PROFESSOR ERIK DEMAINE: All right, so this lecture we talked about fold and one cut, two methods and a little bit about polyhedron flattening, so most of the questions are about fold and one cut. I'll stick to that. First question is, is there an equal software for doing fold-and-cut now? And the answer is yes, there's some software. Maybe not the coolest possible yet, but it's getting there. There's two pieces of software. One is from Final Project after this lecture was given, 2010.

And another one is in SourceForge project called JOrigami, or J Origami. Both are written in Java, I believe. I have this one. This is the swan, which you've seen before. I have it here. We can try it out. It's not online yet, because it could use some improvements, but it's already pretty cool. So you can take something like the angelfish here, and if you like, it has an editor so you can move your polygon around.

And you can say, OK, please give me the straight skeleton, first straight skeleton only, and it will update on the fly, and gives you some nice intuition about how that works. I know that looks weird that it goes up that way, but it is correct, because it's bisecting this edge and some edge, this one I guess. They meet out here, and then you go that way. So you can play with that, and then you can add in the perpendiculars too. Takes a little bit longer, so the refresh may not be as immediate. But it works. It's really complicated behavior in there, some spiraling in, spiraling out, but it's fairly well-behaved.

Here's one of the simpler spiraling examples. This one is stable under perturbation more or less. You move the vertices all a little bit. They should continue spiraling. It looks like that one is a little bit degenerate. So we'll go around, and as you go get bigger and bigger pieces of paper, you'll get more and more creases. I should mention, you can add new vertices as well and draw polygon. You can delete edges. I'm holding all sorts of crazy modifier keys to make this happen, but it does work. The one other thing you can do is snap to a grid. This is probably hard to see, but there's a square grid underneath. If you hold down Alt, it snaps to a square grid.

I have one other example I've prepared. It has save and load, which is pretty cool. And this example is one of the dense instances. Although this one has rational multiples, so it doesn't actually go forever. Because I drew it on the grid so that could get all the horizontal and verticals. And you can see in particular it stops here, because at some point it gives up in reflecting perpendiculars, says that's enough. It won't draw anymore.

So it's fairly robust in that sense. Occasionally it's using some straight skeleton code they did not write, and it has some issues when you have really degenerate situations, which they tried to mitigate. But occasionally it's not perfect. But there are lots of possible follow on projects to this work, improving the user interface. Actually putting it on the web for people to play with, I think, would be super cool. Alternatively, could port it to JavaScript-- right now it's in Java-- and make it even more accessible run on iPhones and things like that.

Still, one of the questions I've posed in lecture was, can you make a nice interface that would let you force degeneracies, make like in-- I think the swan has instances of this, where you'd like more than three skeleton edges to come together at a point. Here they almost do. It would be nice to be able to say, and for it to snap to a position where many things come together. Because that, in general, reduces number of creases substantially.

You could also try to compute the folded state. That would be another interesting project based on what we're going to talk about in a little bit, folding the underlying structure of corridors. And then you can computer a crease pattern. And there you have various choices, and that lets you throw away some of these creases. Like this is much messier than the swan crease pattern that's on my web page, because you don't need all these folds. If you choose the right subset of folds you can save a lot of time. So, still lots of cool projects to do here. It'd be great to get this software online, but that's its current state.

Next question is, what about odd degree vertices? This is actually a pretty natural question. Even degree vertices seem nice because you can kind of-- well, it relates

to a page that I didn't really talk very much about in the lecture, but it was in the lecture notes, this idea of a side assignment. So in general, if you have something like a swan, there's the inside of the swan, the outside of the swan, and generally you have a bunch of regions.

And for each region you'd like to know is it above the cut line or below the cut line? If you imagine the cut line as horizontal. And in general, the side assignment would specify, do I want my region above or below? And you could do whatever you want. Now with even degree vertices, this is great. You could just alternate around and say, above, below, above, below. With odd degree vertices, you can't, so you're going to have two regions that are adjacent to each other which are both above or both below.

So what does that mean? It means it's a little bit hard to cut. And there's actually two models of cuts. There's scissor cuts-- which are in particular what the question poser had in mind, and probably what you might have had in mind in general-- where the cut you're going to make separates material from above and below the line. So this is the cut line here. And it'd be really nice if you had material on both sides, because that's usually how scissors work, they tear apart material.

An alternative is that you have a mathematical cut. And a mathematical cut can cut right along a crease line. So it could be you have two polygons, there's a fold here between them, and you can cut right along that line. Maybe I shouldn't draw scissors. Imagine a laser beam which can zap right along the line, so there's no material on the left side of the line, just material on the right, but yet it separates the two things on the right.

So we call that mathematical cut, because it is the natural definition mathematically. You're erasing a line. But practically it's a little hard to do. So scissor cuts are also nice. This is a more restrictive model. This is general model. So what I was talking about in lecture, implicitly use mathematical cuts. And that's when you could make anything with one cut. But, it would be nice to get scissor cuts when possible. And when possible is basically when you have even degree at every vertex.

And one fun example of that is checkerboard. This is actually an old magic trick. You have a piece of usually tissue paper, so it's really thin, pre colored as checkered squares, both sides matching. And you can fold this, because every vertex here has degree 4, so even degree, you can assign the black squares to be on one side or the blue squares to be one side, the white squares to be on the other side. And so you could make one cut and simultaneously cut out all the white squares and all the black squares which is kind of cool. This is old, decades old.

So in general, you can do something with scissor cuts. Or scissor cuts are going to be possible if and only if you can find a side assignment that sort of alternates between above and below the cut line. Meaning when you have two regions that are adjacent, you don't want them both to be above and below. You'd like one to be above, one to be below. And this is what's called a face 2-coloring in planar graphs. You want to color the faces, the regions of your graph, with two colors such that no two adjacent cells have the same color, no two adjacent faces have the same color, and that turns out to be equivalent to having all vertices of even degree.

So this is why that question was asking about odd degree, because indeed with scissor cuts, you can't do odd degree. There is a fun fact, though, that relates these two things. So if you have even degree like polygons, which is typical case, scissor cuts should be possible. Mathematical cuts can do everything. What if I have something that has odd degree vertices, but I still want to use scissor cuts? Well then, it turns out two cuts are enough, pretty much. That if you have, I'll call it a 2-edge-connected planar graph, equals the union of two even graphs. I should say even subgraphs. It doesn't much matter.

So there's one exception, which is, if you have an edge in your desired set of cuts, that partitions the graph into two parts. So if you could delete an edge and disconnect the graph into more parts, there's no hope of decomposing this into even graphs. I think. Should check that. But as long as you do not have such, these are called bridges, typically. So bridges are forbidden. If you have no bridges, then you're at what's called a 2-edge-connected planar graph. And then you can decompose your graph into two parts, each of which is even. So you could fold,

make one straight cut, and get one even graph, fold, make once straight cut, get the other even graph. Good luck folding it the second time if it decomposed.

But in theory at least, with two scissor cuts, you can make anything that doesn't have bridges. So that's just kind of an aside. Some graph theory that tells you a little bit more about what you can do with two scissor cuts-- pretty much everything. Any more questions about that? So that was odd degree vertices.

Next questions are about folding and how exactly-- so I kind of briefly sketched the proof for linear corridors, how you would fold or how you would prove that this thing actually folds. This is the skeleton method first. We'll go to disk-packing afterwards. So how do you convert a set of linear quarters into a tree? Then once you have that correspondence, how does trees folding flat relate to corridors folding flat?

And so I redrew this figure to make it a little bit clearer. So this is one of the images in the textbook where I just face colored, in this case with three colors, the corridors. Those are the regions of constant width bounded by perpendiculars. This is for making a turtle. That sort of doesn't really matter. And this is the corresponding tree. And this is the folded state of this blue guy here between B and C.

And to really illustrate what's going on here, I folded this thing. It's tricky to fold, let's say. And I added a few paper clips to make it really stay. But if you hold it right, which is a little bit challenging, the projection of this structure is exactly this tree. Maybe let me show you some examples. Out here, for example, this flap is labeled A B, so it corresponds to this flap.

And it corresponds to the material here between A and B. And this A part is at the very tip here, and attached to this is this guy. That would be this unlabeled pink edge, which corresponds to this one. It's unlabeled, because it goes off to infinity in principle. So this would just keep going. This one turns around right here. Then there's this edge, this bit of material. Sorry, it's probably easier for you to see from that side.

And that corresponds to this BC part, which is exactly this folding. And if I laser cut

or even scissor cut down these two lines, and just cut out that little folded part, it would look exactly like this. So conversely, I was sort of begging the question by assuming I could fold this thing, if you don't know how to fold this thing, you can go on the reverse direction and figure out how to fold it by first modeling these corridors as a tree.

How do you do that? It's just like in the tree method, going from crease pattern from the tree method to the shadow tree. Except there, we were given the shadow tree as input. Here, we have to compute it. But it's like what you solved in the problem set. Is it one or two? For each of these corridors, it has fixed width, so you make an edge of that length. It's not quite drawn to scale here. This has been scaled up by roughly a factor of two. So this length becomes this length.

And you label it the same thing. This set of connecting component of perpendiculars A goes to that point. The connected component of perpendiculars here, this thing which branches, all of that is B. It's like a hinge in the tree method. So that's going to be a hinge between this flap and two other flaps, whatever touches the B perpendicular. So there's this pink one, and there's the cyan one. Cyan one is here. It's attached to whatever C is attached to. so here's C.

It branches off here, bounces around. All that's C, and it's adjacent to the blue thing we just did. The yellow and the pink-- that's this yellow and this pink-- connecting to D and G, and so on through the thing. So it's actually really easy to map from here to the tree, just every corridor maps to an edge. Every connected component of the perpendicular graph maps to a point. And this is the projection. It will always look exactly like this.

You'll be able to independently manipulate each of these corridors as a flap. And individually it's pretty easy to fold each corridor. It just looks like this accordion thing. So the faces will be stacked linearly in order from back to front, and be really clean. They'll all line up in this nice vertical strip. So that's easy to prove it exists from sketching the proof now. And then all you have to do is attach them together. So basically first you take this tree view, you fold it flat. Here, I can fold it flat just by

collapsing the top and bottom. Then I replace each of these edges with one of these vertical accordion strips.

And then I just need to check that at each vertex I can actually attach everything that needs to attach without getting crossings. So that's the one part of the proof I'm not going to show you, because it's kind of tedious. But basically, because the structure was planar, because it came from one sheet, you can show there's not going to be any crossings there. All of these layers, each of these edges expands to be many layers, and the layers will nicely connect together. So that's the sketch of that proof. Any questions? This is for just linear corridors, not circular ones.

OK. Cool. Next question is about the bad example that makes a dense set of creases that completely fills the plane, and therefore is completely unfoldable. And the question was, is it really unlikely or is actually very likely that this happens? And there are two things going on. Here's the example, again, from the textbook. So it's got, the dark blue is the desire cut graph, then the black lines are the straight skeleton, and then the dash lines are the beginning of the perpendiculars. And the point was to make this corridor width verses this corridor width verses this quarter width versus this corner width, make those all irrational multiples of each other. And then as this thing spirals around, it never finishes. It never hits itself.

And so it just keeps going. I think this is where this one's currently going. And so the question is, well, irrational multiples are actually very common. If you, for example, randomly perturb all these vertices and you measure the sizes of those corridors, with probability 1, they will be irrational multiples of each other, because irrationals are much more common than rational numbers.

And that's true. But the other thing that this example requires is this outer boundary. We need that none of these perpendiculars can escape out to infinity. If they do, they won't stay in there. Eventually, if there's a tiny, tiny gap here, if these guys didn't quite match up, eventually because this thing is dense, it will find that little gap because it has positive length. When it finds the gap, it's going to spiral around and around and around and go out to infinity instead of being trapped inside.

So the unlikely thing in this example is actually this outer pink polygon, that the perpendicular coming out this way bounced around, did lots of things, and eventually hit the same vertex. That's actually unlikely. Definitely in this example and in general we claim that we do not get cycles of perpendiculars except in one kind of scenario, which I guess I could draw in the software. Let's try it. So whenever you have a vertex of degree more than two, so let's do something like that.

Little too far away. So let me just add a little guy like this. And this a little closer. It's got a lot of spiraling. Let's make it a little bit cleaner here. It's actually kind of degenerate, but the point is, if-- here I have three cut edges. I've kind of made them roughly equal so this doesn't get huge, but in general, if you have any straight skeleton vertex and you reflect it' around these bisectors, you will always come back to where you started.

So this requires that each of these angles is strictly convex, so this is why you need at least three cut edges to come together here. But once you have at least three, you'll always cycle around. You'll always come back to where you started. It actually happens again here. This guy cycles around and comes back. So that's-- well, sorry, that one's actually a little bit special because of what I did with the dragging. In general, it's going to be more like this.

It's a little hard to see. What's going on here is that this innermost guy cycles around, indeed, but everyone else spirals around and keeps going. So that's actually the perpendicular. It comes from here and goes that way, and then it spirals around. It hasn't finished yet, but it would actually spiral all the way out to infinity. So the claim is, in general, the-- this is conjecture. With probability 1, if you randomly perturb the vertices slightly, the only cycles of perpendiculars that you get are around a single vertex.

So something like this example, which we saw where there is a straight skeleton edge here, here, and then this guy-- I guess I should draw the skeleton edges. It's a good test of how accurate I drew it, because I know if this is perpendicular, this is perpendicular, and this is perpendicular, this will always come back to its starting

point. It's just property of reflections. Basically, because we satisfy Kawasaki here, this must be true because these are bisectors of these guys.

So this has to happen. The claim is, that's the only situation it happens. And if this is all that happens, you can prove there's no density, and in fact, it folds flat. So this is the good case. Unfortunately, when we make real examples, we like degeneracies because they reduce the number of folds. So the theory says, avoid degeneracies Let's perturb things slightly, then it's guaranteed to fold. But in practice, you want to add degeneracies, just carefully so that you don't get dense behavior like the weird example I showed you.

So that's clarification why we think this does not happen. Or, sorry, why we think with probability 1 things are good. But sadly we can't prove this. Maybe we'll work on it in the open problem session. I think it's tractable, I just haven't worked on it in a long time. Questions? OK.

So, right. The claim is if you perturb this example, everybody will escape out to infinity like in the spiral. OK. So I think this is the end of the skeleton method, but before we go on, I want to show some examples. So problem set-- we have up to four [INAUDIBLE] later today. One of the questions is design your own folding cut, and draw it. So you probably want to draw it into a program like Inkscape or Adobe Illustrator. It has good snapping, so you can find intersections and things like that, and compute angular bisectors as in ruler and compass.

And these are some examples from 2010, same question. There are lots of them, but I chose three that are particularly cool. This one has a line of symmetry, you get a fish bone by [? Ji, ?] who's a Ph.D. student in the media lab. This is by [? Sarah Eisenstadt ?], who's a Ph.D. student in CSAIL working on folding things. Witch's hat is pretty cool, pretty fairly simple. You have to fold your examples. It can't be too complicated. And then Jason [? Ku ?], who we saw the guest lecture by wasn't sufficiently impressed by my jack o' lantern, so he made a really complicated one and folded it. So those are some inspiration points.

And I thought I'd show you a magic trick which is one of the sources for inspiration

for the fold and one cut problem. So, a piece of paper. So this is a magic trick of unknown origin. It was described by Martin Gardner, I think, probably in the '60s, and then the book appeared in the '90s. And it's a story of two politicians, and let's just be generic. Let's say one politician was very much liked by the people, and the other politician was disliked by the people. Imagine a simple world where it was so simple.

And by a freak accident, both politicians die at the same time. And they both happen to be Christians, so they go to the gates of Heaven. That's how the story goes. And arrive at the gates of Heaven. I guess Saint Peter's the keeper of the gates, and Saint Peter says, well not just anyone can get into heaven. You have to have a ticket. And the good politician being liked by the people has a ticket, which he folds flat. And the bad politician has no ticket.

So the bad politician says, well you know, we've had our disagreements, but maybe you could put in a good word to Saint Peter or do something to help me out. I hear Heaven's a nice place. Maybe we could go there together, resolve our differences, whatever. So the good politician, having a ticket and a pair of scissors like any respecting politician, takes his ticket and makes one complete straight cut. And it's going to get a lot of pieces, so I've got to be, hold this carefully. So, put that down. And he hands all these pieces to the bad politician, says there you go. The bad politician has no idea what they're for, so he hands them to Saint Peter.

Saint Peter starts unfolding the pieces, says, I wonder what shapes I get. I hear there's a cool problem about this. So it's a little hard to do without a table. So I'm going to have to use the board. First, we get the letter H. Then we get-- put these down here. Then we get the letter E. OK? Then we get letter L, and the letter L. And if it was on a table, I'd arrange it, and clearly Saint Peter's not happy, and the bad politician gets sent straight to hell. The good politician hung on to one piece cleverly, and his ticket is still more or less intact, and he gets into heaven.

That's the magic trick. It's very simple folding. I've known this, memorized this even, for years. And you get exactly those pieces. So pretty cool. Of course, from a

mathematical standpoint, it's a little unsatisfying. Because, come on, use three pieces for the H, three pieces for the E. Surely you could do better. And from a rectangle, you can kind of do better. It's a little awkward because the pieces are not all the same size, but is one scissor cut because all the vertices are even degree.

You cut it, and you get all these pieces, which if I were more practiced at this, I could immediately pick out which one's the H. OK, I'll just do them out of order. Here is the cross. Well, kind of a cross. Not quite perfect proportions, but good enough. This is the letter E. This one's actually more impressive when you don't have a table, because you can't tell that they're all different sizes. The letter E. We've got the letter-- these look like L's. And they're small L's, but there you go. They're different orientations. And then I've got the letter H. There you go. OK.

So that's our new and improved version using universality of folding cut. So it gives you some idea of A, where folding cut came from in recent times, is the magic community. I mentioned Harry Houdini did some tricks, and Gerald [? Low ?], but there's a bunch of these tricks around, and kind of cool. So the checkerboard trick and the previous hell trick. So those are some examples. Now we move onto the disk-packing method.

So I have one question about this, which is, how exactly do we go from the disk-packing to-- yeah, cool proof, though, cool proof, bro. So how do we go from the disk-packing to the decomposition into triangles and quadrilaterals. So I thought I'd just review this slide. We start with our graph. We offset it by some tiny epsilon. That's to get things off the line, basically. Then we do this disk-packing. And remember roughly how the disk-packing works.

We put some disks at each of the centers. Here, it's a little awkward because of the offset, but you put one at each of these four corners. Also the corners of the piece of paper. Why not? Then you also pack small enough disks along the edge, so that you cover the edge by diameters of the disks. I do that the same on all the sides. Basically, you try to put a big one in, but if that big one intersects, you decompose it into half. That's how the algorithm works.

So now you've covered the vertices and the edges, but you may have big gaps like this, too many sides on this gap. And so then you just greedily put the largest disk you can in those gaps until all you're left with in these yellow gaps are triangles and quadrilaterals, or three-sided gaps and four-sided gaps. And then all we do is draw this red graph by putting a vertex at the center of each disk.

That's what the question was about. And then whenever two disks touch, we call this kissing disks for historic reasons, because they're just barely touching I guess, at their lips. Got a lot of lips for disks. At least we're in flat land. So then whenever they touch, we draw the edge between the two centers. So that decomposes this piece of paper into parts, and because the gaps are three sides, those will correspond to triangles, or four sides, those will correspond to quadrilaterals.

So that's all. And then we put rabbit ears in here, and line universal quad molecules in there, and it folds flat. It will align all of the red edges plus these black edges on the outside, and it will align all these inner red edges with these black edges, and then we do the sync folds that we mentioned. And that will get one of these out of the way of the other, and we'll end up aligning just these cut edges. So that's a quick review of that method.

So one question is, how do you allocate the disks. Is there sort of a best way? And I guess there could be several measures. Maybe you don't want really tiny disks, because that tends to lead to very tiny folds. But the standard measure here is how many disks do you need, because that will reduce the number of folds. So can you minimize the number of disks, and in general, it's known roughly how many disks you need. And the algorithm I just told you achieves that number of disks.

So first I'll give you the number. Number of disks is proportional to the integral-- I didn't say it was the easy bound, but it is the right answer. OK, there's a lot of notation in here, and this one you should definitely not know unless you've taken a computational geometry class. Even then, you probably wouldn't know it. It's not that common. It mostly comes up in meshing and disk-packing, so it's pretty specific. It's called local feature size.

And this is if, once I define it, it is actually fairly intuitive, this bound. Local feature size at X. So we're imagining some kind of polygon here. And at every point of the polygon-- let's look at a vertex here-- but it works for any point along the boundary. Actually maybe not even on the boundary, but we're interested here-- this notation del p is the boundary of the polygon. So this is del p. That just means edges and vertices, all of those points. So we take some point X on the boundary, and we look at the smallest disk centered at that point that touches another feature.

So that's the feature size. Features here are edges of the polygon. That's all we have. Now of course, no matter what size disk, you hit this feature and you hit this feature, so those don't count. Those are incident features. So in general, it is the smallest radius disk that hits a non incident feature on incident feature, which is an edge. So here, as soon as the disk gets this big, it hits this edge. And so the local feature size is this radius. In general, for every point, you can draw some size disk. So over here, it's going to be a little bit bigger. You can maybe get up to here.

Actually, I guess it should be non-adjacent. That was going to be a little bit awkward. I also don't want to count this edge, because if I'm right here, the feature size is super tiny, and I don't want LFS ever to be 0. So you also have to skip the adjacent edges, which means local feature size of this point is going to be more like-- hard to draw circles-- it's going to be more like that distance. That's when you hit a non-adjacent feature, sorry.

So you have to exclude this edge that you're on and the two adjacent ones, otherwise this definition doesn't quite work out. So then you measure this for all points. There's infinitely many, of course, continuum along the boundary. That's the integral over X in the boundary of P. And you take 1 over the local feature size of X, and you integrate that for all X. It gives you a number, and it turns out, some constant times that number is the right answer.

The algorithm I achieve, which I described, which is put the biggest disk you can or divide in half if you can't will get within a constant factor of the best possible bound of all possible to disk-packings. So this is pretty much known. You get within a

constant factor of the optimal disk-packing if you're counting disks. Not a great number. I can't say they're N disks or N squared disks, because you just can't do that. If you have a piece of paper and some really long-- here's your cut graph. Doesn't quite go to the ends, though-- then you've got to have a tiny disk here, and that's going to require you to have a bunch of disks. Even though this has constant size, you're going to need, I don't know how many disks here.

Especially if your paper's really narrow, you're going to need a lot of disks. That's the best you can hope for disk-packing. I actually brought a little example of something like that where your goal is just to cut out a single segment in the middle of the paper. This is to illustrate odd degree vertices. These are degree 1 vertices, and you might think, wow, it's impossible for me to cut this thing without, I don't know, just cutting. I could cut with my X-ACTO knife from here to here. If I want to make a complete straight cut, you can fold it like this. This is what the skeleton method would give you.

Fold it like this, and then you use your laser to cut right along the line. With scissor cuts, it's not so bad, right? I cut slightly in. It's pretty much cutting along the line. Just not quite. I made a tiny mistake. So I cut off a line, slightly thicker line, and I get my slit in the paper. So mathematical cuts are not really that bad. You just are cutting a slightly larger version than the single slit that you wanted to make. You just have to cut very carefully. OK.

That was number of disks. Next question is, how does all this relate to the tree method? This is kind of a neat question. Both methods, skeleton method and the disk-packing method relate to the tree method in different ways. I thought I would talk about that. So first, let's compare the disk-packing method which we just talked about to the tree method. Both of them use universal molecules. This one only uses universal molecules for triangles and quads.

We know in general, the universal molecule works for any convex polygon. This is an example of a pentagon. And it works. It has a gusset to get the tree lengths right. Here we use gussets to kind of get the tree lengths right. As you may recall, we

want the perpendiculars to go to the disk kissing points, those tangencies. Because then they align with the others, and then we never get the perpendicular spiraling or doing other bad things.

So the number of creases becomes proportional to the number of disks. The number of disks is this messy thing. So that's that. Let's see, what else? Here, we only use disks. Over here, we could just use disks, but there's also the ability to use rivers, so this is more general in that we can do bigger polygons than' just quads, and we can do rivers, not just circles. Here, the input, though, is a tree with specified lengths, and it's actually NP-hard to place the disks correctly. We proved that last time. Over here, the input is a polygon.

It's already been embedded on the sheet of paper. There's no really hard decisions here except you have to fill the paper with disks. But that's not so hard. So you can do this in polynomial time. Polynomial and even, almost, roughly this time. Times a log factor. You can find displacement of disks. So that's not too hard, whereas over here, it's NP-hard to place a disk, optimally at least. Find a displacement you could do.

So basically the inputs differ, and it's using the same backbone or the same, what do you call it, front end. The same back end, I suppose, after we've decomposed into a bunch of molecules, you just fold the molecules and you're basically done. Here we have to do some syncs at the end, but it's using it for different purposes. Otherwise they're very similar. And indeed, this is inspired by that.

OK, so that was the disk-packing method. Then there's the straight skeleton method. These look much more different, but in fact, they are quite related. If you do fold-and-cut on a convex polygon, it gives you roughly the universal molecule, just with no gussets. So here, neither one is more general than the other. The universal molecule is more general in that it has gussets, which lets it control the tree topology you get and control the lengths you get in the tree.

Over here, we saw the tree just happens. You can't control the tree. The shadow tree is just a tool to prove that it falls flat, but you can do non convex polygons,

which is much more general than convex polygons. So they each have their own unique claims to fame. A natural question is, can you combine them. And in fact, you can.

And this was, you may remember a bunch of lectures ago-- I think class four or so-- I showed this slide of the meat of origami design secrets volume two, or the second edition, rather. There's tree theory, which we talked about. There was a box pleating tree theory which we briefly showed. And there's this more general thing called polygon packing, and this is in some sense the fusion of straight skeleton method of fold-and-cut, basically the straight skeleton, plus tree theory.

And this is done jointly between Robert Lang and the domains. And here's an example of what it looks like. It's a little hard to see here, but I will point out if you look at the blue lines, which are the cut lines-- here, of course, they're the active paths. This guy is non convex. And if you look at the red stuff in there, it is exactly the straight skeleton of that polygon. So those red lines are the straight skeleton of this blue thing. So there you go. I've got the fold-and-cut straight skeleton being used in that setting.

You've got perpendiculars all over the place, which is an annoying feature of the skeleton method. It is inherited by this method. And if you look at something like this rectangle. It's convex, but you have gussets. This is a gusset, this is a gusset, gusset, gusset. It's a little hard to see, but we basically split the rectangle into this rectangle, this square, this rectangle, this square, this rectangle.

So this is really the fusion of the two. It's not yet been mathematically formalized, because we're first proving that the tree method works. Then we will finally get to this. But if you want these slightly more informal but practical version, read *Oragami Design Secrets* and it will explain all the different cases and what we believe is a correct algorithm. We just haven't proved all the cases yet. And you can use it to design cool things in the box pleating setting and get more efficient than if you required rectangles or some convex shape. Cool.

So that's that. What else do we have? I have one-- almost done-- one cool open

question posed by you is, what if instead of cutting with a straight line, I cut with a constant curvature line? So a piece of a circular arc. Then, of course, the things I get will have circular arc boundaries, but can I get any circular arc boundary with that fixed curvature? Or are there limitations? And at first I thought the answer is clearly yes, but I do wonder about situations like this.

Can you align these two by folding? I haven't thought about it enough to be sure, but not totally sure. Whereas if I have things like this, I think this is good. I mean, I could fold here, and in general, I could just kind of pretend there's a segment. And the segment I saw fold and cut on the segments, and that will also align the circular arc. That won't work in general, but I feel comfortable if they have the same orientation, it looks good. When they flip orientations, I'm not sure. So a neat problem to think about.

And one final thing is about flattening. This is about this question, this picture. So we said, oh great, because we can solve fold-and-cut, we can also take the boundary of that polygon and flatten it like this. Of course, the folding of the interior here which we're ignoring-- in general, this whole thing goes through three dimensions until we end up here. One dimension up, if we solve the 3D fold-and-cut problem where you have a polyhedron, you fold it through 4D back into 3D which is flat so that you align all the boundaries of the faces of that polyhedron.

And then just take the boundary of that motion. It's like flattening a polyhedron like cereal box or whatever. And indeed, that will find a folded state of a polyhedron that is flat. But of course, you can't really use that motion, because it goes through 4D. This one will go through 3D, so if you want it to flatten this linkage, that's also not valid. So you get a folded state, but you don't get the folding motion if you could solve 3D fold-and-cut.

Our current state is, we don't know how to solve 3D fold-and-cut. We do know how to solve polyhedron flattening. So this relation is just kind of interest, it doesn't imply anything super useful. An open problem is, so we know how to solve polyhedron flattening as a folded state. What we don't know so much about is folding motions.

There is a recent paper from just last year which will continuously flatten by continuous motion any convex polyhedron. But non convex polyhedron we still don't know.

These are roughly hand drawn figures. An interesting project would be to actually animate their motion. I think it would look pretty cool and be much more convincing that it works. I mean, I'm convinced that it works, but it's a lot harder to see visually from these pictures. It'd be really cool to see the motions, the actual continuous animation. And a couple last project ideas. One would be to make a fold-and-cut alphabet, like the maze software that you just used. It would be fun if you just type in a message, and out comes the crease patterns. This is sort of like a special case of a fold-and-cut design tool, but in letters it's a little bit easier.

And a different challenge would be if I just want to be able to fold a single letter at a time, can you always do it with simple folds? So can you design an alphabet that is simply fold-and-cuttable. That would be even nicer. And there's some magicians who worked on that, but they couldn't get all the letters, so it would be nice to do with just a few folds, any letter of the alphabet, any digit. This is one we did for our conference in honor of Martin Gardner, gathering for Gardner five, back in 2002.

And here's another possible idea for a project, although mostly it's an excuse to show you really cool pictures. This is Peter Callesen, and he takes usually four sheets of paper, cuts them probably an X-ACTO knife or something, and then from that material that he acquires, builds a 3D structure on that sheet. So you see all the material that was used, and these are just amazing. And I don't think you could necessarily do this level of fidelity from a fold-and-cut , but it would be cool to do a fold-and-cut and then build a sculpture out of the slits that you made.

And I mean, it's really cool when he can use the cutout parts to be shadows of the things that he builds, especially when those shadows are more complicated than the sheet he actually assembles. That's a fairly cool effect of the monsters in the closet. Pandora's box, or box in a box in a box. You can't see that here, but you see it unfolding. The bones inside the hand, the old versus new cities. Using the positive

and the negative. There's actually a little boat up there, which you can see.

So pretty amazing stuff. And for free versus caged, and so on. Interesting project would be to try to make art out of fold-and-cut, just doing one cut instead of a zillion cuts. That's it. Any more questions? Yeah.

AUDIENCE: In the circle-packing method, how do you assign the mountain-valley [INAUDIBLE]?

PROFESSOR ERIK DEMAINE: OK, in the circle-packing method, how do you assign mountains and valleys? That was briefly covered in lecture. First you take a spanning tree of all of the molecules, and you basically fold each molecule into your parent in the tree. So each one has a direction, and from that you can extract the mountain-valley assignment. You go into the parent. It's a mountain there, everyone else's valley. Basically you're reversing one perpendicular fold per molecule. I think. Maybe two. With triangles, it's just one.

For quads you might be reversing two. So you start with all of the straight skeleton being mountains, all the perpendiculars being valleys, and you have to flip a few to just get them to fit together. Little more complicated than what I said, but I would need a slide to show you the details. It's in the textbook. Anything else? All right, that's fold-and-cut.