has been assigned to a formula that says PK is equal to 14. So this satisfies our condition, and with this condition, P3G is automatically checked.

And finally, change our door panel there. That is how we assign and run these parameters in controlling the visibility of our nested families.

The strategy of creating and assigning parameters to our nested families will depend on what kind of family you are dealing with. This kind of parameter assignment is applicable for doors,

but it may not be applicable for windows and even plumbing fixtures.

So we have to think carefully how we take care of this visibility, or the visibility of these nested families, and their parent families. So that will be all for our intuitive families and our designer's template.

We will next proceed to our programmatic workflows, with focus of Dynamo, and a very brief stop about Revit API.

Let's talk about Dynamo first. You already know that-- excuse me. Anyone who are using Dynamo here?

So you already know that Dynamo was primarily built for a computational design, but since development progresses, it could be also used for other things. Because Dynamo exposes several functions of Revit API, or we call here, Application Programming Interface. We can use Dynamo to manipulate elements in Revit. Whether by creating them, or by editing them. So this also gives us a way to learn the more advance Revit API programming, when you are able to catch the flow of the script, or produce logics. You may become capable of more in the programming. It just requires coding.

But for this time, I just want to show you some of our samples in Dynamo that you can refer, or to when you create your own. Or you can use for some other Dynamo projects.

I'll be showing here our own method of creating a grid and levels. Creating our grid and levels, the monumental Apple that we have here. The example of pattern creations, in this case it is a conditional exterior back panel layout. And two, visualization to develop that vicinity visualization and a simple program that we call Cabbage.

Let's start with the grid and level creation. We would use the numbers here for to set our grids in our Dynamo here. Let's have a demo for that one.

I have here the Dynamo script for creating the grids and levels. So in this portion are those that creates the grids, and this portion is the one that creates the levels. So we have our inputs here, but as you can see here, we have already input our levels. It must reflect in our other data here. We have-- We know what each of them are, so it's already created for us.

So how does it happen? To explain what happens in creating our levels, by the way, our method of inputting our numbers to create our elevations for the rest are like this. All of them

are in string values, text values, not numbers actually. For example in upper floors we have this 4,000 multiplied by 6, and then 3,500 afterwards. So that's our example.

We have to first ask, process using our string splitting, or text splitting. So we split our text here, and then in this portion we produce this list here. So you have their own there are 6 4,000s one 3,500 generated there.

We would be using this list, actually, to obtain the quantity of the items inside the list. The quantity will be used as a prefix to generate our level names. Here's our count here, so there are seven. And then it will be used to create our level names here, which is 2FL, 3FL, to 7FL. And of course I'll be happy to add this other-- because our code up here starts with two. It will be just only six here, and in addition we'll have this outer set, or the roof slab line. You have this, when we created our names for our levels, we have to produce this. Get the sum of per level elevation each, for penthouse upper and lower levels. Revert the difference to R1FL, which is created here. This button is here in the bottom.

From this one level actually, be creating levels with some offsets are generated from this sum this by range, which I created. It's a custom node.

So from 1FL and when we create our 2FL, it will create with an offset of 4,000 millimeters. I'm talking about the metric system here, because in Japan, we use metric system.

In the second to farthest, it is 8,000. So we create from 1FL, 8,000. 8,000 millimeters of elevation that there will be our 3FL. And so on and so forth, until we create all of them. And then just a generic here, our grades using this list combine.

Of course it's also applicable for our penthouse, but in our penthouse, we'll be creating the lowest part of the penthouse from the highest level of the upper floor. So it will be the RSL. We also create the levels from the lower level, from the 1FL, but the direction is different. It's being created below it. The values here actually will be limited by-- Anyway, these are the elevations that are offsets. And then the levels are created from the 1FL view. So that is how we create our labels.

Here's the portion that we will create our reads-- but at this point, we haven't set our reads here, or our offsets here. We will set our offsets here. So for example in intervals along x-axis, we have this 10,000 times 3 and 5,000. In our y-axis, we have here, for example 6,000, 9,000, and 5,000. It is automatically created in our Dynamo interface here. You can see it here. And

also it will be created in our Revit model. In our Revit project. So it's here.

There's an option also, that you can rotate these grids. For example, if you want to rotate the grids to 30 degrees. So we just type your 30 degrees in, all the grids should be rotated 30 degrees counterclockwise. I'll bring this back to zero.

The process of creating these grids, especially in the first process, is just the same as how we create our labels. The only difference is, we use these intervals to create our points here. We use these intervals to create our points, that will in turn create our lines by two points, that will be used to create our guidelines.

Here's for example our horizontal grid lines. Here are the points that were actually created in our x-axis, and here are in the y-axis. And then, we use this point to create our lines here. So here are the lines, and then we can connect a also with the option of a rotate, here. Before we create our grid lines from those lines, here.

After we create our grid lines, we have to assign them with those names. So those names are actually here. As you can see here, we have X1 to X5, and our y-axis we have here, Y1 to Y4. You can also change these values here. This what is can be changed here after you have this x and y, For example. And if you've got these two, a and b.

When we set our vertices for x-axis to a, and our y-axis to b, it will be automatically updated here. For our x-axis will be a, and for our y it will be b.

Of course, we can also eliminate these prefixes if we don't want them. If you don't need them, you'll just select this first, and all those prefixes will be gone. And eventually, in our x-axis here, there will be 1, 2, 3, 4, 5, and in our y-axis will be A, B, C, and D. And when we do that, we'll just change the parameter of those grid lines we have with those things that we assigned here.

So that's how we create our grid and levels.

Next we will go to the apple. Obviously it is a complete computational design sphere. But it's not just something that we play with. This apple is part of our clients request. They want a landmark somewhere in a hospital's premises.

The hospital is located in Aomori prefecture in Japan, where it is the heaven of apples. That's the reason behind the apple shape. Let's have a demo for this one.

[SIDE CONVERSATION]

Here's our profile of our pad for Apple, and we'll just open our Dynamo again.

Here's our scrape for the apple that they're creating here.

All we need here, is just this curve that will be used to create our apple. So here's the curve, and we'll just select this curve here. And it'll automatically generate our apple for a number of seconds. So we have to wait. It won't take a minute, you know.

[LAUGHS]

It's longer than what I expected. OK here's our apple that is created in Revit, and also it is also created in our Dynamo interface.

Let's just have an explanation of how this is produced. So first we had to get the geometry of that curve that we've selected here. And then from that geometry will be copy rotate the pad, and divide those pads into what will produce segments. You produce here the segments, and then we'll just divide the curves by the given number of extract points. So the extract points will be here.

When we enter here 16, the number of curves that will be copy rotated will be also 16. The divisions that will happen here will be also 16. And when we gather those points, we have to extract some points that we need to create the NURBS curve afterwards. As I mentioned here, we get the end points the end curves to collect them. That means, we'll get the first point of the first curve. We get the second point of the second curve. We get the third point and the third curve, and so on and so forth until the last point of the last curve.

Really, when you get those points so it's really here, really be creating our NURBS curve. And this NURBS curve, that we'll be using to create our sweep.

One requirement for our sweep, is to create our profile there. We create our circular profile at the start of the point of our NURBS spot. And we will just create rotated copies of those NURBS spot, and of course our circular profile will be given a number that I mentioned earlier, which is the value of eight.

Also we have to create the mirrored copies for those NURBS spot and profiles. When we

created those NURBS spots, we'll just create the sweeps using the NURBS spots and the profiles. Our sweep here was created using the original copies of our NURBS spot, and those mirrored parts are not situated in this stage. When we created those sweeps, we can actually apply colors if we want. But if we want to export them in Revit, or in a Revit project, we'll just create a list and then flatten them. And then connect an input instance by geometries. That has happened in our example here, actually.

Yes?

AUDIENCE: Will you make the slide show, or better yet your data set, will you make that available, uploading it?

HOMER ANAVE: Yes, yes. Up to the grid and level creation, and this apple, will be available to you afterwards.

But first, even those who update them or enhance them for your liking.

Next will be our--

[SIDE CONVERSATION]

So next will be our conditional exterior panel layout. This is a sample that was not originally created in Dynamo. It is created actually in Grasshopper. But I somehow partially succeeded in translating it to Dynamo.

This is the newly opened LaLaport Mall, in Ebina city in Kanagawa prefecture. Here's a part of the exterior of the mall. Actually, it is the front of the mall. But what I was tasked to do in this project is to generate the patterns on this right side of the exterior of the mall.

The designer requested me to generate the wall pattern randomizer script, and use our patterns to consult with the clients of the owner, or the owner of the mall.

But it was not that easy, as there are rules that are needed to follow. The rules are, no adjacent colors on all sides, except for the brightest color. Rule number two, no dark colors at the sides of the blue color here that surrounds our windows. Rule number three, dark colors reduce as level increases. So in our lower level, there are too many dark colors. But as the level increases, you can see actually that the dark colors are decreasing.

As a matter of fact, I have generated five patterns for this, but unfortunately, none of them are

used here. Anyway, I was able to give the script, and taught the designer how to change the values of the script, so that he can make himself patterns that match his and the owner's liking.

Let's just have a demo for this exterior panel they wrote, but we will just have the first row of the panels created here.

I will just switch to our demo machine.

Here is actually what I did to reproduce our exterior panel creation. First though, all we need here is just a mother curve. But this time, I will be using this-- the longer curve here.

So this just like that and it will regenerate.

[SIDE CONVERSATION]

OK, here we are. Here's our result for our exterior panel layout operation. It's just only one row of panels there. OK.

After selecting the curve, it will just automatically go to our geometry creation. Which in this part, will be creating our panel. We have here the curve, and then we'll just divide the curve to smaller curves that has a fixed length. Our fixed length there is very small. It's 600. We divide this curve to a smaller curve so each has 600 millimeters in length. And then of course we have to produce points from there. Then after producing those points, we will be creating line support for those points. And then we will just offset the lines, or create copies. Create the lines which will be translated above. And the offset for that is 3,400.

Why do I need to have a copy for that? If we need to create some surface out of those pair of lines. We have these pair of lines here, and using this pair of lines, we will be creating our surface.

We have to know how many surfaces were created in our created panels here.

One reason is that we'll have to assign some panels in some particular colors here. We have this color volume control, which means you're just controlling the proportion or distribution of colors, of the panels for color. So we can just control here.

And when we control that, you can see that the colors are changing and panels are changing there. So when we control that, we'll just pass here through our panel number proportion. Which will give us the result of the number of panels that will be using a particular color.

So when we are able to establish that, we'll be going now to evaluating our colors here.

This group of nodes here will have the rules applied for our panels. So first, when we have those number of panels, we have just created indices for the panels out of the total number of panels that we have.

First, we will have this panel still, and then sort them and then apply the rules here. So there are two pattern scripts here that applies those rules. And when we apply those rules, we'll be able to produce the final indices, which in turn will be the indices of the panels that you have painted with a particular color.

When we have it, if you produce these colored panels here.

Actually the only rule that is applied here is the rule number one. So when you have applied the other rules, we need to create another rule above it.

That will be all for our exterior panel creation. You can also apply this whenever you have this part and design, that for example if you have a project like this, you can always refer to this one. So that will be all for Dynamo over here.

Let's move on to our visualization here.

Next with concern to some visualization analysis. The first one of each is visibility visualization. In this example, our objective here is how much we view a particular building or mass, from a particular area in the ground.

To be able for you to understand more of this concept, let me give you one application for this. A particular historical spot in Rizal Park in Manila, Philippines, where the statue of our national hero, Jose Rizal, must capture a beautiful clean scenery, or present a very natural scene on its background. Meaning, nothing in it must be an eyesore. That spot is a view port of interest

here, and the picture of this spot is this. This is the picture of the monument of Rizal Park, shot from the area where we must capture the clean background view.

It's not actually clean anyway. There's a tower there. This was the spot where you can take the best picture of the monument, and feel its harmony with nature.

Until there is a very tall condominium which is being constructed just within the view from the spot. The name of the building is Torre de Manila. The worst is, the construction progressed up to the 46th floor, The highest floor of the building.

So, the current view of this majestic view is now like this.

The building obviously becomes en eyesore. This is the Torre de Manila. So with this fate of the Rizal monument, one architect proposed to construct a huge historically relevant mural behind the monument, so as to hide the eyesore building. This is the proposed design of the mural as the statue's background, designed by a certain Noel Tan of Noel-Studio 300. This was actually published in the internet.

This is what we'll be reproducing in Dynamo, to establish tolerably rough dimensions for this mural wall. We have this movie here, that's our script there. You have five projects to select. First we have to select the base, which is here. Then we select the subject attraction, it is here. And our obstruction will be our mural wall. And of course, we need to select important points to produce our subject facer. And then our, of course, subject mask for the view analysis, which is our tower there. Here.

And when we are able to select done, things that we need will just-- press the run button-- and the result will be this one.

As you can see here, the results are in this very opaque red. So it means that this area, this building is not viewable. So in those areas that are transparent, the building is viewable and those gradient areas are areas that somehow, you can see the building partially.

Here in this area, just display the geometry for our Dynamo interface. And for this area, it's just a sponge board for dividing the subject face to the smaller faces for individual analysis. And the area below here is just for providing the valid faces of the subject math. Which is up there. Which are not facing upwards or downwards. We're just having a notice at their end. This one node here is just responsible for analyzing the valid faces for the mass for the surface that we

have. And this is actually the contents of the custom node. This takes care of the analysis that we have here.

We have to create the gradient colors here, but as you can see not all of surfaces are painted red. So for example, if we want to adjust the height of this obstruction because--

We can make some parts tolerable enough to view. We'll just adjust the height here, and then we'll just reselect our object. And then afterwards, we'll just play the run button.

We'll just have to wait for that. And then when you press the run button here, here will be changed. Our gradients will be changed here. In this portion that it was already transparent, it means that from here, they will be the least viewable. And from those areas here, the building is partially viewable. This may be an error for the movie that I did, but we can always adjust those duos to obtain the optimal height or the dimension for a mural and wall there.

Let's proceed to our last Dynamo sample here, which is Cabbage. So we call this Cabbage. This is a little brother of our main application, Pumpkin, and it is primarily built using visual scripting, tools like Dynamo. This is the application for use of design of the stadium-- OK, I'll just skip through that.

The big brother Pumpkin actually produces the results like this, as you can see in the foreground of the soccer field. That's our subject view. In the background are the stadium bleachers, where there are lost of colored figures. And in addition to that, we have this gage here, that indicates the percentage of how much they view the soccer field.

The one here means 100%, and the 0.2 here means 20%. The processing time that we have here, it is for those Windows PC, with three CPUs it took 110 minutes. But for the faster in house computing server that we have, it just took only 13 minutes. The figure colors have meanings. I have already mentioned that earlier. The colored red are those who can see the soccer field 100%. While the yellow and green colored figures here can see the soccer field fair enough.

But there are those who will say, I don't see anything, and I am nothing. Here they are. So knowing that this fate of case, the designers must work something to make these people something. This is how the concept of Pumpkin works. But what I will be showing to you here actually, has a different approach when it comes to giving the result. Hence having the name different, which is Cabbage.

In Cabbage, I'll be showing to you how this applies to the sample data model we have, and use a particular viewer to tell us how much it views the screen.

Here's a demo for that one. Here's the script that runs our Cabbage, here. We have three selections here, so we first select the viewer here. And of course, we have to select all the obstructions in front of this viewer.

And last but not the least, the subject face, which is the screen here. So after we select those things that we need, we'll just press the run button. And then when you press the run button, here, our result.

Here's our result actually. Here's the viewer that has a number in his back, which is 76. It means that 76% of the screen-- it's the percentage that he can view the screen. So to expand what's going on inside, here we just displayed the elements of the Dynamo interface, which are obstructions here.

And of course we'll need to offset the location point of our viewer's eye position. Because by default, it is here. But we need to offset so that it will be a point for our eye height. And of course we need to divide those to some other faces, and get the center point of each face. I actually used a node here by LunchBox. And then when I establish those surfaces, I'll just extract points from there. And then we'll just upgrade those projecting lines from the eyes move to the points from the surfaces.

So when we created those projecting lines, lead this up past those array, to hit analysis on our group here. So we just passed the sections here, and of course our lines here. And then all will be analyzed from this custom node here.

When we do that, we'll just produce the output text. It is 760, it says here. And then we just change the parameter off our object there. With a name percentage. And it must be updated here. So you have to connect one to another geometry node here, just to make sure that our geometry and our Dynamo interface will be updated. If we want to adjust this height of this viewer, for example. Because 76% is not enough. We need more here.

For example, if we adjust the offset to 175. And then we select this model element, me and we run the program again.

It would be updated to a 95%. so he can see 95% of this screen. That will be all for our

Cabbage, actually.

I have just demonstrated to you the Dynamo scripts, so you can always extend these to any other things of your liking. But Dynamo actually unleashes more than what you know. But if you think Dynamo is not enough for you, why not take it to the next level? You can always try Revit API programming. I will not talk about the pure programming here. I don't want to scare you here. I just want to have some brief explanation about its requirements.

Developing programs may take knowledge in C# .NET or in visual basic .NET But when you have it, you'll be able to learn the Revit API yourself. Also, you do not need to be a Revit expert to develop programs in Revit. Take it from me. I myself started to develop programs with a little intermediate knowledge in Revit. And I'm not totally a Revit expert until now. I come from a C++ environment, and I just learned C++ when I was tasked to develop programs for Revit. So don't lose hope. If you're interested, or if you're a beginner, it might not take months to learn it. But you will progress as you develop programs.

It's just a preview of the screen, the program that I'm developing.

The benefits of the ability to develop programs in Revit will take you to create very useful programs, such as those that I'll be showcasing here right now. We'll be discussing here interference checking, and plan and section detailing with compound structures. All created with Revit API.

Let's proceed to interference checking, using our interference report files.

What is an interference report file? This is a file with an HTML format. That is also an output file when you perform interference checks in objects, or between objects on the project model and the linked model. These can be acquired when you run the interference report. Here's their interference report. And we'll just press the export button there. And there again, you have your HTML4 file, which can be viewed from the browser.

Have you wondered where it shall be used next? Anyone, any ideas? Yes? You have ideas?

So anyway, the basic approach for interference checking inside Revit is this, when you have linked models detected. But once you have exported the HTML file, chances are it will only be just another document for Revit and nothing else. That would be the dead end. Unless you use them in Navisworks. But come on, you can use this in Revit too. You really think it's just by using an add in? Unfortunately, yes.

We need an easier way to read the contents of the interference report, the report file, and do something with the results it will give us.

Let's have a demo for this one. Let's make them Revit-documentable!

So here's our example for our interference checking. I already assigned here in our sample project here, an architectural model with a structural model linked in it. You can see it here, and manage links here.

And with this I'll run our tool here we call generate class views. What this does is create red space, and then position them to the points there were where the interference happened. And of course it will create 3D views from them. So this is the HTML file that we have. It's not a very long process regarding this project. Oh, and that's not 16/13.

Our class report is completed. The results are up to the-- I already mentioned this a while ago-- Here are the red spheres that is located at the points where the interference had occurred.

And of course not only that. We have also created our 3D views here, for example, in this 007. We have here the portion where the class happened, or occurred. When have this 3D views created for us, we have to of course isolate those elements that are actually interfering, or clashing. So we need another command here, which is, isolate clashing elements.

So we just pick that, and it will just leave us the elements that are actually clashing. But in this case after the link, I can't-- Only the elements that are clashing from our linked model. I'm still working on for this one.

When we have this glass piece that we have here, we will next document them. In documenting them, we have to lay out the interference checks used for documentation. First you just make sure that all the class views show clear views of the clashes. When you're done with that, you may want to have a key plan, or sorting the areas of interference, something like this.

Actually, this is a different project. It is a very recent project actually, we have in our office. It is a condominium project, and it shows actually the red spheres here. There are two sizes of red spheres here. There's another smaller red spheres after the--

So when you're done with that, you may want to have-- so you can do it using sheets to lay

them out. So the red spheres are indexed, so you won't lose your way on specifying the marks.

Here are the sample layouts where I've created by my colleague here in Taisei. She actually did very well in creating these views. We use these views now to submit to our designers, or to our engineers, so that they can actually talk and decide on what to do next.

I'd just like to highlight something about this thing. This is also created actually, by another command. After they create that, what process must allow those pipes pass or not? In this red areas here, it indicates that no pipes must pass that areas. On the contrary, these green areas are those areas that allows the pipes to pass through.

But in this case, these two pipes are actually failed, because it's passing underneath the smaller beams there. So by the way, I also have the Dynamo version of this one. These then generate class views. But it's just fails, because Dynamo does not recognize getting elements from the element ID. There are some Dynamo programmers there that actually offers some of those that are created, that gets elements from element ID. But after, all of them are failers.

So this is the only thing I am lacking, but it will succeed here in the near future. I hope Autodesk will address the problem. And when I succeed, I would like to share with you.

OK. Some recommendations however, for interference checking, especially when we are having work-sharing projects here. First, we have to disintegrate the project file from the central file. The purpose of that, if you don't do that, you have to perform interference check and export the interference report every time you open your project. This is because the IDs of the elements always update when you're opening your project. So your old HTML file will become invalid after the update.

You'll also have to save your project under a different file name. Just in case. And bind all linked models. This is the idea of, if for example an architectural team's Revit model and the structural team's Revit model do not have a common origin orientation. Actually, this case is eminent in our design office because in our office, the structural team and the architectural team coordinate, with regards with producing the Revit project.

For example, we have this team, the architectural Revit team for just this project in this orientation. But the structural team actually produces the project in a different orientation. It is rotated in 90 degrees when I am conducting those interference tests, it was really difficult. The

solutions that I just discovered is these articles that I have now given to you. So that will be all for our interference checking.

The last will be detailing our models and drawings. In this part of the session, we will talk out plan and section detailing, using compound structures. But first, let us review what our compound structure is.

Compound structure is a property that exists in walls, floors, ceilings and roofs, and you can access it with the edit type button. And you can see it under the constructions group. You know that already. Here's the image of the window when we enter the compound structure property. So here we assign the layers, and the materials, and all of the functions, and their thicknesses. You do this every time you want to define a configuration example of a wall type. This workflow may be tolerable if you have just a few types to work on, but it will be tedious when you have to do many.

That is because you are doomed to a repetitive task of composing different configurations of compound structures per type. And you have to do many mouse clicks, and you have to use frequently your types selector.

So up to the most companies in Japan including ours, do the configuration of these, especially walls. In order for us not to solve for this in this repetitive task, we developed programs that I will be showing you now. Here's the set of programs that I have here. I actually developed this set of programs since 2011, but I changed the configuration in 2013. And those are outsourcing offices, and our temporary employees are using the program.

We introduced this program to our Revit modelers and outsources since three years ago, and what they say about it is what I will guarantee you here. They never use their type selector, and they never pressed that edit type button again. This will just make you concentrate on configuring the wall types, and not let you mind those types that are being created for you.

Let's have a demo for this one. Here's a very straightforward sample model. It's to solve for a cornered space with a interior wall inside. And it's already have the configuration of the interior in this wall. So we also have a wall tag here, that we use often in Japan.

We'll just press this edit wall type, and then we'll just edit these walls for example. And here you can configure your wall types. excuse me if those characters are in Japanese. Anyway, here is the structure of the core layer of our wall. And the outside portion here, and this part

next to here, has the substrate and the finish. It also has the inside view in the interior side.

So for example, we want to set the thickness of our [INAUDIBLE], or RC, to 200 millimeters. And for example, if we want to set our subject to non combustible GB, to our 12.5, and our finish to wall paper. And we also apply here to our interiors.

And when we press OK, either of these will produce our walls here. This was already configured with what is set earlier. I will just-- I just have a view of what happened inside here.

I just-- good to see, good to show you. These are though, wall types that were created with the command right there. And when we try to enter the properties here, the program did the work for us, here.

You have the structure here, and the finisher. For the layers that don't have thickness, We'll just save the function to a membrane layer. For example, if you want to change the configuration for this one-- Excuse me, I forgot. It also created our wall tags here, anyway.

For example, if we want to change the configuration of these, there's no problem. What we'll just do is select the wall again, and then just press this edit wall type. All of the configurations that in that wall will be reflected when we press that edit wall type command there. For example, if we want to make these two plies of non combustible gypsum board, and semi combustible gypsum board. Then we set this to some undecorated silica board. And then we just set up for those configuration from the exterior to the interior. And when we press OK, it will be just reflected here. Not only-- we do here implants, we can also do in sections, actually.

We just go to the view and then apply it here, for example.

We'll sample another configuration here, and then we'll just press the same button. And then we'll press OK. Oops. I forgot to input the size here.

The walls will be configured as well, these tags will be input. So these are programs, actually the heart of all of the problems that is being introduced here. But there are some programs that are very interesting, or very usable here.

For example, if you want to have to put wall tags here, we just press the wall and then press the part where we have to place the wall tag. I think I have to change this so that you can see that the position of the wall tag would be correct.

I just changed something here. I'll be changing the exterior part. For example we have H here, and then of course these values would be changed also. And then when we click this tab input thing-- the side of this will be adapted with this wall tag that we have here. Of course, if you want to change the materials or the settings for our walls here, and for our details here, we have this export materials using Excel. We just press this button, and we'll just export all the configurations and materials in Excel.

To explain briefly, this is the material list that we created. You can always modify this one, actually. And of course, you can also adapt this drafting and model fill patterns. And then copy and paste here, so that will be reflected to our material when we import them into our project.

Other than that, we have this wall structure, wall substrate and wall finish configurations, as well as both our ceiling and floor. materials.

I guess that will be all for our detailing, Let's have a wrap up for this one. Designer's template may be a revolution for designers, so what do you think? Next, the more you customize with your Revit, the more effectual Revit will be for you. And of course, this generation is a Dynamo generation now, so be active anyway. I might be reluctant for this one, but try considering programming with Revit API. Because more and more capabilities will be unleashed when you start programming with Revit API. So that will be all for our session, but actually I will be-- Regarding the tools that I presented here, regarding these Revit API tools, for example this Inst->Type Converter. Actually I didn't demo this one, but I put it in my blog that I am writing here. This here. I have this blog. I'm not that active in blogging actually. Because my company doesn't allow us to blog, for some network security reasons.

Here's the procedure actually for bridging your design template, for your intuitive families. Just in case you want to have a transition from the designer's template to your usual template, so that you can continue on with your working design.

This is my blog, and there are instructions there. Of course-- let me go back to the original slide-- It's available at the link from the QR code that I will be showing you later on. Regarding this, generate the clash views and isolate the clashing elements. actually, it will just be available at the Autodesk app store. There is this, actually. We'll be in the second quarter of next year. I'm still not sure, I just wrote it, It's still being negotiated

But for the Dynamo scripts that I demoed, for example the grid level creation Apple, it's already available at the link of the QR code that will be available for you later on. It also

includes the custom nodes that I used.

If you have some feed-backs regarding my session, please visit the survey stations, or email, or mobile device. you'll be having yours-- Passes will be awarded daily, actually. So please do give us your feedback.

You have questions? Any questions? Yes?

AUDIENCE: When you used this wall type edge--

HOMER ANAVE: Yes.

AUDIENCE: [INAUDIBLE]?

HOMER ANAVE: For example, when there's a type that is not available, a new type is created. But if the types are available, it will just copy these types and then it would just be adapted to the wall.

AUDIENCE: [INAUDIBLE].

HOMER ANAVE: Yes yes. Yes, yes. First it checks the available types, and when it's available, it will just use the type. When it's not available, the program will create another type for this wall. Any other questions?

[APPLAUSE]

So it's none. Thank you very much on behalf of Taisei Corporation.