PROFESSOR: All right, so this lecture we talked about lots of cool origami design software, all written by Tomohiro Tachi. We had Origamizer, Freeform Origami Designer, Rigid Origami Simulator. And then he talked about a bunch of other things that he's built using that kind of technology, like cylinder folding and so on. And I got a bunch of questions.

So first is an exercise. If you haven't picked up handouts, grab them. Let's fold something from Origamizer. And I made the pretty much the simplest possible thing you can make, which is four squares coming together. And so this is the crease pattern, and we're going to fold it. You can start folding now if you like. So I'll do it too.

So the first step in folding a crease pattern like this is to precrease all the folds. The easiest way to do that is to precrease them all mountain. And these black guys, we're actually going to fold-- I'm going to make a rectangular region here, because I didn't trim the paper to the square. So you've got a fold along these two black lines and all of these guys. So it's a good exercise in folding crease patterns, if you didn't already do one NP set two, which is due today.

When you're mountain folding, you can pretty easily guide along a straight line. So you just fold, unfold, fold, unfold. Don't worry about the mountain valley pattern at this point. It's really hard to precrease a valley.

When you're precreasing the non-horizontal and vertical folds, make sure you only crease the part that's marked. Don't go too far.

How we doing? How many people have precreased? No one, all right. I win. It helps that I've already made a couple of these. This is what it's going to look like when it's folded. On the back, you'll have the four rectangles. Beautiful. And then on the front where all the color is, you see all the crease pattern. And so we're going to-- as Origamizer always does, it adds these tabs, the tuck parts, along each of the edges.

And that the vertex you've got a slightly more complicated thing. So that's what we're going to be making.

This is an example of the tried and tested advanced origami folding of almost anything. First precrease all the creases, then fold all the creases. Then you have your model, pretty much. But the precreasing is usually the really tedious part. When we're feeling high tech, we'll precrease with a sign cutter, which is a computer-controlled robotic knife blade. We'll score along the creases. Or a laser cutter, which can burn halfway through the paper. Or pick your favorite tool. The fanciest is a computer-controlled ball burnisher.

Once you've precreased everything mountain, you want to reverse in particular the black lines. Those are really the only important ones to reverse, I find, in this model. So those are all going to be valley. You want to invert them to be valleys.

Once you've got those guys, you can collapse your flaps. Here's the fun part. Ideally, you can collapse maybe all four of them at once. Let's see. It's been hours since I've folded one of these. I've forgotten how it's done. You get this kind of floppy version. These polygons aren't flat like they're supposed to be. And this is where we have too much material at the vertex here. And so there's these tucks, which are just crimps, as we've been calling them. And the blue here is mountain, the way I've set it up.

So you just do all of those crimps. Mountain valley. Mountain valley. It's kind of like little simple folds but in this three-dimensional state. And when you put them all in, it just makes exactly the desired form. That's because this is has been designed to have exactly the right angles after you do the crimps.

And there's only one-- if your paper was rigid, there'd be only one possible shape this could take on. And there it is. This is supposed to be-- usually, you fold it this way so that all the crease lines that you print are on the backside, so you can't see them. So then you have your perfect model out here. Not so perfect.

You'd just be exposing the origami side with the tucks hidden on the backside.

Usually, you also want to hide the tucks. But of course, if you want to get it the other way around, you just fold the blue valleys. The red's mountains. But I find it's a little easier to fold this way, where you can see what you're doing. You can see all the crease lines you're making.

I can see the advanced folders are already going on to the next one. This one is six squares, all coming together. When folded, it looks something like this. But note that the crease pattern as I've drawn it kind of requires you to cut out this outer shape, because I haven't drawn the creases that are outside there, to make it easier to fold. So if you're folding ahead, ideally, you cut along all the black lines in the outside. Those guys.

And then when you want to put in these tucks-- you see how these are lined up-- you just want to crimp that. All right. You can keep folding at home.

I want to show you a little bit about the process of making that crease pattern. It's pretty easy once you know how, but if you haven't done it before, the first time is a little bit challenging. I use a 3D-- first you draw your 3D squares that you want to make. I use a program called Rhinoceros 3D, which has a pretty cheap academic license, and it's commonly used around here in architecture.

You get a top-down view and a perspective view, and all these great things. In this case, I just wanted to draw four squares in a plane, so it was pretty simple. I should have the file here. Four squares. So it looks like this. I've got my four squares like that. Very exciting.

Here's what it looks like in three dimensions. Wow. Then you export that into DXF format. Or sorry, into OBJ format. So you save as OBJ. You've got a zillion options. You want to polygon mesh. Polylines. UNIX. That works for me, though some variations may also work. Then you've got your OBJ file.

You run Origamizer, which looks like this. And you can drag in your new OBJ file. Here it is. Exciting model here. This is in three dimensions. And then you say develop. Actually, first you should probably do angle condition, and then develop.

Now you've got something like a crease pattern. It's actually just placing the squares, and you can change how the squares are placed here. I spread them out a little bit so that this tuck was not super tiny. This model isn't very constrained. And then you say crease pattern generation, and you get your crease pattern. And you can adjust how spread out you want these, how big you want your tucks to be. I made it nice and square, and something like that size.

Then when you save, you get a crease pattern in DXF format. And then hopefully your drawing program can edit DXF. I think I opened it in Rhino, then exported to Adobe Illustrator, and then open in Illustrator. I removed all this stuff on the outside, because I just wanted this square boundary. But you can do whatever you like.

So that's Origamizer in action. Wow, 900 frames a second. That's fast. Of course, if you do more complicated models, it can be a little more involved.

We've seen the bunny. What haven't we seen? A mask. Never done this one. You should see develop. Boom. There it goes. And spreading them out, trying to solve all the constraints, at some point it will converge. In the lower left, you can see its current error value. When that gets down to zero you have a perfect crease pattern.

Except for these green regions. The green regions means that the tucks in the 3D model, some of the-- it's a little hard to turnaround. Some of the tucks may be intersecting. So if we look closely we can probably find some tucks that are crossing each other. And if you want to deal with that in the software-- not just somehow fiddle around with it with origami-- there's a tool which is split extra wide tucks.

If you look at one of these, the green thing is the edge-tucking molecule. If you look at that, it will subdivide into two edge-tucking molecules. Now they're half this tall. They don't go as deep into this model. And they're less likely to intersect. As long as you've got a green thing, there's potential intersection. When you're done, this is probably a valid crease pattern, at this point. A little bit of green. Hopefully they're OK. You can keep splitting if it continues to be a problem. It just adds more and more creases.

So that's how to use Origamizer, if you haven't used it already. And go back to slides. And the slide progression of that. Cool .

So the next question is about-- essentially, it's a question about what makes a convex vertex versus a concave vertex. Concave is a little bit ambiguous, so usually we say non-convex, to mean the opposite of convex. So I'll use non-convex.

Essentially, there are two or three kinds of vertices, depending on how you count. We've got something like this vertex of a tetrahedron. This would be convex, meaning that if you look at the sum of the angles of material at that vertex, that sum of angles is less than 360. You could also have a flat vertex, where it's equal to 360. That's what we just made. I've got four squares coming together, four 90-degree angles. Sum of those angles is 360.

Or you could have a non-convex vertex. Non-convex, it's bigger than 360. And that's a little harder to draw. So I made what I call the canonical-- it's a nice clean orthogonal, meaning all the bases are horizontal, vertical, or the other way. Non-convex vertex. This has six 90-degree angles coming together. Six times 90 is bigger than 365. It's 540.

So this is, of course, inspired by the video game Q'bert. Play it back in the day. And when you put it into Origamizer, it gives you some kind of layout like this. Then you ask for the creases. And boom, you've got it.

And the thing that I printed out had this removed, which requires you to cut here, unfortunately. I also made the squares go all the way to the tip. Place them differently, and you end up with this crease pattern. And this is a little trickier, because you've got some extra tucks in here. They're quite small. And depending on how accurate you want to be, it's a little hard to fold it in exactly the right shape. Looks pretty good. It's got some-- little bit messy here in the center. If I use better paper, it'll be a little easier.

So that's a non-convex vertex. And in some sense, the point of Origamizer was to deal with non-convex vertices, not just convex ones. Convex ones, you can kind of

wrap around the paper, and just tuck away the extra material.

Non-convex, you really have to tuck away material in a clever way in order to get all of these guys to come together. Because normally, on a sheet of paper, everything looks flat. Everything should add up to 360. But if you hide away material, you can get more corners to come together, and that's what lets you get non-convex vertices.

So that's where that came from. You can't just take a convex vertex and flip it inside, because intrinsically, on the surface, it'll still look like a convex vertex, even if it's popped inside out. Some of the angles won't change. Still be less than 360. Cool.

Next thing I wanted to show is Freeform Origami. In particular, there's a bunch of different modes in Freeform Origami, and they weren't really showing much in the videos. So I'm going to show you a little bit about how it works.

So you download Freeform Origami . All this software is Windows only at the moment. So then you open your favorite model. It can be a 3D model or a 2D model. Miura-ori is a really nice example to work with. This is just straight lines in one direction, and then a zigzag in the other direction. I've got your 3D view on the left and right.

Now these views are not enabled, because I haven't turned on a lot of constraints. Now, as you see, there's a lot of different constraints I can turn on or off. In this case, I will turn on developable, which means that each of these vertices in this 3D model are flat, according to this model, so you want to constrain some of the angles to add up to 360. That means that it came from a sheet of paper. That makes it a folding.

So this is different from the target in Origamizer, where it's just a 3D model. And now you can see up here the crease pattern, which will actually fold into that. Because a developable, you can just locally unfold it, and you'll get a picture like that.

The other thing I want to turn on is flat foldability. This is Kawasaki's condition. So

it's going to enforce that this angle plus this angle equals 180. Or the sum of the odds equals the sum of the evens. When you add that constraint you guarantee a flat folding, and then this picture is the shadow pattern, if you make that flat folding, and just draw them on top of each other.

OK, so those are my constraints, and that turns on all of my views. Now I can do-- currently, I am in simulation mode. This means it's acting like a physical piece of paper. So when I drag on a corner, it'll try to fold that up, or unfold it. But this stuff on the right, the crease pattern, is not changing.

So this model, because it has a lot of boundary edges, it has a bunch of degrees of freedom. So I was like number of degrees-- number of boundary edges minus 3 is the number of degrees of freedom, in this general picture. They're crushed.

So that's the idea. You can also hold down spacebar, and it'll just try to fold everything, kind of uniformly. Or you can hit B, and it'll unfold everything uniformly. So this is all, again, not changing the crease pattern up here. If I move away from simulation mode, if I turn this check box off, now I'm allowing the crease pattern up here to vary.

So if you watch this upper right corner, as I drag on this guy, crease pattern changes. It's now allowing the whole thing to be flexible. And I can do things like, oh, maybe I want to make this really high up here. And this is stuff you could not do with Miura-ori. We're changing the Miura-ori pattern.

Zoom out over here. See what's going on. Maybe I want to bring these guys up as well. I can't make any 3D shape, because I am constrained by-- a little too exciting. You can always hit Control-Z to undo. Sometimes it's hard to satisfy all the constraints that I give it. We can do things like snap there. And wow, cool.

So you have to be a little careful. This requires some finesse. Because the constraints are not always satisfiable. But this, whatever I'm making, at all times will come from one piece of paper-- and you can print out this crease pattern-- and it will be flat foldable.

And the cool theorem by Tomohiro is that if you have a valid 3D state, like the one on the left, and you know it's flat foldable, and it came from a sheet of paper, then it will actually be rigidly foldable. And so we can unfold this thing. Whoa. Or fold it, in theory.

I see. The problem is I should first turn on simulation mode. I don't want the pattern to change. Then I let it fold, or unfold, and then it will be well behaved. This is guaranteed to work. When I have simulation mode on, anything could happen. So it could explode.

But that's how Freeform Origami works. So this question here was-- yeah, if you pull on a point when you're in simulation mode, you won't change the crease pattern. But if you turn off simulation mode, which is called design mode, then you can really change the pattern, and get it to fold into something that you want.

And here's an example of something designed with this method. And then we waterjet cut it with little tabs. And this only folds once. You can't unfold it, or else the tabs will break. But it's pretty cool. And you can just print out these-- this is made from one sheet steel and folded by hand. This was made back when Tomohiro was visiting for that guest lecture.

So first we made a paper model, made sure it looked good. And this one, we'll fold rigidly. And we made another version, which I couldn't find. It was metal, but [INAUDIBLE] ridges folds rigidly, like the videos that he showed.

AUDIENCE: Erik, what is the name of the program you're using?

PROFESSOR: This is called Freeform Origami. Or maybe Freeform Orgami Designer. All of these, if you search for Tomohiro Tachi software. It's also linked in some of these slides. You will find all three of these programs. I haven't yet shown Rigid Origami Simulator. Because it's, in some sense, assumed by Freeform Origami, because Freefrom Origami can also do the folding with keeping all the panels rigid. But they have some differences, which I might talk about now.

Next question is, on the slides, Tomohiro showed there were tons of equations. He didn't talk about any of them, and some people really wanted to know about these great equations or the conditions. What are the constraints that go on in Origamizer, Rigid Origami Simulator, and Freeform Origami. And there are a bunch. And I don't want to go into them in lots of detail, because it can get complicated. But I'll give you a high-level picture of what's going on.

So first one, this is Rigid Origami Simulator, which I didn't show you. But basically, you take in a crease pattern. You can hit spacebar to make everything fold. You can hit B to make everything unfold. And it keeps all the panels rigid. That's its goal. And there's essentially-- this software is written in a way that the geometry of each of these faces is determined by the original crease pattern. So you don't-- that's just given to you.

And the only thing really that's free are the bend angles at each crease. So it parameterizes this 3D model by the bend angles. And when you parameterize by bend angles, there's one key constraint you need, which is that if you walk around a vertex and you say, OK I bend by this. And then I bend by this, and bend, bend, bend. I should end up back where I started. Otherwise, there'll be a tear in the paper, at the corner.

So if you're going to prioritize by bend angles, you have a cyclic constraint around each vertex. And that is the one constraint you have. This was originally described by Belcastro and Hull. I know some of you know. And so around a vertex, basically, every time you have a face of paper, you turn by that amount. There's matrix B. It's a rotation. Then you rotate around that crease by however much the crease angle is. And then you rotate around the face, and you rotate, rotate, rotate. You take the composition of all these rotations. That should end up with the trivial rotation, which is do nothing. Otherwise, there would be a tear here.

So this is a constraint on the angles it's a somewhat complicated constraint. It involves sines and cosines of the angles. But otherwise, if you ignore the sine, cosine, stuff, this is actually linear. This is a bunch of matrices, rotation matrices.

You're just composing them.

So it's relatively clean. And then you get your folding motion. A little tricky to do by hand, but very easy on a computer to solve that linear system.

OK, next we have Freeform Origami Simulator, what I just showed you. This has two constraints. Or there are two constraints that I turned on. There are, in general, more that you could turn on.

One of them is developability. So here, we want to start from a piece of paper. And so we want the sum of the angles to be 360. So that is just a sum constraint. The other condition we want is flat foldability, which is the Kawasaki condition. If you satisfy both of these, we know that you'll be rigidly foldable, and that's kind of what Freeform Origami is about.

You can turn them off. You can turn on other constraints as well. There are bunch in there, but those are kind of the core two that you typically want to use. And so it's always solving these constraints. So those two systems have relatively simple constraint systems, although Freeform Origami has a lot of extra bells and whistles. So you could do cool design. You can try to force two vertices to come together, and so on. You can try to make mountains be folded as mountains, and valleys folded as valleys. You can constrain which way creases go. Those are inequality constraints.

The last one I want to talk about is Origamizer. This has a lot of constraints, and this is where it's probably more insightful to go through them. So remember we're trying to place these polygons into the plane so that these edge-tucking molecules are very simple. They're just a single crease.

So that's our-- first we're going to just sort of paramaterize how things are set up. Suppose you've got two faces, which share an edge in the polyhedron, the thing you're trying to make. We want to place those two faces somewhere in the piece of paper. And there's a rotation. So here, we've separated this edge from this edge. And if we extend those lines, they form some angle. We're going to call that angle

theta ij. That's one of our variables that we get to play with.

The other thing is how distant are they. There's wij. Here, and wji here. And just for that prioritization to make sense, you've got to satisfy a couple of conditions, that if you look at theta ji versus ij, it's negated. And if you look at the w's, you can take the sine of half the angle theta, and that tells you how much this w differs from this w.

So these are two relatively simple constraints. Then, like in the previous two-- like in Rigid Origami Simulator, you have to have closure around a vertex. If we're placing these two parameters, w and theta, denote how this guy's placed relative to this guy. And then you can-- if you look around a vertex where all these faces meet, there's the way this is parameterized with aspect to this, and this to this, and this to this. Those should be consistent.

And in terms of the thetas, it means that you should do one full turn around the vertex. You've got these theta i's. Then you've got these alpha i's, which are the angles of the face. Then you turn by theta. Turn by alpha. Theta, alpha, blah, blah, blah. In the end, you should get 360.

And the equation's written this way because these are the variables that you want to constrain. These quantities are all known. You know all the alphas ahead of time. Those are the angles of your surface. So this is a linear constraint on the thetas.

So there's also a similar constraint on the w's. This is a little bit messier. It involves rotations, involving these angles and this other angle, capital theta, which is the sum of thetas and alphas. But it's essentially saying the same thing, that this closed loop is actually a polygon. It should come back to where it started. So if you do this walk, you end up back at your origin, 0, 0.

Next constraint is the convexity of the piece of paper. So you're trying to-- you want the polygons on the outside to form a nice convex polygon, because you can always fold the convex polygon from a square. And so this is just a very simple constraint that, at the boundary, you have these-- the thetas should be greater or equal to 180. That's very simple.

Next one, these get a little bit more technical to make the molecules guaranteed to work. And so, in particular, an edge-tucking molecule, we want this to be a nice convex polygon. And so this is actually fairly easy to constrain, but all these angles should be in the right range. Don't want any giant angle. You don't want these to basically flip open to be more than 180. That would be bad.

The vertex-tucking molecule is a little trickier. There are two main constraints we need. One is that the thing that you fold, which is kind of floppy and has too much material, you want it to have too much material, not too little material. You want each of these angles in the tabs to be greater than or equal to the desired angle over here, so that you can just add in a tuck, like these guys.

Add in one of these little pleats to reduce the angle to whatever you need. If it's too small, no matter how much you fold it, it'll stay too small. So it's like the guy who keeps cutting the board and he says, "I keep cutting it, but it's still too short." So you want it to be too long initially, so you can cut it to just the right length. The angle to just the right length.

This involves all these angles, which I don't want to define, but you can compute what the angle is here. It's easy to compute what the target angle is. You just measure it on the 3D model after you compute the type proxy. And so you're constraining the thetas, or constraining this fee value.

All right, so then the other constraint is this tuck depth condition, which says this is the non-intersection parts. So you want these tucks to not hit each other. They're not so deep that they penetrate each other. And I don't want to go into the details of how that's specified, but it's another constraint.

Now over all, these constraints are fairly complicated and non-linear. But Origamizer solves them approximately. And if you let it converge, and if it says it's got zero error, it has solved them. But it can take a while.

So one of the questions was, can we do an example by hand to solve all of these systems? And the short answer is no. You really need a computer to solve

something like this. At least I would. The solution method is essentially Newton's method, that you may have seen in some context. But this is a high-dimensional version of Newton's method to solve non-linear systems, and it involves the Jacobian-- I'll just wave my hands-- which is partial derivatives with respect to all the different parameters you have. These are vectors, so this is a big matrix. And then you do a sequence of iterations using this method, which is a little easier to see in this picture.

Essentially there are two things going on. So you're reacting to-- suppose you have a valid solution right now. Then someone drags on a vertex. When they drag on a vertex, let's say they drag it along a straight line. That's a linear motion of a vertex. And that will start violating constraints. If you go in that direction, probably not very good for all these constraints. In Freefrom Origami, you have-- the edge lengths should all stay the same. If you're in simulation mode.

So as you drag crazy, you're invalid. So the first thing you do is project. And this is, I call, an oiler step. You project that direction to be a direction that is perpendicular to all of your constraints, which means that it preserves all the constraints to the first order. And that's, I think, this first red step. Sorry. In general, these green steps would be if you just preserve things to the first order.

But if you keep following motions that are kind of correct-- they're correct to the first order-- you'll eventually drift away from correctness. And so you have to correct with the second derivative-- and that's these yellow steps-- to try to get back to a solution.

So as you're dragging, first, you correct to be correct to the first order. You make a step in that direction. Then you do the sequence of second order steps to get closer and closer to where things are actually correct. If that made sense, great. If not, you should take a course on numerical methods in computer science. A little beyond what we can do here. And so I'm just going to leave it at that. Cool.

Couple other questions about things Tomohiro said. So he said, it seems you don't need to worry about NP completeness of flat foldability. That's actually something

we'll be covering in the next lecture. So if you don't know what that means yet, don't worry. We'll be talking about it.

But it means, at the high level, it says it's competitionally intractable to make things fold flat. And yet, he's solving it. Why is that OK? There's a couple things going on.

In some sense here, we don't care about true flat foldabilities. Sometimes, he'd like to fold all the way to the flat state for compactness, and so on. That would be nice. But in particular, he just wants local flat foldability. He knows that if you have Kawasaki's condition, then you guarantee a rigid motion to fold for a little bit of time, and you can prove that.

And so if you're just trying to get rigidly foldable things, it's enough to have local flat foldability, which we do know how to solve in linear time. And that's the Kawasaki condition, and that's what he's solving. And so, essentially, whatever he makes will fold for at least a little bit of time. And if he's lucky, it'll fold all the way to flat. Sometimes not. Sometimes might get collision in between. So you always get something that folds. And then if it doesn't fall the way, you can try tweaking it until it does.

So that's the high level version. But you can, in some sense, sidestep NP completeness here. I think there's still some interesting open problems. In this setting, it seems like, say, Freeform Origami Designer. It seems like you really-- yeah. I have to leave it at that. I don't know exactly how to formulate the open problem here. But I think there are interesting questions about proving NP completeness doesn't matter as much here.

OK, another cool question. This is getting a bit higher level. This is rather tedious to fold by hand, as you've now learned, especially if you're going to make something like a bunny. Can we make a machine to do this? And so I wanted to show you a couple examples of machines for folding that have sidestepped the printing by hand.

This is an origami robot made at CMU by Devin Balkcom. He was a Ph.D. student at

the time. And he's taking a piece of paper. It's a robot. It's open loop. It has no feedback, has no censors, or anything. It is preprogrammed like an assembly machine to fold. Essentially, it can do simple folds.

So it's got a little suction guy, to move things around, crease. Eventually it accumulates error if it does a ton of steps, so you'd need a closed-loop system with a camera or something to get that. But it actually does a pretty decent job. This is real time. In this case, I think it's finished. One more fold. Crunch.

It's pretty impressive what it can do it. But it can really only do simple folds. It's going to have an issue if things really unfold it a lot. It might accidentally hit something. And this should be a samurai hat. Tweaking it a little bit by hand. Wow, it looks like a tetrahedron.

OK, so that was one example. Here's a more modern example. This was done at Harvard just last year. And this is a process involving laser-cutting individual components, aligning them with these tabs. Sorry, these pins. Assembling them together to make hinges.

So they use laser cutting, and to get two-dimensional surfaces, they use folding to make 3D shapes. Kind of like pop-up cards. This is what a typical hinge looks like. They've got all the different materials here to attach different parts. And these piezoelectric folding actuators. This is their overall design. They're trying to make a bee robotic bee. And this is what the final created thing looks like. It's mostly carbon fiber. And then these are the piezoelectric actuators.

So this is the thing they want to make. They build a scaffold around it that causes the whole thing to fold into its desired 3D shape. So they're taking flat parts. And they want to do things like take this flat part and raise it.

So what do they do? They add two hinges to make this part move along this straight up and down motion. And then each of-- that's just a scaffold. Each of the gray parts they actually want to build. They add the appropriate hinges to cause it to fold in exactly the way they like.

So here, for example, the wing is staying vertical. This part-- it keeps moving around on me-- is turning 90 degrees. You do that with all the parts. You get them all to fold like that.

Here's a prototype in real life. And then here's the final version. This is actually in real time, so it folds really fast. Zoom. And then you've got your assembled thing. One more. And then they add this particular metal that fuses the hinges together, so that they will no longer unfold.

So that's what it looks like locked. It's all done in an automatic process. And then you laser cut all of the scaffold away, and you've got your finish thing. A sense of scale, this is super, super tiny. It's tedious to fold these by hand. And in this way, they can mass produce them.

Here's what it looks like when you connect a battery. Either it will fold at 1 Hertz or at 30 Hertz, which you can barely see, because it's a 30 Hertz video. So you get your robotic bee. It's not yet controllable. It doesn't have a battery attached, but it's extremely lightweight, and very powerful.

This is a 3D printed prototype they made first. And you can use it to mass produce your objects. Essentially, automatic procedure. And it's all by layering up flat layers, and then getting it to fold into 3D things. And so you could imagine something like this to execute some complicated foldings, although that's future work. This is, in some sense, a fairly simple thing to build. And we're working on making more complicated things. So that was some robotic folding for you.

Next question is, any cool open problems here? So I have two related to rigid origami. One of them is, if I give you a crease pattern, tell me whether it is rigidly foldable at least a little bit or to the first order or something.

So I'll just give you something, like this will fold rigidly. I want to say yes or no, does this fold? Seems like a pretty natural question. And indeed, if all the vertices are degree four like this, only four four edges coming together, we can solve it efficiently. But given a more complicated general pattern, characterize when that is

possible. We don't have a good algorithm for that. I don't know if there is one.

The more general question is-- that's kind of an analysis question. The design problem is, I want to design cool rigid origami. And we've seen bunches of examples of rigid origami. Here's a new one I wanted to show. The Hexa Pot.

I believe this is rigid origami, as a kick starter on this. And here is one of them. It folds nice and flat. And it has this 3D state, where you can boil water on your camping stove. And they have a video of cooking noodles. It cooks noodles. It cooks pasta. It cooks sausages. Anything you could imagine, you can cook in here, as long as it fits in this space. That's waterproof, obviously.

We saw the telescope lens. We saw this origami stent. How are these designed? Inspiration. Some human had a cool idea, tried it out, proved that it actually folded rigidly. Great. But can we come up with algorithms to design things like this? Could you close the door?

Here's another just one-off example. You may have seen these. These are called shopping bags, and they're usually paper shopping bags. They're usually folded along this crease pattern. It turns out that's not possible if all the panels are rigid. This thing cannot fold at all. It's rigid, if the panels are made of steel.

And it's actually fairly easy to see that, if you look at this corner, these are four 90-degree angles coming together. And if you look at four 90-degree angles, two straight lines, like in a map, you could fold one of them. But only when you get all the way to 180 can you fold the other way. So right now, this guy is folded 90 degrees. This can't fold at all, which means this fold angle is zero. And we know from Tomohiro's lecture that a degree four vertex has one degree of freedom. So this if this is zero, they all have to be zero. And so the whole thing is rigid.

Of course, if you add extra creases, this is done with that Devin Balkcom and Marty. So the same robotic folding guy. Here's a visual proof of what goes wrong. You end up with a tear here. You can fold everything except one of those guys.

If you add extra creases, you can kind of roll the lip of the bag down, and repeat

that until it's really short. And then once it's below this height of one to two, you can just crush it like a garment box. And so you can do that. You can actually fold this thing flat, and you can undo it and unfold it.

An interesting open question is, these paper bags are manufactured in their flat state. If I give you a flat paper bag, can you open it by adding creases? I don't think we know the answer to that. But we conjecture the answer is yes.

There are a bunch of different designs out there. This is done with-- it's hard to read. But this is with [INAUDIBLE] in particular, who did the origami stent. It's kind of a twisting box. Works up to a cubicle box. And he just had a paper with Woo this year on a more practical folding.

So when we roll the lip, we get a lot of layers of material. This one works for fairly a tall bag. I forget exactly heart how tall. Maybe three to one. And it has a fairly small number of layers. They even built one out of sheet metal to prove this is a practical way to make rigid shopping bags.

And the last question here is, could you make one crease pattern that folds into two different shapes? Could you make an Origamizer that at one point will make one shape? And then you super-impose another crease pattern, ideally sharing lots of creases, to make a different shape? And the answer is, watch the next lecture. Yes, we will be talking about universal hinge patterns, where you take a different subset of the creases. You can fold anything, provided it's made up of little cubes. And that's one answer to that question. Any other questions? Yes.

AUDIENCE: Erik, going back to the rigid foldability, you do understand rigid foldability has a single vertex, right? It's just a global [INAUDIBLE].

PROFESSOR: Right. Rigid foldability of the single vertex is easy. Almost anything is rigidly foldable. But yeah, its general crease pattern [INAUDIBLE].

AUDIENCE: So it's very similar to the flat foldability [INAUDIBLE].

PROFESSOR: Yeah, it's like flat foldability, except for flat foldability, we know that testing a single

vertex is easy. Testing a whole crease pattern is NP hard. What'd we'd like to prove is either NP hardness, or get an algorithm for rigid foldability.

AUDIENCE: There's no such result [INAUDIBLE].

PROFESSOR: Right, there's no such result for rigid foldability yet. Other questions? All right. Enjoy folding.