PAUL: Hi, how are you doing? You guys enjoying the show so far? Not sure I need a mic. My voice is booming anyways, usually, I think, so sorry if I hurt your ears.

Thanks for coming out. So far, I've found the show amazing this year. All kinds of great stuff. Today, I'm going to try and show you some about mechanical rigging and getting around how to start setting up rigs. As I just talk about it to get started, I'll get this playing in the background.

This is a quick model I put together for this class specifically. I knew I wanted to kind of target some mechanical side of the field what goes on here at AU, so instead of the pure character work, which is what I tend to be known for. The model was put together just kind of loosely off some models I found, or least some photographs I found, of some mechanical systems. I think it was from sort of the car industry, but I took a bunch of liberties and changed a bunch of parts so that it had some problems to solve. How's that for the class?

So this class isn't necessarily trying to teach you how to rig every robot. It's going to teach you how to try and understand how you might approach putting these things together. I've done lots for all kinds of companies, mechanical companies, over the years, doing all sorts of things. Lockheed Martin Aerospace dealing with all kinds of crazy systems, and I had to learn a whole new language for it, which was kind of fun because we have a completely different lingo in the animation industry.

And what I realized is, of course, accuracy is paramount, whereas in my industry, it kind of looks OK. Good. Well, off you go. You render away, and you do some compositing, clean it up, make it look beautiful, and you're good to go.

So the idea here is I wanted to try and figure out a rig that can be used, manually animated specifically in this case-- most likely, anyways, I guess. You could have it driven with data if you could do it with position data. But as we start looking at this, we're going to start sort of deciding how we should go about rigging this and how we're going to use it, so let me just pull that down there.

Here's the model. Multiple pivot points, as we can see. It's got to sort of a double armature, the main arm; a piston that would be driving it, obviously, up and down that needs to look like it's driving it up and down; added linkages. The arm on the end extends, as we saw in the

video there, so it extends out. It also twists, and it only twists around single axes.

This is one of the problems I often here and get people coming and asking me how do I deal with this. Because certain axes have to rotate only on single axes. Other ones have to rotate on multiple, and it's not like an arm for character. Arms for a character are actually really, really easy. Ball joint here, planar, ball joint, we stick and IK solver in it, and we get exactly what we want right away.

With mechanical stuff like this, you don't have that. You've got to sort of limit certain joints and make sure certain things aren't going to rotate in certain ways. And I always get the emails, how on earth do I make this function and only do what it's supposed to do and not just anything? And so we're going to take a look at how to sort of limit those things as well, and where to set it up and where to start and how to approach it.

So the model, first off, is the first thing I want to start, and looking at the model and deciding if this is the model you should be using. Again, I get a lot of clients coming, asking me can you rig this? Can you work on this for me? You know the first thing I ask?

Send me your models. I'm not even touching them unless they're actually built properly right off the bat, unless they're clean and what not. Unless you want me to fix them all, and obviously pay me to fix them then. But they need to be clean models and working.

One of the things, right off the bat, is never scale objects at the object level, ever. Just don't do it is what it boils down to. There's times when I cheat this, but generally don't. So I don't know if that's something that's done in the CAD world, but in the 3D world, in Max and Maya, whatnot, you really don't want to be grabbing an object and just scaling them randomly up and down.

The problem is it gets passed on. That information gets passed on through hierarchies, and it'll cause all kinds of problems down the road as we start doing other things, skinning objects and hooking things up. And you'll suddenly unhook something, and it'll go flying off or whatever, and it's because of these scale values being put on there, especially non uniform scale. Never grab something and scale it up along one axis. That's the absolute worst, and you'll find all kinds of issues crop up with that.

So really, you want to make sure you clean your model first and go through and check for those things. I'll scale stuff as I'm modeling, sometimes, but the first thing you want to do is

you want to make sure that you go in and do things like resetting xform. And there's the tool here that you'd reset with, so you'd be able to go in there and grab reset your xforms. It'll reset the scale values, reset everything, but keep your model looking the way you'd had there. You want to go through clean everything right off the bat.

You want to make sure there's no gaps or holes in models. Even though this isn't what, normally, people would call a character, I call it a character anyway. Everything I sort of deal with is a character, in a sense, in my world. We are going to be using skinning and whatnot along here, and if you've got holes in your model that's just sort of haphazardly put together, you're suddenly going to find out about it when you start rendering. You don't want to get that come up, so you want to be checking those.

And again, so a good way to check that stuff is we have the xview here, and you can look for things like open edges and have it actually tell you where the open edges are. Now, it says there's 48 open edges, but if you go around and look, it'll be where I've intentionally left holes in the model inside, so there's no interior pieces or whatever. But if you look around it and make sure there's no gaps or something I didn't get welded, things of that nature, because they will crop up and just cause you problems down the road. So I always strongly say you've got to go and fix those things ahead of time, before you even start to you look at starting to rig and put things together.

Next thing is piecing pieces together. This is a lot of parts in this model. There's a lot of little separate individual bits and pieces. I've currently used Grouping to group it all together.

There's the Group tool in Max under this menu. Be honest with you, I forget where it is because I do it through a hockey. But the Group tool, I have the advanced menu in there. You know, I never use it out of the menu. I just realized that.

But I've contained pieces together, and you want to do that. You want to bring pieces together. One of the best ways to do that is just attach objects. Instead of having all of these loose bits all over the place, just attach them together. If they're just one contained piece that it's going to move all at once, get it down to one simple thing.

I think one of the differences, as far as I understand, you need to have to sort of tell me, being CAD people and whatnot, is there's a big difference between what's sort of done here and what CAD deals with and how you're dealing with CADs. CAD, you want the parts. You want the pieces. You want to know how it gets put together.

Here, we're trying to make stuff look cool. That's really what Max is four. There's a certain amount of technical that can be done, but really, we're looking to make things look good. And things need to be working fast.

You start animating really complex armature structure something, and it's moving slow in the view port, and it's hard to move around or whatever, that just gets more and more frustrating to animate. And then what you do is you start saying, well, that's good enough because it takes me too long to make the changes, and all of a sudden, your animation starts to suffer. So that's why we don't have all the fancy constraints that CAD packages have, where you constrain things in just amazing ways. We don't have those because they're not fast, and we just need fast. So we have to figure out how to put this thing together, make it fast, keep it fluid as you work with it, get it to render fast, all those sorts of things.

Really, what we need to do now is start to determine where our main pivot points were. Where are things going to get linked together, and how things are going to function? That's also going to help us determine how we want to interact with it and how we're going to be able to manipulate this rig and be able to work with this model and be able to animate the whole system. And this is going to vary depending on several things.

The pivots, we've identified pretty much straightforward. Here's, obviously, a main pivot right here, main one up top. These, I call secondary in a sense. There are lots of pivots there, but they're all secondary to how the full thing functions. They all move around those center two, in a sense.

The other main pivots on this are a couple that are a little funky, and this is where it becomes a lot different than a wrist on a character or whatever that just is a ball joint and rotates around in any direction. This one's got multiple ones. The whole thing can actually twist along its length. This piece down the center here goes in and out and can actually rotate around it.

This piece rotates up and down. This piece also spins around it along the end-- its axis along its length, so it's nothing like a wrist. The end result is kind of the same. You've got full motion, but the way you're going to deal with it in animation is completely different, the way we'd set it up.

In a character, I'd have one control object at the wrist that handled the position of the arm, rotation of the hand, and I could place it wherever I want. In a system like this, it's going to be

completely different because you can't just have one pivot because they're actually separated and can only rotate around individual axes all the time. So we're going to have to address that.

So you address the complexities of how the system works. And then you're going to have to stop and think about, well, how do we want to interact over all with the rig? How do we want to make this robot function? What do we need to do?

You might be doing track data, where you're actually getting data from a real machine or something that already exists. And it's spitting out a whole bunch of data to an XML file or something, and then you're going to apply it. At that point, really, what we're going to deal with is each joint is just a rotation. That's often what the case is. You're just pumping in rotation data at every single joint, and it does what the real world one does.

The exact opposite can be needed, though, a lot of time, and that's the case where you're going to have to hand animate it. And with a robot like this, this piece on the end that I currently have selected, that's sort of the business end. On that, you'd have some piece of machinery attached for picking up things or whatever it needs to do, so that's sort of where they bolt on whatever it needs to be for whatever job it needs to do.

So that end needs to be able to get to a specific location, a specific target point, and needs to be able to get to that target point and stay at that target point exactly where you want to put it. So that's really the case that I sort of decided to solve here was we're hand animating this, and we need to make sure that we put that right where we want it. And then if we wanted, we could go about running tools or something that would pull all the rotations out and maybe feed it back to a live robot, a real actual machine.

So again, because this isn't like a typical character, we run into those problems of that wrist joint on the end. We also run into problem-- and I'm just going to turn off something real quick here. As we do this, I'm just going to turn off Flex, and nice little quick way to turn off things, I'm going to open up my listener.

So I always do a little bit of scripting fun with everybody. Oops, [INAUDIBLE]. Sorry, I'm using a Mac keyboard and Windows environment.

And what we're going to do is we're going to do something called getClassInstances, so we have get, class, instances, without an r. And with getClassInstances, you can actually grab just about anything in the scene and do things with it. Great for turning on and off modifiers and

stuff, I use it for all the time.

So Flex happens to be in the scene, and Flex is slowing down my feedback right now because Flex is doing some dynamics for me in the scene right now. So if you actually go getClassInstances Flex, what it's going to do is actually going to go to get every single Flex modifier in the entire scene. Doesn't matter whether they're hidden. Doesn't matter what their on. It's going to get them all, so mighty handy.

Dot enabled, you know the little light bulb on a modifier, equals false. Turn it off. Shift-Enter on the line, and you've just turned off all the Flex modifiers.

If you're going to be rigging stuff, in my opinion, you need to know at least some of the scripting language that goes with the software that you're working with. You need to know a little bit about it, and that little bit can save you huge amounts of time. Because if I had to go turn off the Flex modifiers in here, it'd probably take me, even after building this thing, several minutes to go around and figure out what they're all on, and finding them all, and turning them all off. That's just a waste of your time and production.

Learn a little bit of a coding, in this case Max script. [SNAPS] You know, instantaneous. You suddenly get real fast, and then you can drag and drop buttons out and do whatever you want to be able to get it going again. It'll make my rig a little bit faster, but this laptop doesn't like displaying stuff very well. It doesn't have a very good video card in it.

So you can see one of the things with this rig is as that head moves and swings sideways, we obviously have a planar joint down here. Again, so we have multiple joints sort of taking place of that single ball joint that we've got in a shoulder down at the bottom. So you can think of this as an arm sticking up. The whole base rotates in one axis, and then the main pivot rotates in another to be able to get that motion that we're looking for in the ring.

So how do we solve that as well? So we've identified where all the pivots are going to be and how we want to try and work with this and how we want to try and animate it. Now, we have to try and solve and figure out the needs of the rig and how that's going to actually work. So I'm just going to flick it over here and show sort of a base setup going. And I'll actually hit left here.

So again, we've got our main pivot at the bottom, we'll call it, going up and down. The main base, rotating. One at the top, where the arm sticks out.

If you see, I've actually got a bone in there, and the bone doesn't even line up to the mesh.

You might think, well, wait a minute. So what I get is I get people trying to make their bones fit the mesh.

Forget it, doesn't matter. It's the pivot points that are the thing that matters the most. Get the pivots where the pivots need to be. That's the only concern. Where the actual bone points is irrelevant, and we're going to try and set that up.

So if I just show just the bones here for the sort of main arm, you're going to see we've got a few things happening here. Right down at the bottom, we actually have the main pivot at the bottom. Sorry, I'm just making sure I'm tracking time here. Goes up, hits the top pivot up at the top corner here, and then goes out down the arm.

And again, you see all this mess here, and you think what on Earth. Well, don't forget we've got multiple things the end of that arm does. Could have done it a bunch of different ways, but I thought it might make sense to kind of do it mostly with bones. One of the things the arm also does is the-- we'll call it the forearm if it in a sense, because again, I'm a character guy. Sorry, if there's some proper technical name, let me know later.

It has to extend as well, so it actually pulls out and stretches out. That means that the end of our target is actually moving in and out, so it's completely unlike an arm at this point. And we have multiple pivots on the end that we need to deal with, so what I'm going to do is I'm going to deal with it with just multiple bones to handle each one of them. And we'll discuss how they're going to get rigged up.

So as we work our way down to the end here, you'll see these first two small triangular shaped bones. What I decided was that, to make it go in and out, the one that I currently have picked where you see the pivot, this is the one that's going to slide in and out for me. It's going to do the trombone in and out of the arm.

I have a little one in front of it, and you think why? Why is there one in front of it? What's the point of that?

This previous one doesn't appear to be doing anything at all, but it is. What it's doing is, if you notice, it's aligned orientationally. It's in the right orientation, and its child one, the next one down on the order, is aligned up to it, and they're nicely aligned to one another. Good reason for this is that transforms, position values in 3D space, work off of their parents transform.

So if I want this little bone on the end here to move in and out along its x-axis-- or whatever axis. It should be the x. Go to local. --so along its x-axis here, to move straight in and out, its parent has to be aligned to how we want it to go in and out. I get this one all the time.

Man, Paul, it's broken. I don't know what's going on. I wired up to x. The x isn't moving. It's the zed.

So if you look one up, you find out, well, the zed on its parent is pointing in the direction it's going, so you need to make sure that you're connecting to the ones that you want. So in this case, I've got that extra bone in there just give me an alignment, just to make sure that its child is aligned up and going to move straight in out along the x. It's the only reason for it there. You could do it in a bunch of different ways, but I thought this was going to be pretty straightforward.

The next pivot down, again if we kind of go look at the two here on top of one another, this piece here then is that piece that's got a motor or something in it that rotates up and down. So it's just handling the one axis, rotating up and down. And then the one on the end is giving us the twist on our very end plate, the business end, where everything's going to get attached to. This way, we can actually put that main pivot where the solver is right now. We'll discuss that in a second.

We can take that pivot right here. You can basically move the arm into where we want. We can rotate the end up and down, and we can spin it around its end axis to get it exactly where we want, and it's going to stay put. It's going to go right to where we're looking for sort of thing.

Now, to drive it, what I've done is I've used an IK solver. And an IK solver's got a bit of an issue with just throwing an IK solver on it, and that could be shown really quickly if we just go and do a quick bone setup-- kind of something the same here, oops. --and go and put an IK solver on it, so we're going to go and drop a solver on there. And I should have put my things back to my old menu style.

We have a bunch of different solvers in Max. We've got four different solvers that you can pick and choose from. Oops, stay open. Where's you go? There we are.

History independent solver, a history dependent solver, IK limb solver, and a spline IK solver. Spline IK, going from bottom up, is one for doing cool, noodley cartoon animation stuff, where we can basically get bones to run along a spline. You could also use it for doing things like

flexy pipe work and things of that nature.

IK limb solver, I'll get to. I'm going to skip one up and say history dependent solver. The history dependent solver in Max is the original IK solver that came with Max back in, I think it was 1.2 or version 2 or something. And it is dependent on history, hence the name, meaning that it actually has to know what's come before it to be able to calculate where it currently is.

This has a couple of problems. One, if you're on frame 10,000 of your animation, it's going to be really slow because it's like a simulation happening. It's literally simulating from frame zero all the way up to wherever it is every time you do something.

The other problem is, because it's history dependent, you could say, well, you know what, I want to set up an animation that's got loop in it. It goes around and around. So I want to start here, and then I'm going to animate the thing going around, and I'm going to end there at frame 10,000. And if I hit Play, it should just keep going around and around in circles really cleanly.

Problem is at frame zero, there's no keys to before it. So the solve at frame zero and the solve at frame 10,000 can actually be completely different because it's dependent on what comes before it. So it's really not a good solution, but it does some neat stuff. But realistically, you can get around with it-- get away with not using it for the things it can do well, for the history independent solver.

And the history independent solver is not dependent on history, so it doesn't have those oddities and those problems. It does a slowdown at frame 10,000 on you, but it is a little more limited in how it functions. And again, I'm not going to get into all the issues with it and sort of the advantages maybe a dependent solver because we'd have to go down a whole different road with it.

But take a look at it. It does some neat stuff, but you're always going to run into these headaches down the road. So I just avoid it entirely. I haven't used the history dependent one since the independent one has been added.

So I'm going to say history independent solver and take it from this route down to the point and illustrator our first problem. So I can grab my joint and move it up and down. You say, hey, that's great. I can go and place my arm wherever I need it, and that'll work really well.

But the problem's going to arise when we try and rotate it sideways and start rotating our rig

over sideways. You can see how it's gone and fallen over on us. We just broke the main pivot of the model. We just snapped it off sideways.

And I'm guessing your clients and everything else aren't going to be too happy when you break their models, and you say, hey, this is the way it's going to work. You're not going to make too many sales, at least. So we need to find a way of limiting the rotation of the base of this and making sure it doesn't do this.

Now, there are some limits. So if you go into the bone now, and you go to the Hierarchy panel, and you say IK, you can go and check and say, hey, look, there's these limits. That should work. Go and limit it to certain axes.

It doesn't like doing this. OK, it really doesn't like when you turn these things on and start trying to limit them. It'll cause it to freak out and do some odd things, and it just doesn't like to be limited. So I'm going to suggest just avoid that. Every now and again, I'll use it for something, but I'm not going to use it for anything.

It would work on this planar joint. If we wanted to try and stop it going through specific angles and say, well, we only want you to be able to rotate through a certain amount of degrees and then stop. The problem with an IK solver is, however, you can just keep pulling the IK solver away from the end.

So really, when it comes down to it, your animation's going to be the issue because if I grab the solver on the end here, there's nothing stopping me from pulling it right off the end. So even though the bones might stop at the right part, my animation's still going to be a mess because I went pulled the control right through the object it's supposed to be grabbing and out the other side of it, and it's going to be probably not doing what we want again. So really, it's going to come down the innovator to deal with that and make sure that it's dealt with and sort of cleanly animated as we're working. So again, I tend not to limit those things.

Let's see if I can just get this at a nice angle here. OK, so I pull scrub along again, and you see it moving. Well, you'll see there's a whole lot more stuff going on now. Let me just show the control objects quickly.

So here's my control set that I've decided on. I've got a solver in there. I've got my main pivot going, sort of my main arm's going. How on earth are we going to animate this thing? How do we want to interact with it?

How do we want to actually be able to grab it and move it? In this sort of situation, there's no way you're going to get away with one control at the wrist like I was talking about. You're going to have to have multiple in this case.

The red one in the inside, in the middle here, this is strictly the position. This is where we're going to say this end of the robot arm has got to be here, and it's going to be placed in that position. So that's all that is going to be dealing with. So if I grab it, you can see it works with it. I've locked off the rotations through the Hierarchy panel with link info and said, hey, you can't rotate this, for the animator to deal with.

I've decided what I need is I need another control along the arm, and this control along the arm is going to handle it stretching and moving in and out and allowing it to be pulled in and out. And if you can see those joints, this joint here, this one that I said was going to stretch out, actually doesn't really look like it stretches out so much. It actually pushes the other one away from it. Well, it is stretching out. It's just that there's an IK solver at the end of it, and the IK solver is saying stay there.

So instead of it stretching out, it's pushing the rest of the robot back. This becomes one of those things, what's better for you, and how are you going to interact with this? Because you might be saying, well, when we animate this, we're really going to want to be able to just extend that arm and keep the position the same, and we'd have to rig it differently. If you wanted both scenarios, you're probably going to have to have two rigs that you blend between or decide and pick and choose at any given point in time. How do I need it for this bit of the demonstration of the rig?

To set that up, it was really simple. I wired the exposition of this into this control object via parameter wires. OK, so parameter wires are well documented, so I probably won't get into showing you how to use that. It's able help files and find it out.

What parameter wire does, it connects one value to another value. A is going to drive to B, basically. OK? So all I did was I took the x in and out, hooked it up to the-- looks like I hooked it up to the y for some reason, but I did. So I looked it up to the y of that object, and again, this object's pivot is based on, what? Its parent, so the thing that's being driven and the thing that's doing the driving, if you're wiring two things together, know what its pivot is doing, how it's oriented. OK?

On the end, you're seeing two more controls. This big green one is going to handle just our up and down motion, and the golden color one is going to handle the spin on the end to be able to spin the end of the robot arm around and have it working. So if I were to go and unhide the robot again, you can see that spins the end. This one handles the up and down. That's basically stretching the arm out.

And of course, I thought, well, maybe it should-- I want to block that off. Maybe I haven't. Yeah, sorry, it rotates around, so it actually rotates around that pivot point as well, giving it to sort of multiple ways of being able to spin itself because the endplate actually spins when it's on an angle and spins around this axis. The arm spins along its length, so depending on what you need, and again, I've taken some liberties with the model to make it do things that maybe it wouldn't have designed in. But we're going to stick with that.

I just need to get some of my stuff back here. There we go. And that needs to be like that. So let's get that single axis of rotation happening and figure out how we're going to deal with that.

I'm just going to show the controls again, and so if we grab our big red control, go to our Move, you'll see now that it turns. Doesn't fall over sideways anymore. That's because this pivot isn't handling that anymore. It's not actually rotating it.

Remember, I said you could lock it, but you'll run into problems. I haven't locked it. It's still freely able to fall over sideways, but I'm stopping it because I'm rotating the entire base purposefully and pointing towards the direction of the end. And the way I'm doing that is down the bottom here.

See this big red control-- or point helper? Sorry. It actually has a Look At constraint, and it's directly looking at our target. So when you move it around, you'll see that bottom blue line. That's the Look At constraint looking back at the position control, so it's rotating around. Now, believe it or not, it's rotating around on all three axes, so it's of no use to us to pull a value out of directly because it's freely going to be able to rotate around a bunch of axes as well. OK?

So to solve that, what we use is a nice little tool called an Expose Transform helper. So the Expose Transform is a helper object. It's a point helper, essentially, with a little bit of extra functionality built into it. I asked for it years ago, back in Max 5 or 7 or something, I don't remember what it was, just to make my life easier so I didn't have to use script controllers.

What it does is it's just designed to sit in the scene and pull values out of other objects of

transform, so we can get its rotation around x, y, and z separated out. Because if you're trying to pull a rotation of the bottom of that bone, it's being controlled by an IK solver. There is no rotation joint to get.

That value doesn't exist. It stays fixed. OK? It's kind of being handled at another level in a sense, so we can't get at it.

So the Expose Transform will expose values, and what you do is you drop one in. I tend to keep them aligned to the thing that I'm trying to expose, just for logic reasons. I always turn on the display of the axis tripod so I see where it's local axes are pointing and whatnot, and link it up to the same thing that this one's linked up to, its parent object. And again, I'm very careful and try and make sure that the parent object is aligned the way that I need it because rotations, too, are actually happening in their parent space.

So if I hit the Up and go up one, there's the point helper that's the parent of both of them. Notice how zed's up and not sideways. That's because I'm not interested in a pivot that's angled over. I'm interested in the pivot around here, so that when the boom moves around, I'm getting this axis that's pointing straight up and down.

My Look At helper happens to be pointing up at some crazy angle. That's irrelevant. It's rotating around its parent, essentially. OK?

So Expose Transform helpers, you go and take and say, well, I want to expose the base Look At. That's our big red one. And I want to reference its parent, so I want to get the rotation difference between the one I've exposed and, in this case, its parent.

So you can actually uncheck that and pick anything else in the scene anywhere. It doesn't even need to be in the hierarchy, and say get me the rotation values, or position values or anything else, between these two objects. OK, so I'm saying get me this rotation value, and what it does, it exposes this whole pile of values that we can now wire up and get into.

So if you watch as I rotate it around, you can see the local oiler angles changing. So in its default position, we're looking at zed, so we decided that zed was up, if you remember, when we were looking at this parent object. There's the zed pointing up top here. We're looking for that local zed. We want to know when that point helper points at the target, how is it rotating around the straight up and down zed value?

And again, I have another point helper in here. Use lots of them. I've had arguments with

clients about doing this. Well, there's way too many nodes in there.

It's actually good to break up rigs into little layers of nodes. I think of them as layers. Each step does something different.

If you don't, you'll run into all the little, again, oddities. And once something's rotated over on one axis, what's happening? So it's going to break things up with little point helpers. So the mesh needs to be clean and as simple as possible, but our rig's going to have these little helpers in there that are doing specific little jobs for us along the way.

So again, what I've taken is this point helper. If we look at its parent, its parent is this-- it's actually a control object, and I probably shouldn't have made it one. But just a circle pointing down, the whole thing rotates. Sorry, it's this one that's actually wired. It's, again, linked up to a parent, sorry.

So it's this one that's got the parameter wire on it. I'm just going to go into the Dope Sheet, go down to the rotation values, and maybe it isn't that one. Nope, it isn't that one, so it's probably its parent. I did this quickly. There it is.

So it's linked up to something else, sorry, that's aligned to it, so I may have a few too many points in here, now, by the way. You just caught me out. I got a few extra things in there I probably didn't need when I was throwing this thing together.

So this track here is the zed rotation, and it's wired up to our Exposed Transform's exposed local zed access because we didn't need the other two axes. We just wanted that one zed one that spins around, so it's wired directly up to the [INAUDIBLE] dialogue. And so you can see that there's the zed track that I just picked, and here's the Exposed Transform helper's locals zed. So we can actually pull it out now. So as that head moves around the base, it's going to just keep pointing at it, and everything else is going to rotate with it because they're all linked up to it.

So it's all going to look nice in a line. It really looks like I went and limited the rotations on this joint. I didn't. I just make sure it never had to rotate over sideways. OK, so I don't need to turn anything off or lock anything this way, and so it keeps it nice and clean.

The head becomes a little more troublesome. Let me turn off [INAUDIBLE] that one. Yeah, probably do it like this.

This became a little bit more of a problem, try and figure out a way to rig this up. You've got multiple joints down here, all separated and broken apart. I needed to do something a little funky because what I would expect, as an animator, if I was moving that robot arm from here, let's say, down to pick something up, I wouldn't want the end of it, my hand, to do this. I might want it to stay flat so that I have kind of control over it. So I can kind of get the angle right and then move it wherever I want, and that angle stays the same to the ground, stays relative.

It's easier to control that way, when it comes down to the end. If every single time I moved it and tried to get in a different position, it rotated up and then go, well, I'll rotate that. Oh, I need a little bit higher. Oh, shoot, rotate that down a little bit more. You're doing what's called counter animating, and counter animation's frustrating to the Nth degree.

Because as you adjust the function curve-- let's say, oh, I need to slow it down a little bit as it gets to the end, and the rotation of this will be slowing at a little bit different rate. Your function curve now, and you'll start getting this little waiver in your end thing. It won't look clean and neat, whereas you guys are used to making robots that look just perfect. So we don't want that. We don't want to try and fight it and counter animate it from doing what we don't want it to do.

So in this case, I've taken this end bone, and I've wired it so it actually counter rotates again so that it's always trying to stay level. Now, I had to be very specific here. I couldn't just turn off don't inherit rotate and don't inherit the rotations of your child because then, as the whole thing swings sideways, it would twist sideways. That joint can't do that. It can only do it up and down, so we're back to wire parameters again, Expose Transforms, and Look At constraints.

So you'll notice I've got a point in here that's kind of pointing down on an angle. It's pointing back down to the base of the rig, so the head's actually looking back down through to the base. And then somewhere in here, I think it's the blue one, I have an Exposed Transform helper exposing its rotation values, comparing it to a specific object in this case. In this case, the actual position control object, so it's actually taking that Look At constraint that's linked up to the end, that's looking back down to the base so it's staying aligned to it. But as it goes up and down, the angle changes, so we can see that. If I go and do move up and down, you can see the end is staying level to the ground.

So really, what I want to know is I want to know the angle change between that point helper that's pointing down and this object that is staying level because that point helper pointing

down is actually rotating up and down as it goes up and down. So I mean, you can see that it's rotated around, and it's looking much flatter now than when it's up top. See the blue one pointing down there? So I need to know the difference between those two rotations again.

No problem. Expose Transform helper, expose the rotations between it and our main control. Get the value. There's the one that we need, and you can see the values changing. And

Again, we're going to do some wiring, so I've wired this back into that local value again with a negative. Or maybe it was a positive, depending on how the axes lined up, but think of it as a negative. So if you're rotating, I'm going to just go the other way, so as the difference between the two rotate, it always stays level. So I'm forcing it around that one axis because we can't have it going around multiple ones.

Again, break the model. Break the system that we're developing. So now, only that one axis, it's always going to stay nice and level and flat to the ground no matter what we do. Again, making your life easier in animation.

I can tell you this, and it may be something I didn't say, is if you forgo these little things in your rig and say, oh, I'll just rush ahead, I'll deal with in animation, you'll spend way more time trying to make your animation look good than you would have learning how to make the rig work properly. And then the next time you do it, you're still going to run into the same problem, and it's just going to compound itself. Solve the problem at the rig level. Make your animation life easy.

That actually is usually harder and more time consuming because, now, you've got to put it through a whole bunch of different animations, not one. So make the rig right once. Animate easy from that point on, and you're good to go.

So that solves a little bit of that. This end rotation is nice and simple. It's just blinked up to it basically. Doesn't need to be anything fancier than that. It's the final one on the end.

I have this nice control object because it's nice to see a nice control object, not have to hunt and try to find some bone buried inside the mesh somewhere. So I always make nice, little control objects. Really, the easiest way is you could have this bone linked to this bone here linked to the control linked back in the bone.

And you can say, well, Paul, why don't you just link the mesh up to your control objects? That's a good question. I just don't. I just simply don't. I like to keep things as layers.

I've got a rig. I have a control system. I have a mesh. And then I even have layers of rig going on, and those layers make it much easier to sort of see and be able to deal with what's going on and be able to change things up. I keep them separated on purpose so that if there's a problem or a change needs to happen, it's much easier to deal with.

I also would never directly rig a mesh. I've seen people just going, yeah, you just link the mesh together, and then you stick an IK solver in it. That's a disaster waiting to happen because, all of a sudden, the modeling department comes back and goes we just made a change to the model. Man, I'm going to have to re-rig this because the model's rigged.

Me, I just delete the thing and re-link in the new piece. Merge it in, and the rig's there. The rig's separate entity from the mesh, so never, ever, ever rig the mesh. Again, you're going to waste time at some point in your production when somebody makes that change, that rev on the mesh, and you suddenly have to start over again on something.

So that head's got a little bit of complexity to it. Again, we've just got a wire in here to get this working. What about our secondary rig? So our secondary rig-- kind of go back here and take a look at.

We've got this piston. We've got these linkage joints and whatnot. I call this secondary because I think of things as, and sort of again, in terms of characters all the time, I think if things is the animation rig, again, layers. So I think of the animation rig, and then this secondary rig that's driven by the animation rig to handle muscles in a character, or this case, our muscle is a piston, another arm, linkages that are driving it and whatnot.

So they're a secondary rig driven off the animation rig. Again, as a layer, and if you keep things as your head as layers, it's not complicated anymore because each piece does a task. And you need to solve the one task and not solve them all, basically. So solve them one at a time.

So pistons are kind of neat. I often get asked how do I set those up and the little linkages like this. So this is pretty straightforward. Let me go back to-- where do I want to be? Right there.

So for this extra linkage that's in here, just two bones, IK solver, link the base to one, like it to the other. You say, well, that's a bit of an odd linkage happening there, but it's not, really, when you think about it. There's this piece going up and down. The bone doesn't need to go to

the end. I'm concerned with that pivot, and then where those two pivots are for this piece.

Just think of little triangles. Break it into little triangles. OK, doesn't matter where this end one sticks out. I've stuck a point helper in there and linked that point helper back into the bone so that I've got a pivot for wherever the base of the piston assembly is.

Now, pistons are actually pretty easy to set up, and again, I get questions all the time. How do you get these things to work? And it's just two Look At constraints. The two ends look at each other, and again, I often get from the CAD guys sort of thing, how on Earth do you do this because you want to make a piston work.

Nah, nah, nah, we're in Visual land now. We make it look like a piston's working. We don't make a piston work.

So all we do is we simply take the slider and the [INAUDIBLE], and get them to actually point back and forward to one another, and again, I do it with a separated rig. So three point helpers, we're going to need, and you're going to need three point helpers because you can't have a look at b and b look at a. You've got a dependency loop. OK, it's going to give you a big warning right away and say you can't do that, so we add another one in.

So this one down here is an extra little one, and it's basically at one end. It has a child right in the same place, and we have one of the top. Back down to the bottom child that we talked about, it looks to one the one at the top, and the one at the top looks back down to the parent. No more dependency loop.

We just stepped outside of it. They're not looking at each other up to its parent, so there's no more dependency loop happening. So it's really easy to set up, and again, you don't have to use bones. You don't have to do anything fancy than a couple of point helpers, but do not rig the mesh. Because again, we can now go and make changes to the mesh, no problem on this.

Those pivots stay the same, or if they don't, we can sort of just go and grab-- oh, you want to move that pivot over somewhere else and change the mesh? Well, I can go and make that wherever you want it and go and link in the new model, the new whatever it is, the change the, V7000 piston that has to be replaced in there or whatever, and do whatever you want with it. So again, just a couple of point helpers, and it's nice and clean.

This one's a bit of a fun one. I'll turn on-- again, this wasn't in the original robot for sure, but I thought, well, it'd be fun. So you see, I've got a chain around here. This is a motor of some

kind.

Remember, the shaft rotates and spins around, so it actually rotates. So this motor drives a chain that goes and drive that center spindle somehow. Again, I don't design robots, so this is just to look cool and just a problem to solve.

Chains can be a real-- when you think about how a chain works, man, it's a gazillion little tiny pivot points all over the place. They have to mesh perfectly. Well, they only have to do that if you're going to get real close. Don't they? Why can't we cheat it?

So again, if this is a display thing where, OK, I'm making something to look cool for the clients so that we can sell them robots. Are they going to fly in there and go, hey, that chain link, and start complaining about how the chain-- it's going to be a whole thing going on, and it's this little chain happening. So let's find a cheat way to do it. Cheat way is real simple.

There's my chain. It's one piece. There are no pivots. There's no pivots at all in it. OK?

So I made little chain pieces. I'm a bicycle rider, so I modeled it after a typical bicycle chain. Lots of pivots along it, but it doesn't pivot on anything. There are no pivots. There's just one big long piece.

And you use what's called a path constraint, and the path constraint allows you to be able to make sure that follows a path. And all I have is a spline shaped to go where the chain would go. So realistically, when you're looking at that, even from this angle, does it look like my chain's bent? Not really. You can just start seeing it even at that angle.

So how close are you getting? Because rigging a chain can be a real pain. I've done proper ones with things like tank treads which are, essentially, sort of a chain. They're a pain to set up. It's a long winded process.

Do you need to get close? No? Then do this.

Because I've done this on tank treads as well, and nobody's ever noticed because the tank treads are spinning around real fast, and the tank's flying over rough ground. Who cares if each panel is bending a little bit as it goes around the corner? Nobody noticed.

So again, all I've had it do is go around it, and then the percent long path, you'll notice it's grayed out here. Well, it's just simply wired up to the rotation of my control object, so as the

control object rotates, it's driving the pram wire, it's driving the percent value, and driving the chain along it. How did I figure out how far the chain should move for each rotation? Well, you guys probably are better at math than I am. You can work out the math, I guess, for the length of the path and whatever.

I just kept plugging in point values until it looked right. It doesn't ever have to move that far back and forward again. You know, how far is it really going? How close are we getting? You don't even see the teeth sticking up to it, and guess what, the chain doesn't even fit the actual sprockets perfectly.

OK, so what are you doing it for? How accurate does it need to be? And you guys are probably pulling this stuff out of a CAD package or something, and it would need to be as close as possible. But you can see that my teeth aren't exactly matching.

I wasn't too accurate with it. I could have been. I could have got it right on, but again, I was hurried. But you don't even necessarily notice it.

OK, so yeah, making it more accurate, get the value right, and it would look absolutely perfect. But even at there, you're only just starting to see it that chain's bending. Aren't you? So I like cheats. I like making things work fast.

I'm running out of time. OK, let's talk about a bit of the fun, then. Let me just make sure I haven't missed anything along the way here.

Gears at the bottom, pretty straightforward. So you see, I have this gear in here, and a motor. That's just a param wire again. So all I'm doing is, if you remember, I said that the bottom of the IK chain, we can't get a value out of it. Wild guess how I'm getting the value.

Same way I did for the Look At constraints. I'm using an Expose Transform, getting the rotation of that base joint, how much angle change is there, and I'm driving this motor, spinning around. So everything's backwards.

Something to note, we don't drive things the way they're supposed to be. The motor drives the bottom of the rig? No, no, no, not in our world. The rig, the bottom of the arm actually drives the motor. It makes it look like the motor's driving it.

Just think of things in reverse, often, and it's way, way simpler. Don't be literal when you put this stuff. It needs to look literal in the end result. It doesn't need to be literal.

We're not doing sims to find out how strong that motor is and will it pick up the whatever it is. We're not doing that with it. We're making it look like it's right, so think just think about it in reverse.

I want to end off in a bit of fun here, so I think I've got eight minutes or something. And it has to be noticed we had all that dynamics going, so let's go turn some of that back on again. Let me go back to my-- whoop, you know what I have? I've got my macro recorder on. That's why [INAUDIBLE].

So I want to do my getClassInstances Flex.enabled. You know it's got ClassInstances is in brackets, too. Right? Go and get me this thing, which can be all the instances, and then turn them dot enabled equals true this time. Turn them all back on.

See, and it'll get a little bit heavier and slower at this point, and you can see it's really chugging. It's because it's starting to calculate dynamics as it goes, real time dynamics in the view port. I often get asked how did you do that, man? Which dynamic system, real time dynamic system, or whatever, blah, blah, blah, simulation and this, that, and the other? And it's like, no, I needed to make it look right again.

So I could have gone with mparticles and pflow. That would work really well right now. You could get a really nice sim going on it. You could use any kind of sort of hard bodies and [? change them ?] together, some sort of soft body simulation. You could start setting up that stuff.

The problem I find with simulations is they take time. They tend to be a little bit more work, and then you have to run the simulation often and bake it in or something, and you don't have time for that. Sometimes, you just need little jiggly bits.

Again, I think of it as a character, where I need a little bit of-- I'm a little too soft here, so I jiggle a little bit if I bounced up and down. I don't need real dynamics on there. I just need to see that it's loose. It's got a little bit of flex and give to it, and it's going to bounce a little bit.

So that's what I decided really was needed here. I mean, those cables bouncing up and down really aren't an integral problem with this, or something the client's going to say, hey, wait a minute, that moves too much. I just need to show that they're flexible there, and they're going to bounce around a little bit.

Got some cables on the floor and everything down here. And if you watch those cables on the floor, they actually drag along and drag themselves back into place again, looking nice cables. And I'm using the flex modifier, and it's one of those modifiers people always ignoring and go, ah, that doesn't do anything any good. I've been using it for years and making it do all kinds of really cool things. Some little tricks to it, though.

Let's take a look at this mess up top. So I don't know if you remember in the video that I forgot and closed because it takes a while to load, so I won't load it because I'm running out of time. It needed a flex up and down. Well, again, do I need every cable flexing individually? No, I needed to make it look like there's cables there that are going to flex around.

So instead of doing a sim on every single one of them, what I'm actually doing is I'm going to do the sim on separate individual sim pieces, call it a rig, and I'm just going to make sure that the mesh follows it. OK, so here's how you set this up. This is just a spline. If we go down to the editable spline level, you'll notice it popped all over the place and went and freaked out. Oops, no, I'll go in there.

It has very few points along it. You'll notice they're just the odd points, so I can actually get them in the right places. Now, when I sim this, the problem is I need my points evenly spaced apart and enough that I can get little springs between them to stop it from just blowing up and going all over the place. Little springs that'll hold them all together.

So what I then use is a normalized spline modifier, and the normalized spline modifier actually rebuilds the mesh based on a distance apart of vertices. So you'll see it's set at segment six here. And so actually I probably got about 50 little points along that line, now, if you could see them, and they're evenly spaced. It rebuilds it, and that's a segment link.

So it's saying each vertex needs to be six units apart from one another, so I played around it and got it to the right point. Not too many. Too many, it's going to slow down more. Enough that I can get a nice flex going to it.

I'm going to use the Flex modifier. The Flex modifier is really simple, except most people don't set it up properly. Looks like I just had something really weird happen from what I was just doing. There we go.

Most people put it on and just sort of leave it and don't do any setup in it, and that's not how you deal with a Flex modifier. You have to do a little bit of setup in it. Oh, my computer's

thinking I might have to go into a hand puppet show here.

My poor Mac gets abused with Max here. It's not very fast. We'll see if it returns. Got enough up there that I can show it around for the most part.

So if you notice right away, in the Flex modifier, in the first parameters dialogue, got the flex value at the top, it just says 1 right now. I almost always leave it at 1 and play around with it. Maybe down, but with the way we're going to use it, never turn it above 1. It'll just kind of go crazy. So you can sort of limit the amount of flex, but don't try and take it higher.

Strength and sway, strength is a value that tries to pull the thing back into its position again, so it's how much is it trying to pull the wires back into its position. Sway is badly titled. Sway should be dampening, is what it is. It dampens the effect and tries to slow it down, like you're running in water or whatever. So it makes it slow down sort of thing nicely.

So playing around with those two values, you have to sit there and just mess with them back and forward to get them right, to be able to get the right amount of sort of pull back and the right amount of just-- so it doesn't just go boing. All right, turn up the sway, dampening. It's all I ever say in my head, dampening. As it goes up, it should dampen it, not make it sway more so that it just flex back. It'll pull back, but not just keep springing over and over again.

Right below it-- I don't think I've come back. Shoot, wouldn't you know it? I've been working all day. I restarted before I got here, I never had a problem with it, and of course, during the class, it overwhelms my poor little computer. No, it's not going to come back.

See where it says [INAUDIBLE] there? That's the solver that's being used. It's the math that's being used for it. By default, it'll say Euler, and Euler mathematics are very, very, very simple dynamics. Euler was a mathematician.

And the [INAUDIBLE] is just a more sort of advanced solve, is what it boils down to. I like [INAUDIBLE] helps that little bit. It doesn't make a lot of change. It helps that little bit.

Then you'll notice that there's a samples value. Samples values usually defaulted at five, and for some reason, I've managed to find that four works better. If you turn it too low, it's like turning up your sway almost. It becomes kind of slow and non-reactive. If you turn it up too high, it kind of freaks out and blows up on you.

And I've kind of found four is the sweet spot. Sometimes, three, but usually four. Why? I don't

know, but I don't know the code underneath the hood of this. Unfortunately, now, I can't show you what's going on underneath.

And do I have time to restart a Max copy? No, I don't. Close the program. Sorry about that. See if I can restart Max here, probably not.

At the very bottom of that last panel there, there's a section where you can deal with springs, and this is where it actually is a nice trick. Now, I actually have a whole section on my website about dealing with Flex and setting it up, so go there, and you'll see what I'm doing. By default, essentially what Flex is doing is taking every point and making it spring around. But point one has no understanding about point two, so when point one freaks out and moves around, it doesn't pull this one with it. So the object is left to be able to sort of just go wildly out of shape.

So the springs-- you can pick the vertices and go give me springs. You need to know the distance apart between them, and you can basically say create springs with this distance apart. You want just enough springs that you've got a spring between each and every one of them. And those springs have a spring value, by the way, and you'll actually see it says preferred value, or something, I think it's called. Now, that I can't see it, I can't read it back to you.

Pay attention to that preferred value there and type it in because if the value's too high, it'll blow up. Too low, it's not doing anything. Type in those preferred values, and basically it's handling the strength and the sway of the springs in a sense. Again, those usually don't need to be touched too much. I usually put those at the default value that it tells me to put them at and then start playing with those strength and sway values to be able to get the right kind of flex.

The other trick is the ends of your spline are now springing around. The ends are coming off. Well, how do you stop that? Probably can't see anymore, it says edge vertices, here. If you go into that Subobject mode, again, it's counterintuitive, but once you know how it works, it works.

If you pick a vertex, it's no longer in the sim. It's locked off, so what I did is I picked the two end vertices, have them selected in that subobject, and then there's a couple up around the top here. So there's one right in here for sure. I pick it because it's in an arm that holds it up and out of the way.

I pick it, and now my ends are locked down, my middle points locked down, and the rest are

able to flex around. And they try and hold their shape as best as possible and give you a little bit of flex. I did the same thing on the ones on the ground, and I did the same things going from the back of the thing back down to it, literally just the same thing three times over. It took me minutes to set up, literally just minutes to set up.

And then when you animate it, you can go back and say, you know, it's flexing a bit too much. Let's just pull that flex value down or maybe a bit of strength on that one. You can even animate those values, if you had to, over time. This frame, it suddenly kind of moves too much. Just animate it down and back up again.

OK, so doing any kind of dynamics is always going to cause you a little bit of fun. I believe that's it. Isn't it, guys? So I'll go back to that.

So thank you very much, coming out. I hope you picked up something in all that. There's a lot there. Sorry my box decided to go the way of the dodo.

But again, head over to my website. I handed out my business card. Anybody else needs it, I can get you one. And take a look at some of my little tutorials on there for little tips and tricks and things and ways of making stuff work. Thanks, I hope you enjoy the rest at AU.