JOSE ELIZARDO: Time-- even though it's after lunch, I do have an hour and a half of stuff to show you guys. So-- and out of respect for your early arrivals, we'll start on time.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Sorry?

AUDIENCE: Do you have any preference as far as graphics [INAUDIBLE]

JOSE ELIZARDO: Yeah, I mean, it depends what your output is. Like, this is-- what you're looking at right here is real time. It's a game engine.

So that requires a pretty beefy graphics card. And depending what you're doing too-- if you're doing VR, right, there's a whole other requirement to do virtual reality too. But just a display of these kinds of graphics requires pretty good graphics card, nothing too crazy.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah.

AUDIENCE: I just don't know if it's better for that type [INAUDIBLE]

JOSE ELIZARDO: So for real time, gaming the Geforces are always better. But in terms of content creation with our tools like Max, Maya, Revit, stuff like that, you need-- it's better to have a a QDRO. More memory, more-- I'm running a QDRO.

Like, I'm not running a Geforce at all, yeah. Yeah. Mike, I'm honored. You're in my class. How are you?

[SIDE CONVERSATION]

All right, let me see where we are with time. Three more minutes. Yeah, sure.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: I'm not a CFD expert. I'm the guy who takes the CFD content and visualizes it. But from the demos that I've seen, yeah, I could do AutoCAD simulation. I couldn't provide more context than that but, I have received water simulation.

AUDIENCE: [INAUDIBLE] somebody who can hopefully give me a yes or a no on a--

JOSE ELIZARDO: Well, I have received water simulations from CFD to visualize. So it's doable.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: The water that I've seen has been not so big. But you know, you'd have to-- I could probably find out and see to what extent it's doable. I'm sure it is but, I couldn't provide any more context.

We have other tools like-- I don't know if you've ever seen Maya's Bifrost system. It's an ocean simulator basically. So if you want to do oceans and stuff, that's really-- like, it's like, large scale. They use it in movies and stuff to do ocean simulations, an ocean vis.

But it's not a parameter driven tool like CFD. It's not condition based. It's probably not physically accurate, it's just beautiful to look at.

AUDIENCE: That's more than we're looking for [INAUDIBLE]

JOSE ELIZARDO: Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah, yeah, yeah.

AUDIENCE: --make it look

JOSE ELIZARDO: Yeah. Yeah, well, you could probably look into Bifrost. That would be an interesting-- at least look at the videos and see if it satisfies what you guys are after.

But yeah. But simulation two-- simulation [INAUDIBLE], fairly certain it can do larger stuff than what I got as a file. But you you'd have to find out.

AUDIENCE: OK.

JOSE ELIZARDO: Cool.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Sure. How are we doing on time? Hello. I think I'm going to get started--no, two more minutes.

I started going on. What do you guys want to do? Give it some time? [INAUDIBLE] cool?

Oh, you're not certain. Hesitation. Tall tale.

So what do you guys want to get out of this class? That's what I want to find out first. What do you guys do?

Can you give me a kind of synopsis here and there what you guys do and what you want to get out of this class? What's your goal here? What do you think this class is about? What was that?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: OK. Speak up?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Cool. That's one thing we're going to look at.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Same? Any more? Mike, what do you want to get out of this class?

AUDIENCE: [INAUDIBLE] I guess visualization possibilities.

JOSE ELIZARDO: Sure.

AUDIENCE: You know, [INAUDIBLE]

JOSE ELIZARDO: Yeah, telling a story.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah, cool. I think we should be able to cover that stuff. I was hoping you guys were going to tell me, I want to learn all about simulation. Because I'm not the sim guy. I'm the vis guy.

We are going to do a little bit of sim, though, just to keep it in check. So I'm going to jump right in and start because it's 1 o'clock and you guys are here. So what do you guys think we're looking at on screen?

That's the first thing I want to talk about. And forget this here. Pretend this is not here.

What does this look like? Unfortunately, I can't block that. So-- I mean, I know it's arrows and they're colored and it looks like an acid trip.

But it's not. It's the stadium. What's that?

AUDIENCE: Wind simulation?

JOSE ELIZARDO: Wind simulation-- you got it. So this is-- and that was not a trick question. We often ask ourselves, how can we visualize things like wind. We don't see wind.

And how can you interpret wind in a visual way that actually resonates with people and makes sense? So that was actually just for my own etiquette. Does this look like wind to you? Do you think that this is wind?

We're going to talk about basically this project for the next hour and a half. I'm going to show you guys the nuts and bolts of how to put this together, where this started from, and what the outcome is. The other thing I want to talk about real quick is this is a looping video, sure.

But this is actually outputted from Stingray, the game engine that some of you guys talked about. For those of you not familiar, we own a game engine-- Autodesk. So just like Unreal and Unity, Stingray is a game engine just like those.

And so this is actually a fully interactive environment, a fully interactive project. Right now, we just see wind simulating from the west at nine miles per hour. But we have a whole bunch of other simulations in here that I'll show you guys at the end of the session.

So jumping right in, here's my beautiful title. My name is Jose. I've been at this for about 10 years now.

I've done a few different roles since I've been at the company and I've been on this team for about three years, my third years teaching at AU. And I'm basically a tech specialist. I work with primarily 3ds Max, also Stingray as of recently, other products I dabble in. But those are really my bread and butter, if you will.

So class summary-- I won't ask you to read this and I won't read it to you. They make us put this in here. I'm assuming you read this or else you wouldn't be here. Of course, also our learning objectives-- I think for the most part, we're going to get through this.

They make us write this like eight months ago. So things are in flux and change a lot. But I think for the most part, we're going to get through most of these learning objectives.

And one giant disclaimer-- I've been practicing my session in the last few weeks and I realized I have way too much. And that's probably every speaker's-- every speaker falls victim of that. So I'm going to touch base on some concepts, deep dive into others that I think are more important.

That said, the handout-- did you guys download the handout? Did some of you get it? So the handout covers every single thing that-- it was part of this project, all the details.

And the hand out actually has also embedded little videos. So you need to install the flash plug-in in order to get it to work. But every so often, there's these little embedded videos that highlight some of the workflows.

Other disclaimer-- this room is 90 people and there is like maybe 40-some odd people here. So-- and thank you for making it. That said, let's keep this interactive.

I know everyone says that, but ask me questions. If something's not clear, you want me to deep dive, double click onto something, let's do that. This is really about you guys.

It's not about me. I want you guys to walk away with something here. And so just make it interactive. Ask questions and let's keep it flowing. Cool?

I have a skeleton to follow, a sort of high level overview of things to follow. But for the most part, we can go in any direction you guys want. Before I jump in to the actual workflows, I want to tell you guys a little story about Fox, the sports broadcast channel.

So Fox, back in 2014, they were interested in potentially leveraging wind simulation effects and having those being overlaid on top of their broadcast of Super Bowl 2014. So they worked on a project called Fox Weather Trax.

This Is public information. You can go on Google and YouTube and research this. And at the time, they were using a combination of Autodesk tools, non-Autodesk tools to achieve this. But they had a lot of problems, right?

I mean, these tools are not necessarily connected in any meaningful way, required very advanced expertise to be able to connect all this. It wasn't very flexible. You couldn't really change things interactively or fast.

And so it was a bit of a challenge but they still managed to put something together. This is an early prototype image of what they were able to generate. But this effort got the attention of the 3DS Max development team and they spent the last couple of years building out a bunch of tool sets inside of Max that allow for a much more streamlined approach to visualizing simulation data.

So this is what we're going to look at basically today. This is the proposed ecosystem of products that make this happen today. So we have Max at the center of it all. And I'm going to come back to that in a second. It's an important point.

So I'm going to leverage that Max's SketchUp importer to bring in a model of a stadium. One of the great things about SketchUp-- and this could have been a Revit model. This could have been [INAUDIBLE] model.

This could be any model that you guys have. Great thing about SketchUp models is there's like a whole bunch online, free [INAUDIBLE]. The warehouse is huge.

I don't model stadiums for a living, so I was about to model a stadium. And Max has a very elegant importer for the SketchUp files. So I'm going to show you guys that real quick.

Then we're going to leverage Autodesk CFD. Who uses this tool, by the way? Does anybody use this tool? Does anybody do simulations all together? What tools are you using?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Why aren't you using Autodesk CFD?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: I'm just kidding. Hopefully, you're going to want it by the end of the class. So Autodesk CFD, I'm not a sim guy.

I know how to go into it, do the couple of things that I want-- in my case, wind sim-- bring it back into Max and do some stuff with it. So I will give you guys a quick little overview and we're going to do some stuff in it-- in our case, like I said, wind sim. Then we're also going to leverage another tool called InfraWorks 360.

Do you guys know that tool? A couple of nods, a couple more faces-- cool. So [INAUDIBLE] 360-- again, I'm not a civil engineer. I don't use it for civil engineering tasks.

But just as a tool to generate GIS based cities, real world contact cities, it's effortless. It's all done on a cloud.

It's super easy to come in [INAUDIBLE] files into Max. It's awesome. So we're going to leverage that tool to create the contextual surrounding.

We're going to massage the whole thing in Max and we're going to push it into Stingray. Cool-- that's the workflow we proposed to Fox and they were pretty excited about it. And I want to make a big important point here that 3DS Max really is at the heart of all of this workflow and it's at the heart of a lot of workflows inside of Autodesk all together.

Max is the bridge that connects all these tools together. It's not necessarily just about input and output. It's about bridging all these tools together in ways that no other product can really do. So Max is-- and it's really important, if walk away with one thing from this class, is that Max is your tool of choice for bridging all these technologies together. Right, so I split up the presentation in four segments, if you will.

The first is-- Ben, tardy arrival. How are you? Sorry to single you out.

First we're going to talk about is wind simulation. So for wind simulation, we're going to bring in a model of a stadium. In my case, I'm using SketchUp because that's what I like to use.

Then we're going to look at some very basic modeling tools inside of Max to create a proxy version of the stadium. So the stadium in its current form is way too complex for a simulation in CFD and there's also problems with the model holes that you need to solve. So instead of doing all that, we're going to create, a proxy version or a low res version of that stadium. I'm not going spend too much time on this because the point of this class is not to learn how to model, but it is an important step in the process.

And then of course we're going to do some wind simulations. Let me put this in my pocket and keep going. Some wind simulations with CFD-- so let's start off with the SketchUp importer.

Here we go. So from Max, the application button, you can go to Import and you can locate your SketchUp file and import it. You get the SketchUp importer dialog.

It's still called Google which it probably shouldn't be because it's no longer owned by Google. But you've got a couple of parameters in here. The two that I'm concerned with are split object by layer to keep things tidy in the Max project and the daylight system.

I don't care about the light itself but the daylight system comes with a compass. And the compass is really, really important for this workflow because we want to know what true north is and what true south is, west and east, so that when we create our wind simulations, we know where we're simulating wind from.

So the V model imports this way. We get the SketchUp file there. And when we select our compass, we see that we've got the north, south, east, west telling us what's going on here.

And the model, because it's a SketchUp file, they are geo referenced. And when they come into Max, they are correctly aligned. So I can trust that. And if I Google that particular stadium, which is the Keenan Memorial Stadium in North Carolina, in Google Maps, you see that they are represented the exact same way.

I didn't rotate this model. This is exactly how it was brought into Max. Right, so this is a really good starting point and I can trust that when I do my sim, wind will be accurately coming from the direction that I specify, if you will.

All right, so now we got that. Let's go back into Max and take it from there. So we got the model in.

The next thing we need to do is create a proxy version of this model. Really what we want to do is create volumes that represent the major parts of the stadium. It doesn't have to be super precise, at least not in the case that that I'm working with.

We just want to go ahead and remove all these sort of, you know, really complex pieces of geometry. So for example, this bleacher model here, this area here, it's a curved surface. To recreate that, what I'm going to do is use the spline object inside of Max.

So grab my line object here. I'm going to turn on snapping on the main toolbar. If I right click, I get options. I can snap to vertex. Make sure that's enabled.

Turn on my edges. And I approach a vertex. I get my little snapping cursor.

So now I can start drawing my line out. And I can still pan and zoom in the viewport to facilitate this process. So I'm just going to go ahead and create my line object following the contour of this shape here. And I'm not going to do it super precisely.

The point right now is just to show that this is very easily done and you don't have to be a modeling whiz to be able to achieve-- whoops, that was the wrong one. Let me get closer and keep going. Right, so let's just go ahead and close the loop here.

I've closed my spline object. Oops, I just lost my mouse. There you go.

Close the line, there you go. And when I isolate my spline object, you see we have this nice curved surface that we can start working with. I want to give this guy some volume. So I'm going to go ahead and add an extrude modifier from the modifier list.

Let's look at the extrude modifier, give it a little bit of depth. And now we have this volume that we can work with. All right, so let's go back and look at another way of creating other parts of this object.

The stadium-- that's of course the main component. And if you look at the stadium, really what you have are this exterior shell that you can guide yourself with this part of the model here and then an interior shell that's carved into it to create the bleacher area and the field. So let's go ahead and see how we can create that kind of easily.

So I'm going to create a cylinder, start off with that, put it into my scene like that. I'm going-- whoops. And I'm going to line it up using the align tool to the field so that it's oriented the exact same way.

Let's try and find the field. There we go. We're going to set our orientation x, y, and z to be the same. And we're going to set our reference coordinate system to local. So now when I move the sphere or the-- sorry, the cylinder and take off snaps, I'm moving it within the same alignment as the stadium there.

So let's go ahead and modify this guy. We're going to remove the height segments. We're going to scale it out a little bit so that it fills in roughly that space right there.

If I go into top view and turn on X ray mode, Alt-X, I can scale this guy in a little bit so that he better represents that shape right there. And when I come back in here, I have already the outer shell has been created.

I'm going to adjust my height a little bit so that it's better positioned right there. Now I'm ready to keep going. Next thing I need to do is create the inner volume that I'm going to carve out of the outer volume using Booleans. So to do that, I'm going to scale a copy of this guy inwards and make a copy, not an instance. And then I'm going to adjust the height there so it pops out a little bit so we see it. And I'm just going to push it up a little bit more.

So the next thing we need to do is scale this guy out so that he's representing the inner volume of the stadium. So to do that, I'm going to convert to editable poly. Right click, grab my upper face here, and I can start scaling this guy out.

I can scale in all directions, something like this. And this is kind of rough right now. We're going we're going to fix this as we keep going. Just gives you a rough idea of what we're trying to achieve here.

And then the last thing we're going to do is just take these two guys and isolate them and carve this guy out of this guy. To do that, grab my base model, go into compound objects, grab a Boolean.

This is a brand new Boolean system in Max 2017. I'm going to add an operand, which is this guy right here. I'm going to set it to subtract. And now I've carved out that part of the stadium.

What's really cool is I still have access to that operand itself. So if I double click on the operand, I can still go back and select a face. I can still scale it out to adjust that crave action that I've just done there to really precisely create the effect that I want to create. So Boolean's a really easy way to model the surfaces and these volumes without having to get into edit polys, vertices, edges, and all that complicated stuff.

All right, so the last thing around modeling that we need to do before we send these two guys over-- you know, we would, of course, complete the entire model with different volumes. I want to send this now into simulation CFD to create my wind simulation. But to do that-- so to do that, I'm going to use the IGES file format, which is one of the file formats that Autodesk CFD supports and Max can export.

But before I do that, I need to convert these guys into body objects. Right now, there editable polys and the IGES file format doesn't know what to do with that. So to do that, I'm going to select these volumes right here. I can come down into this list and expand it.

I have body objects as a sub item. And I can just click the body object button right there. And now I turned this guy into a body object. So I would do this to all these volumes that I've created, turn them all into body objects to get further tessellated. And now they're ready to be exported out as an IGES file.

So let me fast forward to a portion that-- to a part that is all modeled. So this is basically my proxy version of the stadium. You see it's kind of crude.

It's a crude representation. It doesn't have to be super precise. The idea is just to recreate those main volumes.

So once I've done all this, I can select all these objects I can go to Export, Export Selected. And then we're going to make sure we select IGES as our file format. Give a name.

Stadium is fine. Override that guy. We're good to go.

So now we can take that IGES file and bring it into CFD to create a design study. Let's go back into PowerPoint and see how that's done.

So if I click this here from within CFD, we can create a new design study, browse to that stadium IGES file on disk, give it a name. It's pretty straightforward. Hit the create.

And you're pretty much good to go. What's really cool about this tool-- and again, I'm not a sim guy, so thank God it's like this or else I wouldn't have been able to do this. It kind of guides you through the general process of creating at least a when simulation.

It's pretty straightforward stuff. So you get this geometry tool dialogue that appears and it's asking you to fix things. It's asking you to merge certain edges that are too close together.

It's asking you to remove small objects are not necessary in the simulation. And then it's asking you to create an exterior volume for the wind tunnel. Now there's rules that you generally want to follow when creating a wind tunnel. I actually documented them in my hand out. So there's some images that you can refer to.

But here, you just drag out your handles, position it how you want, scale it out how you want, hit Create, and your wind tunnel has been generated. The next step-- and what's really cool is if you follow essentially the ribbon, it kind of tells you. You know, you've got geometry tools.

Then we're on to materials and then boundary conditions, et cetera, et cetera. So right now, we're in materials. And basically, what we need to do is define the different materials that are inside of our projects.

Example-- the wind tunnel is going to be defined as a fluid air. And then we can hide it, get it out of the way. The stadium is going to be all steel in my case. Again, this is my project.

If you are using different kinds of surfaces, you would define those as they are-- humans, gypsum board, plastics. The next step is to define some boundary conditions. And this is basically telling the system, hey, where is air coming in from and where is air exiting the tunnel?

So we're going to grab the right hand side of the tunnel because we want air coming in from the east. We're going to define it as velocity set to miles per hour and we're going to give it 20 MPH. So wind is going to be coming into this tunnel at 20 miles per hour.

The next thing is define the exit. So we're going to set this to pressure and pascal. And that is basically telling the system air is exiting this way.

At this point now, we've got to mesh it out. This is all automatic. You hit the Auto Size there and you get-- you mesh out your volume. Your volume is the way the Auto Size does that.

At this point now, you basically just solve. So hit the Solve tool, the Solve button. Set your iterations to whatever you want.

A recommended iteration amount is anywhere between 200 and 300 iterations to get a physically correct simulation. I used 100 just to keep things going. And this actually takes a lot longer than what you see there. That was a sped up video.

But once you have simulated-- this is where things start to get fun-- you come to the Results tab up here and you have different ways of visualizing the simulation, even directly in the tool, in this tool, without even going into any other product. So what we can do is create-- I like to create traces-- so paths that are going to follow through and traverse the stadium. So I set myself up here as traces.

And then there's different ways of creating these traces. I'm going to create a rectangular grid of traces and I'm going to set my seed density to something a little bit lower to create a denser grid of traces, if you will. So once I set these guys up-- and if you expand these, you have different ways of creating your traces. You can create circular patterns.

You can create rings, et cetera. There's a few different ways that you can draw these out. I'm going to hit the Add button. And then I just need to basically click to create this series of red dots.

And basically, the density of these red dots is driven by your seed density up there that I set to 0.5. So let's just do something like this, hit that. And then boom, we have our traces inside of our stadium representing the wind simulation that we just did.

All right, now what you can do with this, a few cool things we can do to visualize this directly inside of CFD. We can animate these guys. I hit the Play button here.

We get them animated. It's going a little bit fast. We can slow that down a little bit.

We can also take our generating path here, our generating shape here, and maybe just drag it out a little bit to create a completely different wind simulation. But it's always following the settings of simulation that you defined a little bit earlier. Let's speed this up a little bit.

What we can also do is modify the look and feel of these traces. Right now, they're just tubes. So if we go to Edit, we have some settings in here that we can change.

For example, we're set the cylinders. Like I said, you can set them to lines. I'm not particularly a fan of the lines.

What I do like, though, are the comments. The comments are kind of cool because they're continuously generating. If I just get my mouse off of it, they turn-- they're colored.

And we can change the size of these guys-- for example, make them a little bit thinner. And one of my personal favorites is setting these guys to spheres. Let's make them smaller spheres and then play with our animation a little bit. No, it's a pretty good animation.

And then this starts to get really-- like, you can start to get some really crazy effects in here. For example, if I just go ahead and hide all my steel here and we can just visualize just the particles themselves-- I can set myself to a perspective view to get a nicer look. Now, I mean, you know, this is just playing around with the actual visualization tools in CFD, but there's not much you can do with this after the fact, right?

Like, you can create an AVI video and that's about it. But you can't comp this onto anything. You can't really visualize this in any meaningful way outside, you know-- I mean, within this product. So what you want to do now is take this and bring it outside of CFD into something else like Max, where you can actually do lots of crazy things with and a lot more flexible of a tool.

So to do that, you go to the application button. You export and you export your nodal results. This is going to save a CSV file on disk that you can then load into Max using the new tool sets that we added in the last couple of releases. So let's go ahead and see how that is done and let me just go back into PowerPoint just to make sure I'm on the right track. Yes I am.

All right, so we're going to-- we're following my skeleton here. So now we're going to take a look at how we can import that simulation data into 3DS Max. We're going to apply a visual style using something called shader effects.

Does anybody know what shader effects is? No, I don't think so. You know a little bit?

You know a little bit. You don't count. What are you here for?

So essentially, shader effects is the shading platform or the material definition platform that exists inside of Stingray. So when you create materials in Stingray, you're creating shader effects materials. A really great thing is that shader effects is also inside of Max. So you can create shader effects materials.

It's basically a node based material editor and you send your assets in. It's one to one. They come and they look the same and the same shading tree is also carried through into Stingray. So this is a really good-- a big advantage when working with Stingray the game engine.

And then we're going to use those shaders to animate that wind data. Cool? So what's next?

Here is a demo. So let's jump back into Max and see how-- whoops, it's not what I wanted. All right, there we go. So let's skip over into this portion here.

So we've got our stadium. We simulated-- we created a proxy version. We simulated the CFD. Now how do we bring this into Max?

So we have some new tools. The first one that I'm going to show you guys is under the create panel. We're going to expand the standard primitives and we're going to locate CFD.

From here, we have a couple of nodes. And there's a lot more that I'm not going to show you today that I just don't have time. The one that I want to show you is this import data node.

This is going to allow us to import that simulation data. So we click it, put it in our scene. And it's really important when using any of these CFD objects and nodes to center them to the world of. Max

You want to make sure that their x, y, and z is set to 0, 0, 0 down here. To do that, I'm going to right click on the spinners, resets them to 0, 0, 0. This is so that everything lines up properly inside of Max.

So once you've got this guy in, you can come to the Modify panel and he's looking for a CSV file. The CSV file is what I outputted it from CFD using that output nodal result. It created a CSV file on disk.

So let's go ahead and grab that in. I have it saved here as my east.csv, so the wind stimulation coming from the east. Hit Open. It takes a few seconds to load.

There's a lot of data. And it's basically going to come in as a point cloud of sorts or a-- the actual real term is a velocity field.

So now that it's in, we still don't see it, right? It's in but we're not looking at it. Why? The reason is because we need to turn on vertex ticks on that object.

To get to that, you can right click on the object in the Scene Explorer, display properties, vertex ticks. So now we get this lovely blue point cloud which is actually a velocity field. And basically, what you're looking at is every point in this point cloud contains simulated information-- in our case, wind direction and wind velocity.

So where is it going and how fast is it going? Right, so once it's in, we no longer need to see it in the viewport. It just needs to exist in the file.

So we can just hide it, get it out of the way. The next thing I want to do is I want to recreate those traces we saw in CFD. I want to create paths that are going to traverse that velocity field to accurately represent the wind as it's flowing through the stadium.

So to do that, I'm going to grab another new tool, under Shapes this time. Expand this guy here. We have a CFD category and we have CFD spline paths.

Again, I'm going to pop it in my scene. This one actually has a viewport representation of a circle. I'm going to zero it out to x, y, z, and come to its parameters.

This guy requires two things. It needs to read that point cloud, that velocity field, and it needs a geometric object to create splines from. So let's go ahead and connect those two together. We're going to click on this button here and locate our import data node in the point cloud.

And then the geometry-- I have this organic piece of plane that's floating off to the side of my screen here, of my model. We're going to use this guy. Basically, what's happening here is every vertex on the geometry that you assign will create a spline and that spline will traverse the point cloud, look at every point and then determine where to go next from each point to point. So it's an accurate representation of the wind.

And there's some settings in here to adjust the look and feel. They're actually documented in my handout as well what they all do and also in the help file, as a matter of fact. So I'm going to set up these values that I know work for my case-- 75 and 2. And now I have much smoother, cleaner curved lines representing the wind.

So the really great thing about this whole thing is that these are just splines like any other spline object inside of Max. You can turn on rendering, make them renderable, give them volume. You can increase the size and set this to 300, make them a little bit thicker.

And you can set them to a rectangular if you wanted to. You can add a modifier, a suite modifier, and completely change the shape itself. You have access to all of Max's tools when you're using these objects because they are just spline objects.

We want to make sure that mapping coordinates are enabled. This is because we will be applying textures to this guy and we don't want to do our own UV mapping. Just turn that on and don't even worry about it again.

And the great thing is that they are connected to this object here. So when I move this object, the splines are going to update to the new position within the velocity field. So they are splines, but they are procedurally connected to the objects that they are used to generate the splines from.

The really great thing is I can make modifications to this guy. For example, let's go ahead and add a tessellate modifier to add more vertices, right? That's what this is about. Add in, boom. You get more lines.

Modify vertices on this plane. Whoop, went a little too fast. And you can really creatively change the look and feel of these splines that you're going to be using to display and visualize wind data. Cool?

All right, so now that we have our wplines, we have our objects, we want to shape these guys.

We want to give these guys a visual style or a visual interpretation. Right now, they're just pink tubes and clearly that's not very interesting.

So to do that, we're going to use shader effects. Now to create a shader effects material, you select any material in the material editor and you set it to a DirectX shader. And then you set it to Stingray here in the dropdown.

Now there's shader effects and Stingray. And that gets a little bit confusing. It confuses some people.

We're going to be using Stingray which is essentially a shader effects material. But it's a subset of a shader effects material. A shader effects material has access to a lot of nodes and a lot of tools that Stingray doesn't necessarily support the whole thing. So for other game engines or for other kinds of use cases, you'll want to use a shader effects material. But if you want to use it to go into a Stingray, you use a Stingray material.

So we set Stingray here and we get these options that appear. We have controls here for reflectivity or roughness, metallic, base color, normals, emissive, AO. This is a default Stingray material, default Stingray UI.

This UI is completely modifiable via the shader effects graph. So if I just double click on this guy to show you guys what this looks like, these nodes here and the connections between them are what make that UI in the material. So when I delete things from here or add things to here or modify things in here, the UI reflects those changes. So you basically build a material UI, if you will.

And this list of notes here on the left hand side is what you have access to in the side of a Stingray shader. If you add a shader effects material, there'd be a lot more nodes, but they woudln't be supported inside of Stingray. Makes sense? All right, so the other point I want to make on this is nothing. Let's just go ahead and assign this guy.

So we're going through-- oh yeah, the other point I want to make is this default material setup here that we see what, you know, your normal maps, your base color, your metallic, your roughness, your emissive-- this is the default setup to create a PBR shader or a physically based shader. So physically based trading requires your base color, your roughness, and your metallic. We're not creating a physically based shader.

We're just going to create some colored arrows flowing through splines. So I don't really care about any of these nodes. This is overkill right now for me.

Right, so what I'm going to do is go ahead and take all these notes right here and just simply delete them. I'm going to keep two nodes, two nodes that are important-- the output note, of course, right? This guy is a standard base. He's got all the inputs. Whatever gets connected into this gets reflected in the UI.

And we have our text coordinate node. This node is actually allowing us to read the UVs that are currently on the object-- so the mapping coordinates on the object. We need this guy-- really, really important.

So we're going to keep these two in and we're going to build out a shader to visualize CFD data. So the thing I want is color. I want to able to give this guy color.

So to do that, I'm going to use a material variable. Another heads up I want to just kind of lay on you guys-- shader effects materials can be a little bit daunting and intimidating and scary. And they were for me but give it time. Stick with it. Don't give up.

There's a lot of terms in here that are kind of strange-- vector threes and construct vectors and it gets a little bit wonky, especially if you're not from the gaming world, which I'm not. But give it time because if you give it time and you get the hang of it, you can create some really kick ass and cool shaders. So I'm going to walk you through the process of creating a basic shader for CFD vis.

So we want base, a base color-- so a diffuse color. We grab our material variable node. We're going to rename it here in the properties to Color-- very simple.

We're going to set type to vector three. Vector three is basically just a color swatch, right? And we're going to connect this guy to our base color. Notice what happens in the material. We get a new color entry that appeared set to black.

It's assigned to the splines array. I can adjust this and change the color of my spline objects. Let's just set it to white.

Because I want to have glowey arrows, I want to also drive the emissiveness or the self illumination of my shader by this color. So I'm going to connect that also to the emissive input. So next thing we need to do is I don't want to just have a white blobby spline or a white blobby shape to represent wind. I want to have arrows in my case or a texture map.

So to do that, we need to leverage the opacity channel of this node here. We need to give it the capability of loading a bitmap. So to do that, we're going to use a node called Texture-- Sample Texture. We're going to bring it in.

We're going to rename this guy to Opacity Map. And this guy requires UVs. What UVs? The UVs from the object. We're going to connect it in. And we're going to pipe this guy into our opacity map just like that.

So now here, we have a new slot called Opacity Map. We can load in our arrow texture or our opacity map. We can load in other maps, but that's the one we're going to load in for now.

Let me just put in an editable spline on top of the sky just to get that white line that's a little bit-- edits, poly. It's fine. No, it didn't do that. Hold on. Wrong one. All right, let me just add an edit spline just to get rid of that white line.

So what actually happened here? We have an arrow texture driving the opacity on these shapes but we're not tiling the texture. We basically stretched that one texture across the entire spline. That's what's happening right now. That one arrow texture is being stretched.

So we need to give tiling capabilities. We need to add the ability to basically tile this texture. So to do that, we're going to grab another material variable.

We're going to call this guy X Tiling and we're going to set it to a scalar, which is a spinner in the UI. A spinner will appear. We're going to give it a maximum of 100 rather than 1.

And then we're going to create the same thing for y tiling. So Control-C, Control-V to copy and paste and just rename this guy to Y Tiling. And then we need to combine these two.

This is where things get a little bit at least strange in my mind. To combine these two, we're going to use something called a construct vector two. Do not ask me what that means.

I'm not a programmer. But it's what you need to do. Cool?

So a construct vector two-- we're going to combine the x and the y. And basically now we're going to multiply this result by the UVs of the object. So to do that, I'm going to grab a multiply node, which we have.

We're going to bring that in here. We're going to say, hey, multiply the result of the xy tiling by the objects UVs and plug that as the new UV input of my opacity map. So now in my UI, I have xy tiling, both set to 0 so everything disappears. We're not doing any tiling.

If I set this to 1 for example, and set this to 50, now we have arrows that appear inside of our guy here. We can tile in both directions. But of course we don't want to tile this way. We want to tile that way.

So we set it to 1. We set it to 50. You can modify this, set it to 20.

It's going to have less arrows. They're going to be more stretched. We could set this to 100-- smaller arrows and a lot more arrows. Cool?

So this is all great, but our arrows are static. They're not moving. This is not what we want. We want to animate these guys.

So to do that, we're going to keep going with this guy. And we're just going to move this all over a little bit so we have some space to work with. Let me go ahead and grab a note called a panner node.

A panner node allows me to animate textures along the surface. All right, so this guy, very straightforward. He needs UVs.

So we're going to grab the UVs from the object. He needs a time so that you can do this over time. We're going to grab a time.

Nothing to do there-- just click and drag it in. We also need to control the speed in which it will be moving over xy. In this case, they call it UV. You can see that it's the same thing.

We need to be able to control the speed in which the textures will be moving. So to do that, yet again, our good old material-- variable node. This time, we're going to rename this guy to Speed U.

We're going to set it to also scaler. We're going to give it maybe 10 as a maximum. Copy and paste this guy for our V, rename it, and connect these two guys into the node.

And then connect the output of all this back into our multiply node, overriding the texture coordinates from the object. And now we're basically multiplying the tiling by the speed. So now when I come back into my shader, I have speed U and speed V. Just to give you guys a little idea of what this does, if I hit maybe one on my speed U and hit play on my timeline, we're animating those textures along the tubes.

This is not what we want of course. We want something going in the other direction. So let's do 0.5, maybe 0.1. Then we can have these kinds of effects.

And now you can really start playing with this. There's just lots of different things you can do here. We can just get rid of the U if we want to just to have long, flowing arrows in our stadium.

Cool? Pretty fun? Cool. It doesn't stop there. There's more.

So another thing-- right now, what we're looking at is basically just wind direction. We're not leveraging the velocity that we simulated at all, right? We're just leveraging the wind direction. But we do have velocity information that we simulated. So we can visualize that as well.

To do that, we're going to apply on our object, on our spline objects, a modifier called color-- CFD color vertex modifier. This is going to read the velocity information from the velocity field at a point cloud that we brought in a little bit earlier. So I'm going to punch that in right there.

And then we need to tell our shader to actually display this in vertex colors is what we call it in the viewport. I have another version of this shader that has support for vertex colors. I have this huge vertex color spinner. I'm going to show you guys how to set that up. It's a little bit redundant at this point.

But if I assign this, now you see that our arrows are colored. I can turn on or off the vertex color display using the spinner here. And I could increase the red amount to just exaggerate that a little bit or have a slightly wider range of coloring. And basically, whatever is a little bit more red is moving faster and whatever is a little bit more blue is moving slowly or slowlier or more slowly.

Cool? Fair enough. So far, so good. You guys with me? Cool, awesome.

All right, so what's next? Now we got our shaders. We got our splines in. We got shaders assigned to them.

Now we want to bring in some context around this whole thing. A stadium is cool but a standing floating in space is a little bit boring. So what we're going to do is move onto the next portion of my presentation. We're going to talk about Infraworks.

So we're going to leverage Infraworks specifically to create GIS based city contexts around--

or contextual surrounding around our stadium. And you know-- anybody us Infraworks? I know I asked that a little earlier.

But you know, it does a lot of civil engineering stuff. Mike [INAUDIBLE] can talk all about it if he wants, but it's not my cup of tea. But for creating, like I said, these real world environments, it does a pretty good job. It does it very fast.

Problem is that these models are not Stingray ready at all. They need like a lot of love, a lot of massaging, a lot of things to consider. And this is what I want to highlight here today and show you guys how to fix it. And also we're going to optimize this whole thing for Stingray.

So from the Infraworks UI, we can hit the model builder right there. This is super simple. Doesn't even warrant a video, but the point is to show you guys how easy it is.

We type in our location, Keenan Memorial Stadium. In my case, you see that we get that stadium right there in the middle of the map. Basically, draw out a region that we want to create the city from, give it a name, hit Create Model.

This gets pushed to the cloud. And a few minutes later, you get a fully textured, fully modeled representation of that area. So this is pretty freaking slick.

Like, I love using this for all kinds of projects. And you see it looks pretty freaking good. I mean, it's ready to be used in other contexts.

I mean, it's cool in [INAUDIBLE]. It's cool in Infraworks. But you can take this out into other areas.

It supports FBS export. So from the export 3D model, we're going to use the entire model. We're going to split it up into individual FBX files using this utility here.

I want to have individual control on the ground and on the buildings because they each pose different problems that need to be solved. So you export the FBX files out and then we can go into Max and import those guys in. So let's go ahead and do that. Infraworks start-- I'm going to say import.

We're going to grab-- first of all, I'm going to grab my ground. That's the first one I want to deal with. The ground comes in.

And there's two main problems to solve. One is it's super tessellated. And this is-- you can make any game engine crawl, not just Stingray, right? [INAUDIBLE] to solve this.

This is a small area. You know, for the demo, I did a small, little area so it's doable. But you know, larger areas, this is a lot worse.

The other problem is that there's a whole bunch of quadrants and objects and they're all called mesh one. So we need to solve all this stuff. All right, so the easiest way to solve all this is grab everything.

Let me just get rid of some of the helpers that come in. I don't like these two guys. They mess me up.

All right, so I'm going to grab the whole thing. And I'm going to, first of all, deal with the tesselation. So I'm going to grab a prooptimizer modifier inside of Max. And this guy allows me to decimate this mesh while still maintaining the integrity of the volume. So it's a really, really powerful tool.

So we need to just set up some things here. I want to exclude the borders of the individual objects. This is so that I don't have seams and artifacts where the quadrants meets. Right, so we want to exclude the borders. Keep them intact.

We want to keep the textures because we have textures. If we don't enable this, we're going to lose all the We're going to have gray models.

I want to keep the UV boundaries-- super important. Once we set this guy up, we hit the calculate button. It runs through, only takes a few seconds. And then we're ready to decimate.

I'm going to use a spinner here to decimate. And you can go wild because-- you know, it depends. Like, I'm going to go wild because I'm never going to see this from close.

I can probably bring this down to 1% if I wanted to. It would still be fine. I'm going to go 10%.

And you see that it does a super good job of decimating the mesh. I mean, the mesh results are not very nice, but it doesn't really matter because we're not going to animate this guy or deform this guy. But the point is that if I get close to this geometry and hide and unhide the pro-optimizer, you see that it didn't really-- it's barely affecting the integrity or the actual volume of the mesh.

So for my case, this is perfect. I mean, if you're decimating something that you're going to see up close, you don't want to go to 10%. But the point is that this is a really good way to get to a much better result for Stingray.

So once I've done this, I can right click convert to poly. And did that work? It did, yes.

The next thing I want to do is merge these into one single object because we have still a whole bunch of quadrants. So I'm going to grab any one of these guys here. In Edit Poly, we have attach with option box.

I'm going to click on this guy and have a list of all the objects I can attach to. I'm going to select them all, Control-A, hit attach, match the material IDs, and boom. That's all I have to do. I have one object now.

Now he's called mesh one. I don't want that, so let's call it ground. And this guy's ready to go into Stingray.

Let's bring in the buildings now and see how to deal with those and what their problems are. Actually, the problems of the buildings are the same problems that we have with the-- one of the problems that we have with the ground that we need to deal with. This problem that we're going to solve is the fact that the images and bitmaps assigned to all the buildings and all the grounds that Infraworks generates have funky characters in their names that Stingray does not handle unfortunately, not yet.

It will eventually in the next release. We fixed that. But for now, if you want to bring this content in, it's going to fail to compile all the bitmaps that came from Infraworks.

And that special character specifically is dot. Any texture name with a dot character in it will fail to import into Stingray, fail to compile. So we need to rename these textures to remove the dots. Now, doing that manually, of course, is an arduous task that no one wants to do.

So what we did is we created a Max script that you guys also have. I uploaded it to the AU website. That's basically what's called the bitmap sanitizer.

It's going to cycle through, run through all the texture maps in this file, and it's going to look at all the ones that have dots. It's going to create a copy of them, rename it, remove the dot so it's not messing around with the original one, and put it into a folder for you and also make the file use that new copy of the texture. So let's go ahead and run that script, scripting, run script.

I have the script [INAUDIBLE] right here. This guy requires me to point to a folder on disk that I will dump the new textures in. So let's go ahead and just do that in the [INAUDIBLE] and sanitizer. Hit OK.

Takes about a second. There, it's done. And if I open up my asset tracker here, I refresh.

You see that all of these bitmap names no longer have dots. All the dots have been replaced with underscores. These were all dots before. So now this is all ready to go into Stingray, ready to be used in the game engine.

So let's go ahead and do that. And before I do that, let's go back into our trusty PowerPoint. Real time interactivity-- so I'm going to talk to you guys about Stingray for the next-- until the end of the class, I guess. And so I'll give you guys a quick little intro into the game engine-- what does the UI look like, what are the different tools available to you, and how to use them.

We're going to look at live linking. So live linking basically connects Max to Stingray. So you can send content really easily with one click and you can modify your content in Max and push those modifications into Stingray also with one click.

So we're going to look at that. We're going to look at how to import animated cameras. So if you animate cameras inside of Max, I'll show you guys how to animate one camera. And then we're going to send them into Stingray and see how to use it.

And then we're going to look at flow. So flow is pretty cool. Flow is basically a node based scripting editor inside of Stingray.

So if you're using Unreal, it's similar to Blueprint. So basically, it removes the need to script everything, all the interactions. When you build the game, you have interactions.

I touch this, something happens. I walk here, something else happens. All that is usually scripted.

With Stingray, you can use flow to remove the scripting. It's node based. It's very user friendly. It's very elegant. And you can extend that with scripting as well if you wanted.

So we're going to take a look at that. And then that will bring us to the end. All right, so in Max, we're going to start sending some content over into Stingray.

Let's go ahead and just switch to this state here and find our stadium model. Now what I recommend as a workflow is don't send the whole project as one file. That's like the worst thing to do.

In fact, if you want to understand or sort of the way to see how-- to make the decision as to what do I send individually, think about it this way. If you want to interact with things, send them separately. Like, if you have a Revit building, for example, and you send the whole thing in, you're going to get a big blob. That's fine.

But if you want to animate the doors or have parts of the building that you want to trigger things have events, send those as separate objects. So in my case here, I'm going to send all of the environments as one object, the stadium as one object, and the CFD splines as one object. All right, so let's go ahead and just select those.

Let's go ahead and just grab that. All right, so let's go ahead and grab our ground, first of all, and our buildings. We're going to go Stingray, send selection.

This is the live linking I mentioned earlier. It opens up a-- actually, you know what? Before I do that, let me show you guys Stingray. That's what I should be doing first. Then we'll send some content in.

So this is the Stingray UI. Who has seen this UI? Who's been playing with Stingray?

So, a few people-- cool. Obviously-- you don't count. You should have raised your hand too.

All right, so is the Stingray game engine. We're all super excited. I'm in this thing every day of my life.

So of course we have the viewport here in the middle where we're going to [INAUDIBLE] the level. We're going to put things into this level and build out some interactions. Level flow-- this is what I told you about.

This is what I told you that-- a node-based scripting editor where you connect nodes together to build out basically game logic or behaviors or interactions. I'm going to just delete these guys because we don't want to use them. By the way, I should mention-- this is maybe nerd talk but it's kind of cool. All of these flow nodes are all Lua based.

So Lua is the scripting language of Stingray-- so very similar to Python. Right, so if you want to extend flow, you can go into Lua. And this is so Lua that even the nodes themselves are all based on Lua scripts.

So I can grab all these guys here and I can say Control-C to copy. Then I can go into Notepad and I can do a Control-V and I get the Lua code equivalent of those nodes and their connections. And a really cool thing-- I can send this to anybody, right?

It's a Notepad. They can edit. They can modify it if they know Lua, send me back some new code. I can copy that code, come back into Stingray-- let's go ahead and delete these guys-- and do a Control-V. And it brings in those new nodes and those connections of those nodes. So it's pretty powerful and extendable, extensible platform here. So let's just delete this. We don't want this.

We have our scene explorer over to the right hand side with all the different components that are inside of Mycenae. Right now, we have a Create panel here where we can create basic helpers, objects, cameras, lights, that sort of thing. We have a Properties editor down here.

So if I select my light, for example, I have all the properties associated with that light located here. This location here, this Translate, Rotate, and Scale area, is really important. We're going to look at this quite a bit and I'll explain to you guys what I mean when we get there.

Then we have an asset preview window over here. I don't have an asset selected, but if I, did I would have a preview of it down here. This is the project folder. So all the assets and files associated with the Stingray project live in here. So I have a folder view of it here but I have a hierarchy view of it here as well.

This is exactly the way the project is represented on disk. So if I right click and I say Show in Explorer, I have the exact same representation here. So I can navigate the files here.

I want to dump files into the folders and windows, I can still do that. Come back into Stingray, just do an F5. It updates and refreshes and takes those new assets that you put into those folders yourself manually. So it's pretty cool-- a lot of flexibility, a lot of really cool things we can do.

So let's go ahead and bring in some content. So back into max, I have my environment selected. I go to Stingray, Send Selection. And you see that we're open in the right location already.

I'm going to go into demo here. I have an Infraworks portion or folder. I'm just going to call this

IW for Infraworks. And then an import dialog box in Stingray will appear. And I'm going to leave the default settings.

I want the unit mash, the textures, you know, cameras if they were there. I don't have any, so it's fine. I'm going to leave everything under unit mesh as default. The Animation tab, we'll comes a little bit later. But for now, unit mash is all we want.

By the way, a unit in Stingray is an object or an entity unlike unit of measurement in other applications. So that confuses people sometimes. It confused me for a while. So when we say unit, we're referring to an object or an entity.

So that object that I sent is quite heavy. There's a lot of textures. Even if we decimated the model itself, there's still a whole bunch of textures that it's bringing in.

So it is going to take a few seconds to bring in. In the meantime, I'm going to start bringing in other assets like my stadium. So let's go ahead and just grab all the components of my high res stadium, say Stingray, send selection, content, demo. We have a folder here for stadium, bring this guy in and hit Save.

Again, we get the import dialog. Same settings there. You see we have a progress bar down below. It's still importing at 0% but it's going to go up, believe me.

And then lastly, we're going to go ahead and grab our CFD splines. Let's go ahead and just grab those objects right there. Let me make sure I have them selected.

Something just happened. I don't know what just happened. All right, yeah. So let's undo whatever I just did there.

And we're going to say Stingray, Send Selection. We're going to go to Content, CFD, and we're going to call this guy CFD. And we're going to import it yet again.

So now back inside of Stingray, we start to have content that's being filled or is filling up these folders. So if I go to Infraworks, for example, I have my Infraworks model here. It's still not fully compiled. So we're going to give it a second or two.

We have all the shaders that are associated with that Infraworks model itself. And we are almost ready to get going. I think we're good.

There we go. Take my model, drag it into my viewport. This is just a drag and drop.

And remember I mentioned that these spinners are super important? So for now-- and this is going to change soon. But if I want everything to line up in Stingray as it is inside of Max-- so the stadium on top of the ground with CFD splines going through-- I need to set my x, y, and z translation, the position essentially of my objects, always to 0, 0, 0. Every object has to be at 0, 0, 0.

All right, so we're going to do that. Hit F to zoom extend out a little bit. And then we can get closer.

And now we have our Infraworks model inside of Stingray. Let's go ahead and grab our stadium. Again, all the materials, all the textures came in.

Grab our stadium model, drag it inside of our scene. Zero it out to x, y, and z so that it lines up to the same location. And now we have our stadium.

And then lastly, what we're going to do is bring in our CFD splines. And we're going to just grab this guy in. And we have our splines.

But now we have a problem. Can anybody determine what the problem is? Splines won't display anything.

The reason is because Stingray and max handle xy differently. Stingray basically inverts it 90 degrees. So what that means is if I grab my CFD shader, you remember that we had-- CFD shader.

Remember that we had x set to 1 and y set to 50. It's actually inverted that. So the fix it, the easiest fix is just to do this. And now we have-- whoa, went a litlte too fast there.

Let me just zoom right back out. Slow down the zooms. We got our arrows up here.

You can actually fix this in the shader itself. And by the way, just like in Max, if I double click on the shader here, I will access the shader effects shader tree. And this is the exact same representation as Max that we just built out.

I can go ahead and add some hooks in here to say, hey, if you're inside of Stingray-- like, a checkbox that says Stingray UVs. And I would just flip it inside the shader. And when I checked that checkbox on, I'm essentially switching the x and y.

I'm not going to do that for now. But the point is that it's very easily fixed and it's something just to remember when bringing in these shader effects materials from Max into Stingray. The shaders need to be-- the xy needs to be inverted however you want to do that.

And then the thing to do is notice that they're not double sided. Right now, we don't see them for the other side. So to fix that, we're going to double click on the shader and we're going to go to the output node right here.

We get the output node's parameters on the right hand side. I'm going to set the face calling to double sided. And if I close this out, save out my shader changes, now I see the arrow textures on both sides of the splines. All right, and all the modifications that I made in shader effects in max, I can do all of that inside of stingray.

It's visa versa. It's the exact same platform. All right, so now we got our splines in.

They're displaying properly. And we have a stadium and we have our city. Let's go ahead and bring in some cameras.

So in max, I'm going to switch to the state here where I have basically this spline object circle that appears up above my stadium. I'm actually going to lower it just because it's a little bit too high. And what I want to do is animate a camera going along this path here, all right?

So to do that, it's super easy. We're going to create a camera. I'm going to create a target camera, just drag it anywhere in my scene. And I'm going to go to animation, constraints, path constraints.

And I'm going to constraint that camera to this guy here. Now, the next thing I need to do is grab the target of the camera and put it somewhere in the middle of the stadium so that it's actually always-- at least always pointing at the stadium. But because I've constrainted to the path, it's automatically animated when I scrub my timeline. I have that automatic animation happening.

All right, so I didn't have to do any funky animation tricks. It's super easy to do. You can constraint to any path.

You can, for example, in silicon texts, a lot of what they'll do is they'll extract the spline of a road and they'll attach a camera to that spline. And basically, you're always following that road from that camera view-- very easy to use, path constraint. All right, so we got the camera in.

And it's been animated frame, frame zero to 30, of course. Actually, what I should do is animate this. So I'm going to set a key here, set a key here.

What's going on? Hold on. Doesn't like what I'm doing for some reason. And set a key here.

Actually, you know what? I grabbed the target. So let me just set a key here.

I go to frame 300 and set a key here as well. So we got our camera. It's animated.

For some reason, it's kind of slowing down. Not sure what's going on. So we got the camera.

And now we want to send this into Stingray. So we're going to select the camera and we're going to go to the application button. And we're going to say Export, Export selected.

And we're going to dump this inside of the folder structure of Stingray. So to get to that, we come to Stingray. And I have a folder here called camera.

I'm going to right click in here and say, show in explorer. This gives me the path that I want. I can copy this path from the Windows Explorer.

I can come back inside of Max. And I'm going to put this path right here. And now I'm going to just call this guy.

Am I in the right place? No? What's going on here?

Export. export selected. There we go.

So in here, I'm going to call this guy camera. And now the camera has been generated. But Stingray is still not aware-- oh, I saved it as an IGES file. That's great.

Let's do that one more time. Export, Export Selected, boom. Set this to FBX because we want to send an FBX file. Stingray only accepts FBX files, by the way.

And we're going to hit Save. What's going on? Ah, hold on one second.

Max does not like what I'm doing for some strange reason. Bring in our path, yes. Oh, We're already there.

OK, I get it. Never mind. Camera-- camera, we don't want an SVF file.

Let's go ahead and make an FBX file. Are we in the right place? Yes, we are. Boom, hit Save.

In the FBX Export dialog, you want to make sure you come to animation, bake animation. You want to bake the animation that's currently on our timeline 0 to 300. And we also want to make sure that our cameras are enabled here because we are exporting a camera. There's no geometry in this FBX file.

So we're going to hit OK. It's going to create the camera file. If I go to this folder here, wherever that was-- let me just go back to Stingray and get to it. So Stingray, open up, show in explorer.

You see that we have a camera FBX file. We also have these two that we're going to delete. That was a mistake. But Stingray is not aware of the camera file or the FBX file.

It hasn't created the necessary assets and files it needs to be able to use this FBX file. So to do that, to make Stingray aware of this FBX file, we take it. We drag and drop it into the Stingray browser, project browser.

Then we get the import dialog. So we do want the unit just like before. But in this case, we want animation because we animated the camera.

So we enable animation. We're going to create an automatic skeleton. You don't have to worry about what that is or what that does. It's just going to do it for you automatically. Stingray needs a skeleton.

We're going to import the clip, the animation clip. That's what we call those animations. When you bring them into a game engine, they're called clips.

And we're going to create an animation folder. Just to keep things simple. So let's go ahead and bring in our camera.

And so we have a camera object. We don't really see it because it's not an object because it's not a piece of geometry. And we have our animation clips and our camera animation clip that we brought in which we can play but we don't really see it here anyway because it's a camera.

So now we want to use this camera in our project. What we want to do is we want to put in our camera and start the game from that camera view but also trigger that animation of it circling the stadium. So to do that, we're going to grab our camera object itself, the unit, drag it into the viewport.

We're going to zero it out to x, y, z. So it's the same place as it is in Max. And we're going to use Flow to make this happen.

So in Flow-- remember Flow back here. Flow needs a Flow representation of anything you want to use in your project. So to make that happen, I'm going to keep the camera selected.

I'm going to come into Flow, right click, and I can say create level unit from camera. This node that you see here is a Flow representation of that camera object. All right, so now we got the camera in.

Now we need to do a couple more things. We need to extract the camera information from the unit itself. So we're going to look for a get unit camera.

From what unit? The unit that we just brought in. We want to set the active camera.

So we're going to look for a set active camera node. What camera? The camera we extracted. From what unit? The unit we brought in.

And now we need to give it an event. Just like everything in Stingray, an event is required in order to tell Stingray when to do this action. In our case, we want it to be when we load the level.

When we hit play, we want to be in that camera view already. So we're going to use a level loaded event. Level loaded.

So when I load my level, I will be viewing the scene through that camera. But I'm still not animating. I haven't used that animation clip yet.

To use that animation clip, I'm going to locate another node-- whoops-- called play animation clip. This node allows me to play an animation clip specified by a given event. So we're going to go ahead and plug in our camera animation clip that we brought in.

And we're going to say loop set to true. When is this going to happen? When we load the level. So this should work-- fingers crossed.

Let's hit play. If that doesn't work, I don't know what's going on. Ben, is this going to work?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Ben's keeping me honest. Not animating. What's going on? Oh, we need to tell the animation clip on what unit to also play that animation on, which is the camera.

Now that should work. All right, I'm going to show you guys one more trick with flow. [INAUDIBLE] a little faster. But now we've got that spinning camera around that is loaded on level load and reviewing the scene through that camera view. Cool?

So I'm going to show yo guys one more trick with regards to the CFD nodes themselves. So now we have one simulation coming from one direction. And the project I showed you at the beginning, there's eight win simulations.

And we wanted to be able to see them individually at different times based on different [INAUDIBLE]. Because now there's a whole bunch of ways of setting this up depending on what you want to achieve as an experience. So I'll show you guys just how to basically hide [INAUDIBLE] hide them and assign that two specific triggers.

So let's go ahead and bring in our CFD node into Flow so I can select it here. I can come into Flow, right-click to create that node representation of that CFD object. And then I have a note here called set unit visibility. And so I'm going to set this-- I'm going to keep this guy to visible true.

On what node? Our CFD node. And when do I want to make this true? When I hit a keyboard button.

What keyboard button? The 1 key just for simplicity. So when I press the 1 key, I'm going to turn the visibility on on this unit.

Let's copy this guy out and let's copy this guy out. Too many. Now when I hit the 2 key, we are going to set the unit visibility to false.

Pretty straight forward, right? So let's go ahead and compile this and see if this works. Now I'm a little far away from the CFD node, so you might not see that happen very easily. Oh, you'll see it.

So if I hit the 1 key, I should see them. I hit the 2 key, they disappear. Hit the 1 key, they come back. 2 key, they disappear, right?

Basic programming. Well, you can take this lot further, of course. You can have it cycle through 1 key. If I hit the 1 key over and over again, it cycles through the visibility of multiple different objects, not just 1 and 2.

You can map it to joysticks. You can map to a VR stick. You could do a lot of different things. Cool?

All right, I'm going to show you guys the final project as I promised. It went a little faster than I expected. I might give you guys some minutes back if that's OK.

Let's go ahead and play this. This is the final kind of project and what you guys saw a little bit earlier. But you see that we have that spinning thing.

Yes, I have wind simulations. And also, by the way, the projector is not nearly as nice as my screen. So just as an FYI, projectors kind of suck.

But-- so we have many different wind simulations-- eight, to be exact, coming from eight different wind directions. And every time I hit a key on my keyboard, I'm hiding all the other wind simulations and just showing one. Then I can swing into camera animations.

So now if you put a VR headset on and you're sitting in the middle of this field, it's like an acid trip. And I have a VR version of this. I just don't have the headset here.

Right, so I have a whole bunch of different camera animations that we set up inside of this along with a whole bunch of different wind simulations. Cool? David, you have so many questions. Do you guys have any questions before we-- I know I'm a little bit early, but eh, 20 minutes.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah, let me show you guys how to do that real quick. Yeah, it's a good point. So Stingray-- if you want to export something out of Stingray, right-- you don't create images.

You don't create videos. You create run times, executables, games. So we have a tab here called the Deployer tab.

And basically we can currently compile for the platforms you see here-- so Windows, Android devices, iOS devices, PS4, and Xboxes. We also support WebGL. That's very new.

It's under the Experimental Features category that's currently disabled. But you can create

WebGL files that you can host on web sites. I actually created a WebGL kitchen aid configurator, kitchen aid mixer configurator, that's hostable on a web site. It's really cool.

So if you want to deploy, let's say, for Windows, for example, you select Windows. You give it a path and you hit Package Project Windows. And it creates-- and I actually have a folder pointing there already. It creates basically this list of files that can be zipped, sent to anybody, right?

You don't have to have Stingray. And you basically just run the executable that is in here. And it'll open up exactly what you've seen when you hit the Play button-- actually, a little bit more because it's running the entire project.

The Play button is testing just a level. But you can create a project with multiple levels with the menu and jump into various different levels. But basically, this is what we just saw. It's exact same thing outside of Stingray now. Yep, yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yes, that was done in Max and I think that's true because we were a team that built this. I think he's triggering them through-- no, so the Autodesk logo is scale form, which is the UI tool inside of Stingray to create UIs and HUDs. This is just geometry that appears and disappears based on triggers done inside of Max. All those little animations and spinning and stuff is all Max as well, yeah.

More-- I wish it looked as good as my screen. I swear it's so much nicer. So any other questions?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Sure.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Not that I know. It's in metric as well and the unit of measurement in Stingray is meters. One unit is one meter.

So you want to make sure that your Max or wherever you are making content is set to meters as well because you could have things out of scale. Yeah, I don't know how to change it inside of Stingray. And Max, you can change your units to anything, but-- any other questions?

Do you guys see value in this? Do you guys think this is cool stuff or not? You can not think it's cool. That's OK too. Yeah.

AUDIENCE: So I'm assuming this is [INAUDIBLE].

JOSE ELIZARDO: Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Oh yeah, you can set up lots of different ways of setting this up. Right now, I'm setting it up as a static camera view that I'm always looking for that camera. But you can set it up as-- I have other experiences, other games where I'm a person walking around an apartment. And I'm turning-- opening and closing doors and I'm turning on the stove and the TV and just, you know, lots of different ways that you can-- lots of different ways to strip a cat, beat a cat or-- skin a cat?

What do you say? My French is coming on. Yep?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Blend between, like-- using the same assets? What I would probably do there is actually have different assets at different simulated speeds and blend between the visibility of each. I wouldn't do it at the shader level. I would do it really at the asset level. Like, the object itself has, you know, faster or slower speeds.

AUDIENCE: But you have, like, multiple speeds [INAUDIBLE] speeds.

JOSE ELIZARDO: Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah, yeah, exactly, yeah. Just switching to the visibility of them. You can probably also-- I mean, so the sort of fake thing that's going on right now is yes, the direction is accurate and yes, the color of the arrows is accurate.

But what's not accurate is when the arrows become more red, they're therefore going faster.

They're actually not moving faster because it's the shader effect. It's not an actual, physical effect. So there's no way-- well, I don't know of a way right now, at least not with this workflow, to say, hey, the arrows also move faster when they approach faster areas of the spline.

Right now, they still have a constant speed. They're just colored differently. So--

AUDIENCE: Can you define [INAUDIBLE] the vertex color [INAUDIBLE]

JOSE ELIZARDO: Possibly. I have to try.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah, I could see how that could work, yeah. Yeah, that's something to explore.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah, oh yeah, absolutely. I mean, I can look up other projects. There's--

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: We have actually projects like that where we have factory layouts where it's like, you know, we approach a machine that's problematic and we fix it and go somewhere else. And we have scenarios where we do fire evacuation scenarios. What is the fastest way to get out of a building based on [INAUDIBLE]? There's like a lot of use cases for this. There's a lot of different experiences you can create for sure.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: The projects themselves? I mean, some probably not because they were done for customers and, like, with customer content. Some of the stuff that I build-- and yeah, I can-- yeah, I built it, so it's not like it's proprietary to anybody except myself. But some of the more sort of-- some of the more elaborate projects that we've built for customers, probably not. And some of that stuff is more customer stuff.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah. Well, we have a really interesting one where it's basically a-- someone else in this room can comment a little better, but it's a full train configurator. You know, the different wagons, the different styles, how many wagons, and what's the cost. It's like this whole cost analysis thing that it does. It's pretty slick, pretty slick.

Yeah So any more questions, comments? If not, I'm going to show you my last slide because I'm supposed to show you this.

Where's PowerPoint? So fill out the survey. It's all I ask. Oh.

If you hate it, that's OK. If you liked it, that's OK too. I always want to get better. So take a minute and fill out the survey please. With that--

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: The survey? I didn't-- sorry, go ahead.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: The glow? So remember how at the start of the whole thing in Shader effects, I connected-- let me go back into Max. I connected-- oh, I connected, in my shader effects tree, I said, hey color, connect yourself also to emissive.

Emissive is basically self illumination in the world of shader effects. And so I'm driving the emissiveness of my material with this color, whatever this color is. So that makes it self illuminated.

And then in Stingray, you basically have, in this midday shading environment node here, this is basically where you define all of the shading effects like fog. You can add exposure, your exposure control here, your tone map mapper, [INAUDIBLE] occlusion, screen space reflections, depth to field, motion blur, lens quality to distort the lens. And you have bloom in here. So when you enable bloom, it's going to take all the really bright surfaces and kind of just bloom them out a little bit, create that glow is what you're looking for.

So you need the self illuminated material and you need-- or something bright, you know. Doesn't have to be a material. Can be light. And then you use the bloom to blow it out a little bit. Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Sure.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yeah.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Oh, good question. Like, anywhere in the world?

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Well, I mean, I've tried in the US and Canada and I've got really great results. I've never gone outside. I'm assuming it's a Bing map.

So-- and it's GIS data. So if it's available, it should work. So-- the question was whether you can create those Infraworks city models from anywhere in the world, I guess. And the answer is I haven't tried.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Yes, you can. He knows. I thought you were nodding in agreement to that question.

AUDIENCE: [INAUDIBLE] the features.

JOSE ELIZARDO: Sure.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Sure, sure, sure.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: OK.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Right.

AUDIENCE: [INAUDIBLE]

JOSE ELIZARDO: Probably not like villages and countryside stuff. But I've done the LA. I've done a part of LA.

I've done Montreal where I'm from.

I've done that area there for the demo, North Carolina. I did another area too. Everything looks really, really good.

Like, even-- the LA area is like these really-- all of New York and New York City. New York City one is pretty freaking cool. I mean, you've got real big buildings and you can create like a race car game if you wanted to just based on that content.

It's really cool. So I didn't repeat all the questions that was asked. I'm sorry.

All right, if you guys got nothing else for me, I'm going to give you guys another 12 minutes back of your life. Cool? Thank you.