PROFESSOR: All right. Welcome back to 6.849. Today we are going over TreeMaker, folding polyhedra, folding checkerboards, and Origamizer. Lots of stuff. And can't fit everything in today, but we'll be doing more Treemaker in some sense from a more practical perspective in the very next lecture video and next class, with a guest lecture by Jason Ku.

So today we have a bunch of questions. We'll start with TreeMaker. So the first question is, wow, you can make all these crazy things with the tree method. Are they really uniaxial? And the answer is yes. And I think it's probably easiest to just go through some examples.

These are all from *Origami Design Secrets*, this book by Robert Lang, which is on the recommended reading list. And I think there's even an electronic copy available through MIT Libraries. So you can read it online or there maybe one in the library. There's two editions. These figures are from the first edition.

So here is a-- the goal is to design "Scorpion varileg." These are all Robert Lang designs. You come up with this stick figure, your tree. TreeMaker makes this crease pattern. You can also see the rivers here and the disks. And then you get this base. And then it folds into that scorpion.

So while the scorpion doesn't look like it has a single axis-- scorpion, maybe. There's kind of an axis down the center. But it's not just this axis. It branches, then it branches again. You can make a tree and it's still uniaxial. So uniaxial refers to this axis in the base. All the flaps are attached to that one spot.

So in this picture, it's vertical. Usually in the-- when we draw all the figures, we think of it as the horizontal plane, or horizontal line. So we imagine this is the floor. And then we have flaps attached to the floor. Something like this.

So in that sense, when you flatten this base, everything lies along this axis. Even though, in terms of branching structure, you could have lots of branches. So that's to clarify what uniaxial means. Maybe not the best term, but I have a few more examples here.

This hat includes a color reversal. Some crazy stick figure. And you get your pattern. Here's a somewhat simpler want. It makes a more three dimensional horse. This Alamo Stallion. So you can go through, and for each of these, I've just reverse engineered from this crease pattern what the stick figure must be. And you're going to do this on PSET2, which will be released in a few minutes.

And so this uniaxial's to contrast from something, the kind of most famous non-uniaxial or biaxial origami bases, with Montroll's Dog Base, which looks like this. So it's very easy to turn this into all sorts of dogs. This is a wiener dog. But you have kind of one axis here and one axis here. Which is different. It's not the same kind of branching tree structure.

Of course, you can make a dog using uniaxial method as well, but it's kind of a fun counter example of the things you might do. So that's uniaxial.

Next question is, do people use TreeMaker and/or Origamizer in practice? And the short answer is no. I would say most origamists don't use software to design things. But they use a lot of the ideas in their own designs.

So in particular, the tree method of origami design, which TreeMaker is implementing, almost every advanced origami designer uses. Not everyone, but most of the complicated-- most complex origami that you see uses the tree method at least to get started. Origamizer is much newer, so it's not used as much. Most of the cool designs-- I'll show you a bunch of both-- are by Tomohiro Tachi, who invented Origamizer. But I think it's still evolving. And hopefully, these more advanced techniques will catch on with time.

And by now, you've seen lots of examples of the tree method. I'll show a few more from Robert Lang's website. This is a local delicacy. And these are all on langorigami.com, on his website. And we'll see even more examples in the next class. And Jason Ku is going to give an overview of various artists.

And in some cases, they publish crease patterns. Not always. This is not quite the tree method. So this example was standard tree method. You've got disks and rivers, which aren't drawn here. You get a crease pattern. This could be done directly from TreeMaker. I don't know for sure whether it was.

This pattern is not done with TreeMaker because it's not regular disks. This is the box pleating version of the tree method, which there's another question about. So we'll talk about that more. But you see all the creases here are horizontal, vertical, or diagonal, 45 degrees. That makes it a lot easier to fold, a lot easier to find reference points. You see it falls nicely on a grid. But still, you get an arbitrary tree structure.

I think I have another example. This one also-- well, this has 22 and 1/2 degree folds. It's a little bit more general. Also very non-stick figure-like models, particularly. It's obviously called the "Fiddler Crab." No, sorry. I have the wrong title here. I will fix that.

I forget this guy's name, but if anyone remembers. He wrote about the division between scientists and people who know literature as two different types of intellectuals that know almost nothing about each other. At least in the early 1900s when he was writing.

And one more example. This is based on a 60 degree grid. So all the creases, I think, lie at 60 degrees to the axis, or horizontal. There might be some 30 degrees. So I'll talk a little bit about this alternate version of the tree method that doesn't just use arbitrary circles, but tries to stay on these nice grids.

Here's just one more example. Jason Ku is the designer of this model called "Pan." and he'll be giving the guest lecture. So this is just one example. But tons and tons of advanced origami artists use tree method of origami design. And you get some pretty cool crease patterns from it.

But I think most people do it by hand, usually on a computer using some drawing programs that can compute intersections and do things with high accuracy. And then they draw these pictures, then they fold them. And then you get your base and then you shape it into the model.

So next I want to show you some practical examples of So there's a bunch on Tomohiro Tachi's Flickr site. This is one of the earliest. This is just making a negative curvature surface that curves in both ways like a saddle, called a hyperbolic paraboloid. We'll see more of those in the future. But this is folded from one square of paper. And there's tabs on the backside that you don't see here.

Here he's making a kind of 3D bell curve. Here's a computer mouse. Apparently the scroll wheel works fine. [LAUGHS] Doesn't support USB, though. Only Bluetooth. We've got a nice mask. These are all-- you're given a 3D model, and you fold exactly that 3D model.

This is a tetrapod. So here it sort of shows Tomohiro's background in architecture. If you haven't seen tetrapods, they're used to hold back the ocean and things from-- or hold from erosion and things like that. So usually made out of concrete. But here you can make them out of paper.

Here's a flat design. So here the mesh, the triangles were all in a plane. And of course, you still get the tabs on the backside. But you make this very exact leaf. You can get the triangulation edges exactly where you want them to express these veins of the leaf.

Here's what it looks like to fold one of these in practice. It's not quite a square of paper, but it could be a square. And this is actually folded at CSAIL. You might recognize some of the furniture.

And so while Tomohiro's folding, he uses various devices-- paper clips and clips and so on-- to hold it in shape. Because until it's completely collapsed, it kind of want to open back up. So it's kind of like having a hundred hands at once. But when he's done, he'll take all of them off, and you'll get-- so this, I think, was about 10 hours and you get your bunny. Easy.

Or this is the one we did a little bit later, just last year. This is a laser cut sheet of steel. And now, here we've cut out bigger holes so there aren't too many accumulation of layers. But essentially the same design. Somewhat coarser mesh of the bunny. And you have to wear gloves, otherwise you'll cut yourself.

So we're perforating the metal at the creases. And then also takes about eight hours. A lot harder to fold steel. And in the end, you get your bunny.

So in principle, out of any sheet of material, you can make any 3D shape you want. That's the exciting thing. So I think Origamizer is really powerful. Obviously, it's hard to fold. You need to be pretty advanced. But there's a lot of potential for designing very non-stick figure-like models.

So that was TreeMaker and Origamizer in practice. So next question is about this box pleating, which is the horizontal, vertical, and 45 degree creases. And TreeMaker, is there some theory for this? And indeed there is.

And we started working on it, me, Marty, and Rob Lang, I think during his first visit here, which was probably 2004 or something. A long time ago. And the best writeup of it currently is in *Origami Design Secrets*, especially Second Edition. So

It has a chapter on basic tree theory, a chapter on box pleating in general. This is sort of classical box pleating. But then there's a chapter on mixing the two. Uniaxial box pleating and polygon packing. These kind of go together.

So if you're interested in this stuff, you want to design something the way that the experts do, check out *Origami Design Secrets*. Or go to an origami convention where Rob was talking about this stuff.

We are working on this giant manuscript. It's currently called the *Mathematics of Origami Design*, which is, in particular, trying to prove the tree method always works. And we'll generalize to things like this. But it's not finished, so we don't have a complete proof yet that it all works. Everything seems fine, but it's tedious to write it all down. So still working on it.

But I thought I'd show you an example of box pleating uniaxial origami design from

Origami Design Secrets. So this is sort of typical tree method of origami design. If you're not necessarily using TreeMaker, but you're doing it by hand, you think about, OK. Suppose I want to make this kind of insect, this stick figure.

Maybe I realize, oh, it's kind of centrally symmetric, so I'd like to make the left half of the paper same as the right half. You can also express that in the TreeMaker software. And then you start thinking about where these leaves correspond to disks. Then you've got to have the corresponding rivers in between them.

So you might start with this kind of layout, and then you try to blow it up until things can't be expanded anymore and there's lots of touching. So that would be the usual approach. Then from that, you could apply the tree method, get the crease pattern.

With the box pleated version, essentially, instead of disks, you have squares. And instead of rivers, you have these orthogonal channels of constant width. And so I'll just wave my hands and say that happens.

Now, you get these weird things-- you get these gaps. You tend to get more gaps in this way. But in this case, instead of just having a square, you can extend it out to be a rectangle. So these guys are actually going to fill in these holes like this.

And then from this, you can start constructing a crease pattern. So you start with these ridge creases. And this is something like a generalization. It's actually something like a straight skeleton, which we'll be covering in a couple of lectures for a different purpose, where it was originally developed for origami purposes. But these are just lots to bisectors. And you'll have to see the general version later.

And then you start putting in the hinge creases. These are these green lines. And they're perpendicular to these. These blue creases are going to be the floor. And so on, you fill it in. Eventually you get your complete crease pattern.

So in a nutshell, that's how it works. You tend to get tabs that are a constant width like this. Constant height, I guess. And so, it's a general technique. It's a little bit much to explain here, but read the book if you want to see it.

And I would say it's probably some of the most common. There's a lot of intuitive ways to do it, and then a lot of the details are worked out in that book. But it's quite common to design bases in this way, because it's just so much easier to fold them. You don't have to construct really weird angles.

A lot cleaner. But a little bit less efficient, because instead of using a disk, which is the minimum amount of paper you need to make a flap, you're using a square. So you're wasting those little corners. But not that much more inefficient. Cool, so that was that method.

Next question I have is about the triangulation algorithm, which I didn't even cover in lecture. So this was you set up-- you have your piece of paper. And you assign each leaf in this tree to some point in the piece of paper. And you satisfy the active path condition so each of these distances measured on the paper should be greater than or equal to the distance measured on the tree, which is the floor of the base.

Great. And so if you happen to have some equalities, you draw in active paths. But what if you don't have any? Or you don't have enough? The tree method in its original form only works when the active paths decompose the piece of paper into convex polygons. This is what you'd like to have happen. Each of the faces here is a convex polygon.

If that happens, great. You can use the universal molecule and you get your folding. If it doesn't happen, you have to modify your tree a little bit. There are a couple ways to do this in the TreeMaker.

You can modify these edge lengths and try to modify them as little as possible so that things touch. But the easy way to prove that something is possible-- because remember, if we add a little bit to the base, we get an extra flap. That doesn't really hurt us. At the end, we can hide the flap. Just fold it against other flaps and just pretend it wasn't there.

So as long as we can add flaps in order to make the active paths decompose into convex polygons, we're happy. And in fact, what we prove is that you can keep adding flaps until you get into triangles. And triangles are always convex. And so they make us happy.

So we will end up adding a bunch of flaps in our tree in order to triangulate with active paths. And then in each of these, we just fill in a rabbit ear molecule. So that's the goal. Now let me tell you how the triangulation actually works.

It was originally described by Lang. It's also in the textbook for this class. I'll try to give an abbreviated version here. It's a little bit technical, but here's the idea. So suppose-- all right. So we've assigned some leaves. And suppose we have some region that's not convex. It's not a triangle. Has more than three sides.

Now, some of these edges may come from active paths, and some of them may be from the boundary of the paper. So maybe the paper is here. So this edge is not active. It's just the boundary of this region because it's the boundary of the paper. So there are two types of edges.

But there should be, I guess, at least one active edge. And there's at least four edges total. So what I'm going to do is look at any of the active edges, and I'm going to imagine-- OK. So that active edge, this is in the piece of paper. But now I'm going to think about it in the tree.

So the tree looks like something. We don't really know what. Any active path here corresponds to an active path in the tree of exactly the same length. So maybe it's from this leaf to this leaf. So it corresponds to-- use a color. This guy corresponds to this path. And the sum of those lengths should be equal to that length. It isn't, obviously, but there's a scale factor in between here. Lambda.

OK. So here's what I'm going to do. I'm going to modify the tree by adding a new leaf somewhere off of this path. OK, where? For convenience, let's assume that this path length here is 1. Then I'm going to measure out some distance x here, which will leave a different distance, 1 minus x, here.

And then I'm going to make this length, I guess, L. And I'm going to call this leaf L. Capital L. OK. This is a modification to a tree. I can do it. I can do it for any value of x between 0 and 1. And i can do it for any value of L greater or equal to 0.

So what I'd like to do is design the tree so that this point ends up in an interesting place on the piece of paper. In fact, I claim that no matter where I draw L here, capital L, as my desired place. Now, what I'd really like is for-- I should give these guys names. This is called U, or say, in the notes, I call it V and W.

So here we have W and V. What I would like is for these two paths to also be active. Meaning the lengths here match the lengths in the tree. I claim that no matter where I put L in the plane, anywhere in the plane, I can choose x and choose L so that these two lengths are exactly correct. Do you believe me?

Not really. How many people think this is obvious? Good. A couple maybe. Let me give you a quick sketch. It's not that interesting, so I just want to do it very briefly. If you let x be a free parameter, but fix l, little l, then what this corresponds to is, in fact, an ellipse with V and W as foci of the ellipse. Because you have to hold the sum of these lengths fixed if you fix-- if you let x vary, this length plus this length is always going to be the same if you fix l.

And so this is called the major axis, the sum of these two lengths is going to be like 1 plus 2l. And so if I have a point, basically, there's some ellipse of the appropriate size that passes through L. That will let me choose little l. And then as I vary x, I walk around the ellipse. So I just choose the appropriate value of x that gets me the desired point on the ellipse. That's it.

The point is, the set of all ellipses with these two foci spans the entire plane. So wherever you want to put L, you can make those two paths active. OK, this is good, because it makes a little triangle. But of course, if I just add an arbitrary triangle, not very interesting.

The good thing is I'm free to put L wherever I want. I'd really like to put L say here and draw active paths like that. Because then I'm kind of decomposing my polygon into triangles. That's not always possible.

So let's just go through the cases. In all cases, we're going to simplify a polygon, make it have fewer sides. And that's on the next page. So the claim is-- well, the L. There's sort of three cases. Going to reorganize this a little bit.

One thing that would be nice is I can place L-- maybe I'll draw another version of this. So I've got UV. I'll draw the active paths in red. So we know UV is active. We've got L. If I could somehow place L so that it's active with two other points, two other leaves. This is V and W. This is U and this is T.

If I can do this, and the region looks something like that, then I'm happy. I'm going to put L there and add in these four edges. And essentially, if you look at any one of these regions that remains, there's a triangle here. That's definitely OK. These regions, each of them will have fewer edges than the original region.

That's pretty easy to prove, because you have-- Because you're connecting to these four vertices, this region won't have this vertex or this vertex. So it has one added vertex and two removed vertices at least, W and T. So this one will be smaller. And this is symmetric for all of them.

OK. So this would be a good case. So what I'm going to do is try to move L around. I'm going to start very close to VW and just start moving off the edge. Initially, nothing else will be active because it's basically right on top of VW. But as I move around, I might get another active path.

OK. Maybe this is Case 1A. Before we get there, Case 1 is you get LU active. Suppose you just to get one additional active path. The other case is, nothing else becomes active. So here's V, here's W. Here's L. Here's U. So this is active, active, active, active, active.

OK, what I'm going to do in this situation is, if I want to keep LU active, I can actually move L on a circle. As long as L stays on this circle, LU remains active. OK? So just move it along the circle.

Now, there are two possibilities. It could be it becomes active with something else. Then I'm done. Or it could be it doesn't, which means it will hit the boundary. So it will hit a boundary point.

So Case 1A-- so this is Case 1. You can make something active. When you move along the circle, either you make another thing active, or you won't and you hit the boundary. Case 1B is you hit the boundary.

And then I claim you're also happy. So in that situation-- draw it one more time-- we've got L on the boundary. And we've got some path here, here, here, here. These are all active. So here's L. OK. I claim again, each of these regions has fewer vertices than before.

Should be pretty obvious. This one is emitting W and this was U. Yeah. It's emitting W and it's emitting whatever's down here, if anything. Yeah, I think it's pretty obvious.

You've got to check all the cases. There's a few details here. But it should be each of these regions has strictly fewer vertices than it did before. And so you're making progress. If You started with all faces having some number of edges, each time you do one of these steps, you decrease it.

OK, the last case, Case 2, is you can't make anything active. This can happen, for example, if you're in the corner of the piece of paper and you have an active path. I guess that's a little less exciting. Maybe your piece of paper is this shape. Be a little bit weirder. So this will work for any convex piece of paper.

So it's not quite a triangle. And you're moving L around here and you just can't get another active path. You have no vertices around. This is W, this is U-- sorry, this is V. And in this case, L can go anywhere here. It's free.

In that case, I'm going to put L on one of the vertices of the paper. So I get this active and this active. And then I've decomposed that region into pieces by, this is what we call a diagonal. We're adding a vertex to vertex edge in this polygon. And that always decreases the number of sides in each of the subregions. So if I just keep doing this, taking any region of size larger than three, I will eventually reduce them all to have size three. So they'll all be triangles.

So that's the triangulation algorithm in a nutshell. You can look at the textbook if you want to see it more detail, but it's pretty simple. This proves that something is possible. I don't think TreeMaker actually implements this algorithm specifically, but you can kind of do it by hand.

So I have a little example here. This is TreeMaker. I drew this really weird tree. We call it a caterpillar tree because you would actually use it to make caterpillars. And if you just plug this into TreeMaker with all the lengths unit, it will give you this error message. I couldn't construct all the polygons because things weren't tight enough.

And if you look carefully-- it's a little hard to see these colors-- but the light green edges, those are the active paths. And here-- so this was a triangle and it was happy. If you look at this green polygon-- maybe actually I'll draw with the tablet. So this thing is an active-- that's a region bounded by active paths. And probably the top edge is not an active path. That's just a boundary.

But it's not convex. And so you're unhappy. And so I just kind of eyeball this and say, OK. Well, probably I should add another leaf here that would maybe add a disk that will probably fill that in. So you don't have to be super-precise. Here we had the very carefully place the disk to make things touch.

But because TreeMaker's always just trying to blow things up and make them touch, it's quite a bit easier in practice. You just say, OK. I will add an extra leaf here, like that, to my tree. And then I hit Optimize, and boom, it works.

In this case, I was lucky. In general, I might get a nonconvex region. I'd guess where to add another leaf, and it works. You can adjust. Did that leaf have to be that bit, or could I get away with a smaller one? But eventually, you will get a base. And if you want, you could carefully monitor this proof and actually always succeed. But in practice, it's usually not that hard.

Questions about that? So that was the triangulation method in a nutshell. The next question is about the universal molecule. And we might spend some more time on this next class. But I thought we could actually look at this example, because it's got a bunch of different universal molecules.

So I will continue to draw. So let me pick, let's say, this universal molecule here. It's a quadrilateral. All right. So those are four active paths in this case. And it corresponds to some tree. So I don't have handy here, but it's going to be a tree with four edges. With four leaves, sorry.

Tree with four leaves is going to look something like this. This is a piece of the bigger tree, which you recall looked something like this. And we added one somewhere. But this particular quadrilateral is doing some particular subtree on four leaves. In particular, there are four leaves here, correspond to four leaves here. I don't know which ones. I don't really need to know. TreeMaker keeps track of it for me.

And I happen to know that these four paths are active. Meaning the lengths in the plane here match exactly the lengths as measured along the tree. So what I do to make this work-- this is basically the floor of the molecule. Those four paths will all lie on the ground level.

What I'm trying to do is slice higher and higher in the base to see what happens when I slice. And if you think about it, as you move up here, in the plane this corresponds to moving parallel to these edges. So I'm shrinking this polygon by parallel offsets.

So let me draw one parallel offset. Approximately parallel. OK, after I shrink a little bit, might look like that. An interesting moment in this case is here. Let me draw this carefully. It should go-- this should go here. Ooh. Oops. I'm going to start over.

This is actually already drawn for us. It's this guy. OK, this is a parallel offset. And in this case, something interesting happens, which is this path. So it turns out this middle edge, in this case, becomes active. So what that means is, originally, if you looked at these two points, these two leaves, they had the wrong length. OK?

So that's going to correspond to something like these two leaves. They're opposite corners of the quad. And you measure the length along the tree, you get something. And the claim was, in the plane, it was too big.

Now, as you do this parallel offset, as you shrink the polygon, at some point it will be just the right length. This could happen. Doesn't have to happen. If it happens, you have to stop. Because if you kept going, it would become too short. And we know every length here must be greater than or equal to length over here.

So when it becomes equal, we have to stop, and we have to make this a crease. Making it a crease basically says, ah, this is exactly the right length. I have to split here. And where'd my chalk go? And I have to sort of grow now two different flaps from that point. The crease sort of makes it horizontal.

And then we end up shrinking in two different parts. We shrink in this triangle, which gives us a rabbit ear. And we shrink up there in that triangle. Gives us another rabbit ear. So it will end up giving these angular bisectors.

In general, as you do the shrinking, you watch where the vertices go. Those are your ridge creases. So this one went along an angular bisector. That one did as well. But once we do this split operation, because of this newly active path, which we also called a gusset in the universal molecule, then these vertices actually split and go in two directions. One for this thing angular bisector and one for that angular bisector.

So in general, for a universal molecule, there, are two things that can happen. You can be shrinking your polygon and suddenly discover there's a newly active path. And then you have to divide it into two polygons and start shrinking those separately. Or vertices can disappear.

So it could be you're just shrinking, shrinking, shrinking merrily along the way. And then suddenly, these two vertices collide with each other. So you get to ridge creases here. In this case, you just treat those two vertices now as one. So you'll start shrinking like this. And so these vertices will now start going that way.

So in general, as you're shrinking, these are the only two types of events that can happen. Either two vertices merge, or you get a new diagonal here that becomes active. When that happens, you just divide. Now, this will work for any convex polygon and you get your molecule.

It's hard to just draw a picture of this without using a computer tool or really keeping track of what you're doing. Because how do you tell when something becomes active? You have to look at your tree. You have to measure lengths. It's tricky to do without a computer tool. And that's why TreeMaker was made. But it can be done. Probably even easier with physical paper if you know exactly what lengths you're trying to match.

So that was a quick overview of how universal molecule works in more detail. Next, I want to talk about a few different open problems. So one of them was-- these are called gift wrapping problems.

So we have things like given a square, a unit square, let's say, what's the largest regular tetrahedron you can wrap? That's still open. All these problems are still open except for the squared to a cube which talked about in lecture. And I realize equilateral triangle to tetrahedron. That's also really easy.

If you can do it without any paper wasted, just clearly optimal. But it would be very cool to study these. I think these would make a good project or open problem for the open problem session.

Folding a given rectangle with given aspect ratio into a cube. Also open. Pretty much any version you can think of is open except the ones that we've already seen.

Also wanted to mention for checkerboards, a fun problem, kind of in this spirit, is, what would be the best way to fold a two by two checkerboard? Even for a two by two, we have no idea how to argue any kind of lower bound about how bad you must do. That you can't just take a square and fold a two by two checkerboard of the same size. Surely you can't do that. But we don't know how to prove that. So it would be a nice target.

We obviously have upper bounds. We have constructions that make two by two checkerboards. I think from a 3 by 3 grid, you can make a two by two checkerboard. But is that optimal? We have no idea.

So a couple questions about the checkerboard folding. So we had this picture of how you would take a square root of paper-- you start with a square paper. You fold into this shape with these long slits. And then these guys fold over. And then there are also tab sticking up here, which are not drawn. And they fall over and give you the color reversal.

Question was, how do you actually build this thing? This does not look uniaxial. And indeed, we do not use uniaxial techniques for this. We use a separate set of gadgets which are kind of based on pleating. And this will actually be very similar to something we see in a couple of lectures when we do box pleating to make cubes and stuff.

But you just compose these gadgets. So this is kind of a general tab gadget. You collapse this crease pattern, and do a couple more folds. And you end up with this tab sticking out here, and you can flip it up or down.

And you see that it has these pleats running off to the side. This mountain valley, mountain valley, and then valley mountain, valley mountain. And so those have to go all the way through the paper. And they come into this construction, which basically lets you make a big slit in the paper. Or you can also turn a corner.

And so you just prove that these gadgets compose. I mean, it's pretty that they compose. You just have to analyze how much of the paper you're using. And it turns out to be really good. That a short version. If you want to try it out, fold some gadgets.

Fun project idea from you guys is, given an image, sample at low resolution, make it black and white. And then come up with the crease pattern that would fold, by this checkerboard technique, into exactly that two color pixel pattern. You could make all sorts of useful things like Space Invaders and stuff like that.

You'd never want to fold them, I think-- although maybe with low enough resolution you could fold them. But I think they'd be cool just as crease patterns by themselves. You input a different pattern-- I'd like web applet. You change the pixel pattern. And boom, it gives you the crease pattern. I think this would be a fun project if you're into implementing things. It's essentially the algorithm we already have, but it needs to be just written out in detail and coded up.

OK. Here's what it looks like to fold an eight by eight checkerboard with this method. These are all by Robert Lang. So a lot of precreasing along this huge grid. I thinks it's roughly 48 by 48. And then some tape to hold things shut, mostly for photographing. And to collapse, now we're down by a factor of two or so. Now you can see the slits in one direction.

I think the particular method being used here only has slits in one direction. Oh, no. Here we've got slits in the vertical direction as well. You could see the tabs sticking out here. They're going to fall down. There's some more of the tabs. Start folding over, and boom, you've got your eight by eight checkerboard.

Robert Lang says, wow. That was not one of the easier things I've done. And this is folded from a 48 by 42 checkerboard. In principle, this method can go down to 36 plus epsilon by 36 plus epsilon, but it gets messier crease pattern-wise.

And 36 would be better than the best known eight by eight board if you want seamless. Notice these are seamless squares. No crease lines through them. This one's not necessarily-- this is not better than known techniques, but it follows our algorithm. And that's what the crease pattern looks like. Cool.

Next we go to Origamizer. So as I mentioned in lecture, there's two versions of Origamizer. There's the version that's implemented in the software, and there's a version that we're proving is always correct. These are different because the version in the software doesn't always work.

And it remains that way because the software version is more practical. And also, the theoretical version is still a moving target. We change it every few weeks to fix part of the proof. It's almost done, but it's been almost done for a couple of years. So we're still working on it. Tomohiro was just visiting a couple weeks ago, as I mentioned.

And we're closing in. I actually have with me the current draft of the paper. And it's not nearly as long as this one with the tree method, but it's still growing and working up, making sure all the details check out.

And the theoretical version is kind of complicated, so I thought, in particular, given this question, we'll talk about the software version because it's actually a lot simpler. Only catch is, it doesn't always work. And it's described in this paper if you want to read it, by Tomohiro, about Origamizer 2010. And I have a few figures from that paper.

Well, this is still from Tomohiro's Flickr. So we had this example where you wanted to fold this hyperbolic paraboloid. And here's what the crease pattern looks like. It's actually pretty simple. You've got these white polygons, which are polygons from the surface. That's what you need to fold.

And the goal is to lay them out on the piece of paper-- this square is the piece of paper-- so that I can-- for example, I need to bring this polygon to touch this polygon. This edge has to touch this edge. Wouldn't it be great if I could just fold the bisector of those two edges, and this would come right onto here?

Sometimes that happens, but for example, if this polygon is way down here, that won't happen. It will fold over and they won't align. There's two issues. First you have to get the angles to match. And also there's this vertical shifting.

If you place the polygons in the plane in a good way, this will work. You always get alignment. And what Origamizer implements is nonconvex optimization, a constraint projection, to make all of the edges work. And sometimes that's possible. It's actually possible fairly often. And that's when the Origamizer software works.

If it's not possible, you're screwed. This particular method won't work. But when it's possible, things are great. So essentially, you have to bring the edges together.

Then you also have to bring the vertices together. Like these four vertices come together to a point.

And so you get two kinds of gadgets, which are the vertex tucking molecule to bring vertices together, and edge tucking molecule to bring two edges together. Edge tucking molecule is a single crease. Trivial in this construction.

Vertex tucking molecule is complicated. And in general, what happens is for these points, we construct what's called a Voronoi diagram, which is essentially, for each of these points, if you grow a disk at equal speeds around all of them, and when the disks meet, you stop them. So then these two disks will meet along this perpendicular bisector. And they'll keep growing until they kind of all die out. And you'll get this tree structure, in general.

Those are your main creases. You follow that structure and you do some stuff. Essentially, what happens when you fold along this Voronoi diagram, you also have to add in these creases from the Voronoi diagram to the points. When you do that, you'll get a kind of mushrooming structure that has too much material.

If you set it up right, all the angles of paper that you have are larger than what you need in the folded state, which would look like this. And so what you do are lots of little pleats to reduce the angle. . If you've got a big angle of paper, you can do a mountain and a valley and make it a smaller angle. And that's what all these creases are doing. You've got these pleats to reduce the angle here, reduce the amount of material here and here in order to match the 3D structure.

So overall, what the algorithm does is first, given the surface, it constructs a suitable 3D structure of where to put all the extra tabs. Then it designs things so that the angles are bigger here than what they need to be there. This is, again, a constraint projection. Also constrains these edges to meet up perfectly.

And then it just applies all these crimps-- it's easier in a computer than to see it here-- to make all the angles correct to match the 3D model. Yeah, question?

AUDIENCE: So on the sheet metal one that you showed, do you just cut out the vertex shrinking

[INAUDIBLE]?

PROFESSOR: Right. So for the sheet metal bunny, we just cut out these polygons. Because otherwise, it's a mess and you get lots of layers. And the point was to make it of one sheet of material, but not necessarily a square. So if you have really thick material, recommend cutting those out.

And we were laser cutting to score the lines anyway, so why not cut out some holes as well. Yeah. So that way, you just have edge tucking molecules, which are really easy. Mountain, valley, mountain. And so it actually is pretty practical to fold these things out of generalized sheets with holes.

So the non-software version of Origamizer works the same way, except it does the full generality of constructing this tuck proxy, where the tabs should go. And that can get very messy in general. It's just tedious to explain. There are all these crazy spherical diagrams that you saw in lecture. But it's not that exciting.

The more interesting part is that the edge tucking molecules can't just be a single crease anymore. They have to, in general, kind of follow some path to get to where they need to go. But it's similar. You just do a bunch of different folds. You can sweep them one way or the other in order to navigate this edge to be where it needs to be.

So again you place the faces in the plane. In the mathematical version, you can place them anywhere you want. It doesn't matter. And then you have to shrink them until they're small enough that things work. Then you route these paths to get from each edge to each corresponding edge.

The edge tucking molecules are fairly straightforward. The vertex tucking molecules become even messier. We still use a Voronoi diagram in the end. It's just there's a lot more points. Here we're just using the corners of the triangles as the source points for growing stuff. In the general case, you have to add lots of points in the middle too.

But it turns out not to matter. You could throw in tons of points. You could always fold a Voronoi diagram. You get this mushroomy thing. And then you can just fold away the parts you don't need. You have to make them really small in order to make these tabs not too big, because if the tabs are too big, they collide with each other.

That already happened in the software version. Even in the software version, this might not work because this tab may be huge. But you can-- in this case, you just add a few more and make this pleat up and down, and you'll avoid collision.

So that was a quick, but a little bit more clear, overview of Origamizer and how it works. Any more questions? Cool. That's it for today.