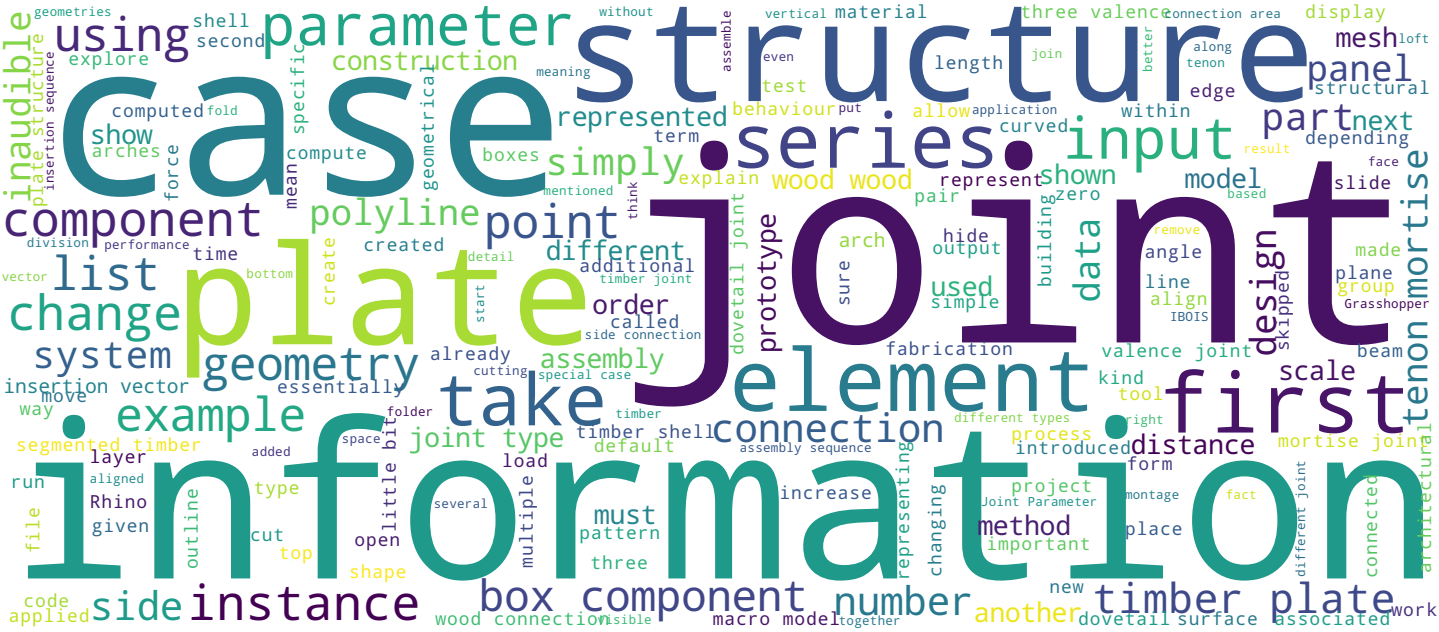
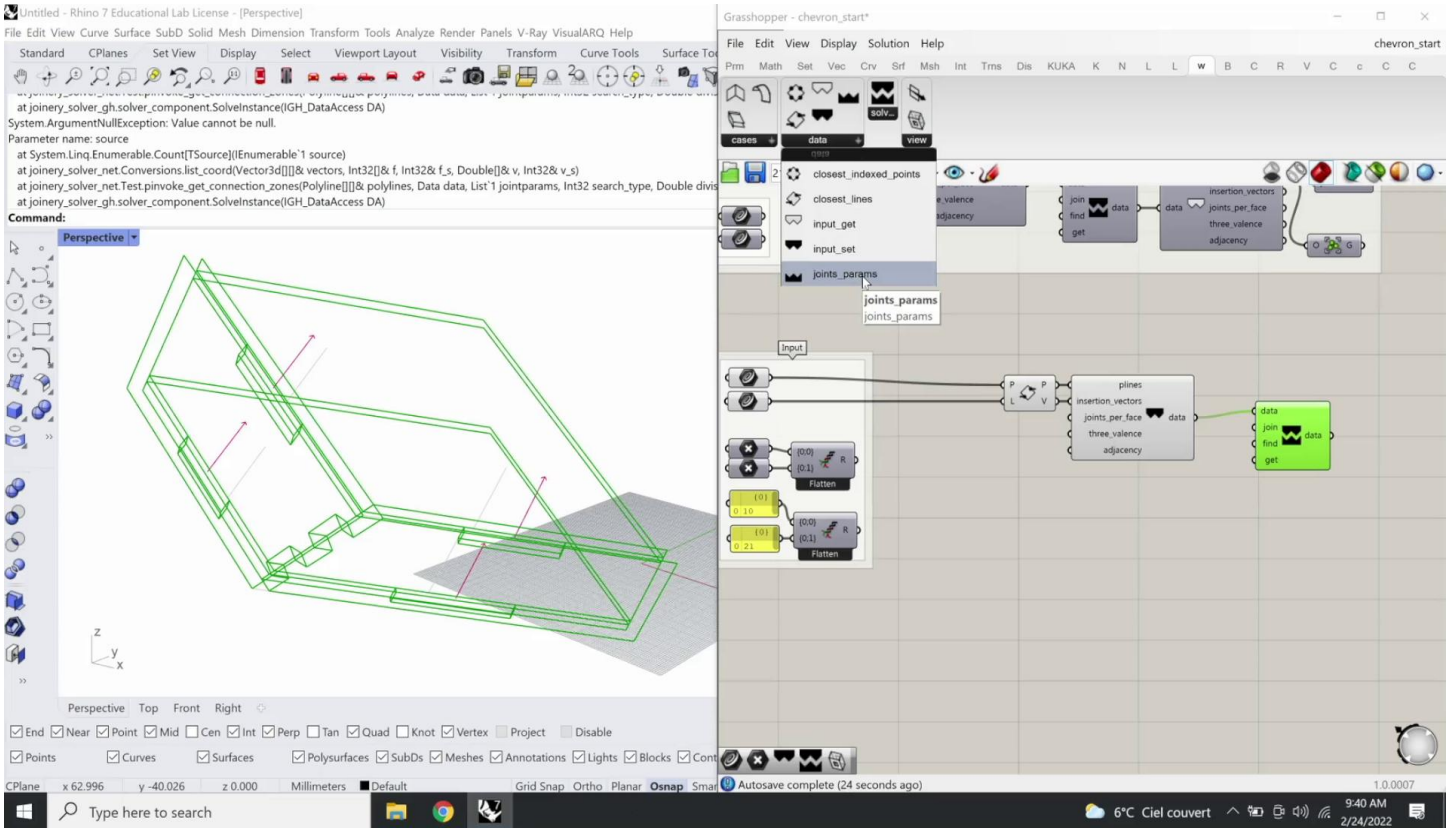

EPFL
iBOIS
Double-layer
Timber Plate Shell
Tutorial
Rhino3D and Grasshopper
Part. 2
Petras Vestartas, Ph.D.


case
structure
joint
plate
information
element
series
parameter
input
component
polyline
point
list
change
system
geometry
example
take
first
connection
box
instance
side
number
timber plate
using
panel
part
model
design
data
tenon
mortise
assembly
joint type
timber shell
distance
scale
curved
represented
simply
wood
shown
construction
inaudible
material
mesh
structural
next
depending
slide
made
plane
group
Grasshopper
IBOIS
beam
tool
kind
valence joint
fabrication
prototype
dovetail joint
different
created
order
called
side connection
already
cutting
special case
essentially
insertion vector
move
way
segmented timber
layer
run
aligned
Rhino
given
file
outline
type
cut
top
open
shape
little bit
multiple
several
must
pattern
three
load
default
assembly sequence
increase
representing
changing
project
Joint Parameter
method
important
place
montage
form
process
introduced
mortise joint
architectural
connected
different joint
new
another
dovetail surface
associated
work
code
applied
wood connection
visible
macro model
together
Search MOOC
Video
EPFL

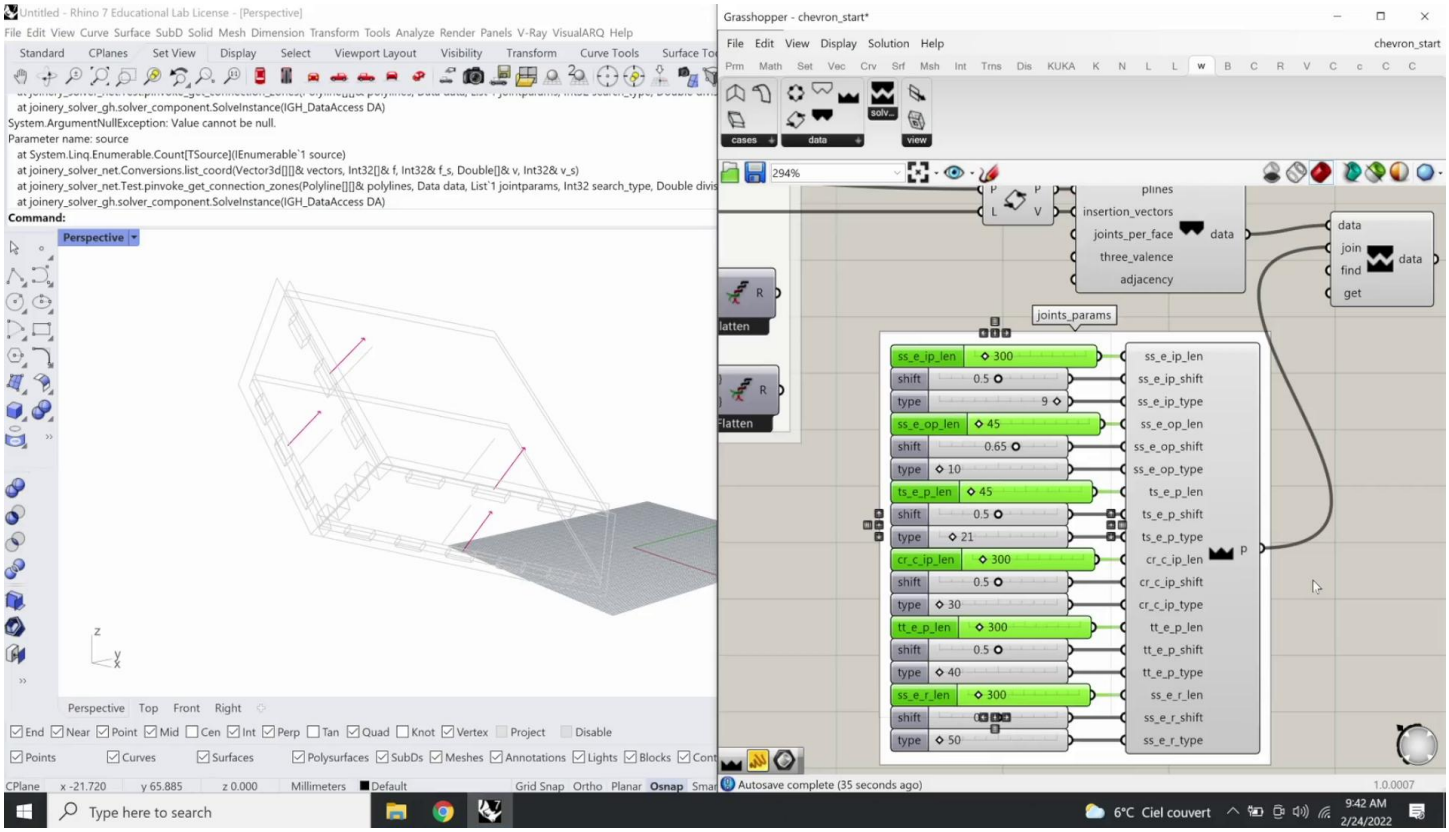


Now, the second case is an actual geometry coming from the segmented timber shell. We have, again, two vectors of insertion. We have a series of outlines. There are, in total, eight outlines which means there are four plates. And now, we need to get some sort of information for the insertion but also we need to have information for defining the timber joints We can actually repeat the same process again just for iteration. We can use the component Closest Lines. We take a list of quarter lines and we take a list of plines, and we compute those vectors per plate. Now, we can also use the component to group the data into one data structure that is represented for the joinery solver. Then we actually can put this data into the solver to cook with the information. You can see here that there is a small little problem is that this model is actually different in scale, and the scale is important because the joints are dependent on the length. Division, we need to change those parameters. For instance, we can use another component that is called Joint Parameters. It is a big component but it simply creates a series of numbers that controls these joints because we can also use different joint types, but in this case, we are going to use just a simple joint represented for the segmented timber shell.

Notes

Summary

0m 04s

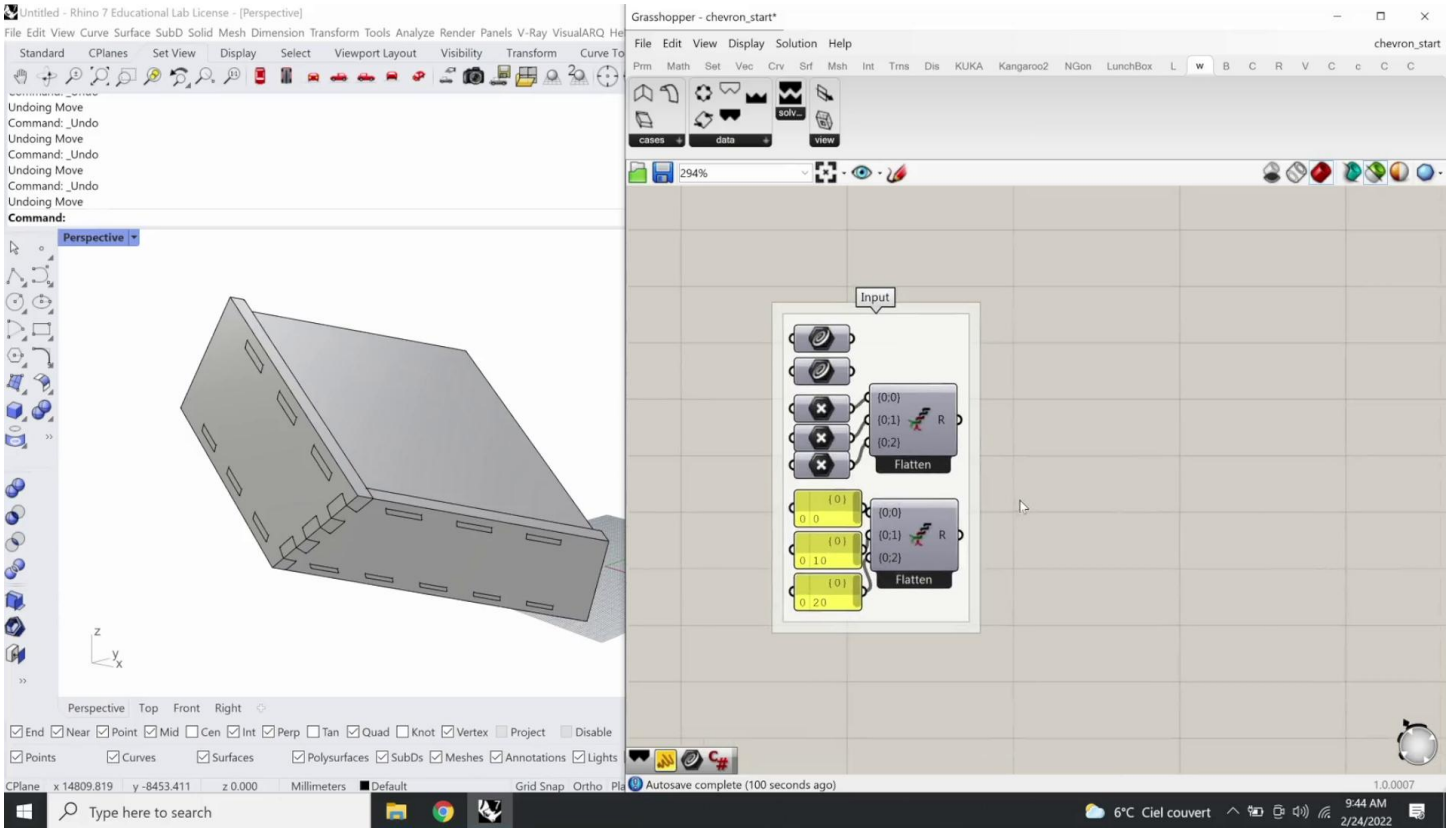


For instance, if we would place the join, input to the P, it represents the parameters. We can actually change those joints. For instance, the first thing what we need to do, we can change the length of the actual tenon-mortise. For instance, we can change the out-of-plane connection which is actually the tenon-mortise joint. If you see that if I move the joint, the [inaudible 00:02:31] the dovetail joint, you can see that the scale is actually changing. Let's take it, like, 4-5 units, and then we can also change the type of the length of the tenon-mortise which is top and side connection to a smaller number, let's say, 45. You see that it pretty much works real-time with changing the joints. There'll be another parameters for different cases but the data is that we have six groups of join that can be represented by this tool. Meaning that we can represent side to side, in-print connections side to side, out-of-print connections, then top side connections, then cross connections, and top-top connections, and other rotated connections. Within those types, we actually can have multiple subcategories because they can be different types of tenon-mortise, different types of dovetails, another Japanese giants and so on, that we actually will explore in the second session.

Notes

Summary

1m 54s

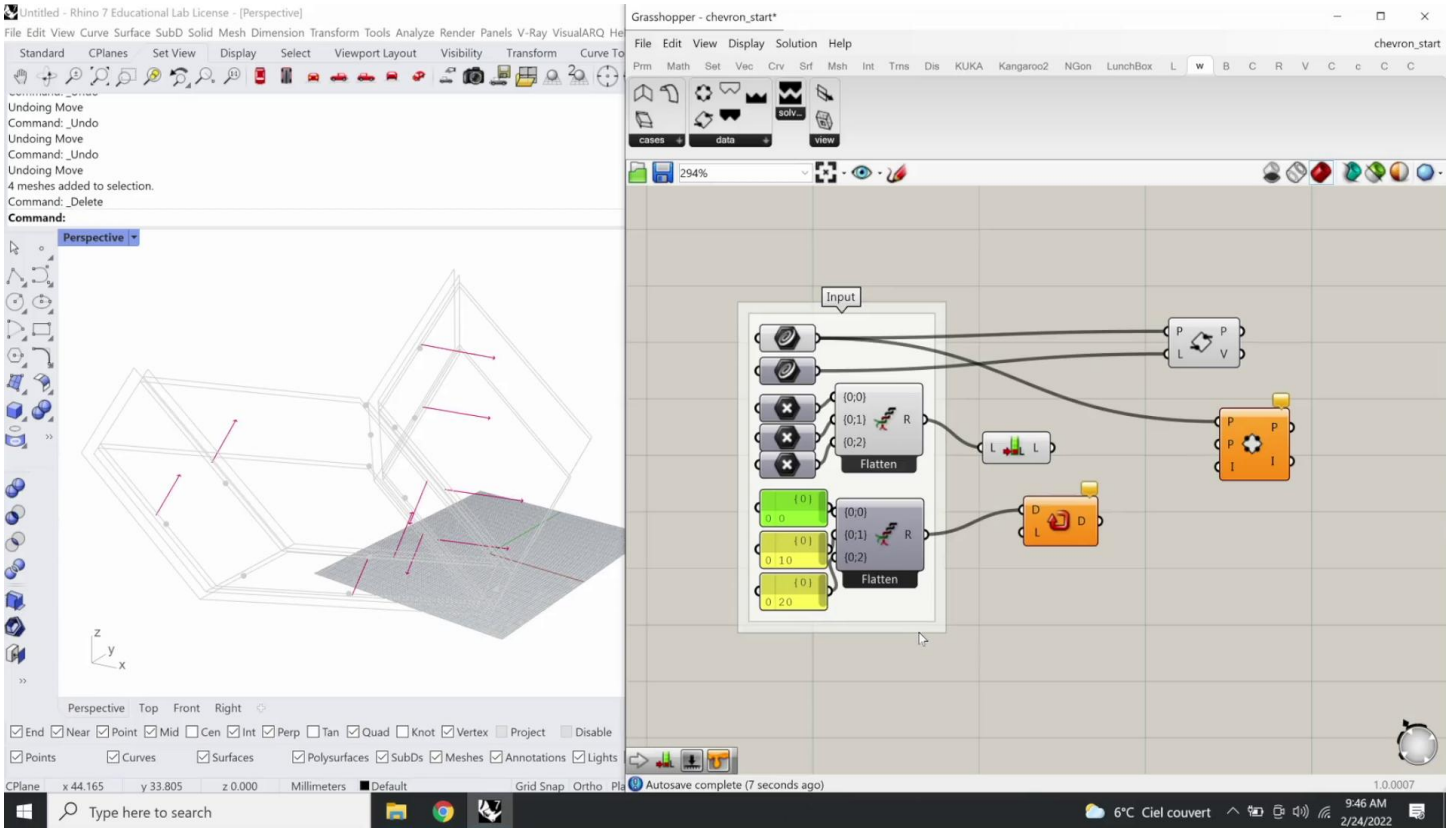


And then what we can do next is actually we can loft the geometry. First, we need actually to output this information. See that we have the full geometry representation. We can again bake it into Rhino and then cheque the assembly is correct. For instance, I can move these joints, and you see that they actually all following one insertion sequence. They are inserted by one plate. If we actually remove this plate as well, we can disassemble this dovetail joint in another direction. How it works in reality, first, those two plates assembled together and then goes one plate and goes another plate and we actually have the box component. For sure when we have multiple box components, we need to assemble them together. I'll show how to do this next. For now, we can skip those parameters, and I'll explain in the next example how we can actually use them. This example will be over, and I will simply disable this information. You can also disable this information. At this point, we don't need those parameters for the sake of very simple example, would be enough to show this very basic change of parameters depending on the [inaudible 00:05:41]. Now, let's go to the case where I introduced two box components.

Notes

Summary

3m 45s

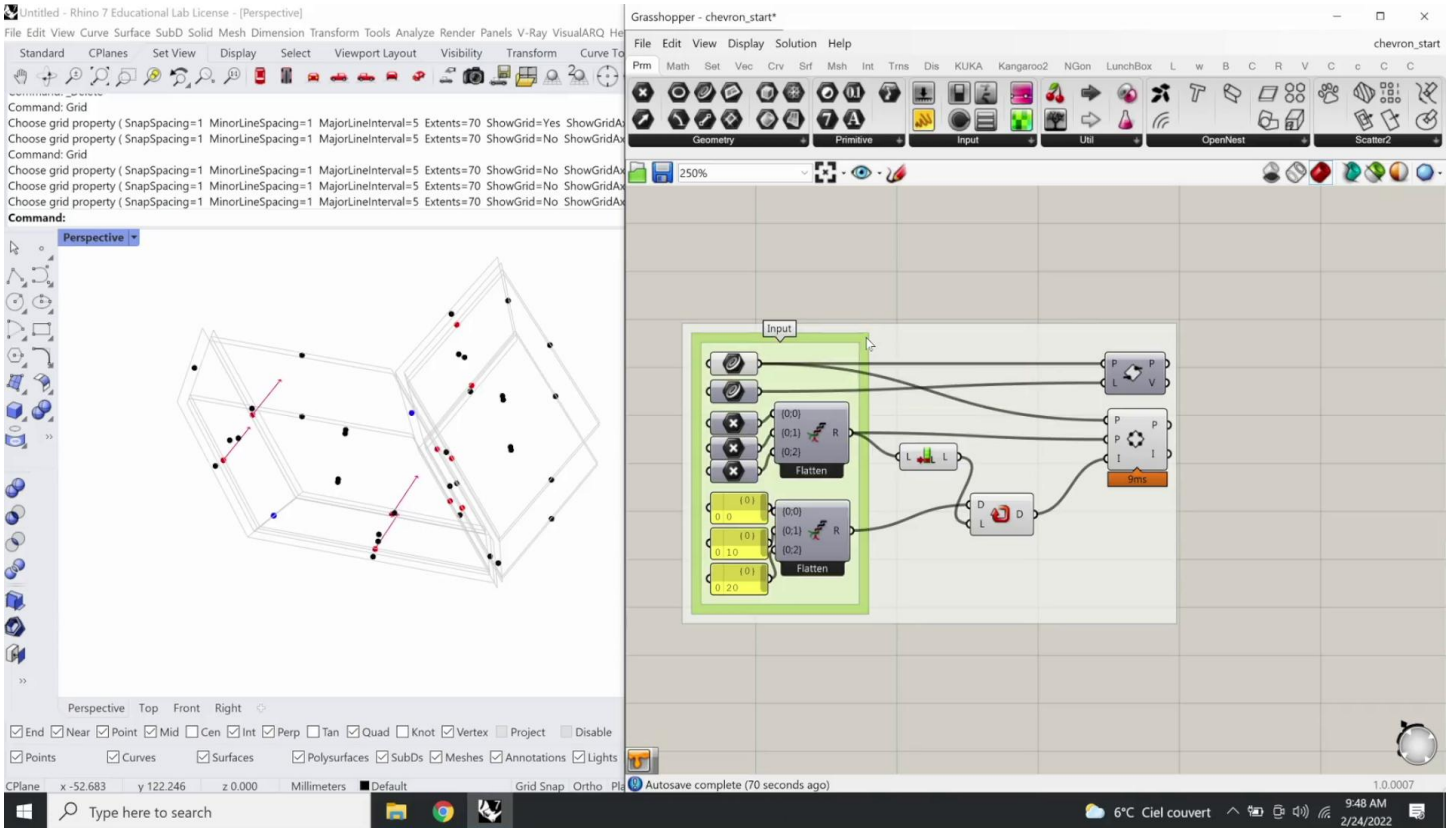


I will delete this information because it's not necessary anymore. You see we have two box components. The question is how we can connect those two box components together, because we already know how to make one box component. The first case, as we started before, we need to create insertion vectors. We have borderlines and we have lines. We created a series of insertion vectors. And then we need to have actually a special case. We need to assign individual joints depending on the edge length. In my case, I simply draw a series of points. You see these points here. These points will try to find and assign a special joint type per each edge, the same as this component did. To explain it more better, I will use a component called Closest Index Points. It finds closest points to each plate side. I will also input the parameters which represent plates. Then the two other parameters are the points and the numbers. I will use List Length to measure how many points do we have. I will also use Repeat Data to repeat actually number of three different joint types. This is actually the joint type that must be skipped. If you input zero, all the joints that will be assigned to zero will be skipped completely.

Notes

Summary

5m 48s

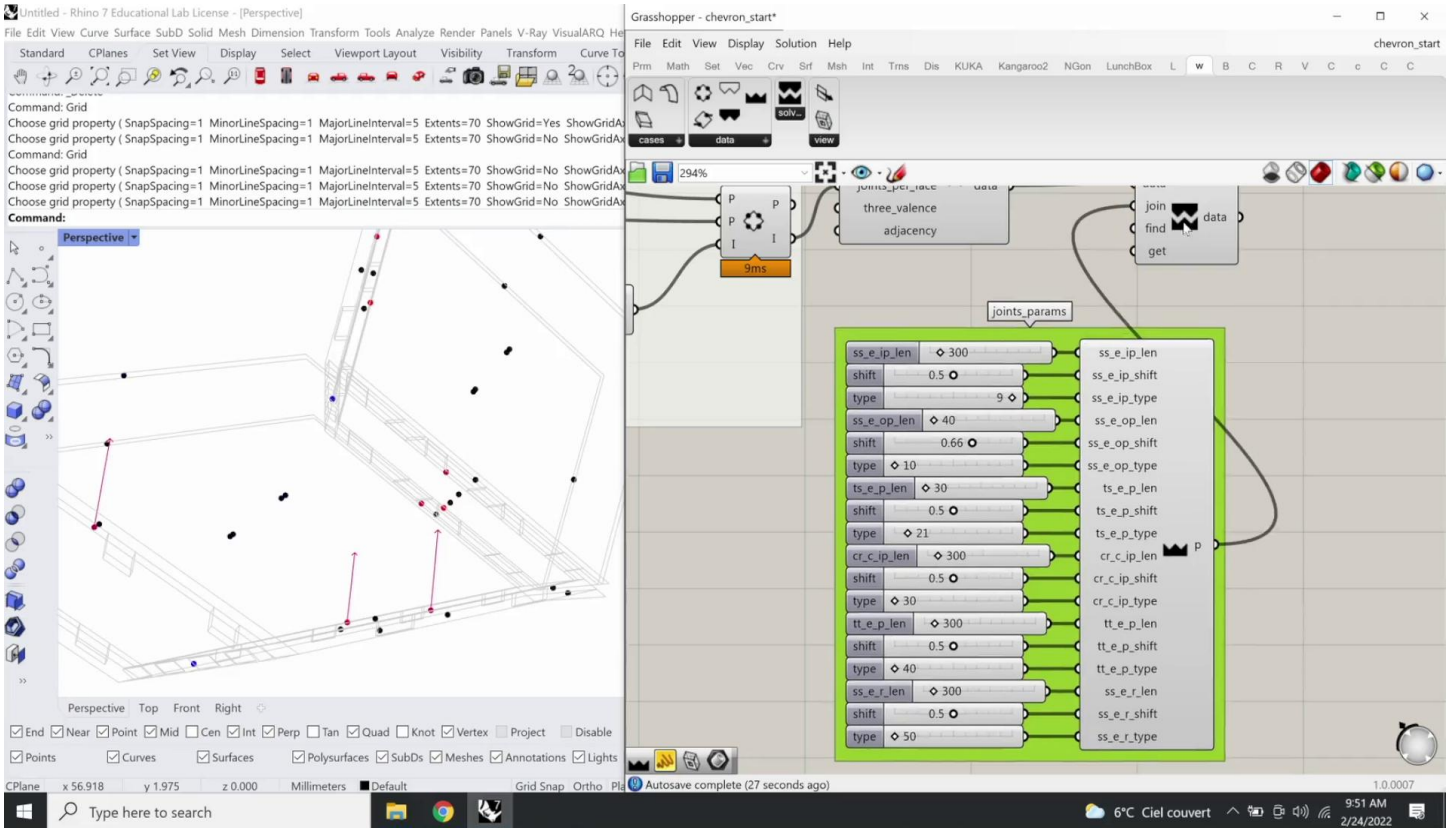


Then we have another joint type. Then we have a first joint type which is dovetail and a tenon-mortise joint. In this case, what we just created, we created a list of numbers. Then we created a list of points, meaning we have one point that must be skipped. In this connection area, nothing must be added, so it should be skipped. That's why it's zero. Then we have another connection point, which is actually the dovetail, as you can see here. Then we have a series of tenon-mortise joints that are visible here. If you actually input these numbers here, and then you input points here, you can see the assignment. There are three groups of points. Black points represents that no joint must be added here, nothing should be computed. Then the violet ones would be the dovetail joints. And then actually the red ones would be the joints that must be computed as tenon-mortise joints. I'll try to hide all the information so it's not overlaying. We have this kind of block. Also, we hide the grid. Now, we are almost ready to give all the inputs to our example. For sure, there is one last trick that I will need to show. We'll call it the input of the overall definition. I can delete this one.

Notes

Summary

7m 33s

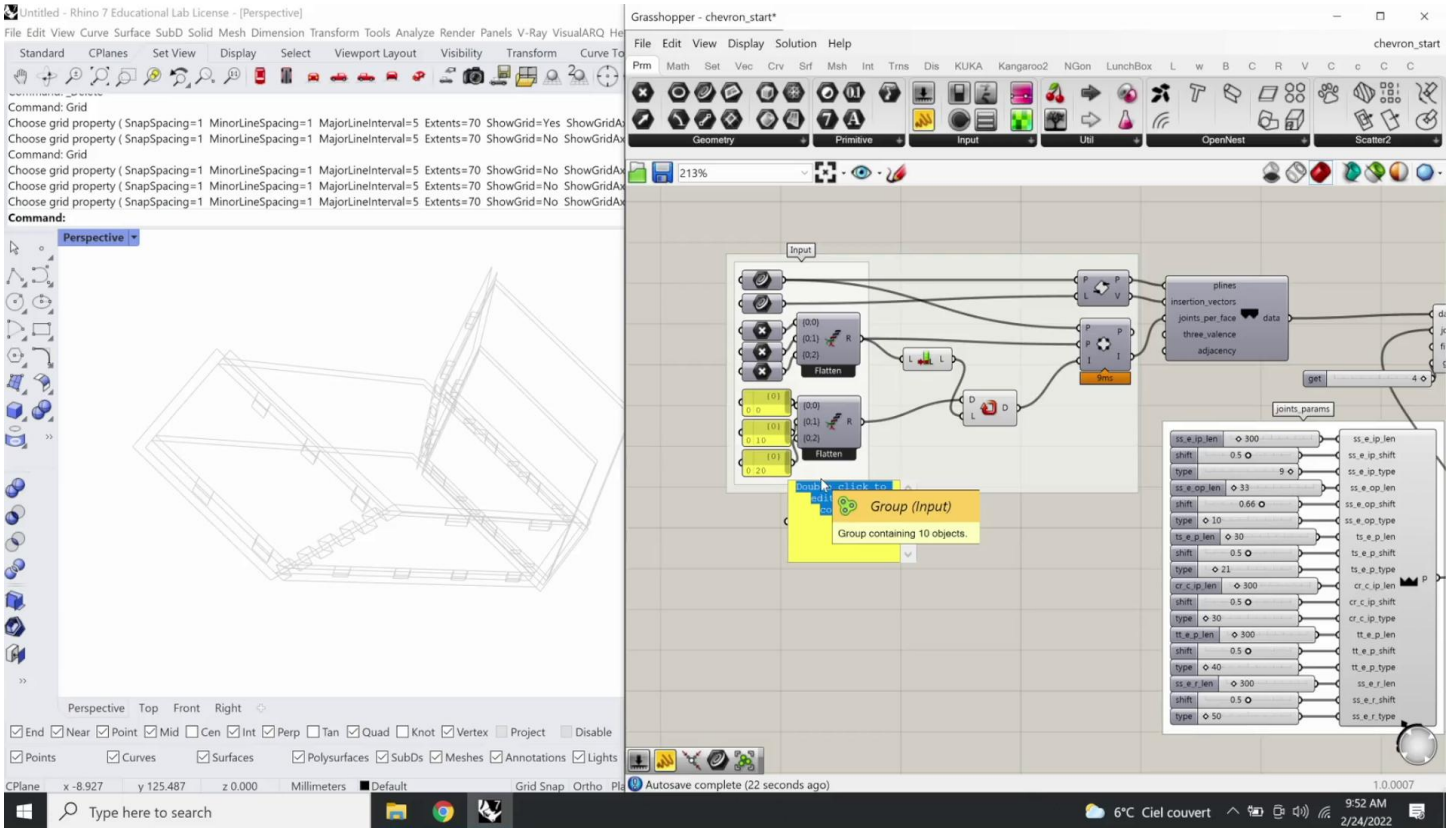


We have the input information. And as we remember before, we can simply have the input set, which is a list of plines, a list of insertion vectors, and then actually joints per face, which is the full parameter that we are going to explore. Now, we also need to compute the joints by simply inputting the data. You see that we have this first definition of a series of joints. Now, we can tweak a little bit the distance because it's not completely correct since I'm going to use the Joint Parameters component, then place it to joins. Now, we can change a little bit the information. For instance, we can change the distance to 45. Then also the distance to 45 here as well, so we can have more joints. Then you can see a little small problem that those joints are not really aligned properly here. Actually, we'll try to change the distance between those elements Then you can see that you can also increase the division number. There is like a special case in this example that depending on how those faces attaching each other. This connection area, they can be wrongly aligned. We have the case of the three valence joint as I mentioned before. The key question is how we can align.

Notes

Summary

9m 29s

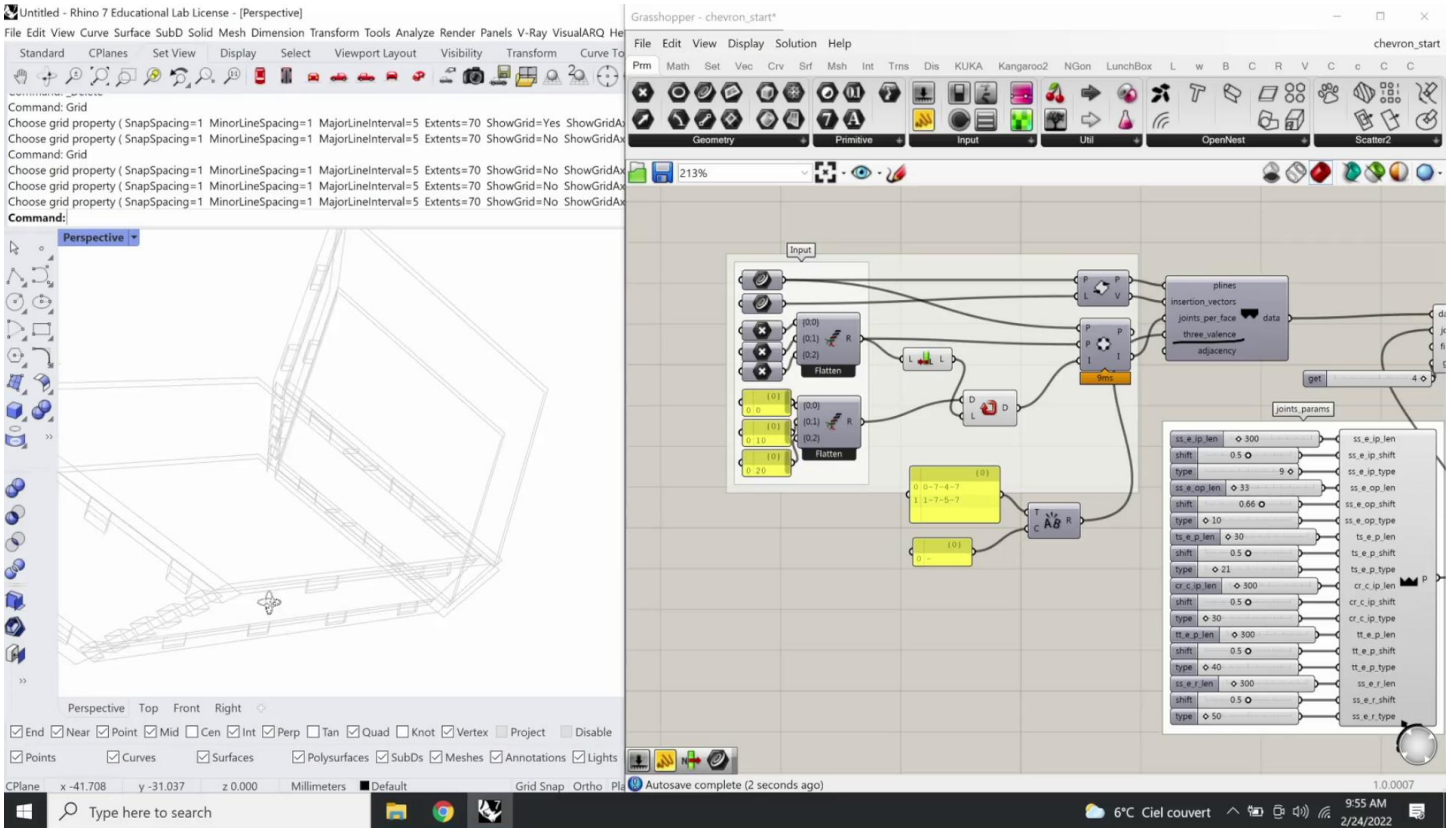


I will hide the previous information, that will not interfere. I think I need to hide as well this one. To give you some brief, there is a series of other inputs. Since if I would take this definition and try to output another information, you would see that… In this case, you can see these little central lines. These little central lines are representing how the joints were computed. Depending on the length of these segments, the joint distribution depends on the location, position, orientation, and so on. So if they are rotated depending on each other, they actually are wrongly positioned. By default, the input is 4 to get the joints. But there is like connection zones, connection volumes. Then the final output is polylines, which is not merged at this time. This one is the polylines that can be merged. If I will try to mesh this information, it will be actually wrong. Let's try to align. For this, we are going to explore the third input which is called the three valence joint. For this, we are going to simply take a list of indices. I will write indices of the plate. Let's say 0, which is representing the first plate, then the seventh plate, which is the side plate, then the fourth plate, which is another top plate, and then, again, 7.

Notes

Summary

11m 23s

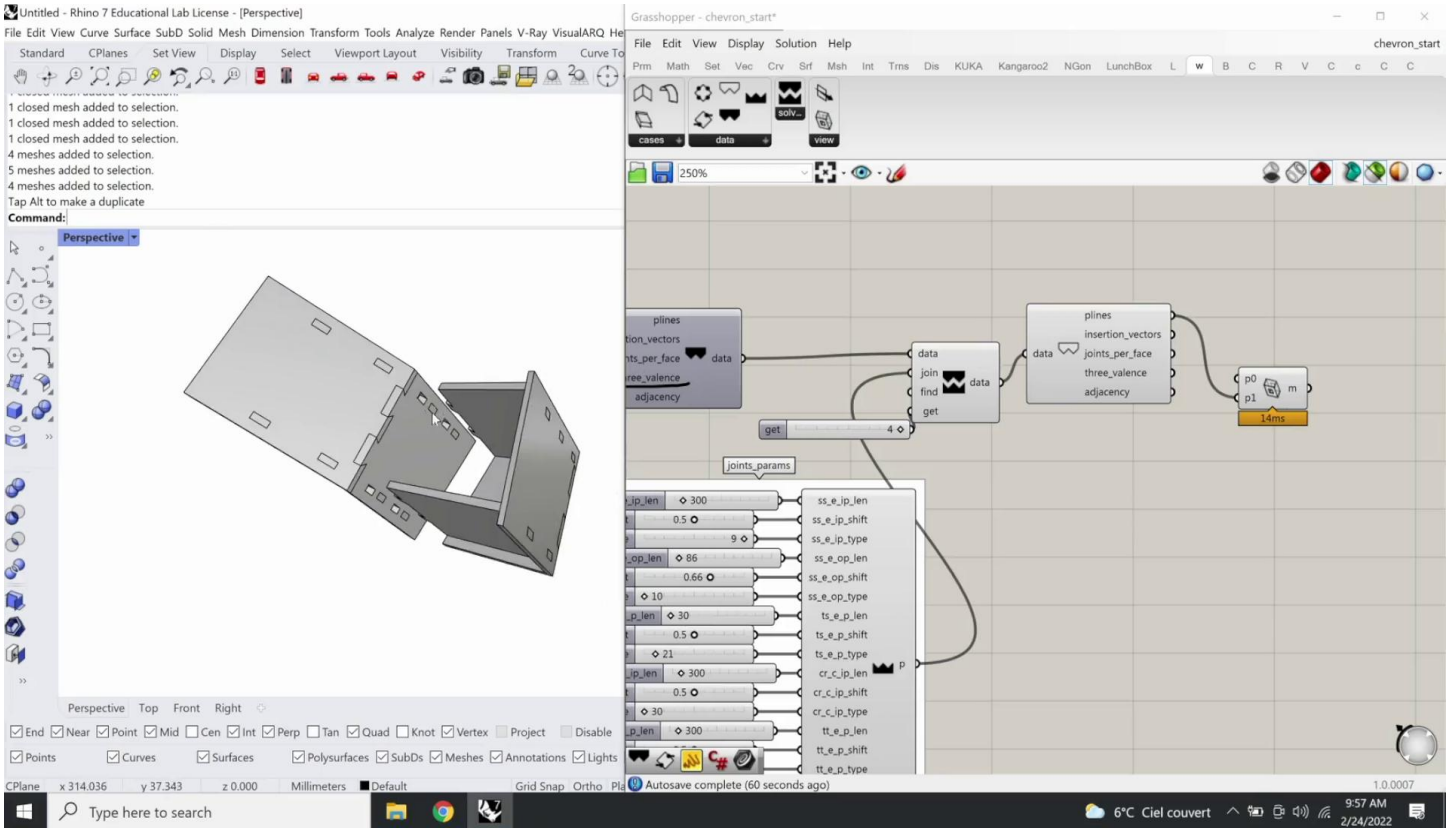


These are two pairs, one pair and another pair. Then I also do it for the bottom outlines. I will say that there is 1 and 7, 5 and 7, which is representing the other pair of outlines that must be represented. We can take this information. We can convert it to a proper data tree using the Split Tax component. This three valence input requires for a series of numbers which represents the outlines. To give you the understanding what it actually is, I can take a list of borderlines from here. I can simplify, and I can also take this branch here [inaudible 00:14:14] can take the first one. You see that this is actually pier 0-7 and a 4-7, which is represented 0-7, 7-4. This is the first pier, then comes another pier, this one, which is, again, the 1-7 and a 5-7. This is indexing you need to support to align those numbers. Depending on the structure, you might not need this kind of three valence joint but it is very specific to the segmented timber shell, that's why it's actually shown here. We don't need these components. We simply can input the three valence joint to this part. You can now see that the outlines were correctly aligned.

Notes

Summary

13m 12s

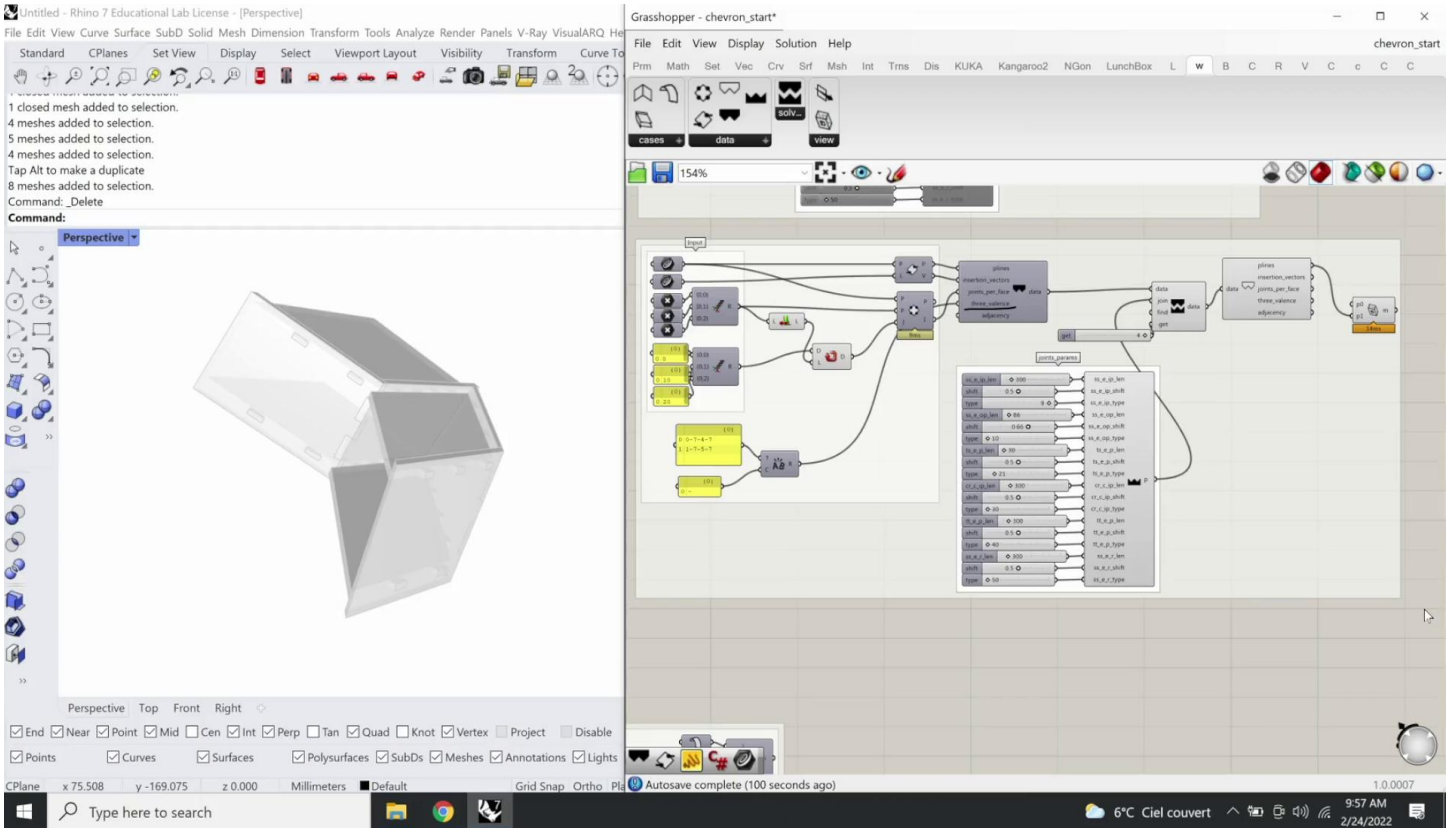


For instance, you can see the information here that these kind of aligned joints would be not aligned if I would remove this component. This is a wrong alignment, and this is a correct alignment. Simply what it does, it tries to have the same division align for the joints. If you have this information as the full input information, we can loft this data and we can display. For instance, I will use this component and simply try to mesh it. For sure, we cannot mesh it using the data. You need to unwind the information. That it's graphically readable Then you can see that you can have this information. You can also tweak again the divisions. You can have more dovetails if you need it, and so on and so forth. You have one box component, and then you have another box component. Simply saying that they can be moved together by one insertion sequence. In the other case, you can see that those box components can be disassembled or assembled based on the component basis by using one insertion vector. For sure, this is like simplified example, and we need to consider another case for the bigger structure. In the next example actually, we're going to export the full segmented timber shell because we will try to automate these parameters that we just input it manually.

Notes

Summary

15m 09s



It has to be a bit specially designed for the full [inaudible 00:17:20] case. I will group this part. I represented in this example the free one and joint and the alignment joint. And then now we are going to go to the next sample which is the case of the full annual structure or segment timbershell.

Notes

Summary

17m 16s