ERIC CYLWIK: Good afternoon, everyone. Thank you, guys, so much for taking the time to attend Using Reusable Subassemblies today. First question out of my class-- how many of you attended the other version of this last year? Wow, only a few. OK, cool. So last year, I did a class called Building Reusable Subassemblies, and we are literally picking up from where that class left off.

If you didn't catch it, it's not a big deal. We're not going to be building subassemblies today. We're doing what you do after that. So you don't feel like you're missing out on too much. And if for some reason, you decide you hate building subassemblies, I have all of this stuff posted online so you can actually download it and never have to build a subassembly. But you will miss out on so many great opportunities in life if you don't build subassemblies.

So first of all, this course is for lazy engineers. How many of you are lazy engineers? OK, I'm looking at the people that didn't raise your hands, because you guys are the really, really lazy engineers. And that's who this course is for. So-- what was that?

AUDIENCE: There are some technicians in the room.

ERIC CYLWIK: Oh, you guys were like sub-lazy. You guys are way below that. Actually, you guys probably do all the real work, right? There you go. OK.

So Civil 3D is a powerful platform. The corridor can be used to solve many design, engineering, and construction problems. By using reusable subassemblies, one can leverage the power without creating content from scratch every time, the idea that you spend a bunch of time-- not maybe a bunch, but maybe a day-- making something that's awesome, and you get to use it for a year or two and save yourself a ton of time.

So at the end of this course, you'll be able to understand subassembly parameter passing. I'll show you guys how to do that and kind of explain what it is. You'll understand custom codes and styles and how important that is to really making corridors do what you want and need them to do. Learn how to create dynamic assemblies. That's what most of the class will be. And then also at the very end, we will generate a quantity report so that way you guys can kind of see one example of what we can do with this kind of information.

I will say that, at the end of the keynote, they had the junior in high school up there and how much he accomplished. And I don't think that's anything compared to this guy right here.

[VIDEO PLAYBACK]

-

[INAUDIBLE]

. Block

tower. 4

minutes.

[END PLAYBACK]

ERIC CYLWIK: Of course, that might be the wrong unit of measurement. I don't think that block tower exists in minutes. Maybe it's inches. But just the idea that our future is bright, and if that doesn't inspire you, I don't know what will.

So quickly, for the agenda today, we're going to be talking about the review of CI11634, which was the class last year, cover some of the concepts that are important to what we're doing today. The rules of subassembly reuse-- there are some things you can do, some things you can't do. We'll be talking about that briefly. We're going to go in and use three or four subassemblies today to create dynamic assemblies, and then ultimately, we will be quantifying quickly at the end.

So the review of last year's class-- we created the quadrilateral shape, which is a fancy mathematical term for a four-sided shape. And in this particular case, you have the ability to use parameters which are exposed to the user inside of the Civil 3D environment to change slopes, to change widths, to change the height of it and all of these kinds of things. And because there's different parameters that control the top slope and the bottom slope and the inside height and the outside height, you can truly make something that's extremely flexible, right?

And beyond that, one of the key lessons from last time was you can then pass that information to other subassemblies. So if you have a roadway target that goes out to the edge of paved line work, you can pass that roadway width through your subassemblies to the layers below. So that way, you only have to use one target when you're using targets in your corridor, which is a huge timesaving feature that I don't know too many people are taking advantage of.

The other subassembly we created was a math operator. And this is kind of a-- not a visual subassembly but a pure mathematical subassembly. It literally has a point, and you tell it add, subtract, multiply, or divide, and you give it A and B. So what this allows us to do is if your sub base material is one foot outside of your roadway width, you say, well, grab the roadway width from the

previous subassembly. Add one. And make that the width of this new subassembly.

And so then you don't need to go in and do all these custom things. And as that width changes, you don't need to play around with targets and da, da, da. You can just go in and have all of that stuff done at runtime when you're building your corridor and really make it truly dynamic and responsive.

Here were some examples of what drove Sundt to build this particular assembly. This is a runway airfield. And this is using just the quadrilateral shape. But you can see how unique each of these pieces of geometry can be, and as the profile height changes, different layers expand or contract to be able to absorb the profile in existing grade differences.

So you can do some really, really neat and complicated things with this. And I kind of showed a little bit of that last year, but we're going to be focusing on this year, probably things that are more applicable to the middle range, where we're not doing something that crazy, but we're also doing things that'll be used much more frequently.

So the ideals last year were abstract base shapes. From a quadrilateral, you can compose roadway sections. You can compose runway sections. That was based on very, very flexible geometry as well as codes. That runway section that I showed had a different code for each portion of those shapes. So that way, when you go in and quantify your materials after the fact, it doesn't matter that you're using the same subassembly over and over again. You can change its code based on what particular material that's representing. We talked about parameter passing and then that runtime math calculator.

So this year, why reuse? Well, I'd argue that it's ultimately allows you to do reduction in troubleshooting. How many of you have built a custom subassembly? How many of you have then rolled that into of Civil 3D model and then had to go back into Subassembly Composer to fix something? Yeah, it happens a lot. The Preview window inside of Subassembly Composer is great for what it can do. But there's lots of things that it can't do.

And so sometimes you get it into a Civil 3D environment that doesn't perform exactly like you expected it to. Well, with these particular shapes, you're building these components within the Civil 3D environment so you can see how it reacts. You can instantly throw it into a corridor. And then you also never have to go back and troubleshoot this base shape, because it's so solid because you've used it already.

It allows a lot of consistent-- I'm sorry-- allows for a lot of consistency and standardization. We're

going to be building an MSE wall and a curb today. And then I'll show you how to transfer that to a different file and reuse that MSE and reuse that curb. And they're all based on that quadrilateral shape.

And ultimately, for us, it's important that a project that happened a year ago-- if the owner says they've decided they need an as-built model for something, we can go back in and we still have that packet file, and it's still the same packet file we're using today. So it really allows us to be able to dive into other things. And then also as we're starting to work on different projects across the organization, we don't have to distribute a new subassembly library every week. We have that same one, and those assemblies can change on a per-project basis. And ultimately, this allows our engineers to continue to focus on modeling instead of building subassemblies, which is ultimately what most firms are paid for.

So some ground rules for using the Copy, Mirror and Insert functions. How many people have ever used this and found something that was wrong after you did this? So there was a pretty big bug up until 2017-- it's fixed-- where if you mirrored multiple subassemblies, it would mess up some of the logic. And so it's fixed now, but we'll talk about some of those things.

So one of the ground rules is you have to select the logical connections. And what that means is over here, on the left we have our Prospector tab view of an assembly, and then on the right, we have our Assembly Properties view of that same setup.

And so if we're going to select logical connections-- for example, showing on this Prospector tab, I can select these four subassemblies, and that's OK. If I copy them, if I mirror them, that is totally fine, and that's what the tool is built for. But if I try to, say, select these two and then these two and skip over this one, the entire logic breaks down, and it won't let you mirror it. It won't let you copy it, because it knows that you shouldn't copy these two without this one in the middle.

So there are certain restrictions on how you can flip subassemblies and different things that you can do with them. So logical connections is the first ground rule. The second one is to always check the logic after you mirror or copy. It's worth just spinning through the Assembly Properties window on the Construction tab just to be able to look at it and make sure that it did do what you intended it to do.

Updating subassembly names is important if you have two subassemblies that have the exact same name in the assembly and you're passing a parameter from one of those, Civil 3 only looks at the name that the user's typed in. So if both of them are called "road" or "lane," it's not going to know which one actually you're trying to reference, and it'll cause issues where you go to build your corridor

and your subbases, 2-foot-wide instead of 22 feet wide. And you end up with some really weird circumstances.

So here's an example of the issue I was referencing about mirroring. If we go in and look at the composition of this particular assembly, we see that this left shoulder ACP item is right here. It's kind of about the middle of this first chunk. And then two below that is the math operator. And the math operator is looking to that left shoulder to grab the final width of that particular piece of the roadway, OK?

So what we're going to do is we're now going to select the entire side of that subassembly. And we're going to mirror it to this assembly down here. And then we're going to go look at the Construction tab in that other assembly. And what happens is that left shoulder ACP has dropped to the very bottom.

But if we click on that Math operator, according to this, it's still looking for that left shoulder ACP, but it actually hasn't been calculated yet. One fun thing is if you click on that, it'll automatically select None, because it realizes that it can't grab it. So just looking at the logic of, OK, this happens, this happens, this happens, after you mirror it is worth checking. Like I said, in 2017, that's not nearly as much of an issue.

For updating subassemblies-- so this is not dealing with the assembly anymore. This is particular subassemblies. There are certain things you're allowed to do. You are not allowed to change the packet name. Inside of your Packet Settings tab, you can type in the little name there. That's actually used to generate some kind of code in the back-end that helps Civil 3D identify which parts and pieces you're using in your assembly. So you may never change those if you want to update a subassembly.

You may never change a parameter name, remove, or add parameters to your assembly. So if you're like, ah, man, my subassembly-- I don't need this parameter, but it's already used in a model that's in production, don't delete that parameter. That would be a very bad thing to do. Just leave it alone until the users potentially doesn't do anything or disconnected from actually doing anything.

You're allowed to go in and change formulas. So if your thing is from 0.1, go 2 feet right, and you've decided you now want to make it 3 feet right, you can still go in and change those things, and it will automatically-- after you import subassembly again-- it will automatically update that subassembly for you the next time you update your preview.

You may add and change variables and target parameters. So if you need to, oh, man, I need to go

and have this target the surface instead of a profile, that's something that you can do. And you can also do-- variables are just internal to the subassembly and they're not exposed to the user, so you're allowed to adjust those.

You may adjust the order of your build. So instead of going 0.1, 0.2, 0.3, you're like, ah, I actually need to calculate 0.3 first, you can readjust the flow of your subassembly. That won't affect your ability to update it in Civil 3D. You may adjust the defaults for the parameters.

And one thing that surprised me was you can actually add new geometry. So if you have a subassembly that's there, and you're like, oh, man I need to add a new shape over here, you can actually go in and add a shape and re-import that subassembly, and Civil 3D will automatically fix that the next time you update that preview or rebuild your corridor.

So in this particular case, this is the window for Subassembly Composer. Kind of a way to think about it is you can change just about anything over here. You can change anything in this portion of the window. This will automatically update. And then you can change anything on these tabs.

The things that you can't change are on these two tabs right here. Don't change the things on those other than your default values over here. So this is kind of your sketchy area, and then don't change those. Cool?

So if you do change those two areas that are marked as red, you will receive that when you go to update the subassembly or update your corridor. And if you're updating your corridor, you'll probably get thousands of those messages, which is re-affirming and makes your day wonderful.

So process of updating a subassembly-- we're looking at an assembly right here that has our quadrilateral from last year, and it's currently in my Preview window and also built in the corridor that's not shown at the moment. We've gone into Subassembly Composer, and we have added this little new triangle point. We added this point and then also, two links and a shape. So we've kind of changed the geometry pretty drastically of that particular subassembly.

I'm going to delete that subassembly from my tool palette, and it's going to say, are you sure? Yes, I'm sure. I'm to go ahead and close my Civil 3D file. I don't need to save my changes. Close Civil 3D. And then I'm going to make sure that my name hasn't changed. So I'm checking those things that I said, don't change, making sure that they're not changed.

I'm going to save over my old packet file, and I'm going to hit Save, yes. And then I'm going to restart Civil 3D, and this'll be the fastest Civil 3D start ever. That's the 2018 version, guys, don't worry. It's

that fast.

But so now the Civil 3D's done starting. We're going to go to the tool pallet. And before I open my other drawing, I'm going to import that subassembly that we just overwrote the packet file for. And when I pull it in, it's going to put it over on the Pallet tab. And then I'm going to place it. I'm going to refresh the image so that way, I can kind of see a visual for what I'm looking at.

And then I'm also going to insert it into the drawing 1 that appears on Civil 3D as a detached subassembly. And what that does is when I refresh it and insert it like this-- it actually pulls in that new packet, the new DLL, the new code that's created. And then when I go and open up my other file, it's already swapped out.

So Civil 3D doesn't even really realize that I've changed my geometry, but as soon as I select this and change a property or a parameter on it, it's going to go to that code and execute it. And even though I've added new code, it doesn't seem to care or really even notice. So as soon as I hit Enter, we can see I've added this extra link right here, and it's added that shape over there.

So that was one thing that-- I'd kind of been preparing for this class, and I was like, oh, I didn't even realize I could add shapes, because I went through and looked at everything you could do to a subassembly to figure out what you could and couldn't do. So that being said, that could be really helpful on projects as you guys are getting in and starting to realize that there might be a scope [? change or ?] something.

So we are going to make one subassembly today. This is a very basic subassembly. This is going to be our super elevation to slope. And this is a subassembly I've been using in the past year pretty frequently. It allows you to specify a super elevation at the beginning of your execution of your assembly. And then you can pass it as a parameter. So that way not all of your subassemblies have to support superelevation, because sometimes it gets pretty tricky on how you do that, and it's been a little bit of a easier ride to be able to do that.

So again, we're just able to select the superelevation once, and then we can pass it to everything else that we need. So in Subassembly Composer, I am going to come over here, and we're going to call this Super_Elevation_To_Slope and "converts a super elevation to slope and passes a parameter. Longest description in the world.

And we're going to come over here to the Input/Output tab, and I'm going to add a few variables. And I am going to change the type of these. This one is going to be a superelevation type. This one is

going to be a string. This one is going to be a slope. That's going to be an output. So this is where we're actually going to send out that superelevation information as a slope. And then this one's actually going to be a slope as well.

The string-- we're going to call it code_point, because we want to be able to allow the user to choose what this point is represented as in the assembly. We're going to give it a default of marked point, which is kind of in the default Civil 3D template.

We're going to change this name to slope_out so that way we know it's going out to the world. And then this one is going to be called "default slope." So I'm going to just add a point to my window here. And I'm going to give it code_point for the point codes.

And so when I go and build the subassembly, it's not just going to put code_point as the style. It's going to look at this parameter that the user can change. And in this case, it's going to add it as marked point. The other thing I'm going to do is say Set Output Parameter. And then I can choose that Slope_Out parameter that I created just a second ago.

And the other thing that's already in the subassembly is this used superelevation one. Right now it's grabbing the left inside lane. And I have this really cool formula that I created that looks just like this. If you guys want to take notes, go ahead and jot that down real quick.

That is posted in the handout, but all it does is it goes through and verifies that what the users selected actually exists on the alignment. So that way, if they select left inside lane but there is no superelevation for left inside lane, then it uses the default that the users entered. So that way if you're at a point where you don't have that information yet, it's not going to cause your subassembly to throw a whole bunch of errors.

So I'm going to put that into this [? inter ?] [? VB ?] expression. And then it is good to go. So I'm going to go ahead and save this as this subassembly right here. And then now we are going to go over to Civil 3D and begin the fun.

So the first thing we're going to do is we're going to create a new assembly. And this is a brand new Civil 3D Drawing 2 file. I just created it as you guys were walking in. I'm going to call this one [? "right ?] [? curb." ?] Click anywhere. Press OK.

And then we're going to come over to our tool palette. I have a tool palette here that I'm calling AU just to be able to sandbox some of these things so you don't have to see all the other subassemblies

that are going on here. These are just the generic ones that I'll usually pull in since I end up using those at some point in the assembly.

I'm going to right-click on my tool palette name over here, the name of the tab I'm going to say "import subassemblies." And I'm going to go ahead and import all the subassemblies that we need. So again, these subassemblies were created in last year's class. I'm going to come over here and refresh these images, just so that way, we can see a little picture here instead of a giant white box. Refresh image.

And then this vertical deflection is literally a copy of the vertical deflection that you are given as a standard subassembly in Civil 3D. One thing that doesn't do is it doesn't perhaps pass a parameter for the total height of that vertical deflection, which they have a beginning and an end, but even if you run those two values and subtract them, you don't get the actual height of it. So I'm not sure exactly what they're passing, but I'm not 100% confident they're labeled properly.

So this vertical deflection-- again, it's up on the website, but it just is the same exact thing, looks exactly the same, behaves the same, except for it passes a few more parameters that are more useful if you're doing that. So the first thing I'm going to do is place a superelevation to slope subassembly. And I just placed it, and it's got that little triangle. So I'm going call this R-curve SE. And I'm going to go ahead and give it a negative 2% for that default slope.

And then I'm going to place the quadrilateral we created from last year-- I'm going to place it on that superelevation point and place another one. And I'm going to do three of these, in this case. I am going to select this first one. I'm going to name it Gutter. This one is going to be called "lip." And then this one's going to be called "top."

And then I'm going to select all of them. And all of them right now have this outside slope defined as a one-to-one. So I'm going to change the outside slope to a zero-to-one so that way it doesn't slope out like that.

I'm going to change the rest of it in just a second. So here's our subassembly-- or sorry, our assembly the way it looks at the moment. I'm going to come into the Assembly Properties window, and we are going to click on our gutter. I'm going to give it a width of 1. And a inside height of 0.5 is perfect. And then I'm going to take the top slope and have it equal that [? superelevation ?] [? to ?] slope out parameter that we created just a few minutes ago. I'm going to tell the bottom slope to do the exact same thing.

Now going to come to the lip subassembly. I am going to give the top slope a value of 0.66 to 1, because we want it to go up as the gutter lip should. I'm going to change the width here to 0.33. And the inside height-- I'm going to tell it to grab it from that gutter that we just created. I'm going to say, grab it from the outside height of that gutter. And then the bottom slope should be whatever that gutter bottom slope was.

So I'm going to hit Apply and OK. And instantly you saw it shrunk there. And now it's starting to behave dynamically. If I select the superelevation piece and I change this to a positive 5%, you'll notice that both the slope of this gutter piece as well as the lip bottom updates, but that top holds true to that 0.66 to 1.

I'm now going to come in and set up the final passing for this top. So in this case, that top-- we're going to set the inside hide of the top to be equal to the outside height of that lip that we created. And I'm going to set the slopes equal to the bottom slope of the lip. And then we are going to set the width to 0.5.

So now I have a curb piece here that's made out of this one subassembly. And I can change it to a negative 5% slope here. And you'll see that all of that responds just like that.

And that's a really basic example. That's nothing that's too crazy. But it's this idea that you can take one or two subassemblies and be able to use those in that math operator [? difference-- ?] in this case the superelevation and slope-- to be able to create something that's a little bit more complicated.

So the next thing we're going to create is a left MSE wall. Before we get too far into it, I need to go in, and I'm going to add a new code for shapes here. And I'm going to add a fill material, and we're going to give it the name of "engineered fill." Apply and OK.

OK, so on this one, we're going to start out with our vertical deflection subassembly. I'm going to place two of them, one and two. I'm going to call this first one, call it "exposed wall height." And then this one, we'll call "embeddment depth."

I'm going to set this to negative 2. It seems to be pretty common depth for soil embeddment. And then we are going to change this to a negative 10.

And so those two pieces'll just vertically deflect based on whatever property we give it there. We're going to go in and place another quadrilateral here. And one thing that's important right here is it's much easier to actually click on the links when you're placing new subassembly components than it is

to click on the points, because you'll have four points that are stacked on top of each other. But if you click on the link that represents the subassembly you're trying to connect it to, it'll actually snap it to the closest point.

So I'm going to click right there, and even though I clicked on the link, it's going to snap it to that point right there. Here, I'm going to make the coping cap. So I can actually type in "horizontal," and it will make that top slope completely flat. We want to do the same thing with the bottom slope. I want the outside slope to be vertical. And I'm going to change the top width here to 0.85 and the inside height to 0.33.

I'm going to copy this piece right here, and I'm going to name this one "CopingCap Top." This one'll be called "CopingCap Leg." And we're going to give this a-- and so I copied this CopingCap Top to this new location. So that way, I don't have to adjust the slopes again since I'm trying to build another rectangular shape. But the top width in this case is going to be 0.2, and the inside height is going to be 0.5.

OK, so now we have this little [? pre-cast ?] coping cap just out of two of those quadrilaterals. And then now we're going to go in and use that math operator that was from last year's class. I'm going to place two of them on top of each other. And I'm going to go to the assembly properties to rename them.

So two of them are on top of each other, and they look exactly the same. It's really difficult to tell the order of them when they're on top of each other like that. But since they have the same name, if I pass a parameter and try to connect to one of these, it's going to freak out on me.

So I'm going to open up the Assembly Properties here. And I'm going to call this one "math op." And then I'm going to call it "total height." And then I'm going to call this one "math op." And then I'm going to call this "th" for total height. And I'm going to say "less cap height." I'm going to hit Apply and OK.

And now when I go back into my subassembly, I can pass all the parameters, and it'll show me those new names that I just gave it. So in this case, I'm going to say I want my parameter A to be the delta Y from the exposed wall height. And to that, I would like to add the delta Y from the embeddment depth.

And I'm going to go to the next math operator, and I'm going to say, grab the result from that total height addition. And this time, I'm going to subtract the CopingCap Top inside height. I'm going to hit Apply and OK.

Now I'm going to again copy this subassembly. I'm going to place it directly on top of that math operator that we just set up to receive some parameters. And I'm going to come in here, and I'm going to call this one "MSE panels." And I'm going to set the top width to 0.33. And the inside height-- I'm going to have it grab from that total height less coping cap. I'm going to hit Apply and OK.

And now I have this piece that stretches down and goes to exactly where the bottom of the embeddment depth was. If I select that embeddment and change it to a negative 5, that MSE panel subassembly automatically stretches with it. If I change that to a negative 2.

And same thing happens if I come in here and adjust this-- Negative 8-- or if I come in and just adjust the coping cap for some reason. The supplier said, oh, it's actually not an inside height of 0.3. It's an inside height of 0.5. OK, well, I change that, and again, all of that stuff is calculated at runtime as it's building the assembly.

I'm now going to select this coping cap. I'm going to copy it down to the bottom of the MSE panel. And in this case, I'm just going to move the subassembly, and it's going to apply a constant offset to it. I'm going to call this one the LevelingPad.

And I'm going to give it a width of 1 foot. Actually, I lied. I'm going to move this now. I need it to snap from middle to middle. And that height of 0.5 is perfect.

So I've got this MSE wall that's responding now. The next thing we need to do behind it is the big strap area where all the engineered fill goes. And as a contractor, to me, that's the stuff that's really expensive. It's very time-consuming. I have to do it in a lot of lifts. And so that's where a lot of our quantification efforts go into is to figure out exactly what that looks like. How does that compare to the existing grade, and all that kind of stuff.

So what I'm going to do is add a one more math operator. I'm going to place it right there. And I'm going to select it, and we're going to call it, MathOp. I'm going to call it StrapLength.

So I'm going to go into the Assembly Properties. I'm going to say the A is coming from the MSE panel's inside height. And I want to multiply that by-- we'll start out at 1.2. So I'm choosing multiply from the dropdown.

So it's going to take whatever that inside height is of those panels and multiply it by 1.2. I'm going to hit, OK. I'm going to select this again and place it on top of that math operator we just did. I'm going to call this one, [? Engineered ?] [? Film ?] [? StrapZone. ?]

And I'm going to give it a shape code of engineered fill, which is what we added to our code set style earlier. So now it's got that fill pattern on it. I'm going to come into the Assembly Properties and say, the engineered fill strap zone is going to grab the top width from the strap length multiplier, this math operator that I have, and then it's going to grab the height from that MSE panel subassembly. I'm going to grab the outside height. I'm going to hit Apply and OK.

And so now, as I flex the height of the wall from 8 feet to let's go to 12 feet, it's automatically growing that one subassembly. Again, these are all quadrilateral components. But this one's automatically stretching horizontally. It's automatically stretching vertically. It's responding as if it was a really complicated subassembly. But really, it's one dynamic subassembly and that math operator working together.

So I can also come in here and select just this math operator. And since this is just a value I entered, I can change the strap length to a multiplier of 1 of the height. I can change it to 0.8. I can do all that kind of stuff.

The other thing I'm going to do is I want to model that zone behind the StrapZone. That's not really strap, but it's just some overfill. So we're going to call this, freezone. And then we're going to pass it.

We want to tell it to match the height of that previous StrapZone. So the inside height. And we're just going to set the width of this one to 2. And so now, we have this subassembly that if we go in and change the embeddment depth, it'll go in and respond dynamically, all that kind of fun stuff. But that StrapZone on the backside, the free area, stays at that 2 feet. So it's a good example of a very complicated component that often is in corridors. It's something that needs to be able to respond to what the existing grade is going to do and all that kind of stuff.

So now we can go into our tool palette. We're going to add a separator just by right-clicking. And then I'm going to click on-- how many people have added an assembly to the tool palette before? OK, so a few of you guys. How many people didn't know you could do that? OK, we have three people that are telling the truth. Good job.

No, so this was something that caught me by surprise. I didn't think you could do this. And it's to me one of the least intuitive things. You select the red marker or whatever your style is showing for the assembly. And then you have to mouse over just that red line. And you click and hold on it. And then it looks like you're moving it in AutoCAD, but if you just keep on going and mouse over, it flips out for a minute. But then it lets you add it to-- oh, hold on. I got to save real quick.

So if you click and drag it and then you just let go on your tool palette, it actually adds that entire assembly composition-- all the names, all the parameters you've passed, all the styles you've entered-- it adds all of that to your tool palette. So instead of building a subassembly curb, we've just created a subassembly-- or sorry we've created an assembly called curb, and now we're going to do this with our MSE wall as well.

We're going to go ahead and drop it right there. And then this is, again, literally the drawing we created last year. And we're going to select the assembly. And I'm going to click and drag it over to our tool palette.

And we had it called typical section outs last year. It's not the most elegant name. But I just wanted to be able to pull that specific assembly in.

So now we're in this new file. I have a really basic alignment here. I created a very rigid surface out of some 3D polylines. We're showing two profiles up there. One's proposed. One's the existing. And then I have three sections over there on the right.

And we're going to go in and create a corridor based on the three assemblies that we just imported into our tool palette. So I'm going to start out by placing the typical section roadway assembly here. And one thing we didn't have last year was that superelevation to slope. So the only way to do that is to go in and manually change your slope. Well, so if you had superelevation, you were kind of out of luck.

Well, if I use the superelevation to slope and I choose this Insert option-- and I'm going to choose Before. And then I'm going to zoom in to this top layer of my roadway. And again, I'm going to click on the line of that subassembly and not that point. And then I'm going to hit Enter to accept that. And if I look at my assembly properties-- I lied. I didn't place it right there. Superelevation slope. There we go. That is definitely it.

So now if I go look at my assembly properties-- I didn't choose before. Hold on. Bear with me. Insert, and I'm going to choose Before. And then I'm going to click on this subassembly. And I'm going to accept it and except it again.

And then as I look at the assembly properties, it now has that superelevation to slope right at the very beginning of that. So I'm going to press F2, and I'm actually going to go through and rename all of these right now, and I'm going to call them [? right-, ?] so that way, later, when we mirror this, it doesn't go too crazy on me.

And then-- I've been told that I speak quickly. Would you guys agree with that?

AUDIENCE: [LAUGHTER]

ERIC CYLWIK: OK, so I did check before the class that they are recording this, and it will be up. I also have a handout that shows screenshots of all the stuff that I'm passing and how I'm doing all this stuff. So my big thing that I want you guys to take away is that you can make a flexible subassembly and create really cool components that are dynamic, and you don't have to manage a whole bunch of subassembly libraries and all this kind of stuff. You have one or two or four subassemblies that are really cool that can do everything, and that can save a ton of time as far as distributing amongst a large user base, be able to go in and quantify stuff.

So I hope-- it's pretty clear that I'm trying to show that? Yes? OK, cool. So if you guys are like, I didn't see exactly what he passed there, don't worry about it. You can go, and I think they'll have like a podcast mode where you can play it at half-speed, and then my voice will sound really deep, too. So don't be afraid to go back and look at that. I'd actually recommend it. I'd seen some of you sleeping, so that would be a great opportunity to pretend that you were paying attention the whole time.

OK, so I've got all of these named. I'm going to hit Apply and OK and come back into them, come back into that Assembly Properties window. And I'm going to change this top slope from a 2 to 1. I'm going to change it to read that value that's coming out of that superelevation to slope value. So I'm going to hit Apply and OK.

And then again, I'm going to change this superelevation value. In this case, we want to go to the right outside lane slope. And we're going to change it from a 2 to 1 to a negative-- sorry 2% to a negative 2%. And again, you'll see that all of that responds dynamically. In this case, we're even adjusting the slope on the fly.

So if I come in here and change this to a 10%, which is really exaggerated, you'll be able to see that this is following a specific slope based on the layer height. So you can get in and do some pretty neat stuff with some of these things. I'll change this back to a negative 2%. And then once we go to build our corridor, it'll actually read the right outside lane slope and override that default value of 2. But again, this is a good way for me to be able to get in and mess around with a preview and what it looks like.

I'm now going to select all of those subassemblies. So again, that graphic from earlier-- all of these

are logically in order. So I'm able to select all of them and then press this Mirror button right here. And I'm going to mirror it over to the other side of the assembly. I'm going to come into Assembly Properties. And now I have this cool Left category. And I'm going to change all of these to Ls, if you'll bear with me.

And then I will show you why I keep on-- after I rename stuff-- why keep on closing this window and then reopening. It's because if I come here and I say-- even though I've renamed this already, I haven't actually applied the changes to the model. Wow, I'm making a liar-- oh, so if I click on this dropdown list, this dropdown list is populated at the time I pull open the window, not what's actually there.

So notice how that option is that R-superelevation, because that's what it was named originally. Oh, I just messed it up. But if I hit Apply and OK and then come back into it and look at that again, I can say, I want to go to the left one.

So you'll see me come in and either rename it in the viewport. Or I'll come in and rename it all in that assembly window. There and then I will I'm update it by Apply, OK and then going back into it.

I'm going to change this to the left outside lane slope. And I'm going to change it to a 2% slope, so that we don't have a crown in the middle. And now we have this really cool kind of dynamic roadway with a few different materials in there.

But we just built that really cool curb. I'm going to go ahead click on that to grab it, and now I have this curb component. In this case, I'm only going to select these three subassemblies, because I don't want that superelevation one. I want to pass the roadway slope to the curb.

So I'm going to select those three and hit Copy. I'm going to paste it right there. And then I'm going to come in and tell it that this gutter needs to grab the slope from the AC1, which is the top layer. I want it to grab the top slope and pass it to both of these things.

I'm going to hit Apply. Uh-oh, that doesn't look good. Didn't pass it to the slope, did I. I did not. I passed it to the inside height, which it doesn't like. And the inside right on that particular piece was supposed to be 0.5.

So I'm going to pass it to the bottom slope. And I'm going to say, get the top slope of the roadway. And I'm going to hit Apply and OK. And one thing that's kind of neat about when you copy and mirror, if you select all of the logical grouping all in order, it keeps what parameters are being passed within that selection. So if you select four of them and the second one grabs something from the first one,

it'll keep all that association. So the only thing I have to do is change the slope on that first piece because it's passing to those other ones.

So now if I come in and I select my superelevation little subassembly for this right side, and I'm going to change this to a 5%, that slope is now passed to that curb. Change it back to a negative 2%. Now I'm going to select these ones, and I'm going to go ahead and mirror them to the other side of the roadway. And then I just need to go in and again set up just this gutter to be looking at the top left roadway slope.

Oh, jeez. Like I said-- top slope-- so I passed the width. Here we go. There we go. That's a much more feasible curb, huh? OK, that other one was part traffic barrier. That's what it wanted to be when it grew up.

Ok, does anybody have any questions so far? I've been going very quickly, and we will have lots of time for questions at the end.

AUDIENCE: When you track the complete assembly onto your palettes, how would you pass those to other users?

ERIC CYLWIK: Great question. So the question was, how do you pass your palette to other users? And this is a little bit tricky, but if you go into your custom profile settings within Windows, there's actually-- I forget what the palette is. It's a dot-- Does anybody here know? It's a dot--

AUDIENCE: aws?

ERIC CYLWIK: So it might be a .aws. I know in Civil it's a little bit different. I wanted to say there was an x in it for some reason. But anyways, I can send out a message or post it in the additional files, and I can give you the link to that as well. But it's in the app data folder. And each user has their own little profile of what their tool palettes look like. And so we have our custom this is what it is.

And then as we're building new palettes for each project, we actually save those out, because I've noticed that when Civil crashes sometimes, it takes a tool pallet with it, so it's really handy to have all of those backed up somewhere. But then if you just distribute that little palette and put it in the right folder in their computer, all they need to do is restart Civil, and it'll have that whole assembly composition there if it's using that same subassembly. Great question.

OK, I'm going to go back to talking fast.

AUDIENCE: [LAUGHTER]

ERIC CYLWIK: OK, so here, we just placed our MSE subassembly. And again, if I change any of these parameters, everything automatically updates. What I'm going to do now is select all of the subassemblies on that assembly, and I'm going to hit Copy, and I'm going to paste them right there, because I want it to go from that point.

I'm going to select the first subassembly that everything else is based on, just this one, and I'm going to move it from the top right corner of the MSE wall-- or sorry, of the [? pre-cast ?] coping cap-- to the point where the actual material for the roadway terminates. And then sometimes that update automatically happens. Other times you have to type in, [? regen, ?] which is different than the [? Region ?] command.

So we regenerated that portion of the model. And then I'm going to come back to my tool palette now, and for this other side of the roadway, I'm going to do a length slope to surface from this curb. And I'm just going to do a 3 to 1. So now we have this complete roadway composition. All of it is using the quadrilateral, the math operator, the superelevation to slope, and then the vertical deflection over here.

So with those four subassemblies, I'm going to create a corridor. And we'll call this, AU in class. And basic style's fine. We're going to choose that alignment. We do want our layout 1. I spent a lot of time naming that profile, as you can tell.

And I'm going to use this typical section outs-- because it's that one that we've been working on, so it's going to be this guy right here-- for my target surface, which is what our vertical deflections are going to be targeting. I'm actually leave that blank, because I only want that first vertical target to target the surface. I am going to leave the set baseline and region parameters checked.

I'm going to come in here to the target. And for the exposed wall height, I'm going to choose that existing grade surface, also cleverly named, [? surface ?] [? 1. ?] And then I'm going to hit OK.

And if I look at here at the section, sorry, and add in my AU in-class corridor-- we're going to give it this all-code style that we're using. Hit OK.

Nope, I didn't use my dtm for the the link slope to surface. Let me add that. OK, much better. Less angry.

So now we can see that this roadway is responding to those superelevation changes. Here, we have a really extreme example where it's-- I geared it up to 12%, just to kind of show that it is truly

responding to it. Then we have a 3 to 1 slope over there.

And then if we come in and change the superelevation, we'll change it to a negative 12% on this side, so it's got a giant crown in it. Or actually, we'll change this side to 12, flip it around on itself, rebuild our corridor. Nope, didn't take it, but it's got that giant crown in it now.

And so also, as the closer this gets to it, that MSE wall, again, is changing the amount of engineered fill that it has. And I will say, if you get in and play around with the math operator a little bit, it is very possible to make it respect a minimum strap length. I'm not going to cover that today, but that is possible to do.

So the other thing we can do is-- just like last year when we went in and targeted our 1, this top-- all right, we want to go with the left side in this case. We're just choosing the top roadway material. I can have it target a crazy little link that I put in here that I can't find. Where is it?

I'm going to draw a new one. We're going to go off the beaten path here. I'm going to go ahead and just draw this new line in and make it look like it's connecting back to the curb, select my corridor, and add a new target for the left side of this roadway.

So this AC1, which is the top material-- go ahead and do that. OK and OK. And as the corridor rebuilds, the entire thing stretches now to be able to get out over to that point.

And then if I go in and make that much closer to the corridor and rebuild, just by changing that one thing, our MSE quantities or the engineered fill quantity's going to change and update and all that kind of stuff as well. So now for quantities-- let me change-- this superelevation is killing me.

So now I'm going in and looking at the quantities. And we can go to the Analyze tab. We're going to click on Compute Materials.

And we're going to choose the materials list. And in this case, I just looked at my corridor and figured out, hey, you have these different shape names. And we're going to call the pavement material Pave1. The base material's going to be Base. And then the subbase is going to be creatively called Subbase We're going to hit OK.

And then it has automatically calculated the materials. You can add it to your section views to show incrementally as you're moving along the alignment. But in this case, we just want to look at our volume report. I'm going to choose my alignment, my sample lines, and then you need to select the Select Material style sheet. And so I did that.

It's not quite clear that this is actually changing what your report looks like and what information is displayed. But you do need to click that Select Style Sheet little folder button. And then I'm going to hit OK.

And then now we can start to see that it's calculating the area of the section of the engineered fill at each station and then also the volume incrementally as well as cumulatively. And that's true for all of my roadway materials as well. And again, all that differentiation between those subassemblies, all that different coding for all those styles, all comes together in this one assembly that you can reuse.

So now if we have a different project that has a curb on each side of the roadway and needs an MSE wall on the left or we need something on the right, you can conditionally kind of combine that with conditional subassemblies inside of Civil 3D to create some really complex things, go in and quantify it, pull out information about maybe where the gutter flow line's supposed to be, all that kind of fun stuff.

So that is being able to get in and use these reusable subassemblies. Again, we have really, really complicated examples like that runway model that we had at the very beginning of the class and then the middle-level complication that we just showed here. We're through all the live demos. Does anyone have any questions about the live demo really quick, and then I'll jump on after that?

OK, cool, so today we spoke about being able to get in and take a look at what happened last year, what we covered. If you guys want to go down that route, that class is are available online for streaming. You can go through, and I talked just as fast, if not faster, last year, so something I've worked on.

But again, we talked about the ability to pass parameters between subassemblies and how important that is, the importance of being able to have this flexible subassemblies, and how important it is to have that math operator, so that way, you can make consistent adjustments at time of execution. We can look at the rules. We've looked at the subassembly use ground rules as far as what you're allowed use, what you're not allowed to use.

We used those four assemblies today, and we created that fourth and final one, that superelevation to slope. We created a sloping curb assembly, the basic MSE wall assembly. Ultimately, we compiled those different assemblies together in an actual full roadway assembly that was dynamic and responded to different superelevation and changing surface information. And then we ultimately pulled values out of those.

So thank you guys so much for attending today. If you are done, don't have any questions, feel free to leave. I appreciate your guys's time and hope that you guys are able to download this from the AU web site and use it on your guys's projects. Thank you.