

The Tensela manual

Weave the design. Parametric design an architect understands.

Made from the manual at tensela.com on 2026-11-25. This manual is under the Creative Commons Attribution-ShareAlike 4.0 licence.

Contents

1. **Getting started**
    1. Welcome
    2. What is Tensela
    3. What you need
    4. Install
    5. Your first project
    6. Coming from other programs
    7. Free, offline and yours
    8. Who makes Tensela
2. **Key ideas**
    1. Nodes and cables
    2. Lists with names
    3. Two kinds of nodes
    4. Draw the form, then cut it
    5. Conditions and attractors
    6. Groups
3. **The window**
    1. Overview
    2. The canvas
    3. The 3D view
    4. The panels
    5. Exporting
    6. Find the best
    7. Preferences
4. **Recipes**
    1. Recipes
    2. A facade that breathes
    3. A rib wall
    4. A waffle model ready to cut
    5. A roof of hyperbolic paraboloids
    6. A Voronoi skin
    7. A tower that turns
    8. A sun study
    9. A vault that finds its own shape
5. **Extensions**
    1. Extensions
    2. Write an extension
    3. Publish an extension

6. **Nodes**
    1. All the nodes
    2. Values
    3. Shapes
    4. Curves
    5. Surfaces
    6. Divide
    7. Measure
    8. Transform
    9. Meshes
    10. Fabrication
    11. Environment
    12. Simulation
    13. Groups
7. **Theory**
    1. Overview
    2. Visual algorithms
    3. Units and directions
    4. NURBS curves and surfaces
    5. Curvature and developable surfaces
    6. Meshes
    7. Form finding and the best value
    8. How Tensela thinks
    9. Named lists, inside
    10. How a project is calculated
    11. Lists in Tensela, Grasshopper and Dynamo
8. **More**
    1. Common stumbles

Getting started

Welcome to the manual

This manual is written for architects, not for programmers. Every page tries to answer first **what for**, then **how**, and only at the end **how it works inside**.

What is Tensela
Start here: what it does and what it is not.

Your first project
A facade with sun fins in about ten minutes.

Coming from other programs
Grasshopper, Dynamo or Geometry Nodes: what changes.

Download this manual
As a web page to keep, or as a PDF.

How the manual is organised

Getting started
What Tensela is, how to install it and a first project.

Key ideas
Six ideas that explain how everything else works.

The window
The canvas, the 3D view, the panels, exporting and the preferences.

Recipes
How to get a result: a facade, a roof, a structure, a cut file.

Nodes
Every node, in the same order: what for, how, what it does, and tips.

Theory
How it works inside, for whoever wants to know more.

What is Tensela

Tensela is a program for **parametric design, made for architects**.

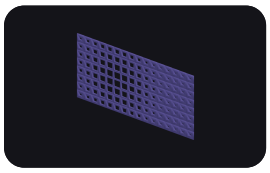
Instead of drawing every piece, you describe how the design is built: *a facade 20 m wide, divided into bays of 1 m, with a sun fin in each bay that turns towards the sun*. Each step is a **node**, a small card, and you join the nodes with **cables**. Change one number, the width of the bays or the angle of the fins, and the whole design updates in front of you.

It works like a **recipe**: the values are the ingredients, the nodes are the steps, and the form is the dish. Change an ingredient and the same recipe gives you another dish. That is what an algorithm is, and in Tensela you see it instead of writing it.

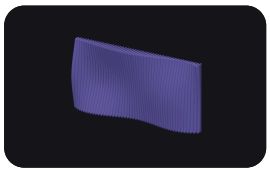
What you can do with it

- **Facades and sun fins:** panels, fins that open and close by distance or by sun, skins of rectangles, hexagons or Voronoi cells.
- **Roofs and surfaces:** surfaces through curves or points, hyperbolic paraboloids, ribs, grids and diagrids on a surface.
- **Structures:** hanging cables and membranes that find their own shape, space frames, a first check of bars.
- **Sun and shadow:** the sun for any place and date, shadows, sun hours on a facade, daylight.
- **Fabrication:** sections and waffle structures, pieces laid flat on sheets with their numbers, a file ready for the laser cutter, a check before 3D printing.
- **Export:** OBJ, STL and 3MF for 3D, DXF, SVG and PDF for cutting and drawing, PNG for images.

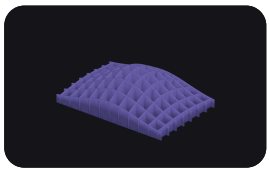
Eight results made with Tensela, each one a recipe of this manual to make it step by step:



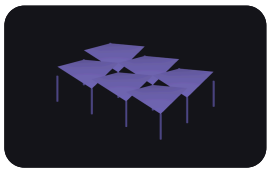
A facade that breathes



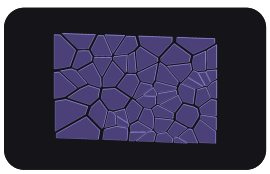
A rib wall



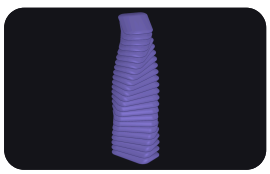
A waffle model ready to cut



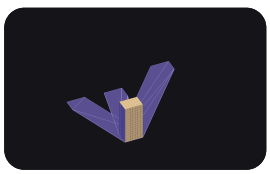
A roof of hyperbolic paraboloids



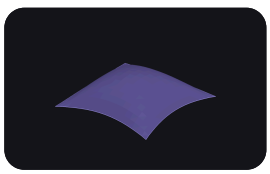
A Voronoi skin



A tower that turns



A sun study



A vault that finds its own shape

What Tensela is not

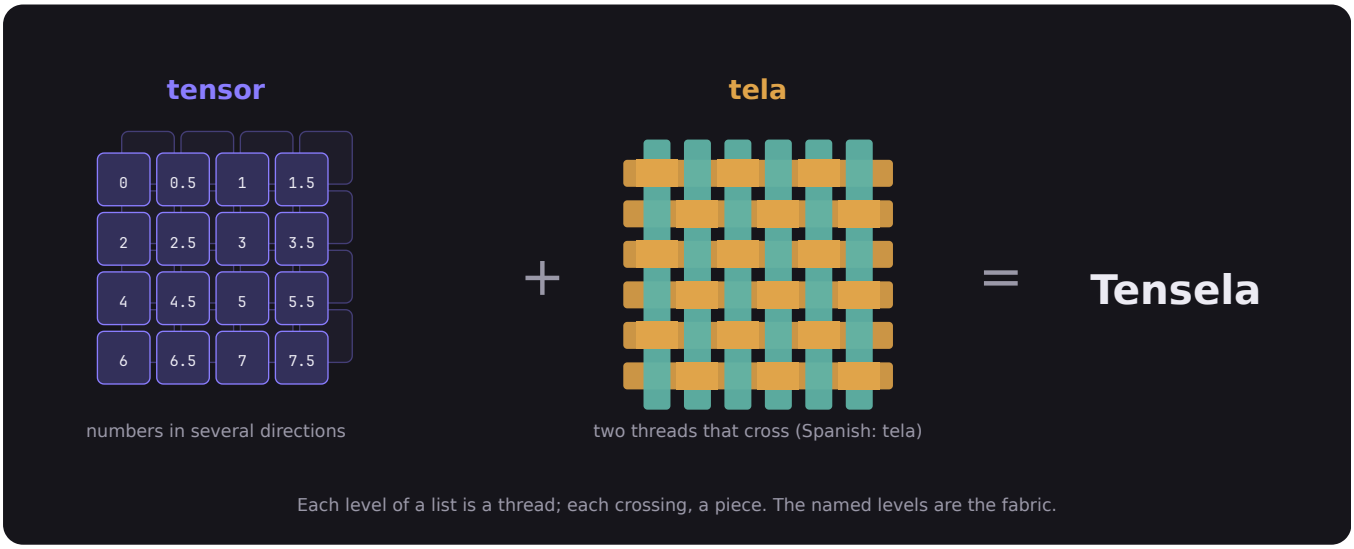
To avoid misunderstandings from the start:

Tensela is not...	Because...
A drawing program	You can draw curves and points in the view, but its strength is in rules, not in drawing every line by hand.
A plugin	It works on its own. You do not need any other program installed.
BIM	It has no walls, doors or quantity lists. It makes geometry that you then take to your BIM or CAD program.
A renderer	For photographic images, export the model to the renderer you already use.

Who it is for

Architects and students who want to design with rules **without learning to program first**, with a gentle learning curve. If you already use a visual programming tool, read Coming from other programs: most of what you know still applies.

Where the name comes from



Tensela joins two words. A **tensor** is, in mathematics, a block of numbers with several directions: that is how Tensela keeps your lists. *Tela* is Spanish for **fabric**, threads that cross: each list is a thread, and each crossing is a piece of your design. More in How Tensela thinks.

Five words you will see everywhere

Word	What it means
Node	A card that does one thing: a rectangle, a grid, a distance, a rotation.
Cable	The line that takes what one node gives into another node.
Input	A value a node needs, like the width of a rectangle. You type it or bring it by cable.
List	Several values at once: 20 heights, 300 panels.
Level	The name of a list: *floor*, *bay*, *panel*. It tells Tensela how to combine lists. More about it.

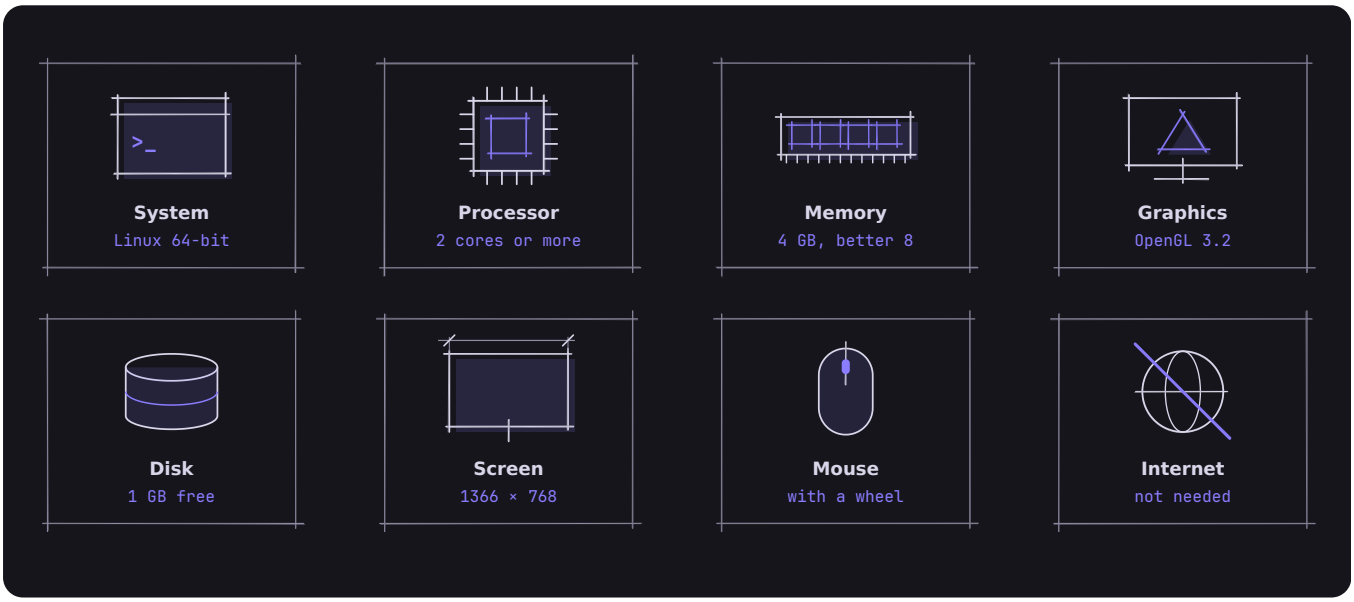
Where it runs, and what it costs

- **Linux**, on a 64-bit PC; a modest computer is enough. Windows and macOS are not available yet. What you need.
- **Free:** it costs nothing and it is free software. What you make with it is yours.
- **Offline:** it works without internet, with no account, and Tensela never sends your files anywhere. Free, offline and yours.

What you need

> **Note**
>
> The first version of Tensela is on its way. The figures on this page were measured with the version in development; the exact size of the file will be on the Download page.

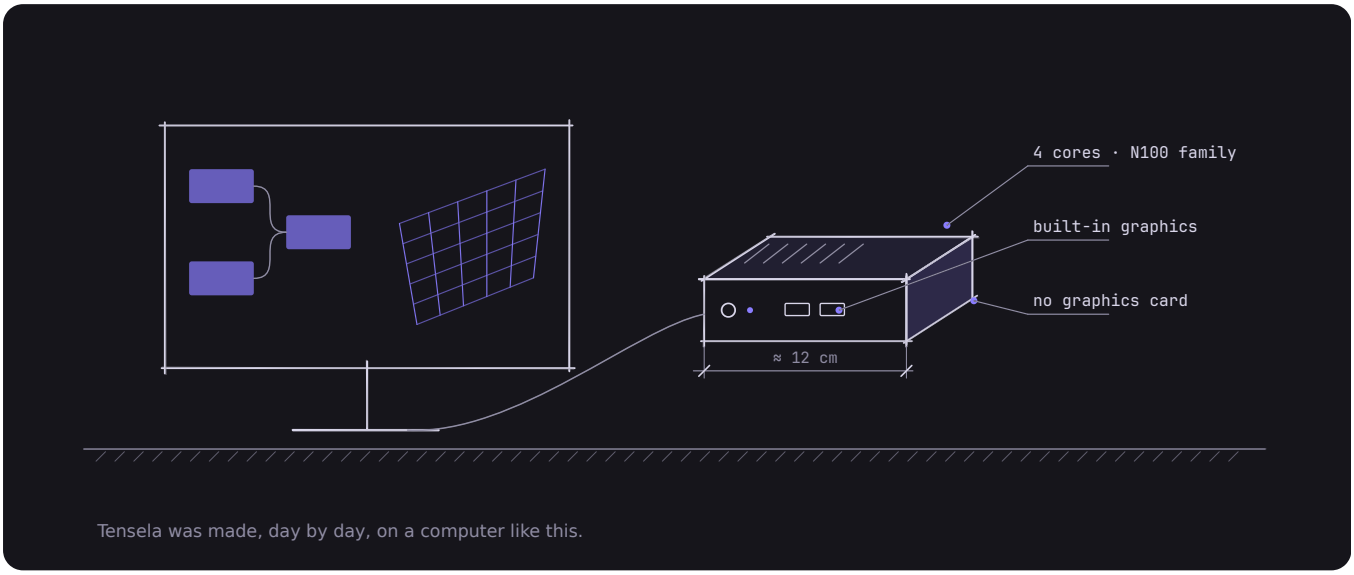
In short



	Enough	Better
System	Linux, 64-bit, on a PC (x86-64): Ubuntu 22.04, Debian 12, Fedora 35, Linux Mint 21 or newer	the latest version of your distribution
Processor	2 cores	4 cores or more
Memory (RAM)	4 GB	8 GB or more
Graphics	OpenGL 3.2: the graphics built into the processor are enough	any graphics card
Disk	1 GB free	
Screen	1366 × 768	1920 × 1080 or more
Mouse	with a wheel	three buttons (a wheel you can click)
Internet	not needed	

Measured on a modest computer

Tensela was developed and tested every day on a modest computer, on purpose: a small desktop computer with a 4-core Intel processor of the N100 family, the graphics built into the processor and **no graphics card**.



Tensela was made, day by day, on a computer like this.

> **If it runs well here, it should run well on an ordinary computer that meets the system above.**

What it used on that computer:

	Memory	Time
The window open, empty	about 450 MB	opens in a few seconds
The window with a project, from the first project to the largest examples	450 to 520 MB	
The heaviest calculations of the examples (topology optimisation of a bridge or a tree)	under 200 MB more	4 to 7 seconds

So a computer with 4 GB of memory is enough for Tensela and the system; with 8 GB you can keep a browser and other programs open beside it.

The system

- **Linux on a PC (x86-64).** Tensela needs the system libraries of Ubuntu 22.04, Debian 12, Fedora 35, Linux Mint 21 or any newer distribution (glibc 2.34). Computers with Arm processors (a Raspberry Pi, a Mac with an Apple processor) are not supported yet.

- **Wayland and X11** both work. On GNOME with Wayland, Tensela goes through X11, so its panels can be dragged out of the window; the small libraries it needs for that come inside Tensela.
- **In a virtual machine**, turn on its 3D acceleration.
- **Windows and macOS: not yet.** Tensela was born for Linux; versions for Windows and macOS are planned for when there are resources to do them well (who makes Tensela).

The graphics

The 3D view needs **OpenGL 3.2 or newer**, which almost all graphics of the last ten years have, including those built into the processor. To check yours, in a terminal: `glxinfo -B` and look at *OpenGL core profile version* (on Ubuntu and Debian `glxinfo` comes in the `mesa-utils` package). The drivers that come with your distribution are enough.

Internet

Tensela does not need internet to work: everything happens on your computer. More in Free, offline and yours.

Install

> **Note**
>
> The first version of Tensela is on its way. These are the steps it will have.

Tensela comes as a single file, an **AppImage**: usually nothing to install and nothing to set up. There is also a portable folder. Before you start, check What you need: a 64-bit Linux PC with 4 GB of memory is enough.

1. **Download** the file `Tensela-….AppImage` from the Download page.
2. **Allow it to run:** right-click the file, open **Properties**, and turn on **Allow executing file as program**. From a terminal: `chmod +x Tensela-*.AppImage`.
3. **Open it** with a double-click. The first time, Tensela offers to put itself in the **applications menu** of your desktop, with its icon, and to open `.tela` files with a double-click. If you say no, it is in **Preferences → General → Applications menu**.

The portable folder

For whoever cannot use an AppImage, or wants it in `/opt`: the same Tensela as a folder, in `Tensela-….tar.gz`.

1. **Unpack it** wherever you like (right-click → Extract here, or `tar -xzf Tensela-*.tar.gz`).
2. **Open it** with `./tensela`, inside the folder.
3. To have it in the menu: **Preferences → General → Applications menu**.

Check the download

The Download page says the **fingerprint** (SHA-256) of each file, and offers the file `SHA256SUMS`. With both in the same folder, in a terminal:

```
sha256sum -c SHA256SUMS --ignore-missing
```

If it says **OK**, the file is the one that was published.

The fingerprints say the file arrived whole. To know **who published them**, `SHA256SUMS` comes with a signature, `SHA256SUMS.minisig`, made with a key that only Tensela has. Check it with minisign (`sudo apt install minisign`), with both files in the same folder. The Download page has the command with the public key already in it, ready to copy:

```
minisign -Vm SHA256SUMS -P <the public key of Tensela>
```

If it says **Signature and comment signature verified**, the fingerprints are the ones Tensela published. The same public key is in the repository, in `tensela.pub`.

Your files

- Your projects end in `.tela`. Open them from **File → Open**, or with a double-click once Tensela is in your menu.
- What you design with Tensela is yours: the program's licence (GPL-3.0, free software) covers the program, not your projects. Explained in plain words in Free, offline and yours.

If it does not open

See Common stumbles → Installing and opening.

Your first project

You will make a **facade of 20 × 12 m**, divided into panels of 1 m, with a **vertical sun fin in each panel**. The fins near a point turn open; the ones far from it stay closed. Seven nodes, no formulas.

Before you start: two gestures

- **Add a node:** double-click on an empty spot of the canvas, or press **Tab**. Type its name and press **Enter**.
- **Connect two nodes:** drag from the small circle on the **right** of a card (what the node gives) to an input of another card.

Step by step

1. **The facade.** Add a **Rectangle**. In its card, write **width** `20`, **height** `12` and choose **plane: elevation**. A rectangle standing up appears in the view. If you do not see it, turn on the eye of the card.

2. **The panels.** Add a **Grid** and connect the Rectangle to its input **rectangle**. Write **cell width** `1` and **cell height** `1`. The cable now reads `row 12 × column 20 = 240`: 240 panels.

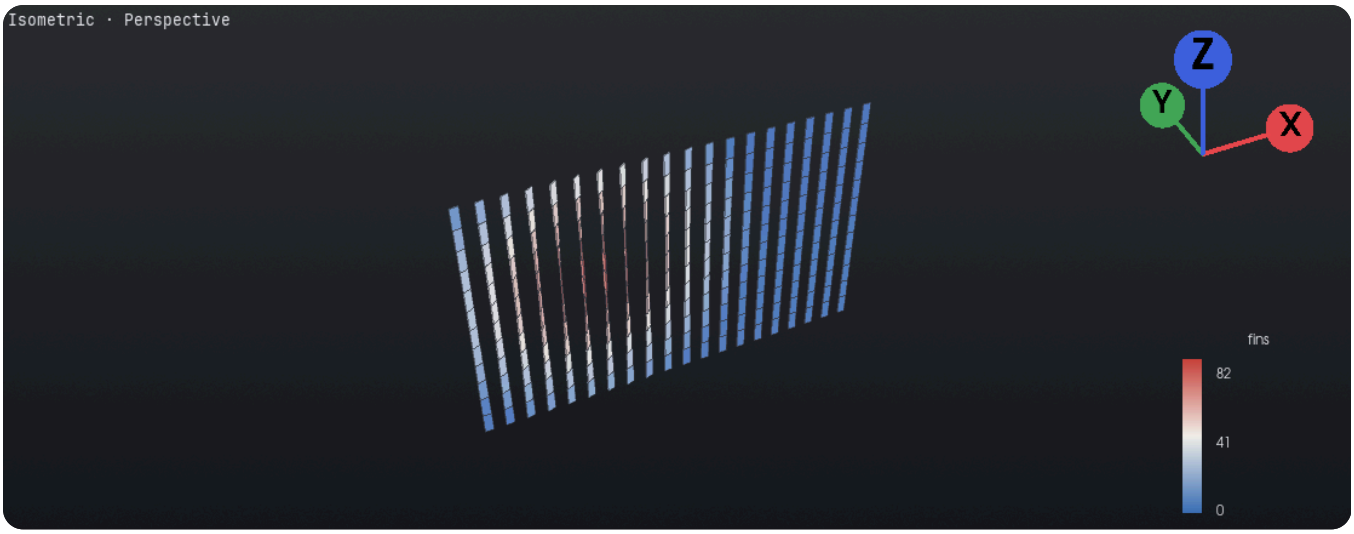
3. **The centre of each panel.** Add a **Centre** and connect the Grid to its input **geometry**. You get 240 points, one in the middle of each panel.

4. **The point that attracts.** Add a **Point** with **x** `6`, **y** `0` and **z** `7`: a spot on the facade, 6 m from the left and 7 m up.

5. **An angle for each panel.** Add an **Attractor**. Connect the Centre to **where** and the Point to **attractors**. Write **reach** `8`, **near** `90` and **far** `0`. Each panel now has an angle: 90° right next to the point, 0° at 8 m or farther.

6. **The fin.** Add a **Box** with **width** `0.4`, **depth** `0.05` and **height** `1`: a thin vertical blade.

7 **A fin in every panel.** Add a **Place**. Connect the Box to **geometry**, the Centre to **at** and the Attractor to **rotation**. 240 fins appear: open around the point, closed far from it.



*Here the fins are painted by their angle (**Colour → by a value**), to see which ones open.*

Now play with it

This is where parametric design pays off. Change any number and the whole facade answers:

- Move the **Point**: change its **x** or **z** and watch the opening travel across the facade.
- Change the **reach** of the Attractor: a small reach opens only a few fins, a large one opens most of them.
- Swap **near** and **far**: the fins close around the point and open everywhere else.
- Change the **cell width** of the Grid to `0.5`: twice as many fins, without touching anything else.

Tips

- **See only what matters.** Turn off the eye of the Rectangle, the Grid and the Centre, so the view shows only the fins.
- **Fins in front of the facade.** The fins sit on the plane of the facade. To bring them 30 cm forward, put a **Move** between the Centre and the Place with **vector** `0; -0.3; 0`.
- **Save** your project with **File → Save**. Tensela projects end in `.tela`.

What you just learnt

- A design is a chain of small steps, each one a node. Nodes and cables
- The Grid made a list of 240 panels, and every node after it worked on all 240 at once. Lists with names
- The Attractor made each panel different on purpose. Conditions and attractors

Coming from other programs

If you already design with nodes, you will feel at home quickly: a canvas, cards, cables, numbers that change the model live. This page tells you what changes, so you do not look for things that work differently here.

The biggest difference: lists with names

In other programs, the data is **arranged** before combining it: trees with numbered branches, lists inside lists, choosing how two lists are matched. In Tensela every list carries a **name**, its level: *floor*, *bay*, *panel*. When a node gets two lists, it looks at their names:

- **same name:** they go one with one, the first with the first, the second with the second;
- **different names:** every one is combined with every other one.

There are no branches to flatten, graft or transpose. Read Lists with names for the full idea, with examples, and Lists in Tensela, Grasshopper and Dynamo for the same tasks done in the three tools.

If you come from Grasshopper

Grasshopper is a visual programming tool inside Rhino. Tensela works on its own, without Rhino.

In Grasshopper	In Tensela
Number Slider	**Number** (a slider on the card)
Expression	**Formula**
Python or C# script	**Code** (Python, running apart and with limits)
Panel	**Info**
Divide Curve	**Points on curve**
Nurbs Curve	**NURBS Curve**
Series	**Series**
Graft, Cross Reference	nothing: lists with different names are combined by themselves
Flip Matrix	nothing: levels have names, not an order
Simplify, Shift Paths	nothing: there are no paths
Shift List, Relative Item	**Reorder**, shift, on one level
Cull Pattern, Cull Nth	**Filter**, pattern or one every *n*
List Item, Sub List	**Pick**
Partition	**Split**
Merge	**Merge lists** (one after the other)
Entwine	**Merge** (side by side, with a new level)
Flatten	**Join levels**

If you come from Dynamo

Dynamo is a visual programming tool for Revit. It works with BIM elements: walls, floors, families. Tensela is not BIM: it makes the geometry, and you take it to your BIM program as OBJ or DXF.

In Dynamo	In Tensela
Lacing: shortest	lists with the same name
Lacing: cross product, replication guides	lists with different names
List levels (`@L1` , `@L2`)	the level, chosen by its name
List.Transpose	nothing: levels have names, not an order
List.ShiftIndices	**Reorder**, shift
List.FilterByBoolMask, List.TakeEveryNthItem	**Filter**
List.GetItemAtIndex, List.Slice	**Pick**
List.Chop	**Split**
List.Join	**Merge lists**
List.Create	**Merge**
List.Flatten	**Join levels**

If you come from Geometry Nodes

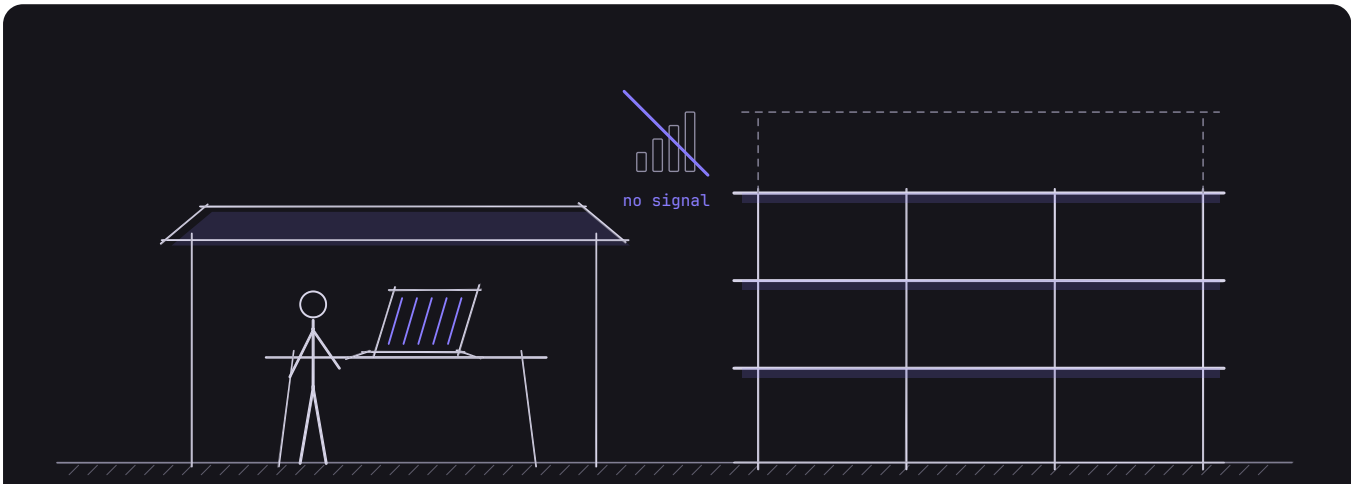
Geometry Nodes is part of Blender, a complete 3D program. Its nodes build any kind of 3D content, from scenes to effects. Tensela is only for design with rules, and it speaks the language of architecture: facades, floors, bays, sun fins, sheets for cutting. You can send what you make to Blender as OBJ.

Rhino and Grasshopper are trademarks of Robert McNeel & Associates. Dynamo and Revit are trademarks of Autodesk. Blender is a trademark of the Blender Foundation. Tensela is not associated with them.

Free, offline and yours

Three things worth knowing before you start:

- **Free:** you can use Tensela for anything, at no cost.
- **Offline:** it works entirely on your computer.
- **Yours:** what you make with it belongs to you.



On site, without signal: Tensela keeps working, because everything happens on your computer.

Free: what the licence lets you do

Tensela is **free software**: it costs nothing, and its licence is the GNU General Public License, version 3 or later (GPL-3.0), the licence of many well-known programs of Linux. In plain words:

You can	What it means
Use it for anything	Your studies, your professional work, paid commissions, your classes. Without paying, without an account, with no time limit.
Install it on all your computers	And give copies to whoever you want.
See how it is made	Its source code is public, so anyone can check what it does.
Change it	And share your changes, under the same licence.

The one condition: if you share Tensela, or a version you changed, you share its source code too, with the same licence. That way Tensela, and every version made from it, stays open for everyone.

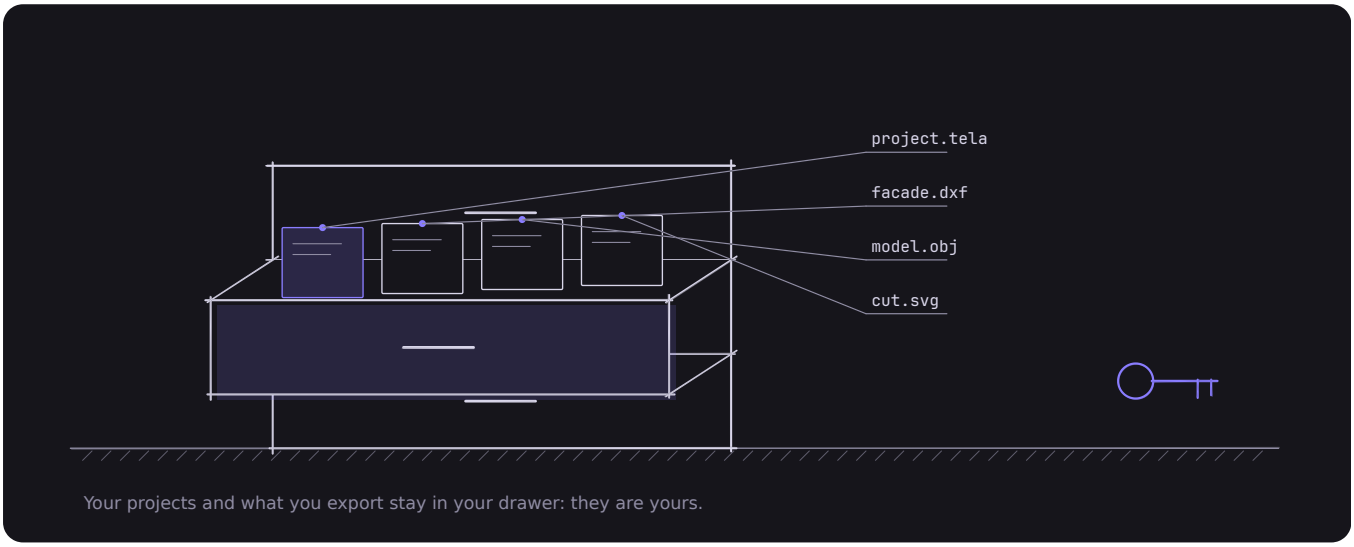
Like all free software, Tensela comes **without warranty**: it is offered as it is. Check the results that carry responsibility with a professional; Bar analysis, for instance, is a first check, not a structural

calculation. The licence, the warranty and privacy, all on one page: Licence, warranty and privacy.

Your projects are yours

> **The licence covers the program, not what you make with it.**

- Your **projects** (`.tela`), the **files you export** (OBJ, DXF, SVG, PDF…) and your **images** are yours.
- Use them, sell them, publish them as you like.
- You do not need to mention Tensela, although it is always appreciated.



Your projects and what you export stay in your drawer: they are yours.

The manual and the examples

- **This manual** is under the Creative Commons Attribution-ShareAlike 4.0 licence (CC BY-SA 4.0): copy it, translate it, use it in your classes, saying it comes from Tensela and sharing your changes with the same licence.
- **The example projects** (the recipes and the examples that come with Tensela) are CC0: use them as you like, with no conditions, not even saying where they come from.
- The full texts of the three licences come with Tensela, in its `LICENSE` files.

The name and the logo

The name *Tensela* and its logo are not under the GPL. Anyone can make their own version of the program, but with another name and another logo, so nobody confuses it with the original. The details are in TRADEMARKS.md.

Offline: everything on your computer

> **Tensela never sends your projects anywhere.**

- **Tensela itself does not connect to the internet.** No account, no sign-in, no updates in the background, no advertising, no statistics of how you use it.
- **The sun** is calculated with astronomical formulas and a list of cities inside the program, or the latitude and longitude you type. No weather file to download.
- **The Code node** is made not to reach the network: Tensela checks the code before running it, and stops it if it tries to open files, reach the network or start other programs.
- **Your files stay where you save them.** Tensela does not upload them anywhere. (An extension can, only if its card says so in amber.)
- **You only need internet** to download Tensela, this manual and the new versions, when you decide.

Why it matters:

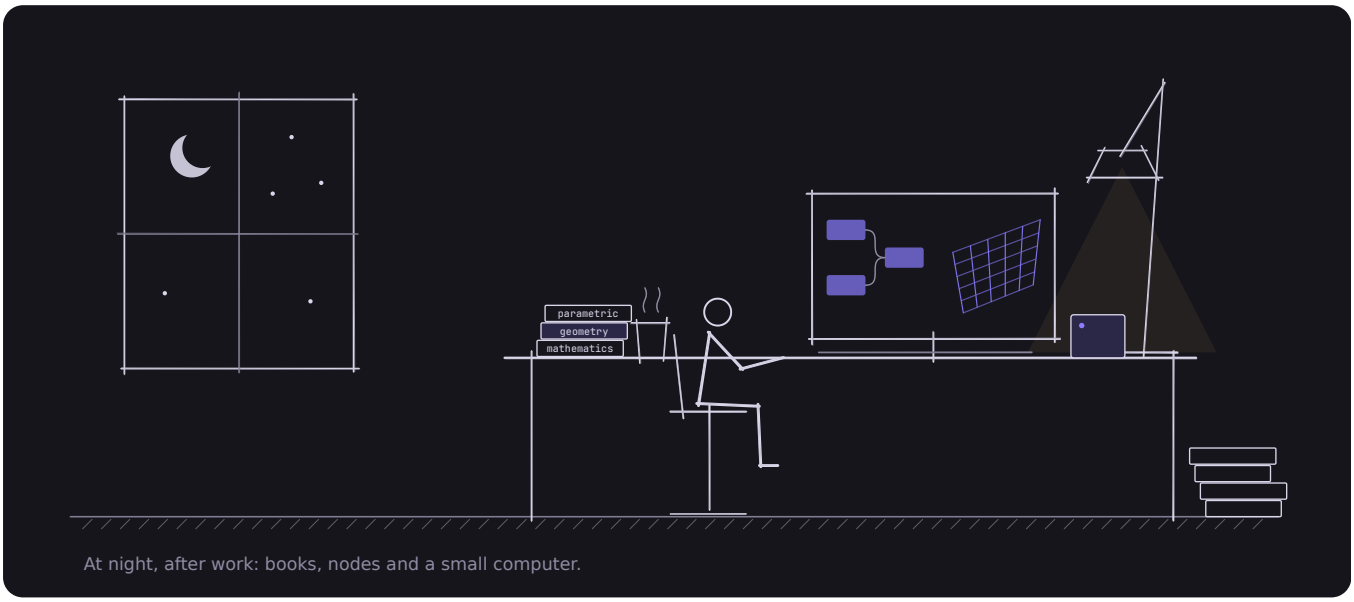
- It works **in a site office without signal**, on a trip, or in a classroom with a poor connection.
- Your projects **stay with you**: nobody else sees them.
- Once it is on your computer, it **does not depend on any server** to keep working.

Free of charge, and supported by donations

Tensela is free, and its licence makes sure nobody can take away the version you have. It is sustained by the people who find it useful: if it helps you, you can support it with a donation.

Who makes Tensela

I am **Victor Crespo**, an architect and a designer who specialises in digital products, and I love parametric design. Tensela is a personal project: I develop it with artificial intelligence as my development team.



At night, after work: books, nodes and a small computer.

Where the idea comes from

Since **2007** I have been trying to work as an architect on Linux. At first it was very hard: there was almost nothing for our trade. Inkscape was in its first versions and Blender was not what it is today. I have worked on Linux ever since: I started with Ubuntu, and today I move between Ubuntu and Debian. I like how it is thought out, how folders and packages are ordered, and the community around it.

I say this without zeal: Linux does not solve everything. The big design programs, those of Adobe and Autodesk, were and still are very good. Nearly all of them run on Windows, many also on Mac, and they need a powerful machine. Architects and designers who work professionally on Linux are still few.

What I love most is **parametric design**: visual algorithms, the idea that an algorithm is a **recipe**, a series of steps you can change to get another result. I wanted to do it on Linux, and I could not:

- **Rhino**, which I use, only ran on Linux inside a virtual machine with Windows, with almost no graphics acceleration.
- **Dynamo** was made to be used with Revit, which does not run on Linux.
- **Grasshopper** is a great program, but I got lost in it: for what I wanted, I struggled to find the way. The same happened to me with Blender’s nodes for architecture.

For me, it was very hard to move forward with those tools. And for an architect a facade, a structure or a form is not an abstract number: it has a height, a thickness, a geometry. Something thought out that way was missing, and something that ran on Linux.

An idea that comes from books

The idea of Tensela was not born with artificial intelligence. It started long before, when it did not exist or I had no access to it: in books. I am a reader, and I still keep many, on architecture, on parametric design and on mathematics. Tensela was tested with exercises of its own, born from what I learned in books more than ten years old, written when there were no AI assistants. Architecture has a lot of mathematics in it, and I like mathematics.

Behind it there are years of experience: what I read, what I tried, what I learned along the way. I still have a lot to learn.

> **Artificial intelligence does not do it all by itself. It helps to build what you already understand, and what you are willing to check again and again.**

How it was made

I am not a programmer: my programming knowledge is basic, not enough to do this alone. Artificial intelligence opened that door for me, and I used it as a **development team**.

I know that many new programs made with AI are appearing today, and that it is easy to think this is one more. So I say it plainly: Tensela **did not come out of a single request**. It is months of iteration: testing each node with real exercises of facades, roofs, structures and fabrication, checking the result, correcting it and testing again. The mathematical base (the surfaces, the form finding, the sun, the optimisation) was built and checked step by step. Every stumble along the way became a change in the program or a tip in this manual.

It is many early mornings, nights and weekends, outside my office job and my time with my family. My own time and money went into it, without big resources.

Tensela was born for Linux because that is what was missing. Later I want to bring it to Windows and macOS, when there are resources to do it well.

Small on purpose

Next to great programs such as Rhino and Grasshopper, Dynamo or Blender’s nodes, Tensela is very small. And partly it is small on purpose: it is meant as the synthesis we look for in architecture, what is really needed.

It is like a camera full of options, or a device with hundreds of buttons: we hardly ever use everything. Tensela prefers fewer nodes that are well understood.

What I think is worth something is **the way of thinking** it offers: each step is a node you can understand, lists have names, and what you draw has the measures of architecture. If that way of thinking helps you, even if you later take it to another program, Tensela has done its job. It will grow, without losing that.

The story of how Tensela was designed is told as a case study on 3dvic.com.

Support Tensela

Tensela takes many hours of work, and more are coming: new nodes, recipes, videos. If it helps you in your work or your studies, you can support it with a donation.

Key ideas

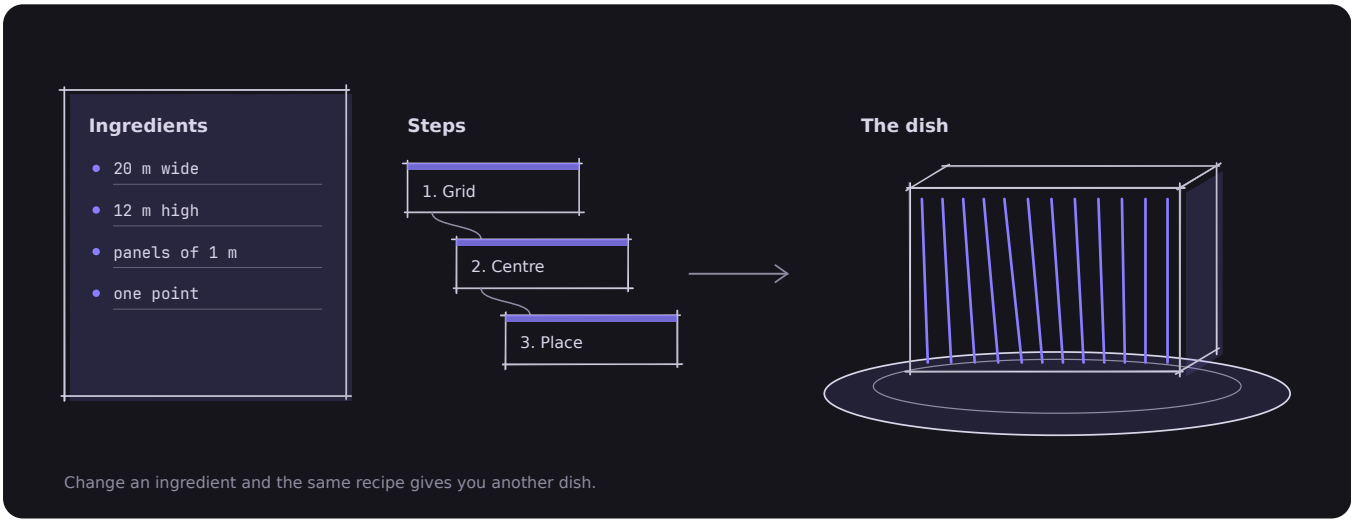
Nodes and cables

What for

To describe **how** your design is made instead of drawing the result. Once the steps are there, you change any of them, the width of the facade, the number of floors, the angle of a fin, and everything that depends on it is calculated again.

An algorithm is a recipe

A recipe has **ingredients**, **steps** in an order, and a **dish** at the end. A Tensela project is the same:



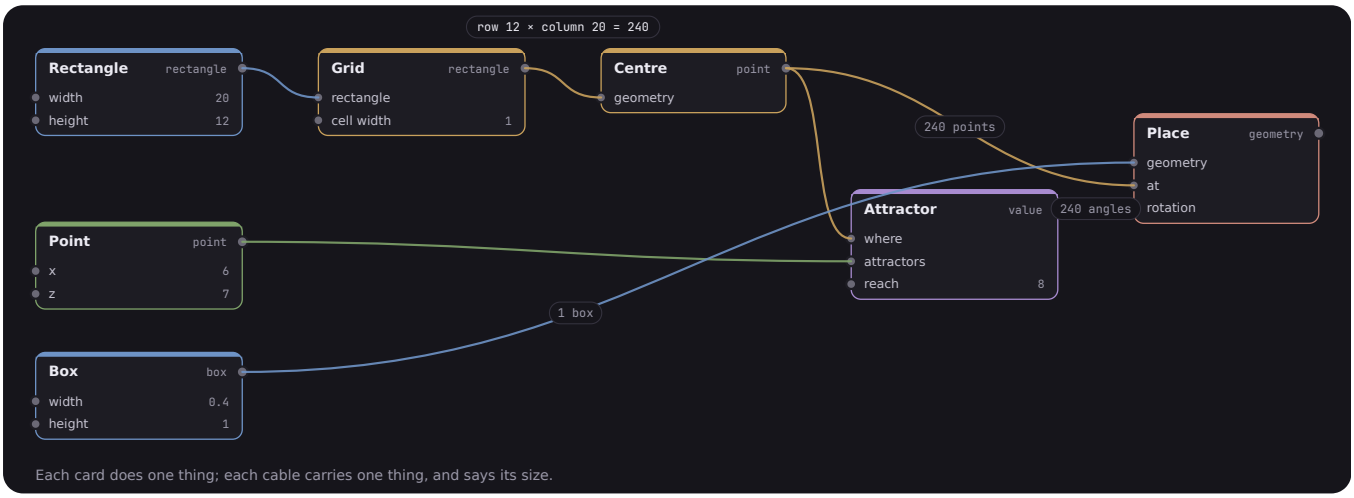
In a recipe	In Tensela	In the facade with sun fins
The ingredients	the values you type	20 m wide, 12 m high, panels of 1 m
The steps	the nodes	a rectangle, a grid, a fin in each panel
The order of the steps	the cables	the grid needs the rectangle first
The dish	the form in the 3D view	the facade with its 240 fins

Change an ingredient, a taller facade or narrower panels, and the same recipe gives you another dish. You do not draw the facade again: you change the recipe. And like a recipe, a project can be shared: whoever opens it gets the same steps and can try their own ingredients. That is why the projects of this manual are called recipes.

How

- **Add a node:** double-click on an empty spot of the canvas, or press **Tab**, and type. You can type the name of the node (*Prism*) or what you want to make (*volume*, *building*): the search finds it either way.
- **Connect:** drag from the small circle on the **right** of a card, which is what the node gives, to an input of another card.
- **Give values:** each input of a card can be **typed** (a number, a point, a list) or **brought by a cable** from another node. When a cable arrives, the field takes its value from it.

What it does



- **A cable carries one thing** from one node to another: a rectangle, 240 points, a list of angles. On the cable you read its size, for example `row 12 × column 20 = 240`.
- **Change a value and everything after it updates**, in order, in the view.
- **Each family of nodes has its colour**, the same on the strip of the card, on its icon and in the search:

Family	Colour	Examples
Values	green	Number, Series, Formula, Yes or no
Shapes	blue	Point, Box, Prism, Rectangle
Curves	teal	Circle, Arc, NURBS Curve, Line
Surfaces	petrol blue	Loft, Extrude, Ribs, Surface from points
Divide	ochre	Grid, Faces, Vertices, Points on curve
Measure	violet	Area, Distance, Attractor, Filter
Transform	terracotta	Move, Rotate, Scale, Place

Tips

- **A card with a problem says it in words**: what is missing or what does not fit, and usually how to fix it.
- **Numbers:** decimals can be written with a point or a comma, `3.5` or `3,5`. Several values are always separated by a semicolon: `1.5; 2; 3`. A point is written `0 0 0`, `8 0 2.5` or `(0; 0; 1.5)`.
- **Inside a card field**, Tab jumps to the next value and Delete erases letters, not the node.

Lists with names

This is the most important idea in Tensela. Once it clicks, the rest becomes much clearer.

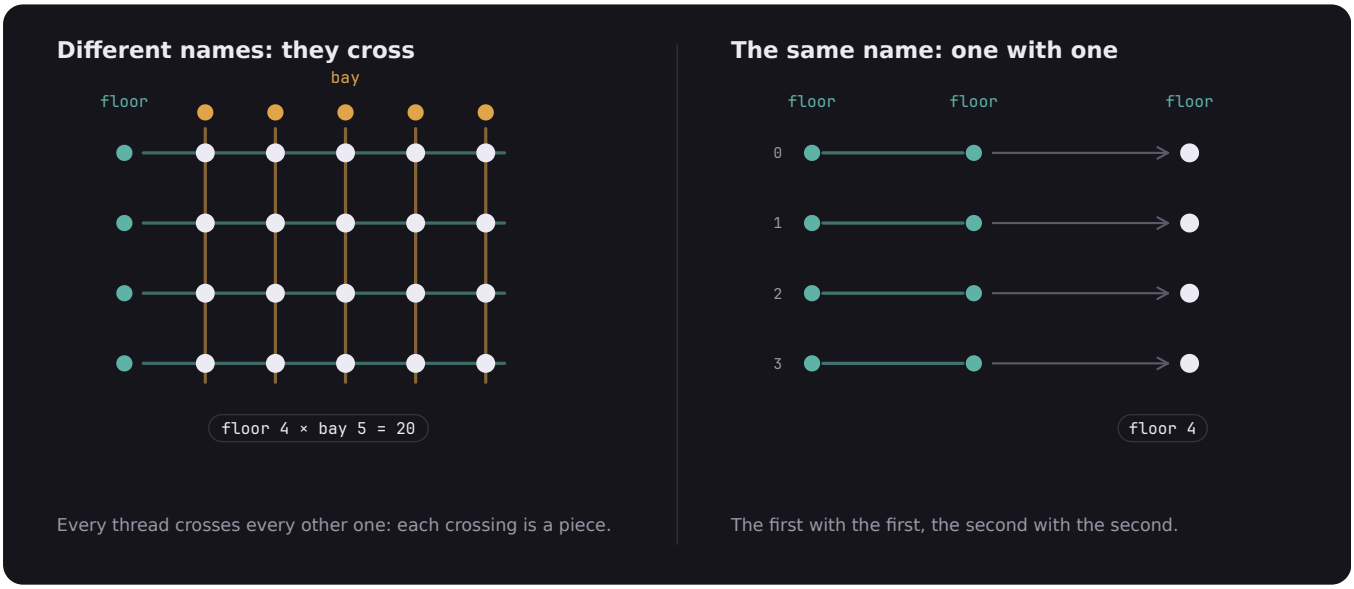
What for

To work with **many things at once**, 16 floors, 20 bays, 320 panels, without arranging the data by hand before combining it.

The idea: threads and crossings

Every list in Tensela carries a **name**, its **level**: *floor*, *bay*, *panel*, *rib*. When a node receives two lists, it looks at their names:

- **Different names: every one with every other one.** 16 floors and 20 bays give 320 panels, like the threads of a fabric: every vertical thread crosses every horizontal one, and each crossing is a piece.
- **The same name: one with one.** The first with the first, the second with the second. Two lists of 16 floors give 16 results, not 256.



That is all. There is nothing to flatten, graft or transpose.

How

- Nodes that make lists, such as **Series**, **Grid**, **Points on curve** or **Split**, have a field to name the level. A Series of 60 numbers named `rib` means *these 60 values are 60 ribs*.

- **The name is free.** What matters is to write it **the same** where you want lists to pair one with one, and **different** where you want them to cross.
- **Tip:** use the singular of what you count, *rib*, *floor*, *panel*, because it reads naturally: *rib 3*.

What it does

- **The cable shows the shape:** `rib 60 × height 60 = 3 600`.
- **Levels have no order.** `floor × bay` and `bay × floor` are the same thing.
- **Operations work on a level, by its name:** reverse the floors, keep one bay out of two, shift the floors one up, split into groups of four. **Reorder**, **Filter**, **Pick** and **Split** ask which level, by name.
- **If two lists with the same name have different sizes**, a notice says so instead of guessing.

An example to try

Make two **Series**: one of 5 numbers named `x` and one of 5 named `y`, and connect them to a **Point** (one to x, the other to y). You get 25 points in a square: the names are different, so they cross. Now name both `x`: you get 5 points on a diagonal, because they pair one with one.

To go further

How a list with levels is kept and combined, with code: Named lists, inside. The same tasks in Grasshopper and Dynamo: Lists in Tensela, Grasshopper and Dynamo.

Two kinds of nodes

What for

To know at a glance where to look for a result: in the **3D view** or **on the card**.

The two kinds



	Make geometry	Only calculate
Examples	Rectangle, Grid, Loft, Place, Ribs	Area, Volume, Distance, Formula, Number, Info
Where you see the result	in the 3D view	on the card
On the card	an eye to show or hide it, colour, opacity, numbering, direction	an ⓘ that unfolds the values on the card

Tips

- **Turn eyes off** to see only what matters: often you only want to see the last node of the chain.
- **Info** shows what any cable carries, as numbers or as a list, without changing anything.
- **Paint by a value:** to colour panels by their area or by the sun they get, use the panel of the geometry (**Colour → by a value**). It is not a separate node.

Draw the form, then cut it

What for

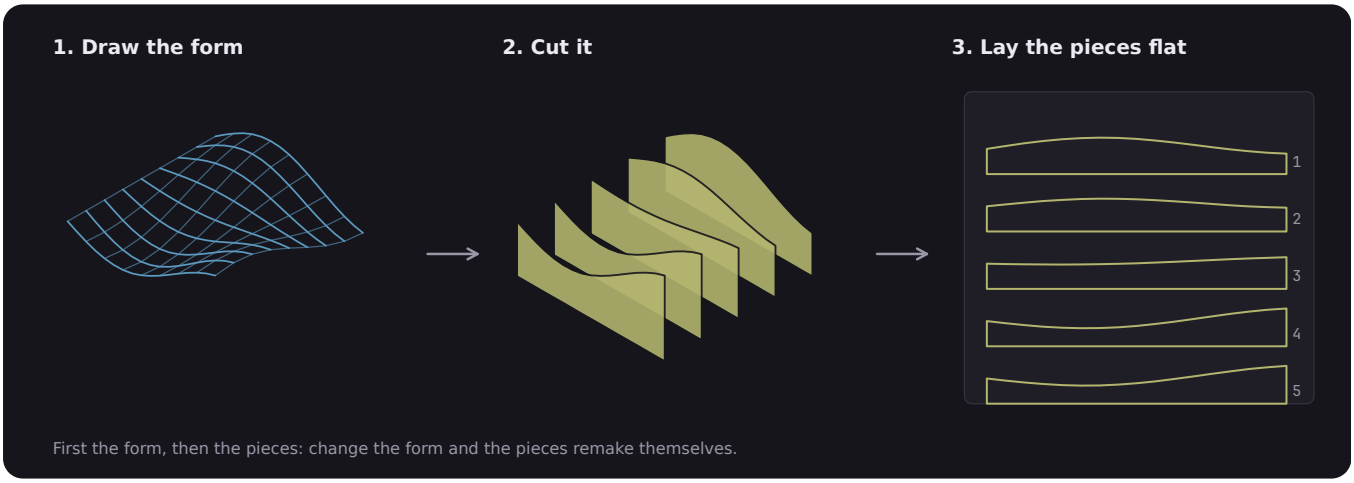
To reach a complex form, a ribbed wall, a waffle roof, a sliced pavilion, with **few nodes and no formulas**.

How

Instead of calculating every point of the form, **draw the form first and cut it afterwards**:

1. **Draw the form** with a few curves and a surface through them, for example a **Loft** through three or four curves.
2. **Cut it** with **Ribs**, which slices it into as many boards as you ask for, with the thickness you give.

A ribbed wall made this way needs about six nodes. Change the curves and all the ribs follow.



First the form, then the pieces: change the form and the pieces remake themselves.

What it does

- **Ribs** cuts the surface with parallel planes and closes each rib against a back plane, the wall behind it. The **back** input is how far that plane is: with `0`, each rib goes from the form to the axis; with more, each rib keeps a straight backing.
- The same idea works for fabrication: **Sections** cuts a body into slices that you then lay flat on sheets.

When to go further

Building the form **point by point**, with Series, a Point and a Formula or a Graph, gives total control over each rib. It is the next step, and it is a very good way to learn Lists with names. Start by drawing

and cutting; go point by point when you need it.

Conditions and attractors

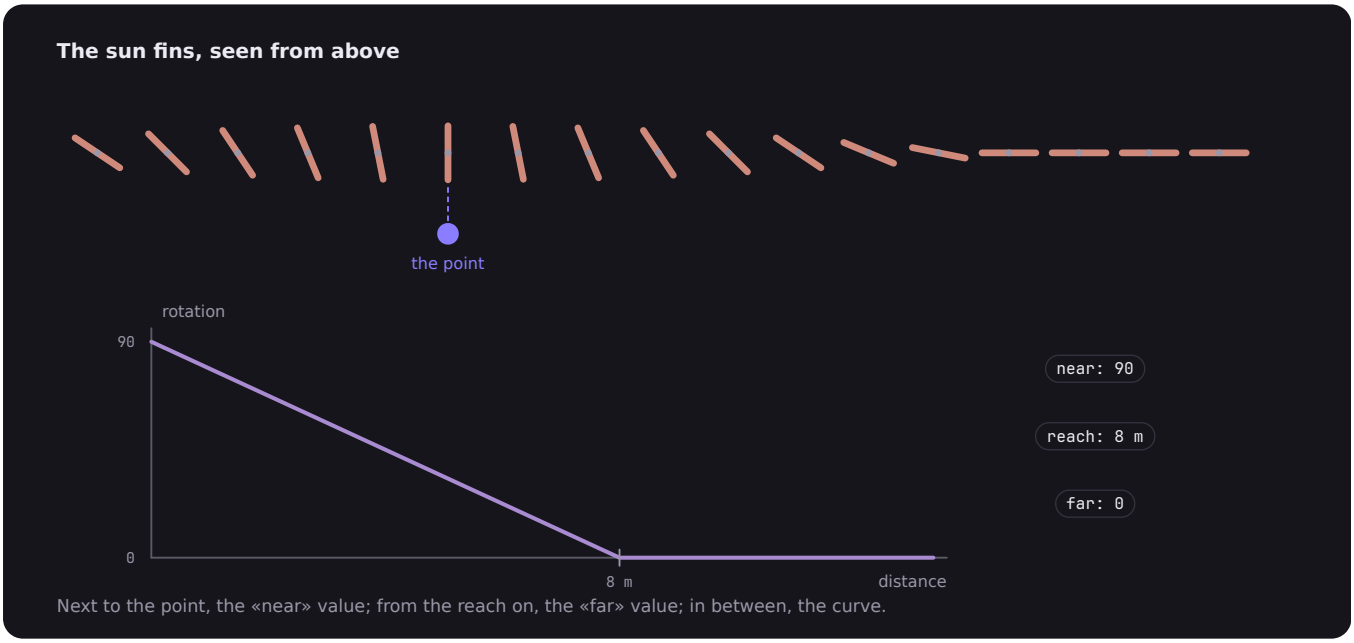
What for

So that each panel, fin or rib is **different on purpose**: larger near an entrance, open towards the sun, closed above a certain floor.

Attractors

An **Attractor** gives each element a value according to how close it is to some points or curves.

- **near:** the value right next to the attractor (for example, `90` degrees).
- **far:** the value at the **reach** or beyond (for example, `0`).
- **reach:** how far the effect goes, in metres. Empty: up to the farthest element.
- **attractors:** one or more points, or curves: a path, a riverbank, the edge of a square.
- **several:** with several attractors, either the closest one rules or their effects add up.
- **curve:** how it goes from near to far: linear, smooth, accelerating or slowing down.



Next to the point, the «near» value; from the reach on, the «far» value; in between, the curve.

Connect what comes out to whatever you want to vary: the **rotation** of a Place, the **factor** of a Scale, the **distance** in a Move.

Tip: to see what the Attractor does inside, build the same thing by hand once: a **Distance** from each element to the point, and a **Remap** from that distance to the range you want, with **limit** on.

Conditions

- **Choose** gives **one result per element**, one thing or the other: an open panel or a closed one.
- **Filter keeps** only the elements that pass, and leaves the others out. With **remove: yes**, the ones that stay are numbered again (0, 1, 2…); with **no**, the others stay in their place, empty.
- **Compare** gives a yes or no per element: *a* is less than, greater than, equal to or different from *b*. **Combine conditions** joins two with *and* or *or*, or turns one around with *not*.

Formulas in your language

A **Formula** works like a spreadsheet cell, and it understands Spanish and English, even mixed:

- `si(a < 7; 3; a < 16; 4,5; 6)` is the same as `if(a < 7, 3, a < 16, 4.5, 6)`.
- `y`, `o`, `no` or `and`, `or`, `not`.
- Spaces do not matter: `a/2` and `a / 2` are the same.

Groups

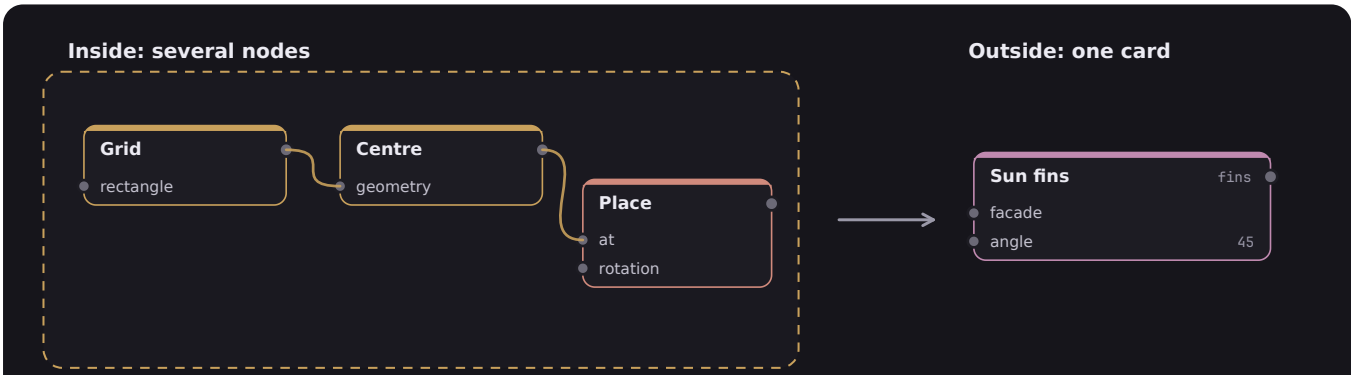
What for

- **Tidy up:** a long chain becomes one card with a clear name, like a chapter.
- **Make your own node:** a group is a **new kind of node**, a *sun fin* or a *panel with frame*, that you can use as many times as you want.
- **Repeat a step:** a step whose result goes back in, several times (a fractal, a curve that subdivides).

How

1. Select the nodes, dragging a rectangle or with Ctrl + click, and press **Ctrl + G**.
2. Give the group a name: that name is the **new kind of node**.
3. **Double-click** the card of the group to go in; **Leave group** goes back up. **Ungroup** (Ctrl + Shift + G) turns the nodes loose again.

What it does



A group keeps a part of the project as a new node, with its own inputs and output.

- **What enters:** the cables that reached the nodes from outside become the inputs of the group.
- **What leaves:** the card of the group gives one result. If two of the nodes are used outside, a **Group result** card appears next to it with the name of the node it replaces. The group is calculated only once for both.
- **Change it once, change it everywhere:** what you change inside a group applies to every node of that kind.
- **Groups inside groups:** inside a group you can group, ungroup and go into another group, as deep as you want.
- **Eyes inside:** inside a group, turning on the eye of a node shows what it calculates with what reaches that card; the outside fades.

Repeat

A group can **repeat** itself: write **repeat** `3` on its card. The first time it receives what is connected; after that, its own result.

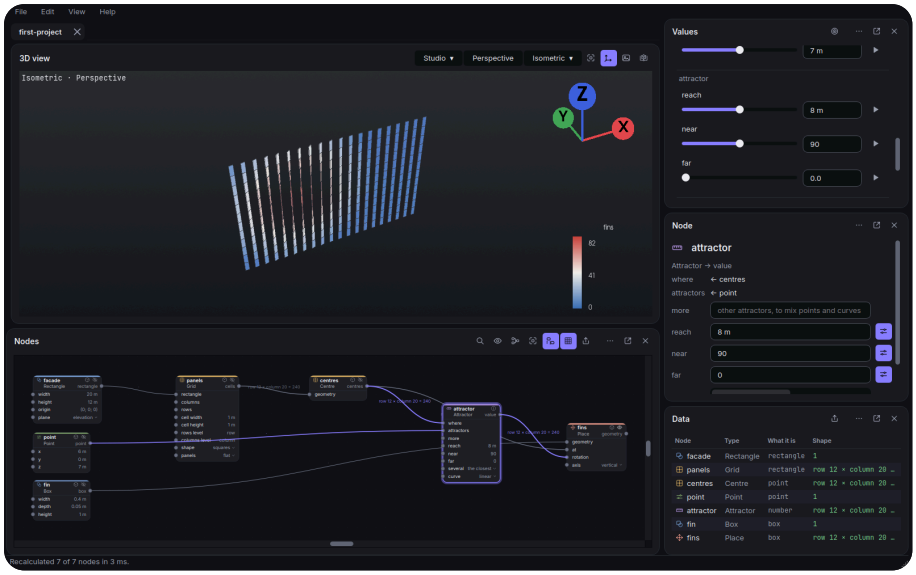
- **back into:** which input the result goes back in through (empty: the first one). The other inputs receive the same thing at every step.
- **keep:** only the last step, or all the steps along a new level, *step*, with the start first (step 0).
- An input of the group named `step` receives the number of each step: 0, 1, 2…

Tip: what comes out has to be able to go back in where the start went in: a line for a line. If not, a notice says at which step it failed.

The window

The window

The window has two main areas and a few panels around them.



Area	What it is
The canvas	where the nodes and cables are. See The canvas.
The 3D view	what the nodes make, updated as you work. See The 3D view.
The panels	*Node* (the settings of the chosen node), *Values* (sliders), *Data*, *Nodes*, *Library* and *Croquis*. See The panels.
The tabs	each open project is a tab, like in a browser: right-click a tab to close, rename, duplicate or find its file.

Arranging the window

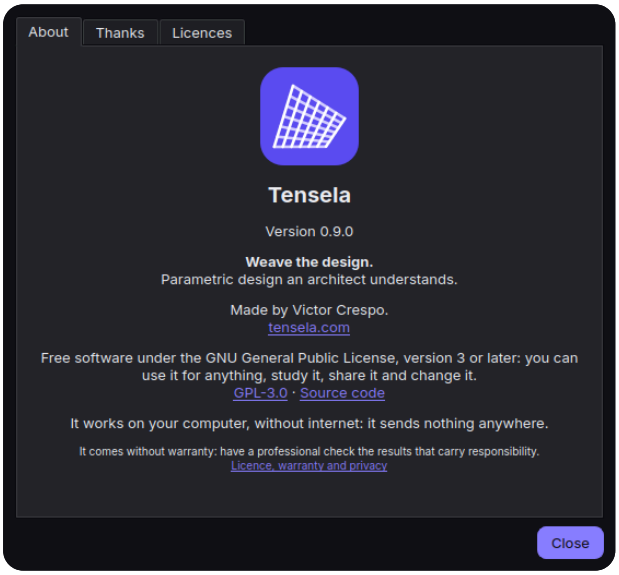
- Panels can be dragged by their title, stacked as tabs, or floated. **View → Panels** shows or hides each one.
- **View → Workspaces** saves the current arrangement with a name, to come back to it; **View → Workspaces → Factory layout** puts everything back as it came.
- **View → Mode** switches between light and dark; **Preferences** has the rest of the appearance. See Preferences.

The menus at a glance

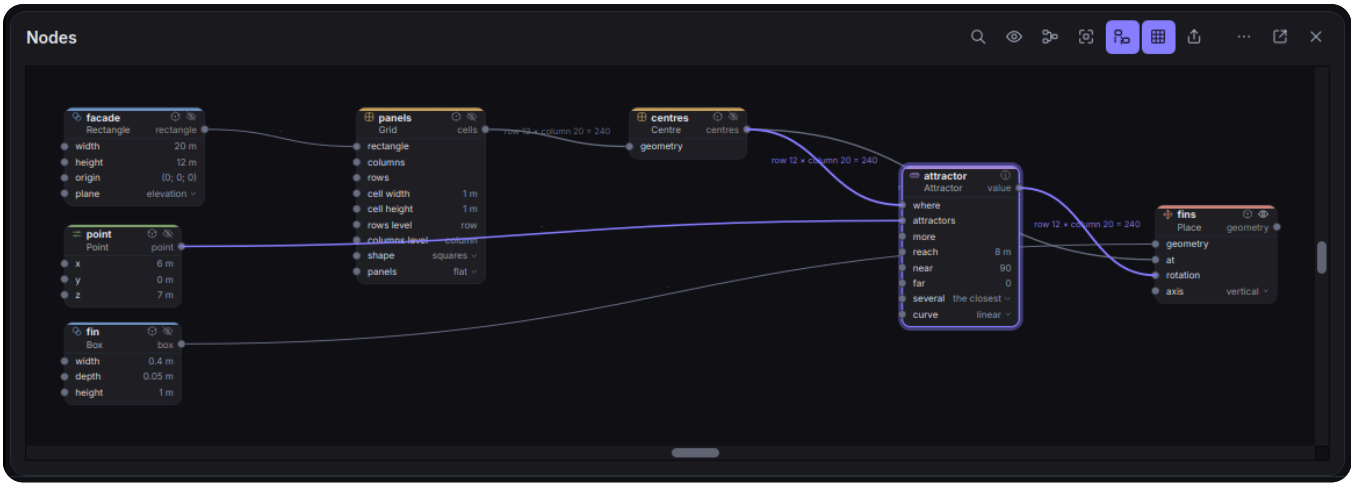
Menu	What is there
File	new, open, recent, save, close; **Export** to images, drawings, 3D, 3D printing and cutting; **Open in** another program
Edit	undo and redo, add a node, find a node, copy and paste, group, **Find the best…**, **Preferences**
View	frame, isolate, show and hide nodes, arrange and align nodes, the grid and guides, colour legends, panels, workspaces, mode
Help	the manual (without internet), help for the chosen node (**F1**), shortcuts, the PDF, community, report a problem, support Tensela, **About Tensela**

Help

Help → Manual opens this manual from your computer, without internet. With a node chosen, **F1** opens it at the part about that node. **Help → About Tensela** shows its version, its licence, the people to thank and the libraries it is built with.



The canvas



Nodes and cables

- **Add a node:** double-click on an empty spot, or **Tab**. Type its name or what you want to make.
- **Repeat the last node: Shift + R** places the last kind of node you used again, where the mouse is.
- **Connect:** drag from the small circle on the right of a card to an input of another card.
- **Choose:** click a card; drag a rectangle, or Ctrl + click, to choose several. With several chosen, the Node panel changes them all at once.

Right-click on a card

Option	What it does
Rename…	gives the node a name of its own: *facade*, *fins*
Show in 3D / **Show only this one**	its eye, or only its eye
Find its best value…	opens *Find the best* for that number. See Find the best
Enter group / **Leave group**	go into a group card, or back up
Save to library…	keeps a group to use it in any project

Arranging

- **Ctrl + L** arranges all the nodes in order, from left to right.
- **View → Organize nodes** aligns and distributes the chosen cards.
- **View → Snap to grid** and **Smart guides** help to place cards by hand.

Annotations

Right-click on an empty spot of the canvas, **Annotate here**: a text, an arrow or a frame. They do not change the calculation: they explain the project to whoever opens it. A frame is not a group.

Shortcuts

Keys	What they do
Tab	add a node
Shift + R	repeat the last node
Ctrl + F	find a node by name
Ctrl + C, **Ctrl + X**, **Ctrl + V**, **Ctrl + D**	copy, cut, paste, duplicate
Ctrl + A	choose all
Ctrl + G, **Ctrl + Shift + G**	group, ungroup
Ctrl + Z, **Ctrl + Shift + Z**	undo, redo
Shift + F	frame the nodes
Ctrl + L	arrange the nodes
Ctrl + ,	preferences

Tip. In a field of a card, the keys write: Tab goes to the next value and Delete erases letters. Copy and paste work on the canvas when the focus is on it.

The 3D view

Moving around

The view always orbits around what is under the pointer, with the vertical always vertical. The wheel zooms towards the pointer (or towards the centre, in Preferences).

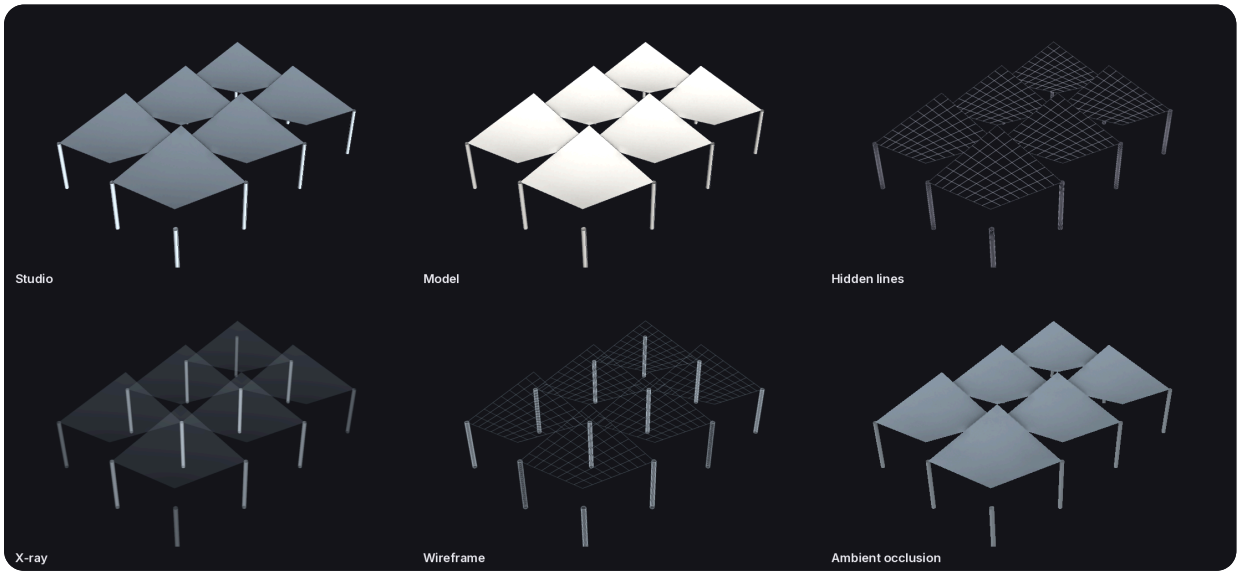
Scheme	Orbit	Pan	Zoom
Tensela (by default)	left button	Shift + left, or middle	right, or wheel
Blender	middle	Shift + middle	Ctrl + middle, or wheel
Rhino	right	Shift + right	Ctrl + right, or wheel
SketchUp	middle	Shift + middle	wheel

Choose the scheme you are used to in **Preferences → Interaction → Mouse in the 3D view**.

Looking from a side

- The **Views** menu of the view: front, sides, above, below, or isometric. The numeric keypad does the same.
- Key **5** switches between **perspective** and **parallel**, like a technical drawing.
- **F** frames everything.

Styles



Style	What it shows
Studio	each object with its colour, lit softly, with darker corners and thin outlines
Model	everything white with outlines, like a foam board model
Matcap	each face lit like a material (plaster, clay, graphite, ceramic): parts close together read one by one
Wireframe	only the edges
Hidden lines	like a line drawing: what is behind is hidden
X-ray	everything semi-transparent
Ambient occlusion	solid, with corners and junctions darker, like an overcast day

The style is changed in the header of the view or with a right-click on it.

What is shown

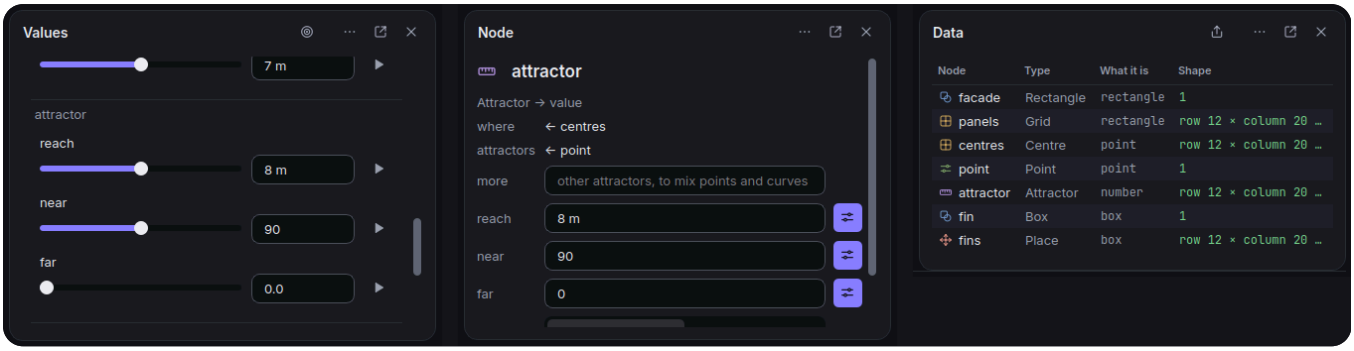
Each node that makes geometry has an **eye** on its card: on, it is shown and exported.

Keys	What they do
H	turn off the chosen ones
Shift + H	show only the chosen ones
Alt + H	show all
Alt + Shift + H	hide all
/	isolate the chosen node, framed; again to go back

Colour and numbers

In the Node panel, each geometry has its **colour** and **opacity**, or is painted **by a value** (the area of each panel, the sun it gets) with a legend; **View → Colour legends** shows them in the view, for a capture. **Number in 3D** writes the number of each element, and **Show direction** draws which way curves go.

The panels



Node

The settings of the chosen node: each input with its value, its options and its help, and, for geometry, its colour, opacity and numbering.

- **More settings** opens the inputs that few people need: the defaults work almost always.
- With several nodes chosen, a value typed here goes to all of them; dragging up or down over a number changes them all at once.
- Some nodes add their own tools here: the graph of a **Graph**, the editor of a **NURBS Curve** (*Edit curve*), the **Move supports** of an **Equilibrium**, the report of a **Print check** or a **Lay flat**.

Values

The **sliders** of the project, all together: the numbers you want to play with, without looking for their nodes. A value of any node becomes a slider with **Show this value as a slider in "Values"**, in the Node panel. Each slider keeps its range and step.

Data

The values of the chosen node as a table: every element with its numbers, along its levels. **Save as CSV…** takes them to a spreadsheet; **Copy as text** to anywhere.

Nodes

The list of every node of the project, with its kind, its family, what it gives and whether it is up to date. A click chooses it on the canvas: handy in large projects.

Library

Your own groups, saved with **Save to library…**, to use in any project, as a list or as a mosaic. **Ctrl + Alt + L** shows or hides it. **Edit → Open the library folder** opens where they are kept.

Croquis

The 3D view drawn as a freehand architect’s sketch: ink, marker, watercolour, hatching. It follows the view as you turn it, and **File → Export → Freehand sketch** saves it as PNG, SVG or PDF.

Exporting

What is exported is what is shown: the nodes with their **eye** on.

File → Export	For	Keys
Image (PNG)	a picture of the 3D view, at the size you choose	Ctrl + Shift + E
Freehand sketch (PNG, SVG, PDF)	the view as a hand sketch (see Croquis)	Ctrl + K
DXF (3D or 2D)	drawings and models for CAD programs	Ctrl + Shift + D
OBJ	3D for any program: renderers, BIM, other modellers	Ctrl + E
3D print (STL), **3D print (3MF)**	the printer's slicer, in millimetres; it checks each piece first	
Cutting file (DXF or SVG)	the laser or the router: one file per sheet, red to cut, blue to engrave	
Blender (.blend)	a Blender scene	Ctrl + Shift + B
IngeTrazo (.igz)	an IngeTrazo model	Ctrl + Shift + I

File → Open in sends the model straight to Blender (Ctrl + B) or IngeTrazo (Ctrl + I), if they are installed.

Tips.

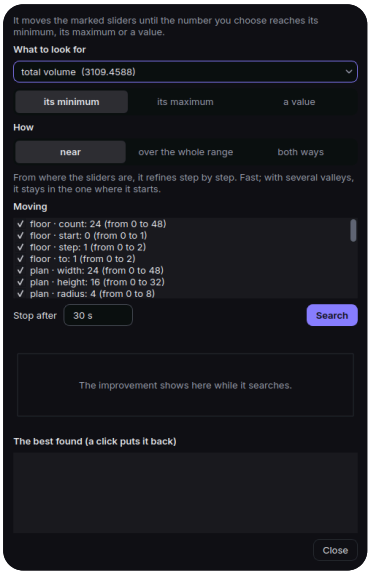
- For a cut file, lay the pieces on sheets first with **Lay flat**. See Fabrication.
- The **Data** panel exports numbers: **Save as CSV...**.

Find the best

What for

To find the values that make a design best by some number: the least material, the most sun in winter on a façade, an area as close as possible to 500 m², the lightest structure that passes the Bar analysis.

How



1. Open **Edit → Find the best…**, or right-click the node that gives the number, **Find its best value…**. It opens beside the window, so you keep seeing the model.
2. **Above, what is searched for:** a node that gives a single number (a Length, an Area, a Formula, the total weight of a Bar analysis…) and whether you want its **minimum**, its **maximum** or a **value to reach**.
3. **Below, which sliders it moves:** those that change that number, all marked to start with. Unmark those it should leave alone. If some number has no slider yet, **Show the marked ones as sliders** adds them to *Values*.
4. **Search.** Each time it finds something better, the sliders and the 3D view show it. **Stop** keeps the best so far.

At the end, the best values stay in the sliders, and **Ctrl + Z** goes back to how they were. The list keeps the best ones found: a click puts any of them back.

Near, or over the whole range

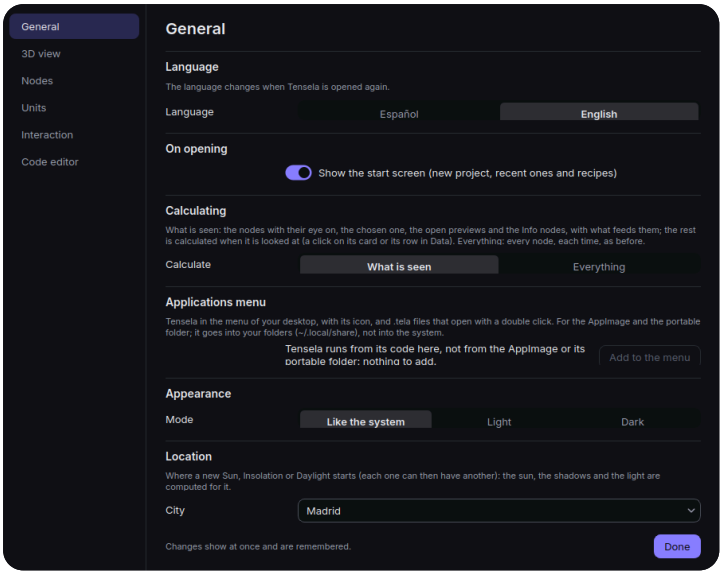
- **Near** refines from where the sliders are: quick, and it finds the best close to the start.

- **Over the whole range** explores everything with a population of candidates, generation after generation: slower, and better when there are several good answers far apart.

Tip. The search works on a copy of the project, so the window keeps responding while it runs. How it searches is told in Form finding and the best value.

Preferences

Edit → Preferences... (Ctrl + ,). Changes show at once and are remembered.



Section	What you set
General	the **language** (English or Español; it changes when Tensela is opened again); the **mode**, light, dark or like the system; the **location** where a new Sun, Insolation or Daylight starts
3D view	the **style** on opening, the **background**, whether to show the axes
Nodes	how the **node cards** are drawn: neutral with a strip of colour, or the header fully coloured; the **canvas** background and pattern (dots, square grid or plain), the grid and the guides
Units	the unit **lengths** are shown in (m, cm, mm, feet or inches), the **decimals**, and the inch fraction
Interaction	the **mouse** scheme of the 3D view (Tensela, Blender, Rhino or SketchUp), where the wheel **zooms**, and the **keyboard shortcuts**: click one and press the new keys
Code editor	the colours, the letter and its size, and the helps while writing in a **Code** node

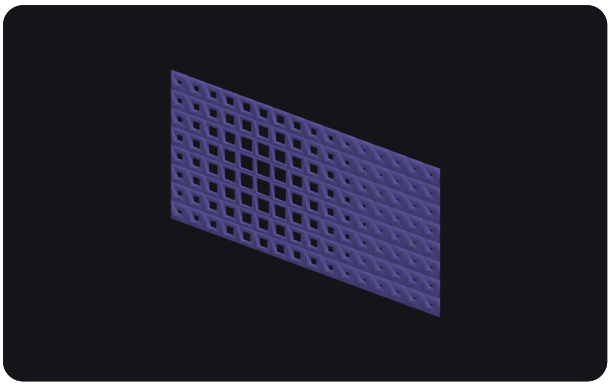
Units

You can type a length in **any unit**, `120 cm`, `4'`, `6 1/2"`, and a plain number is read in the unit you chose. The project is always saved in metres, so it opens the same for anyone.

Recipes

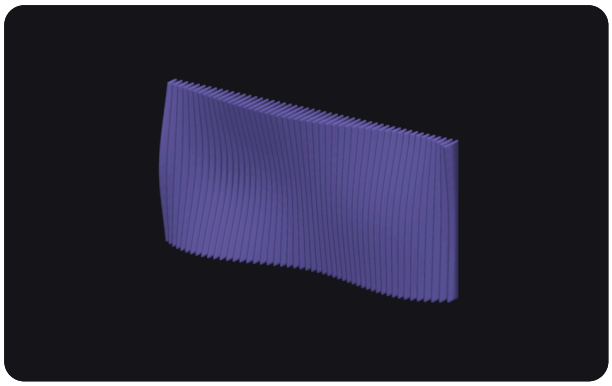
Recipes

Each recipe builds one thing from start to finish, with its nodes and values, and ends with a project file you can open and change. They use only a few nodes each: look them up in all the nodes when you want to know more.



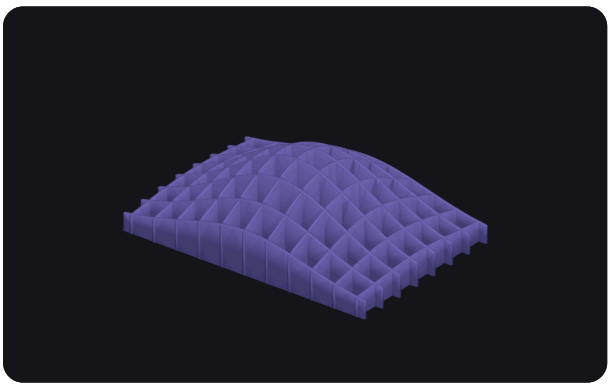
A facade that breathes

Frames whose openings grow near a point and close far from it.



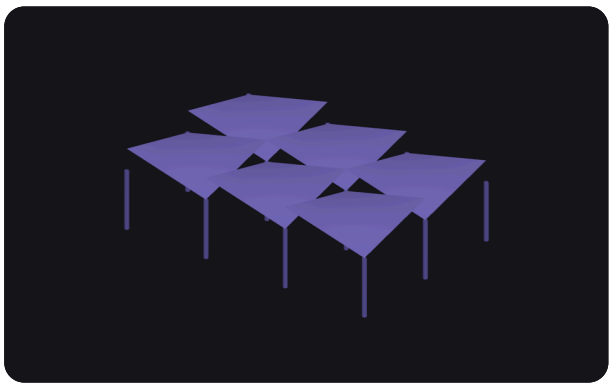
A rib wall

A wavy wall of timber ribs from three hand-drawn curves.



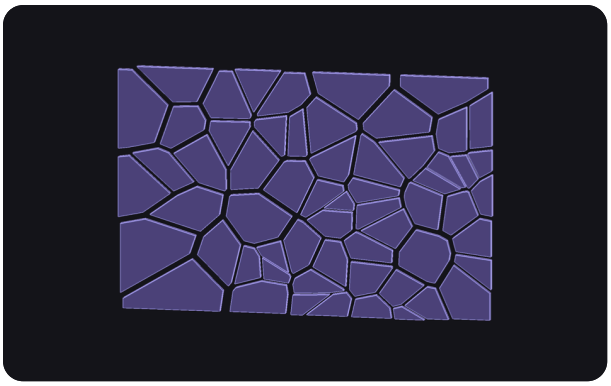
A waffle model ready to cut

A model of a hill as two families of boards that slot into each other, laid on sheets for the laser cutter.



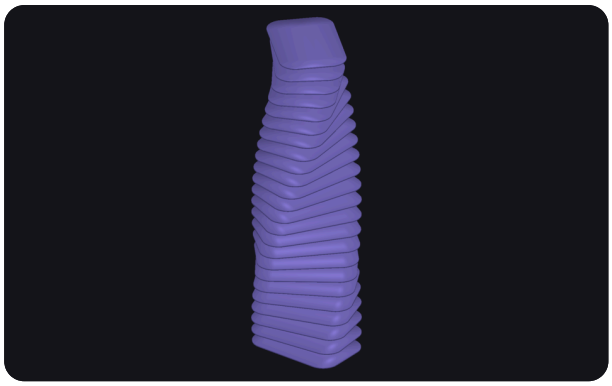
A roof of hyperbolic paraboloids

Six bays of a roof, each folded into a saddle of straight lines, on columns.



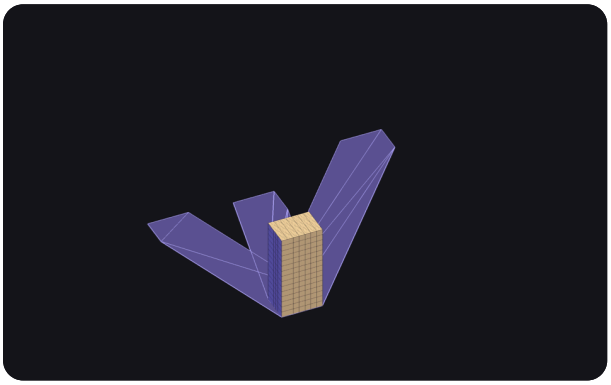
A Voronoi skin

A facade of irregular cells with open joints between them.



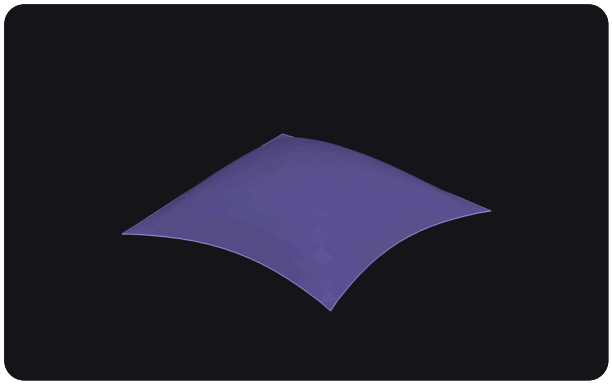
A tower that turns

Twenty-four floors, each turned a little more than the one below.



A sun study

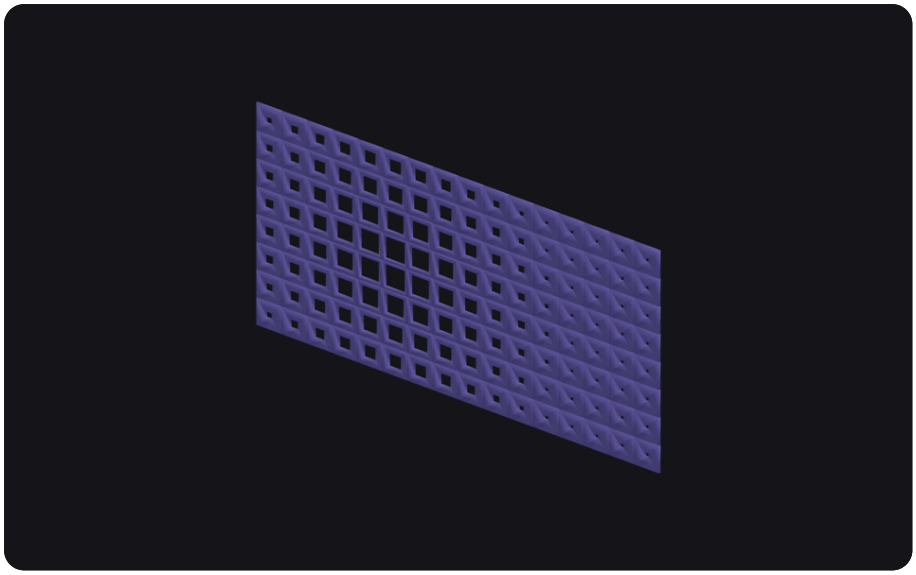
The winter sun on a building in Montevideo: its shadows at three hours and the sun hours on each panel.



A vault that finds its own shape

A membrane hung from its corners and turned upside down: a vault that works mainly in compression.

A facade that breathes



A 24 × 12 m facade of 128 frames. Near a point the openings are wide; far from it they almost close. Move the point and the facade breathes with it.

5 nodes · Download `breathing-facade.tela`

Step by step

1. **The facade.** Add a **Rectangle**: **width** `24`, **height** `12`, **plane** *elevation*.

2. **The panels.** Add a **Grid** and connect the Rectangle to **rectangle**. Write **columns** `16` and **rows** `8`: 128 panels of 1.5 m.

3. **The point.** Add a **Point** with **x** `8` and **z** `6`.

4. **How open each panel is.** Add an **Attractor**. Connect the Grid to **where** and the Point to **attractors**. Write **reach** `10`, **near** `0.85` and **far** `0.15`.

5. **The frames.** Add a **Frame**. Connect the Grid to **panel** and the Attractor to **opening**. Write **thickness** `0.12`.

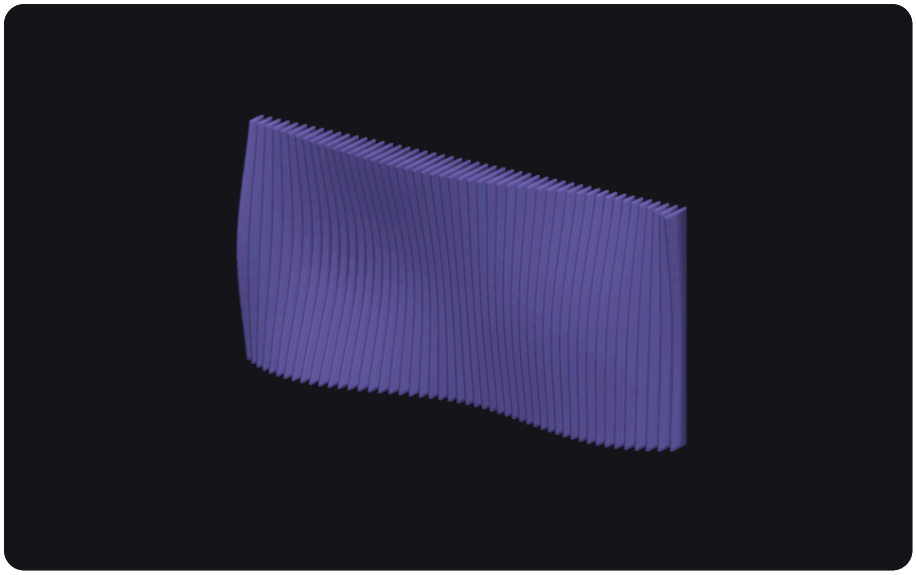
Play with it

- Move the **Point** across the facade.
- Swap **near** and **far**: closed around the point, open everywhere else.
- Add a second Point and connect it to **more** of the Attractor.

Download the project

Download `breathing-facade.tela` · Open it in Tensela: **File → Open**.

A rib wall



A wall 6 m long and 3 m high made of 50 timber ribs. The wave comes from three curves drawn by hand: no formulas. This is the idea of Draw the form, then cut it.

8 nodes · Download `rib-wall.tela`

Step by step

1. **Three curves.** Add three **Drawn curves** and draw each one in the plan view: five points along x, from `0` to `6` m, a little in front of the wall line (y between `-0.2` and `-0.6`). Make them wave differently.

2. **At their heights.** Add two **Move** nodes: one takes the second curve up a **distance** of `1.5`, the other takes the third up `3`.

3. **In order.** Add a **Merge**. Connect the first curve to **a**, the second (moved) to **b** and the third to **c**. Name its **level** `section`.

4. **The front of the wall.** Add a **Loft**: connect the Merge to **curves** and write **along** `section`.

5. **The ribs.** Add **Ribs**: connect the Loft to **shape**, and write **count** `50` and **weight** `0.04` (4 cm boards).

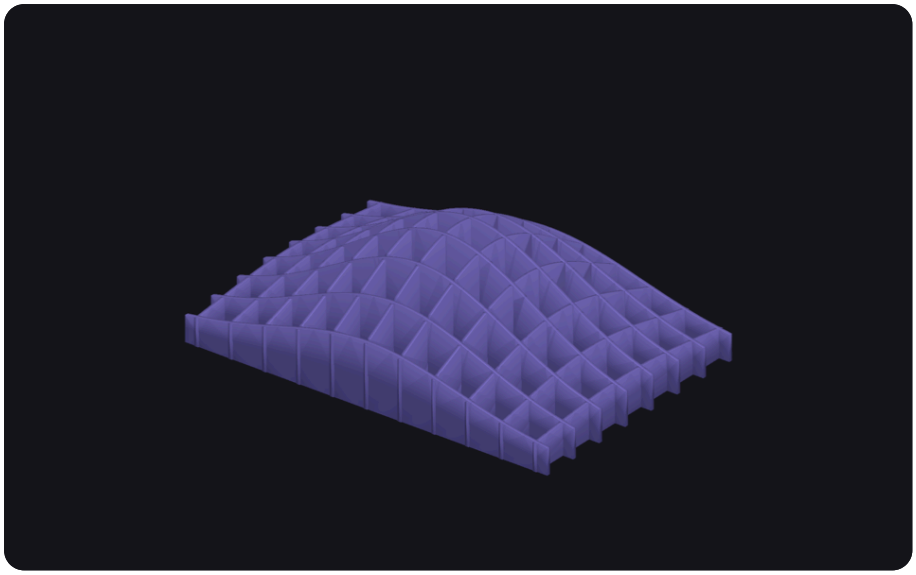
Play with it

- Drag the points of any curve: all the ribs follow.
- Change **count** and **weight**.
- Add a **Volume** and a **Summarise** to know the m³ of timber.

Download the project

Download `rib-wall.tela` · Open it in Tensela: **File → Open**.

A waffle model ready to cut



A 30 × 20 cm model of a hill in 3 mm board: boards across x and across y, slotted halfway so they slide into each other, laid on 600 × 400 mm sheets with their numbers. All measures in metres.

11 nodes · Download `waffle-model.tela`

Step by step

1. **The hill.** Make a grid of points: two **Series** (`13` values from `-0.15` step `0.025` , level `u` ; `13` from `-0.1` step `0.0167` , level `v`), a **Function** *hill* with them as **x** and **y**, **amplitude** `0.08` and **width** `0.3` , and a **Point** with x, y and z from them. Join them with a **Surface from points**, **smooth** on.

2. **Boards across x.** Add **Sections**: the surface to **shape**, **thickness** `0.003` , **gap** `0.025` , **cuts** *x*, **piece** *to base*, **depth** `0.02` .

3. **Boards across y.** The same with **cuts** *y*.

4. **The slots.** Add **Slots** with the boards across x as **first** and across y as **second**, **thickness** `0.003` : it gives the first family, slotted. Add a second one the same with **show** *second* for the other family, and a third with **show** *boards* to see them fitted.

5 **On sheets.** Add **Lay flat**: the first two Slots, the first to **pieces** and the second to **more**, **width** `0.6`, **height** `0.4`.

6 **The file. File → Export → Cutting file**: one file per sheet, red to cut, blue to engrave the numbers.

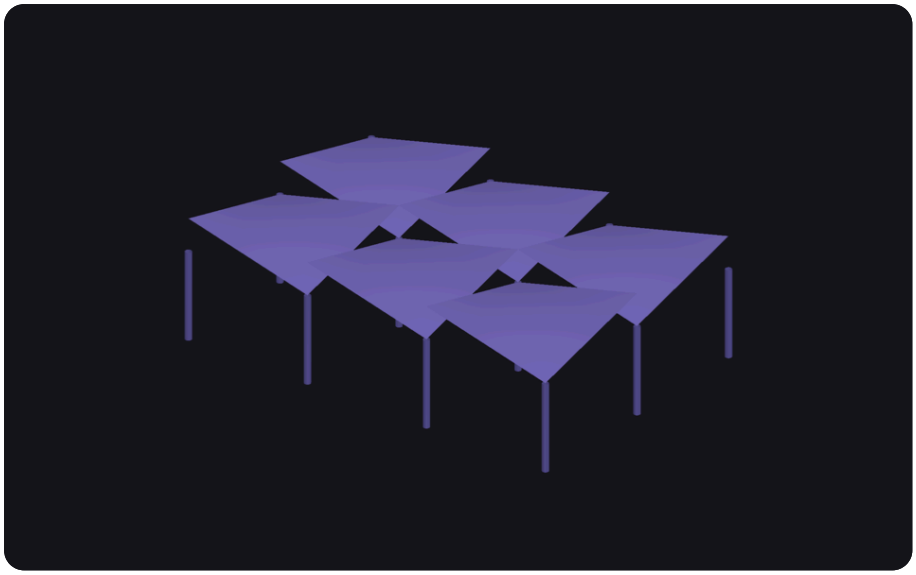
Play with it

- Change the **gap** between boards: fewer, sturdier boards, or many thin ones.
- Change the hill with the **amplitude** of the Function.
- Use **clearance** in Slots if your laser burns more.

Download the project

Download `waffle-model.tela` · Open it in Tensela: **File → Open**.

A roof of hyperbolic paraboloids



An 18 × 12 m roof at 4 m in six bays. Each bay rises at two opposite corners and becomes a hyperbolic paraboloid, a saddle made only of straight lines, like the shells of Félix Candela.

6 nodes · Download `hypar-roof.tela`

Step by step

1. **The roof plan.** Add a **Rectangle**: **width** `18`, **height** `12`, **origin** `0 0 4`.

2. **The bays.** Add a **Grid** with **columns** `3` and **rows** `2`.

3. **The saddles.** Add a **Hypar**: the Grid to **panels**, **lift** `1.5`.

4. **Where the columns go.** Add **Vertices** with the Grid: the corners of every bay.

5. **The columns.** Add a **Line by direction** from the corners, **direction** `0 0 -1`, **length** `4`, and a **Pipe** around those lines with **radius** `0.15`.

Play with it

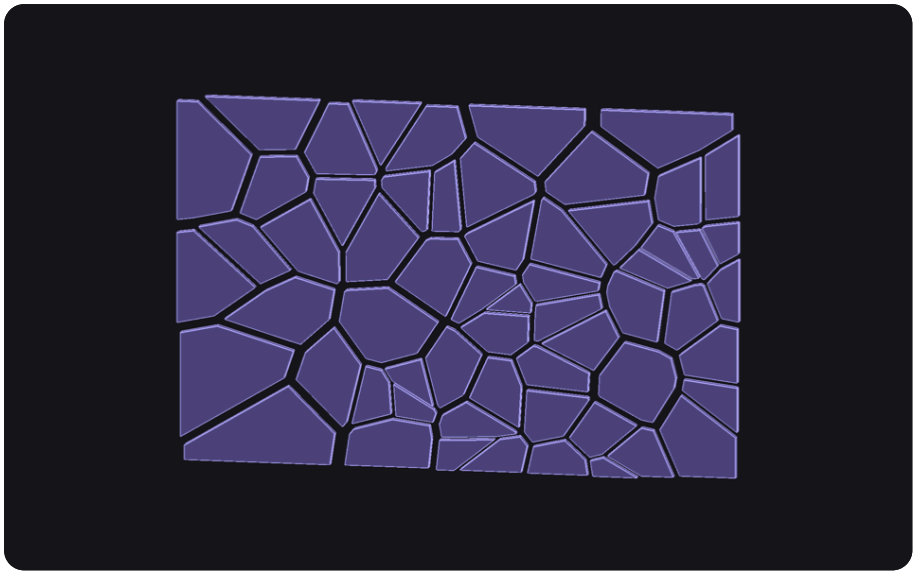
- Change the **lift**: flat at `0`, dramatic at `3`.

- Choose the other two **corners** to rise.
- With **show** *flat piece*, each bay opened along its fold into two triangles, to cut it from a sheet for a model.

Download the project

Download `hypar-roof.tela` · Open it in Tensela: **File → Open**.

A Voronoi skin



A 12 × 8 m facade covered with 60 irregular panels, each one a Voronoi cell, shrunk a little to leave an open joint, and given a thickness.

5 nodes · Download `voronoi-skin.tela`

Step by step

1. **The facade.** Add a **Rectangle**: **width** `12`, **height** `8`, **plane** *elevation*.
2. **The seeds.** Add **Random points** with the Rectangle as **region**, **count** `60`.
3. **The cells.** Add a **Voronoi**: the points to **points**, the Rectangle to **boundary**.
4. **The joints.** Add a **Scale** with **factor** `0.88`: each cell shrinks about its own centre.
5. **The panels.** Add a **Thicken** with **thickness** `0.12`.

Play with it

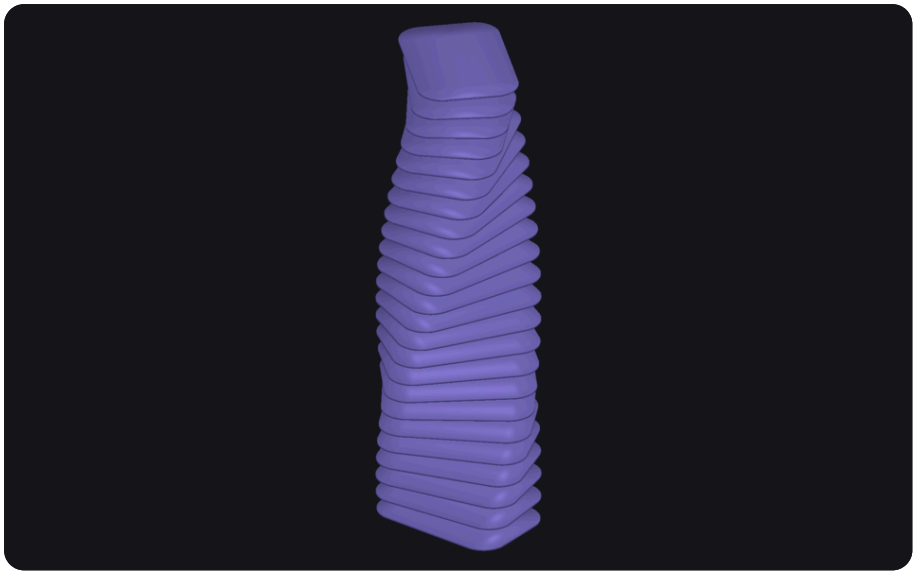
- Try other values of **seed** in Random points until you like the pattern.
- More **count**, smaller cells.

- Drive the **factor** of Scale with an **Attractor**: wider joints near a point.

Download the project

Download `voronoi-skin.tela` · Open it in Tensela: **File → Open**.

A tower that turns



A tower of 24 floors, 3.5 m apart. Every floor is the same rounded plan, turned 6° more than the one below: 138° from the ground to the top.

7 nodes · Download `twisting-tower.tela`

Step by step

1. **The floors.** Add a **Series**: **count** `24`, step `1`, **level** `floor`.

2. **Their heights.** Add a **Formula** `a * 3.5` with the Series as **a**.

3. **Their turns.** Add another **Formula** `a * 6` with the Series as **a**: degrees for each floor.

4. **The plan.** Add a **Rounded rectangle**: **width** `24`, **height** `16`, **radius** `4`, **origin** `-12 -8 0` (centred on the axis).

5. **Up and around.** Add a **Move** with the heights as **distance**, then a **Rotate** with the turns as **degrees**.

6. **The slabs.** Add an **Extrude** with **vector** `0 0 0.35` and **caps** on.

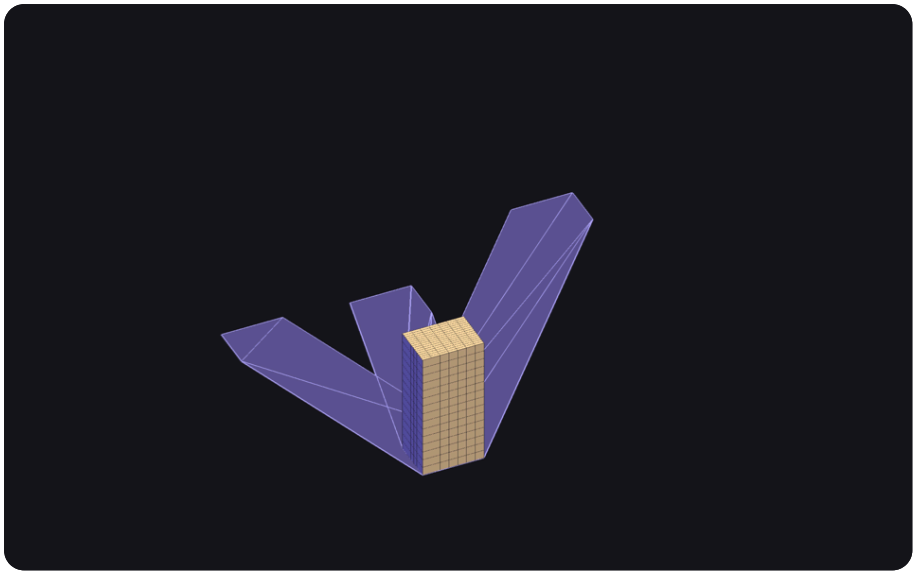
Play with it

- Change the `6` of the turns: a calm tower or a spiral.
- Make the turn grow towards the top with `a * a * 0.25`.
- Scale each floor with a third Formula for a tower that tapers.

Download the project

Download `twisting-tower.tela` · Open it in Tensela: **File → Open**.

A sun study



A 14 × 10 m building, 30 m tall, in Montevideo on the 21st of June, the winter solstice there: where its shadow falls at 10, 12 and 14 h, and how many hours of sun each panel of its faces, roof included, gets over the winter.

6 nodes · Download `sun-study.tela`

Step by step

1. **The building.** Add a **Prism**: **width** `14`, **depth** `10`, **height** `30`.

2. **Its panels.** Add **Faces** with **which** *all faces* (the façades, the roof and the base), and a **Grid** on them with **columns** `7` and **rows** `15`.

3. **The sun.** Add a **Sun**: **location** *Montevideo*, **month** `6`, **day** `21`, **hour** `10; 12; 14`, **show** *ray*.

4. **The shadows.** Add a **Shadow**: the Prism to **geometry**, the Sun to **ray**.

5. **The sun hours.** Add an **Insolation**: the Grid to **faces**, **location** *Montevideo*, **period** *winter*, **show** *sun hours*. Paint the Grid by this value in its panel (**Colour → by a value**).

Play with it

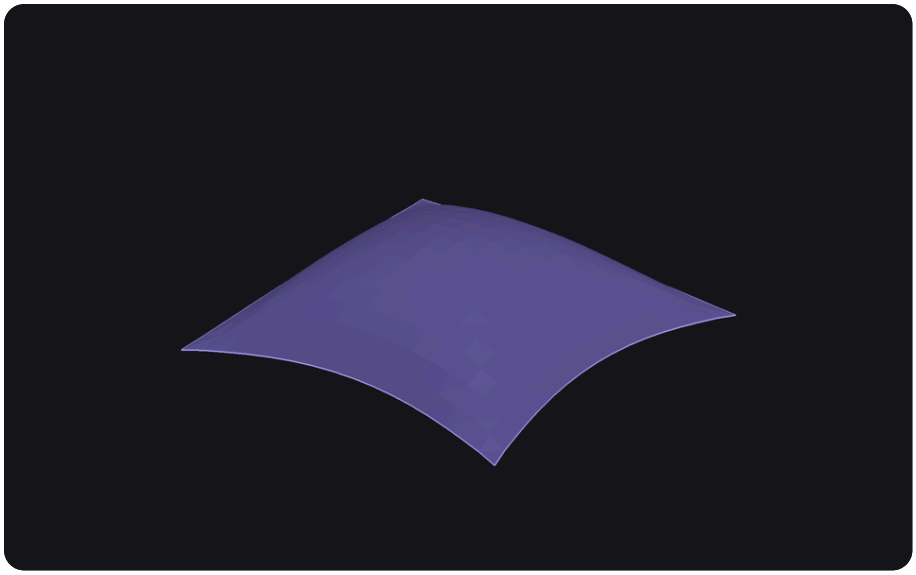
- Press the ▶ of the hour in the Sun to see the shadow move through the day.
- Add the neighbouring buildings to **obstructions** of the Insolation.
- Change **period** to *summer* and compare the north and south façades.

A design study: for regulations or certificates, check it with a specialist.

Download the project

Download `sun-study.tela` · Open it in Tensela: **File → Open**.

A vault that finds its own shape



A 12 × 12 m membrane held at its four corners hangs under its own weight. Turned upside down, that shape is a vault that works mainly in compression: the method of the hanging models of Gaudí and Isler.

It is a form study, not a structural calculation: an engineer has to check it before it is built.

3 nodes · Download `hanging-vault.tela`

Step by step

1. **The plan.** Add a **Rectangle**: **width** `12`, **height** `12`.

2. **The membrane.** Add a **Mesh surface** with **count u** `16` and **count v** `16`.

3. **The vault.** Add an **Equilibrium**: the mesh to **shape**, **self weight** `300`, **gravity** `0 0 1` (upwards: the hanging shape turned over).

Play with it

- Change **supports** to *border*: a vault resting on its four edges.
- Add **more supports** inside: a vault on columns.
- Set **show** to *forces* to see the compression in each span.

Download the project

Download `hanging-vault.tela` · Open it in Tensela: **File → Open**.

Extensions

Extensions

What for

An extension adds to Tensela what it does not bring: a node for a system of your office, a way of exporting to the format of a machine, a panel with a tool, a connection with another program. It is added without changing Tensela, and it goes away when it is turned off.

How

1. Find it on the Extensions page of tensela.com, or get it from whoever makes it. It is a `.zip`.
2. **Extensions → Manage extensions… → Install from a file…** and choose the `.zip`.
3. Read its card before installing: what it does, who makes it, its licence and **what it can do**. What it does not ask for is in green; what it asks for is in amber, each one with **why**, in the words of whoever made it.
4. **Install.** It is on at once: its nodes are in the finder (Tab), its entries in the menus, its panels in the **Extensions** menu.

In **Manage extensions…**, each card has its **switch** (on or off, at once) and its ⋯ (its website, where to report a problem, its code, its folder, Remove). On top, a search, which ones to show and in what order, and two views: detailed (one card each) and compact (one line each, which opens with a click).

What it can do

Every extension computes inside Tensela. Besides that, its card says if it may:

In amber	It means
Internet	it connects to the internet (to download the weather of a place, a map…)
Files outside the project	it reads or writes files that are not of the project
Other programs	it starts other programs of the computer
Talks with other programs of the computer	another program of the computer talks with Tensela, inside it, never through the internet

An extension is a program: Tensela shows what it says it does, but cannot check it. Install only those that come from someone you trust. Those of the Extensions page of tensela.com have their code public and were reviewed by a person before being published: that lowers the risk, but it is not a guarantee. Install only the ones you need.

Update one

Install the new `.zip` over it. Its card says what happens: **it updates the one installed** (1.0 → 1.1), **it is older** (in amber: you go back to an earlier version) or **it is the same**. If it was on, it stays on; if it was off, it stays off.

The two of Tensela

- **Animation** comes with Tensela, off. Turned on, **Extensions → Animation** opens its panel: a slider from one value to another in a number of frames, the 3D view as it is seen at each one, as a GIF or PNG images.
- **AI connection** comes apart (on the Extensions page). An AI program of your choice looks at the open project and changes it. Tensela talks with it inside your computer; what that program does with what it reads depends on it. Each step is undone with Ctrl + Z, and Code nodes and files ask you first.

Tips

- A project with nodes of an extension that is not installed (or is off) opens anyway: those nodes are kept, marked in red, and they compute again when the extension is back.
- An extension that fails when it is turned on stays off, and Tensela says why.
- Its panels move and go to a separate window like the others; a separate window stays above Tensela.
- To write your own: Write an extension.

Write an extension

What for

For what Tensela does not bring and you need again and again: the module of a façade system of your office, a node that reads the data of a supplier, the cutting format of a machine, a panel with a check of your own. If it is only once, a Code node is enough; if it is for you and others, or every week, it is an extension.

How

You need Tensela and a text editor. It helps to know a little Python: an extension is a folder with its card and a small Python package.

1. Start from the template

Copy the repository extension-template (on GitHub, **Use this template**, or download it as a `.zip`). It brings a node (**Fold**, the zigzag of a folded plate roof), an action, a right-click entry, a panel, an icon, its tests and what publishes it.

```
facade-tools/                    the folder: the id of the extension
    extension.toml               its card
    facade_tools/                its package: the id with "_" instead of "-"
        __init__.py              def register(tensela): …
        fold.svg                 its icon
    tests/test_my_extension.py
```

In `extension.toml`, change the `id` (lowercase letters, digits and `-`), its name, your name, what it does, its licence and its links. Rename the package folder as the id with `_`.

2. Load it from its folder

1. **Extensions → Manage extensions… → Load a folder…** and choose the folder. It is on at once, without installing it; its card says «in development».
2. Change the code and save. On its card, **Reload** (the round arrow): Tensela reads it again, without closing it.
3. When you are done, in its ⋯, **Stop developing it**. The folder stays as it is.

3. A node

```
from tensela.nodes import Input
```

```
def fold(length, folds, rise, name=None):
    ...                       # it gives a Data: numbers, points, curves, surfaces…

def register(tensela):
    tensela.node("Fold", "Curves",
                 (Input("length", 12.0, single=True), Input("folds", 4, single=True),
                  Input("rise", 1.2, single=True)),
                 "fold", fold, "A folded line, as in a folded plate roof.", icon="fold")
```

The function receives the inputs by their name (and `name`, the node's) and gives its result. It must only compute: it runs also in the process that computes, apart from the window, and the same inputs must give the same result.

4. Menus, panels, icons

```
tensela.icon("fold", "facade_tools/fold.svg")              # 24 × 24, drawn with lines
tensela.action("Count the nodes", count_nodes)             # Extensions menu: count_nodes(window)
tensela.menu("node", "Add a Fold", add_fold)               # right click on a node:
add_fold(window, node)
tensela.panel("Facade check", build_panel)                 # Extensions menu: build_panel(window)
gives a QWidget
tensela.on("selected", selected)                           # selected(window, node): a node chosen,
or None
tensela.export("Machine format…", ".cut", write_cut)       # File → Export: write_cut(path,
objects)
```

5. Change the project

Through the window, so Ctrl + Z undoes it in one step:

```
def add_fold(window, node):
    def change(project):                                   # a copy of the open project
        new = project.graph.node("Fold", None, length=10, folds=4, rise=1)
        project.export.append(new.name)                    # its eye on
    window.edit_project(change, "A Fold node")
```

If `change` fails, nothing changed.

6. In Spanish (and other languages)

Write every text in English and translate it in the card:

```
[es]
name = "Herramientas de fachada"
description = "…"

[translations.es]
"Fold" = "Pliegue"
"length" = "largo"
```

7. Test it and pack it

```
python -m pytest tests
tensela-app --pack-extension .
```

With the AppImage: `Tensela.AppImage --pack-extension .`. It checks the card, leaves `<id>-<version>.zip` beside the folder and says its SHA-256, its fingerprint. Try the `.zip` with **Install from a file...**, as the others will.

To share it with everyone: Publish an extension.

What it does

Its card

Key	
`id`	lowercase letters, digits and `-`; its folder and, with `_`, its package
`name`, `version`, `author`, `description`	what the card says
`tensela`	the oldest Tensela it works with (`"0.9"`)
`licence`	its licence, by its SPDX name (`"GPL-3.0-or-later"`, `"MIT"`...)
`website`, `issues`, `source`, `contact`	its links: where it is told about, where to report a problem, its code
`[permissions]`	what it may do besides computing, and why: `network`, `files`, `programs`, `local`
`[es]`, `[es.permissions]`	the card in Spanish
`[translations.es]`	its texts in Spanish

The door: what `register(tensela)` receives

	Its function receives
`tensela.node(name, category, inputs, output, function, help, icon=)`	the inputs by name, and `name`
`tensela.export(label, suffix, function, help)`	the path and the visible objects (name, Data, style)
`tensela.action(label, function, help, icon=)`	the window
`tensela.menu(where, label, function, help, icon=)`	the window, and the node (`where="node"`); `"canvas"` , `"view"`
`tensela.panel(title, build, help)`	the window; it gives the panel's content
`tensela.icon(name, file)`	(an SVG of its folder)
`tensela.on(event, function)`	the window; `"window"` , `"computed"` , `"selected"` (+ node), `"project"` , `"sliders"`
`tensela.command(function, help)`	the arguments: `--extension <id> …` , without the window

`unregister()` , if the package has one, is called when it is turned off. Everything it added goes away by itself.

What it may use of the window

`window.project`	the open project: `graph` (its nodes), `export` (those seen), `controls` (the sliders); None if none
`window.edit_project(change, what)`	to change it, as one step of Ctrl + Z
`window.last_errors`	what failed in the last computation: node → message
`window.also_calculate`	nodes to calculate too, though they are not seen (empty it after)
`window.show_extension_panel(title)`	to open one of its panels
`window.statusBar().showMessage(text)`	to say something at the bottom
`window.viewer.save_image(path)`	the 3D view as a PNG

Tips

- Keep the function of a node pure: no files, no network, no window. What needs them goes in an action or a panel.
- A long task in an action blocks the window: split it, or say what it does in the status bar while it works.

- The icon is tinted with the colour of the theme: draw it with lines (`stroke`), without fixed colours.
- The animation and the AI connection extensions are also examples: their code is in the repository of Tensela.

How it runs

An extension is Python that Tensela imports: it can do anything Python does, which is why its card says what it does and why the catalogue reviews its code. Its nodes are registered in the window and in the process that computes (a separate process, so a long computation does not freeze the window); that is why a node only computes. When it is turned off, Tensela takes away what it registered and forgets its package, so **Reload** reads it again from its folder.

Publish an extension

What for

So that anyone finds it on the Extensions page of tensela.com, with its card, what it may do, its `.zip` and its fingerprint.

How

1. **Its code, in a public repository.** The catalogue asks for it: anyone can read what it does.
2. **A release with its `.zip`.** With the template, tag the version (`git tag v0.1.0`, `git push --tags`): its workflow runs the tests, packs it and attaches the `.zip` and `SHA256SUMS`.
3. **Its file for the catalogue**, made from the `.zip` with the tool of the repository tensela-extensions:

   ```
   python tools/catalogue.py entry facade-tools-0.1.0.zip https://github.com/…/facade-tools-0.1.0.zip > extensions/facade-tools.json
   ```

   It copies what its card says and computes its fingerprint. Read it: it is what the page shows.
4. **A pull request** with that file. A check runs by itself (the `.zip` is the one its fingerprint says, and its card agrees with its file), and then a person reviews it.

What is reviewed

- Its code is public, and the `.zip` has what that code says.
- Its permissions make sense for what it does, and each one says why.
- Nothing hidden: no code downloaded while it runs, no data of the person sent anywhere without saying so, nothing obfuscated.
- Its card is complete: name, what it does, who makes it, licence, code, where to report a problem.

The problems of an extension are reported where its card says, to whoever makes it.

Update it

Change `version` in its card, tag the new release, and a pull request that changes `version`, `download`, `sha256` and `updated` in its file. Whoever installs the new `.zip` sees «It updates the one installed».

Check a .zip you downloaded

Its page on tensela.com says its fingerprint (SHA-256). In a terminal:

```
sha256sum facade-tools-0.1.0.zip
```

If the 64 characters are the same, the `.zip` is the one that was reviewed.

Nodes

All the nodes

Each family has its colour on the canvas. Each page tells every node in the same order: **what for**, **how**, **what it does** (each input in plain words) and **tips**.

See them all at once on the map of all the nodes.

Family	What it is for
Values	numbers, lists, formulas and switches that drive the design
Shapes	rectangles, boxes, prisms, spheres and points to start from
Curves	lines, arcs, circles, catenaries, free curves, and how to divide, cut and label them
Surfaces	surfaces through curves, points or edges, structures on surfaces, unrolling
Divide	grids, Voronoi, faces, vertices, and picking, splitting and reordering lists
Measure	distances, attractors, remapping, areas and volumes, conditions
Transform	move, place, orient, rotate, scale, twist, morph, thicken, frame
Meshes	meshes from surfaces, points or triangles, joining, smoothing, editing by hand
Fabrication	sections, slots, plates, flat panels, sheets with numbers, bevels, print check
Environment	sun, shadow, incidence, insolation, daylight, climate
Simulation	hanging shapes, bar analysis, topology optimization
Groups	the group result, and the nodes inside a group

Values

Values are the numbers and lists that drive a design: the width of a bay, 16 floors, an angle that depends on a distance. They are green on the canvas.

Number

What for. A value you want to change by hand: the height of a floor, the depth of a fin. Or a short list typed by hand: floors of different heights, bays of different widths.

How. Write the **value**. In the window it becomes a slider you can drag.

Input	What it means
value	one number, or several separated by `;`: `3.5; 3; 3; 4`
level	the name of the list when there are several values: *floor*, *bay*, *axis*

Tips.

- With several values, give the list a name, so it pairs or crosses well with others. Lists with names

Series

What for. Many numbers that follow a rule: 25 floors every 3.2 m, 12 steps, 40 fins spread across a facade.

How. Write how many (**count**) and how they advance: **with step** (from the start, so much each time) or **between two** (spread evenly from the start to the end).

Input	What it means
count	how many numbers
mode	*with step* or *between two*
start	the first number
step	how much each one adds (with step)
to	the last number, included (between two)
level	the name of the new list: *floor*, *step*, *axis*

Tips.

- A Series is the usual way to start a list. Name its level from the start.
- `count 5, start 0, step 3.2` gives `0; 3.2; 6.4; 9.6; 12.8`: the levels of five floors.

Formula

What for. A calculation, like a spreadsheet cell: an angle from a distance, a height that grows floor by floor.

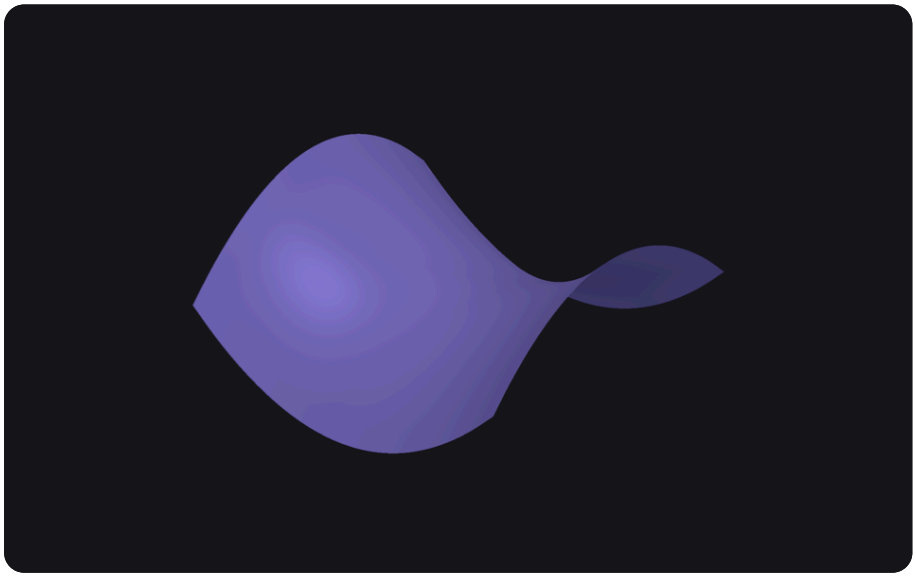
How. Write the **formula** with the letters **a** to **h**, and connect a value to each letter you use. The letters **e** to **h** appear when you write them.

Input	What it means
formula	as in a spreadsheet: `a * 3.5`, `if(a < 5; 90; 0)`, `sin(a)`
a … h	the values the formula uses

Tips.

- Angles are in **degrees**: `sin(30)` is 0.5.
- It understands Spanish and English, even mixed: `si(a < 7; 3; 6)` = `if(a < 7, 3, 6)`.
- If **a** is a list of 240 values, the formula is calculated 240 times, once for each.

Function



What for. Ready-made shapes for a number: a wave, an arch, a catenary, a hill, a saddle. To give a facade a wave or a roof a swell without writing a formula.

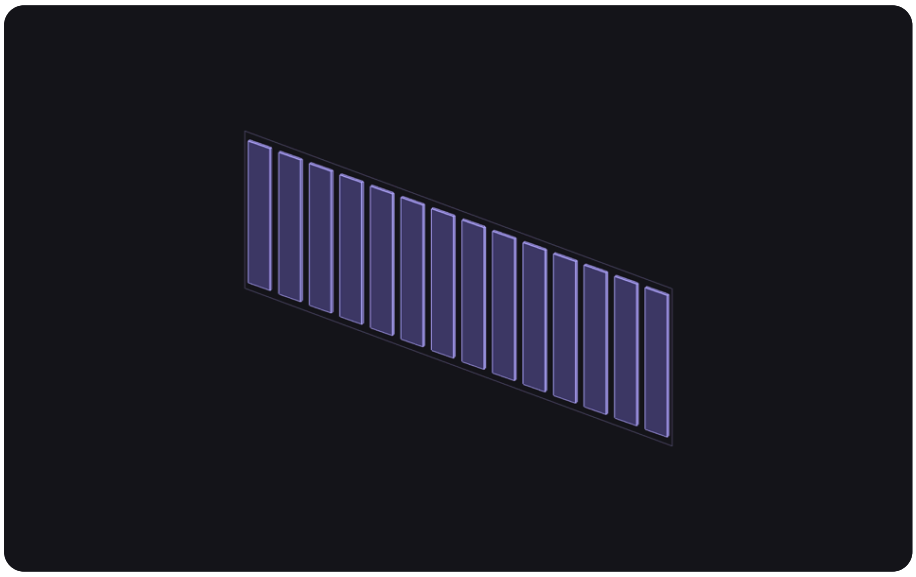
How. Choose the **shape**, connect where it is calculated (**x**, and **y** for the shapes of two variables), and set its **amplitude** and **width**.

Input	What it means
shape	of one variable: *wave*, *ramp*, *parabola*, *catenary*, *bell*, *stair step*, *sawtooth* · of two: *swell*, *ripples*, *hill*, *saddle*, *dome*, *pyramid*
x	where it is calculated: a Series, some distances, the x of some points
y	the second variable, for the shapes of two: another Series, with another level name
amplitude	how high it rises
width	how often it repeats, or how big the shape is

Tips.

- The *saddle* is the hyperbolic paraboloid, the shape of many shells by Félix Candela.
- The panel of the node says what each shape is good for.

Random



What for. A bit of variety: fins of slightly different angles, panels of random colours, trees of different heights.

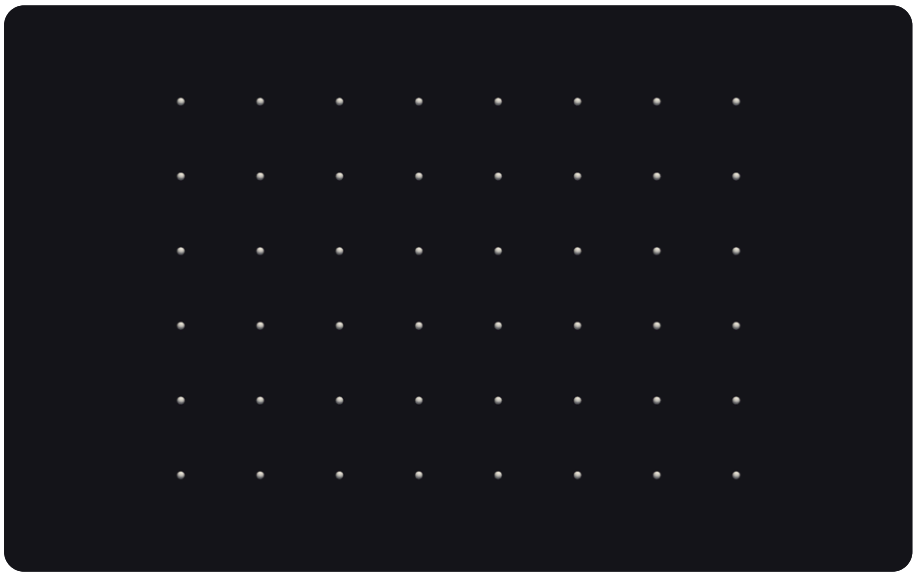
How. Per element: one random number for each element of some data. **Count:** a loose amount of numbers.

Input	What it means
mode	*per element* or *count*
data	the elements that get a number each (per element)
along	one number for each… cell, row; empty: the first level
count	how many numbers (count)
level	the name of the new list; the same name as another list, to pair them
minimum, **maximum**	the range
integers	yes: whole numbers, from minimum to maximum, both included
seed	the same seed always gives the same numbers; another seed, others

Tips.

- Random here is repeatable: the design does not change every time you open it, only when you change the seed.

Point



What for. A point in space: an attractor, a corner, the centre of something.

How. Write its **x**, **y** and **z**.

Tips.

- If a coordinate is a list, you get many points. Two lists with different names give a grid of points. Lists with names
- X goes east, Y goes north, Z goes up.

Yes or no

What for. A switch: to turn a part of the model on or off, or to compare two variants.

How. Connect it to the **condition** of a **Choose**, which then gives one variant or the other.

Info

What for. To read what a cable carries: numbers, a list, points. It passes it on unchanged, so you can put it in the middle of a chain.

How. Connect anything to its **value** and read it on the card.

Repeat

What for. To make a short list go with a long one: a pattern of three bay widths repeated along a facade of 20.

How. Connect the **data**, choose the **level** and write the final **length**.

Input	What it means
data	the list to repeat
level	which level, if there are several; empty: the only one
length	how many elements it must have in the end

Tips.

- `2; 6; 1` repeated up to 5 gives `2; 6; 1; 2; 6`.

Merge

What for. To put two to eight things **side by side**, with a new level: a curve and its footprint, to join them with a surface; the start, the middle and the end of each line, for a curve through the three.

How. Connect **a** and **b**; connect the last one and another input appears, up to **h**.

Input	What it means
a … h	what you merge
level	the name of the new level (*part* by default)

Merge lists

What for. To put lists **one after the other** in a single list: the edges of a box and some extra bars.

How. Connect **a** and **b**; another input appears each time, up to **h**.

Tips.

- **Merge** puts things side by side and adds a level; **Merge lists** puts them in a row in one list.

Code

What for. For what the nodes do not reach: a loop, a function that calls itself, a rule of your own. Most designs never need it.

How. Write a short piece of **Python**. What you connect to **a … h** reaches the code under those letters, and what the code leaves in `result` comes out of the node.

Input	What it means
code	up to 200 lines of Python that leave their answer in `result`
a … h	the values the code uses
levels	the name of the list the code gives; for lists inside lists, one name each, separated by `;`

Tips.

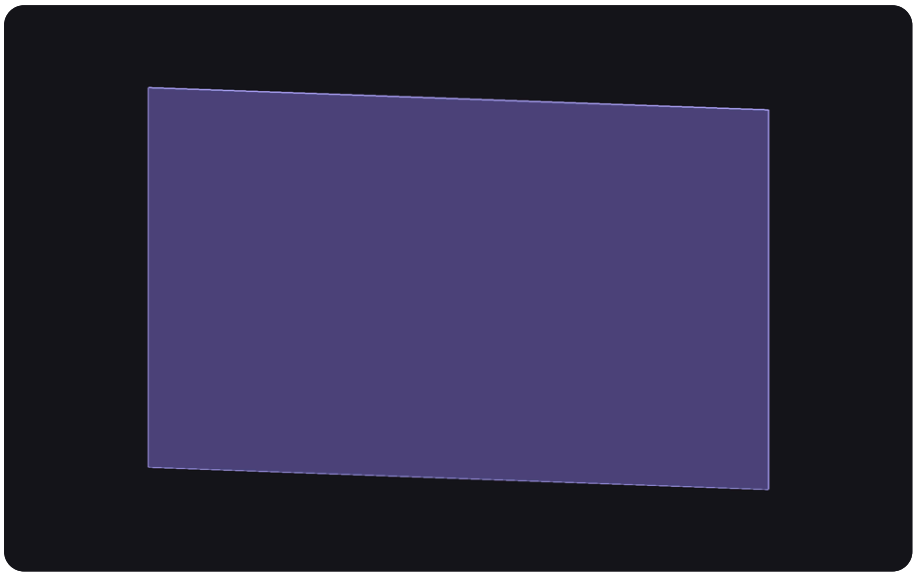
- The code runs **apart from the app and with limits**: it works with numbers, lists, points, lines and meshes, and it is made not to open files, use the network or run other programs.

Shapes

Shapes are where most designs start: a facade, a plot, a block, a fin. They are blue on the canvas.

Every input can take several values. A Rectangle with **width** `4; 6; 8` gives three rectangles, and a Prism whose **height** comes from a Series of 5 floors gives five volumes. See Lists with names.

Rectangle



What for. A facade, a plot, a slab, a sheet to cut: the base of many designs.

How. Write its **width** and **height** and choose the **plane**: *plan* for a floor or a plot, *elevation* for a facade.

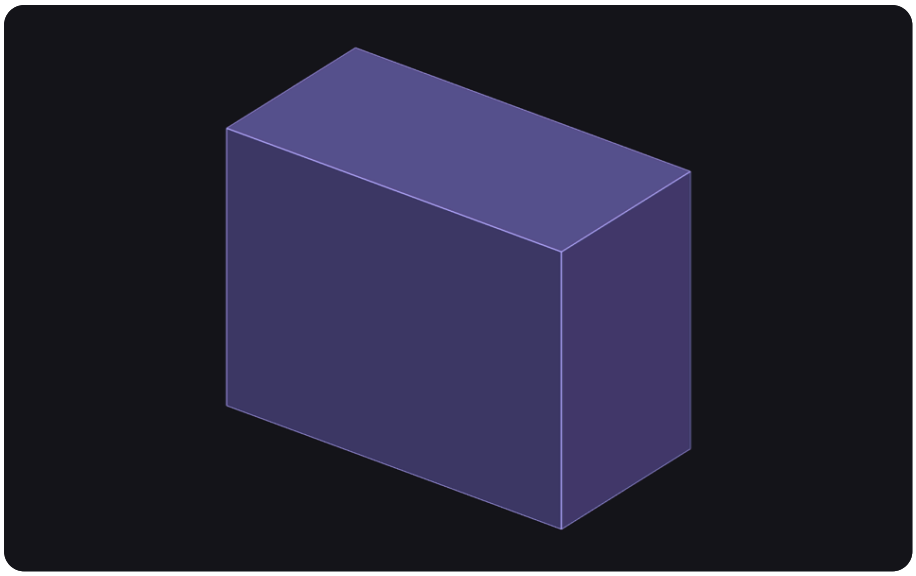
What it does.

Input	What it means
width	the horizontal side: along X (along Y in *side elevation*)
height	the other side: north-south in plan, upwards in elevation
origin	where its lower-left corner is (`0 0 0` by default)
plane	*plan* (lying down), *elevation* (standing, facing south) or *side elevation* (standing, facing east)

Tips.

- A rectangle is a flat outline, not a body: it has an area but no volume. **Extrude** it with caps to get a volume.
- Divide it into panels with a **Grid**, as in Your first project.

Box



What for. A fin, a beam, a brick, a block that you repeat many times.

How. Write its **width**, **depth** and **height**, then take it where you want with **Place** or **Move**.

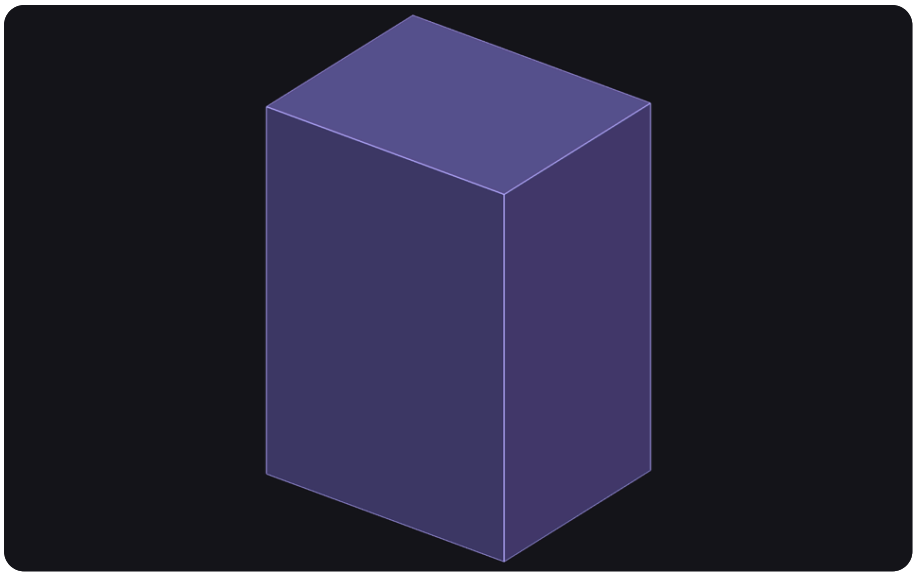
What it does.

Input	What it means
width	along X
depth	along Y
height	along Z, upwards

Tips.

- A Box is **centred on the origin**, so when **Place** turns it, it turns around its own middle: just right for sun fins.
- For a building that sits on a corner, use a **Prism** instead.

Prism



What for. A building volume, a tower, a block of a master plan.

How. Write its **width** (east-west), **depth** (north-south) and **height**. The **origin** is its lower south-west corner.

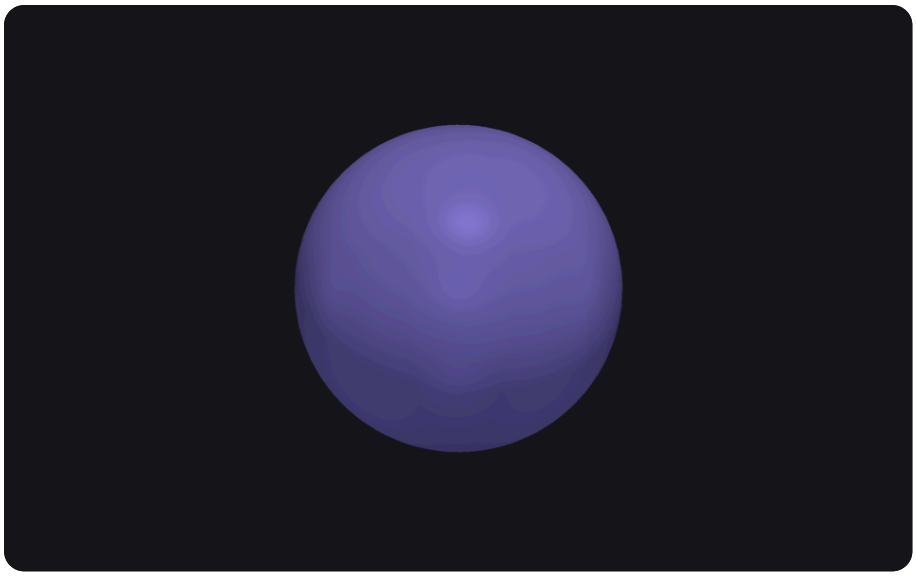
What it does.

Input	What it means
width	east-west, along X
depth	north-south, along Y
height	upwards
origin	the lower south-west corner

Tips.

- Ask for its faces **by name** with **Faces**: *south*, *east*, *north*, *west*, *roof*, *base*, or all the *façades* at once. Then divide a façade with a **Grid** to put panels on it.
- **Faces** works on a **Box** too.

Sphere



What for. A dome, a reference volume, a zone made visible.

How. Write its **centre** and its **radius**.

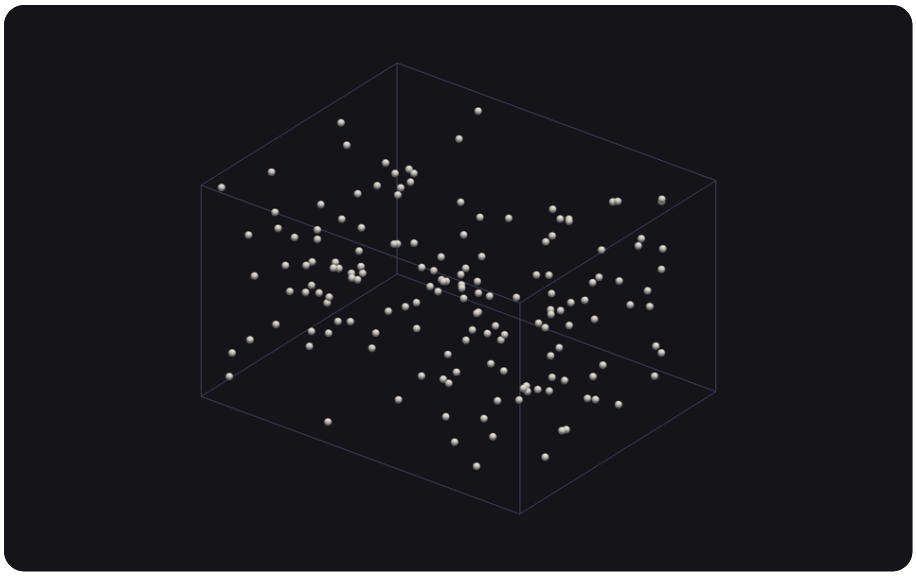
What it does.

Input	What it means
centre	the point in the middle (`0 0 0` by default)
radius	in metres

Tips.

- Several radii give several spheres around the same centre.

Random points



What for. Trees on a plot, holes scattered on a panel, people in a square, the seeds of a Voronoi.

How. Connect a **region**, a box, a prism or a rectangle, and write how many points you want.

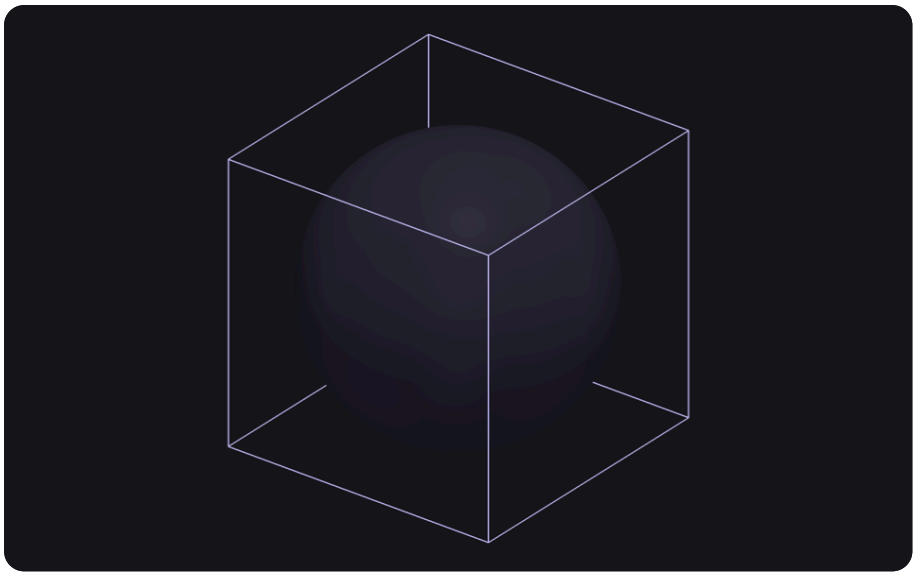
What it does.

Input	What it means
region	where the points fall: inside a volume, or on a rectangle in plan or in elevation
count	how many points
seed	the same seed always gives the same points; another seed, another layout
level	the name of the list of points (`point` by default)

Tips.

- Try a few **seeds** until you like the layout: it will not change again unless you change the seed.
- Connect the points to a **Voronoi** to turn them into cells.

Bounding box



What for. To know how big something is and where it is, or to make the cell that deforms a geometry with **Morph**.

How. Connect the **geometry** to wrap.

What it does.

Input	What it means
geometry	what to wrap
of	*all*: one box around everything · *each*: one box around each element

Tips.

- With **Morph**, move the corners of the box and the geometry inside follows: a quick way to taper or twist a form.

Curves

Curves are axes, edges, profiles, paths and cables. They are teal on the canvas.

On a curve, a position is a fraction or a distance. Many nodes here ask *where* on the curve: a **fraction** goes from `0` (the start) to `1` (the end), so `0.5` is the middle; a **distance** is in metres from the start, and negative from the end.

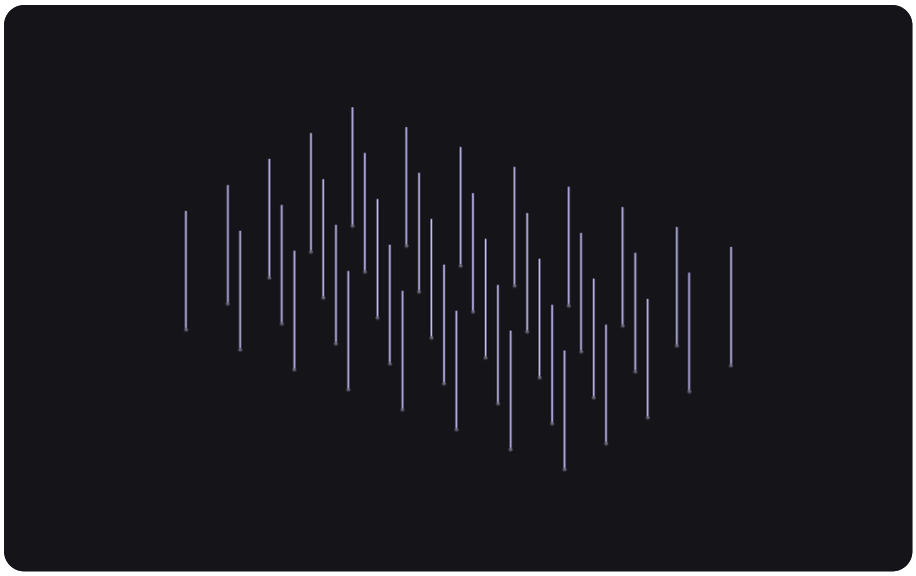
Line



What for. A beam, an edge, an axis between two points.

How. Connect a point to **from** and another to **to**. If the points are lists, you get many lines.

Line by direction

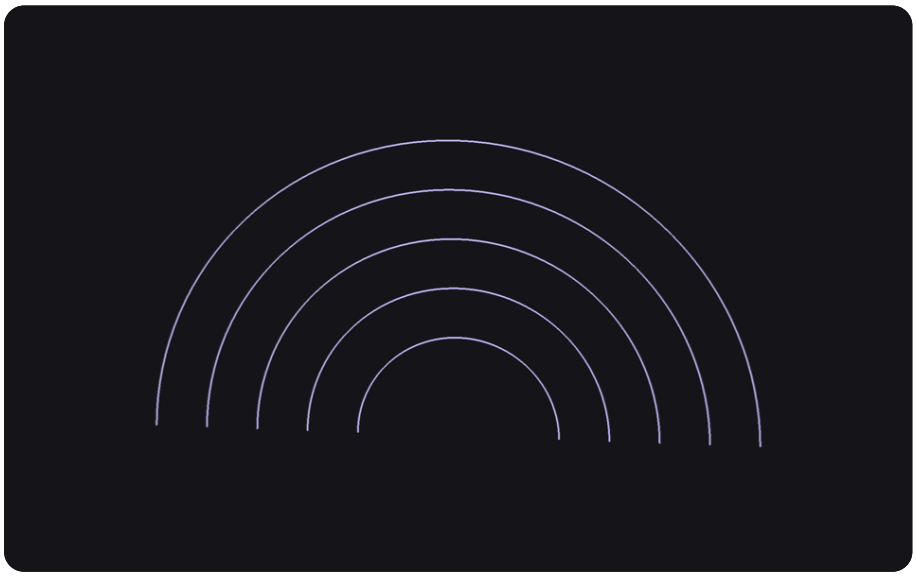


What for. A mast, a column, a perpendicular: a line that starts at a point and goes some way.

How. Connect the point to **from**, give a **direction** and a **length**.

Input	What it means
from	where it starts
direction	which way it goes; upwards by default (`0 0 1`)
length	how long; negative, the other way

Arc



What for. A curved edge, a vault profile, a bend that continues a curve without a kink.

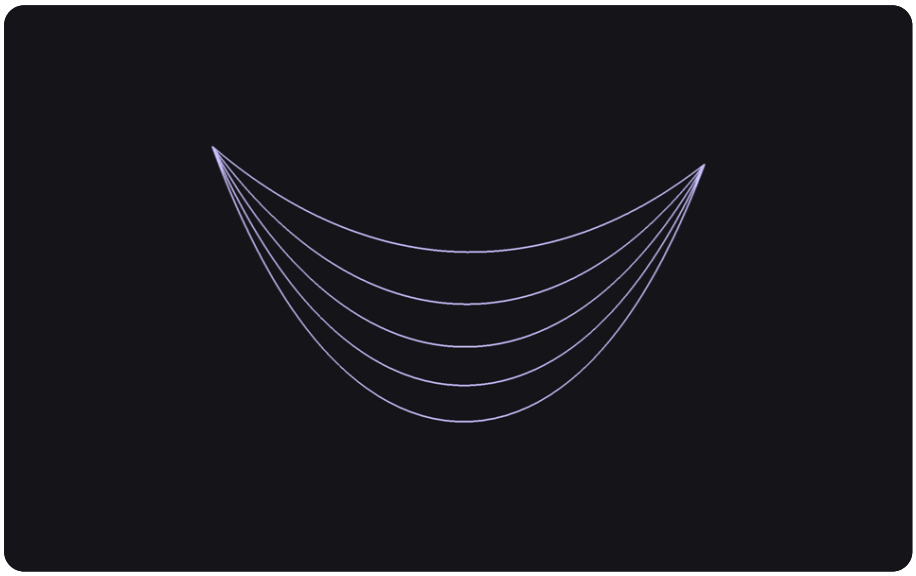
How. Choose the **mode** and give what it asks for.

Mode	What you give
three points	the **start**, a **middle** point it passes through, and the **end**
centre and angles	the **centre**, the **radius**, and the angles **from** and **to** (degrees; 0° towards +X, counter-clockwise)
start, direction and end	the **start**, the **direction** it leaves in, and the **end**: it continues a curve without a kink

Tips.

- To continue a curve smoothly, connect the **Tangent** of its end to **direction**.

Catenary



What for. The exact shape of a chain or a cable hanging between two points. Turned upside down, the arch or vault that works mainly in compression, as Gaudí did with his hanging models.

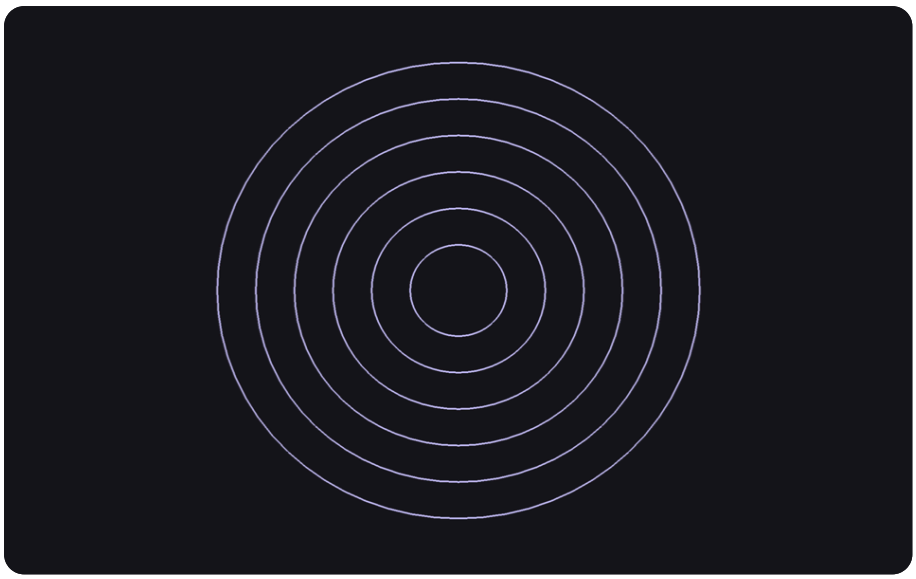
How. Give the **start**, the **end** and the **length** of the cable.

Input	What it means
start, **end**	where the cable hangs from
length	longer, it sags more; shorter than the distance, it stays taut
length in	*metres*, or *times the distance* between the ends (`1.2`: a fifth longer), for many cables at different spans
gravity	where it hangs towards; upwards (`0 0 1`) gives the arch

Tips.

- Its points are evenly spaced along it, ready for **Points on curve** or for hangers.

Circle

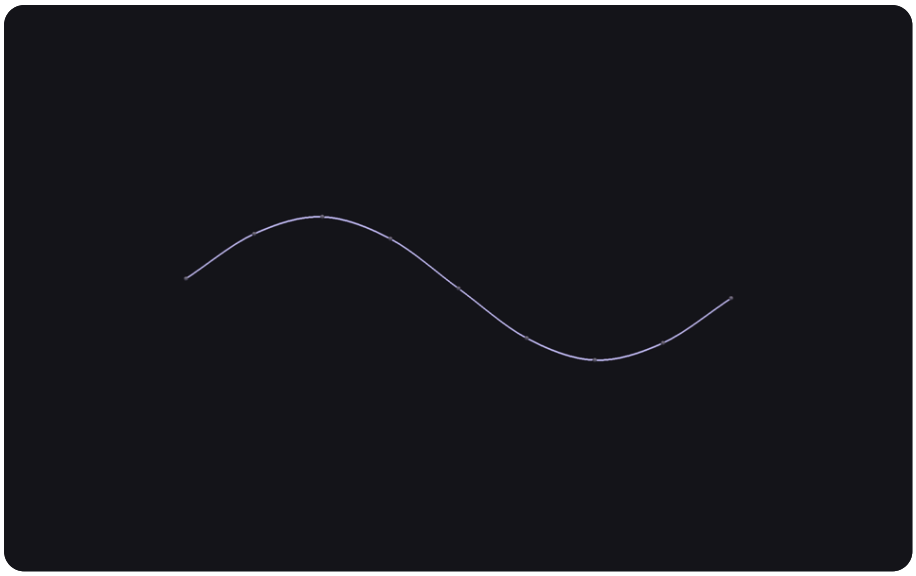


What for. A column section, a round window, an ellipse, a circle touching a curve.

How. *Centre and radius*, or *tangent to a curve*.

Input	What it means
mode	*centre and radius* or *tangent to a curve* (it touches the curve at a point without crossing it)
centre, **radius**	the usual circle
radius 2	empty: a circle; with a value: an ellipse
curve, **at**, **size**, **side**	tangent mode: which curve, where (a fraction or a distance), and on which side
plane	*plan*, *elevation* or *side elevation*
normal	which way it faces; the normals of a surface lay it on the surface

Curve through points



What for. To join points into a curve: a profile, a path, the edge of a roof.

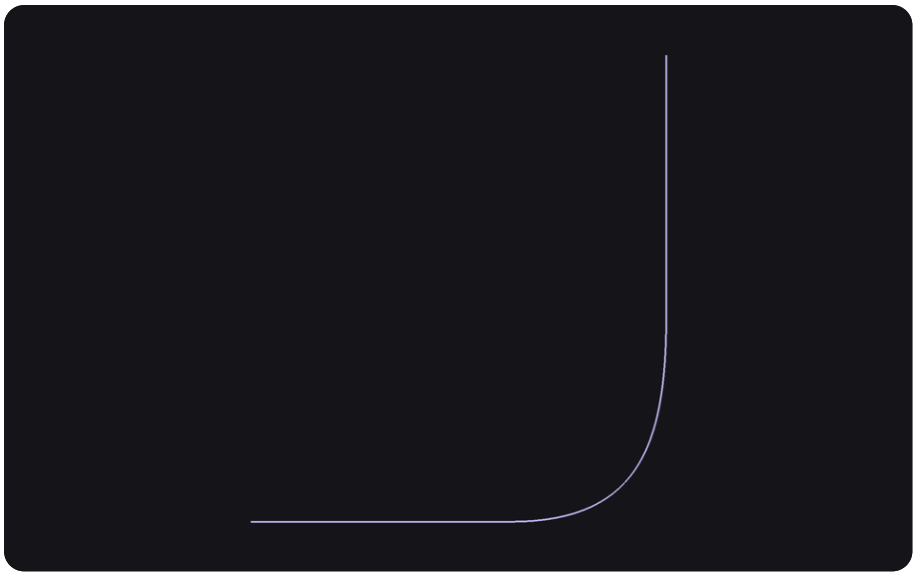
How. Connect the **points**. They are joined along a level, the last one if you leave **along** empty.

Input	What it means
points	the points to join
along	the level of the points to join; empty: the last
smooth	yes: a smooth curve; no: a polyline with corners
closed	yes: back to the first point

Tips.

- With a grid of points, `row × column`, you get one curve per row or per column, depending on **along**.

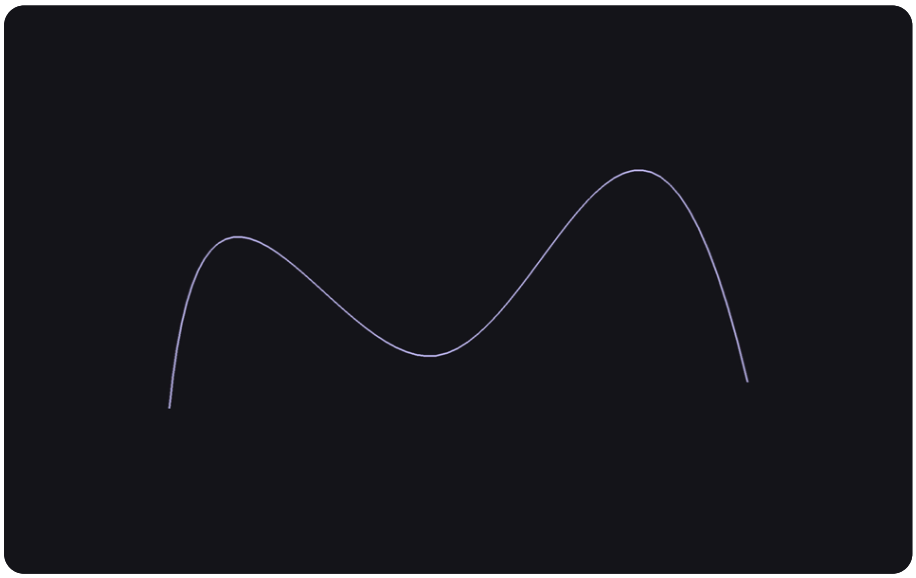
Transition curve



What for. To join the end of one curve to the start of another, smoothly, without kinks.

Input	What it means
a, **b**	the curve it starts from and the one it arrives at
bulge	`0.1`: almost straight · `1`: very wide
join	yes: gives *a + transition + b* as one curve

Drawn curve

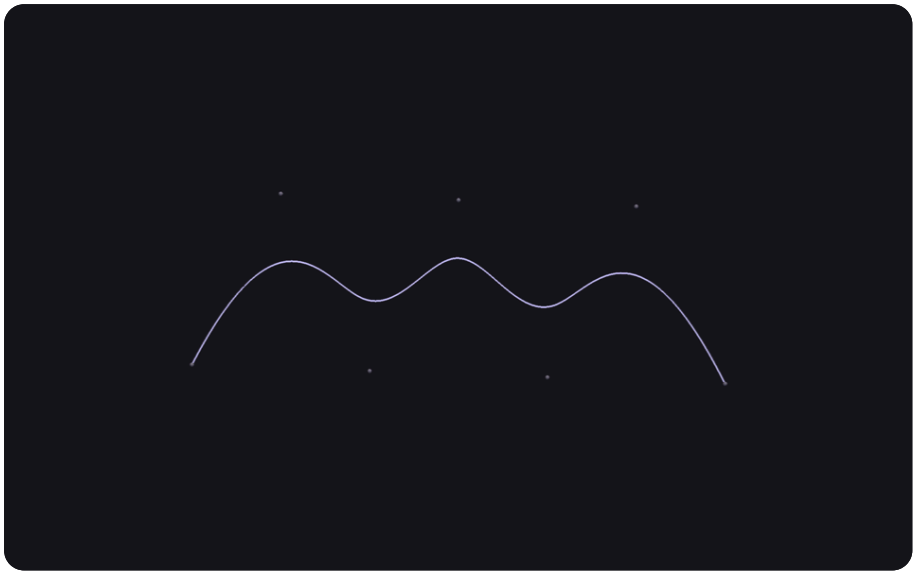


What for. To draw a curve with the mouse, right in the 3D view.

How. Click in the view to place its points, then drag them.

Input	What it means
mode	*through points*: the curve passes through them · *by control points*: they pull it like magnets
degree	`1`: a polyline · `3`: smooth · up to `5`: tauter
closed	yes: a closed curve

NURBS Curve

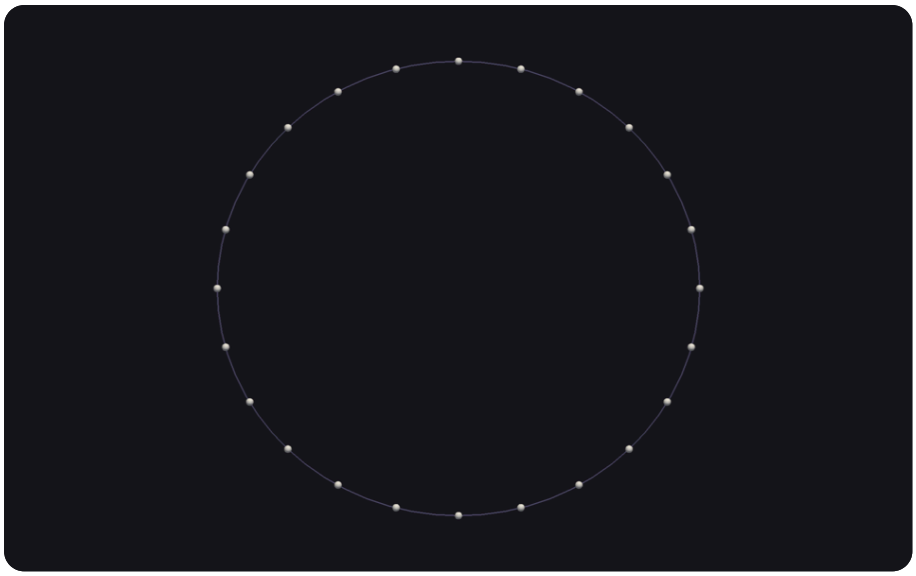


What for. Full control of a free curve: its control points pull it like magnets, with degree, weights and knots.

How. Connect some **points**, or draw them with **Edit curve**. In *pen* mode, place anchors it passes through, with handles that shape it.

Input	What it means
mode	*control points* or *pen*
degree	`1`: the polyline of the points · `2`: arcs · `3`: smooth · up to `7`: tauter
weights	how much each point pulls: `1` is normal, more brings the curve closer
knots	how the curve is spread between the points: evenly, by distance, halfway, or by hand
closed	yes: a closed curve

Points on curve



What for. Points along a curve: posts every 3 m, 20 equal divisions, the middle of an edge.

How. Connect the **curve** and choose the **mode**.

Mode	What it gives
parts	**count** points, evenly spread along the curve
every	one point every **distance** metres
fraction	the **positions** as fractions: `0.5` · `0.25; 0.75`
distance	the **positions** in metres: `2; -2` (2 m from each end)

Tips.

- Name the **level** (*post*, *rib*), so the points combine well later.

Tangent



What for. The direction of a curve at some points: where it goes, or its perpendicular. To orient pieces along a path, or to continue a curve.

How. Connect the **curve**, choose **what** (*tangent* or *normal*) and where, as in Points on curve.

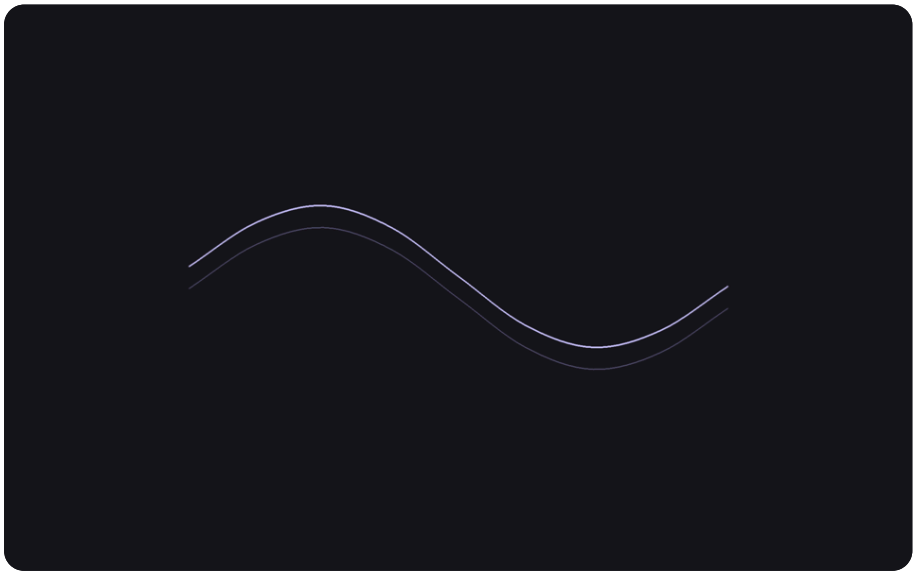
Tips.

- With **Points on curve** at the same positions, you get each point and its direction.

Flip curve

What for. To reverse curves, so they start where they ended. With a **direction**, all of them end up going the same way: before a **Loft**, so it does not come out twisted.

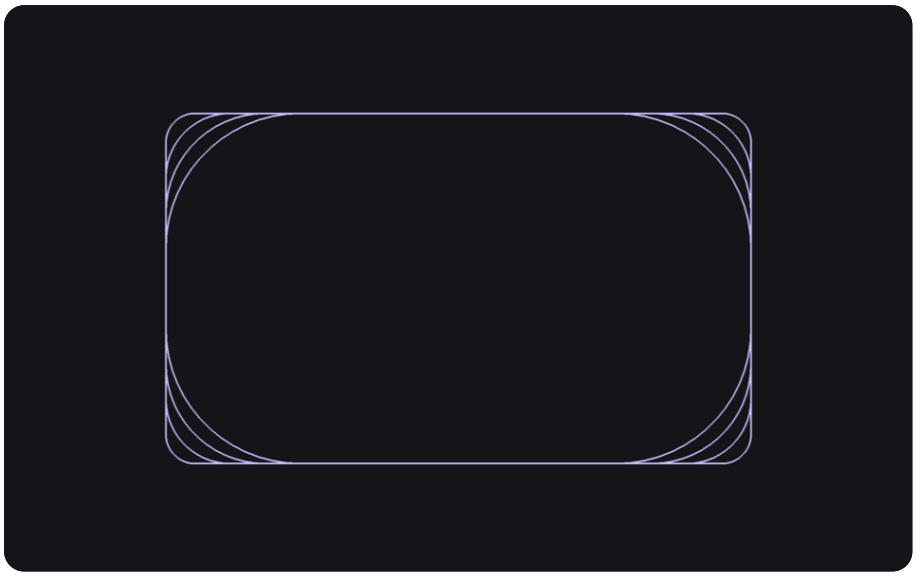
Cut curve



What for. To cut each curve into pieces and work on them one by one.

Input	What it means
mode	*parts*, *every*, *fraction*, *distance*, or *at points* (where some points fall)
count, **distance**, **positions**, **points**	the values of each mode
level	the name of the list of pieces (*segment* by default)

Rounded rectangle



What for. The outline of a rectangle with rounded corners, in plan: a table, a pool, a plot.

Input	What it means
width, **height**	its size
radius	of the corners
origin	the lower-left corner

Part of curve



What for. The piece of a curve between two fractions of its length: an arc of a circle, a piece of an edge.

How. Give **from** and **to**, from `0` (the start) to `1` (the end). On a closed curve it can wrap around.

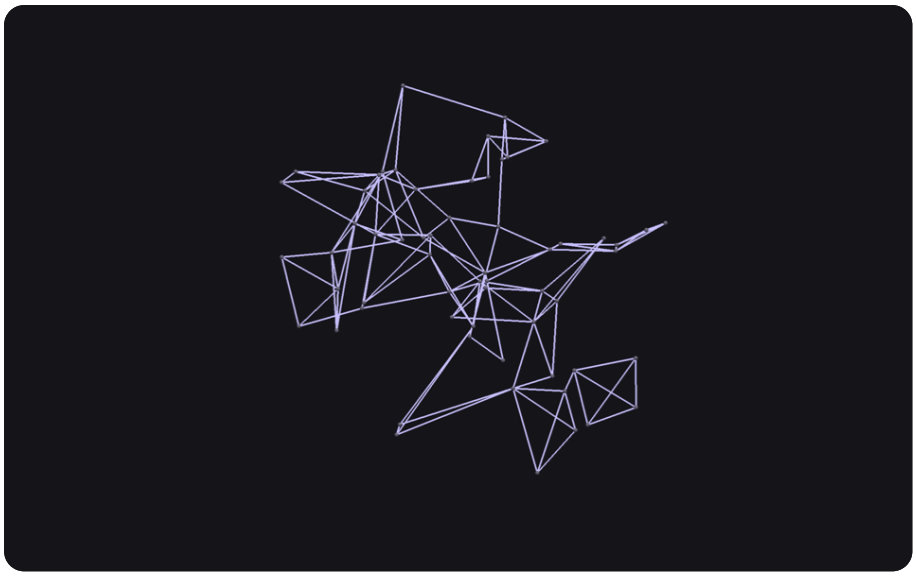
Arrow



What for. To see a vector, a direction or a force, as an arrow at a point. It also works as a line.

Input	What it means
from	where the arrow starts
vector	the direction and size
scale	to see it longer or shorter

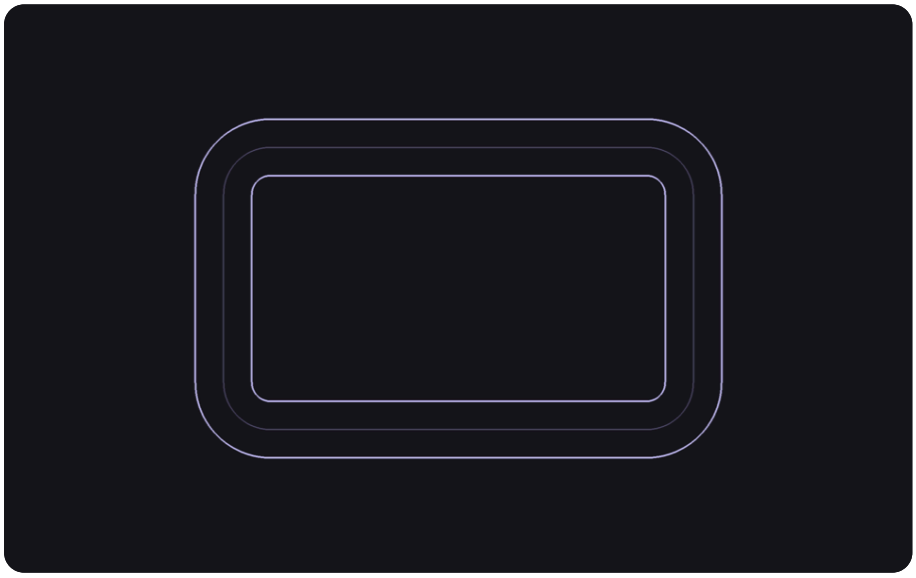
Join nearest



What for. A network of bars: each point joined to its nearest neighbours.

How. Connect the **points** and say how many neighbours (**count**) each one joins.

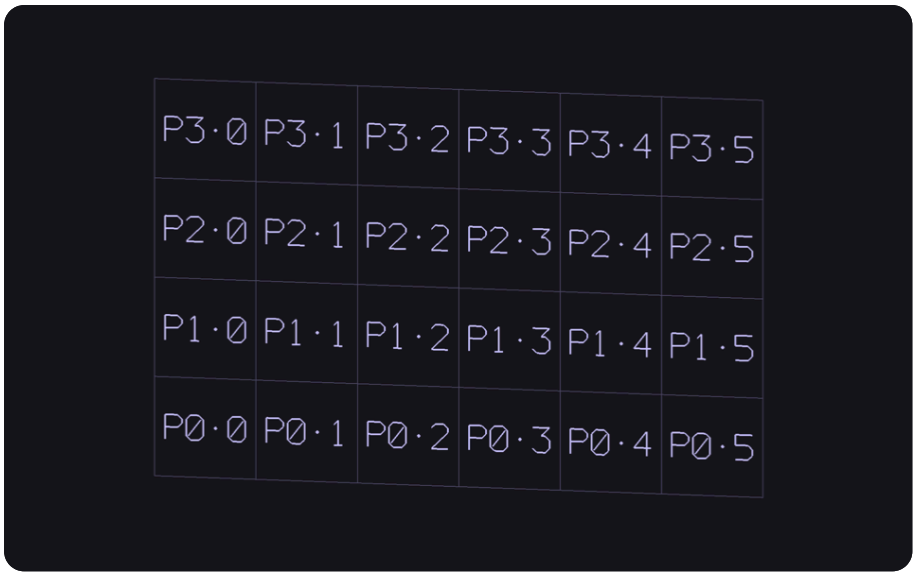
Offset curve



What for. The outline of a wall from its axis, a border around a plan, the second side of a piece to cut.

Input	What it means
curve	one or more flat curves
distance	closed curve: outwards (negative: inwards) · open curve: to its left, seen from above (negative: to its right)
corners	for closed curves: *round* or *sharp*

Text



What for. A label on each element: the number of a panel, the letter of a brick type, an area. Seen in 3D, and exported in drawings and cutting files.

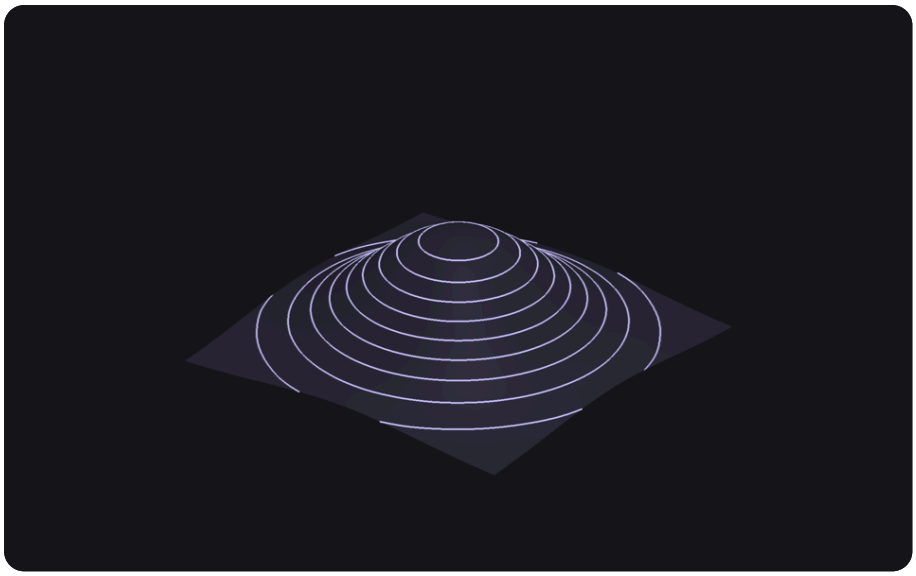
How. Connect **at** (points, or the panels, bodies or curves to label) and write the **text**.

Input	What it means
text	`{n}` : the number of each element · `{floor}` , `{row}` …: its number in that level · `{value}` : the number given in **value** · `A; B; C` : one text per element · `|` : a new line
value, **decimals**	the numbers to write with `{value}` , and how many decimals
height	of the capital letters
plane	*plan*, *elevation*, *side elevation*, or *on the face* (lying on each panel)
lettering	*plain* or *rounded*: single-stroke letters, to engrave · *font*: an installed font, as outlines
font, **align**	which font, and how lines align

Tips.

- `P{n}` writes `P0, P1, P2…` ; `F{floor}-{bay}` writes `F0-0, F0-1…` ; `{value} m2` writes each area.

Contour



What for. Cuts a surface or a volume with parallel planes at a set spacing: contour lines of a terrain, slabs of a tower, rings.

Input	What it means
geometry	a surface, a volume or a mesh
distance	how often to cut
direction	which way the cuts advance; upwards: horizontal cuts
start	the height of the first cut; empty: the lowest point
level	the name of the list of cuts (*cut* by default)

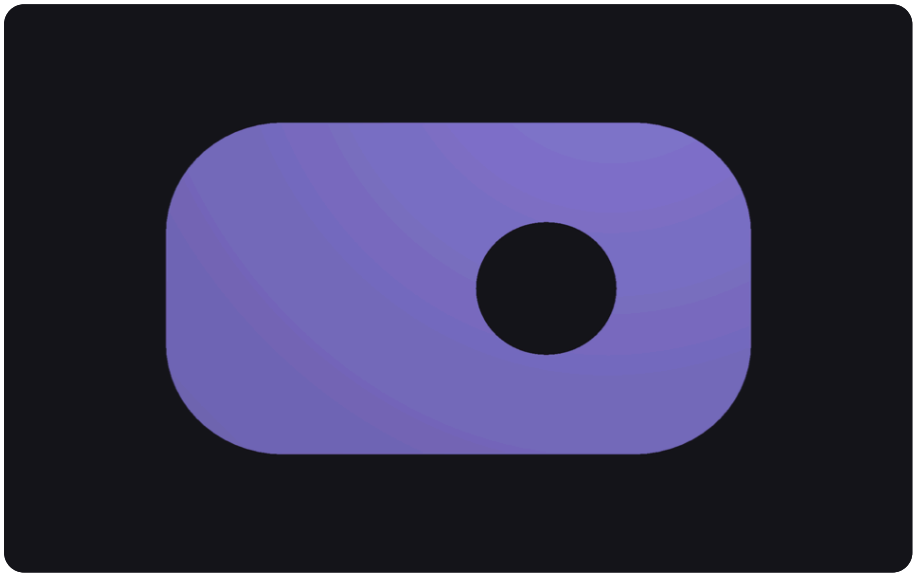
Surfaces

Surfaces are roofs, walls, shells, skins and slabs. They are petrol blue on the canvas.

On a surface, a position is u and v. A surface has two directions, like the rows and columns of a sheet: **u** along the rows and **v** along the columns, each from `0` (one edge) to `1` (the other). `u 0.5, v 0.5` is the middle.

Make a surface

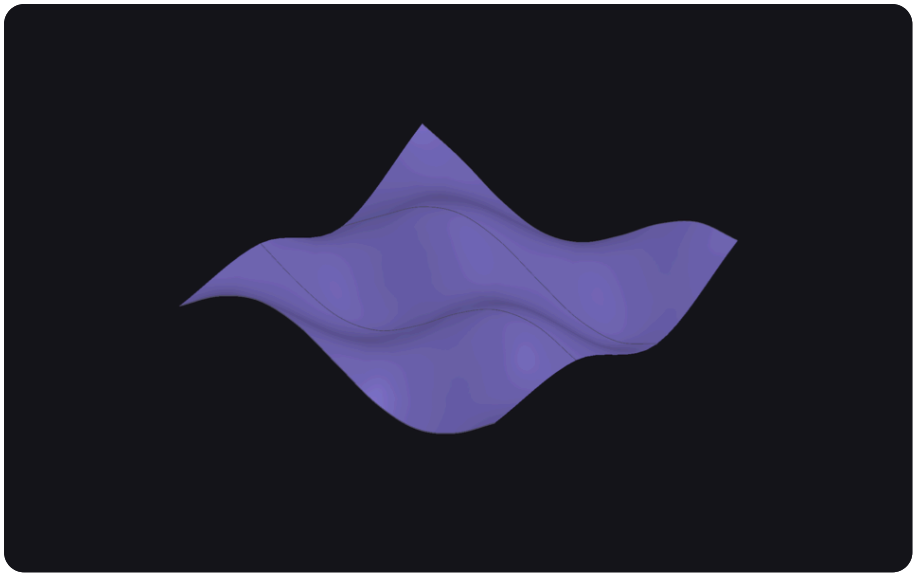
Cap



What for. To close a closed curve with a surface: a roof, a free-form slab, a slab with a courtyard.

Input	What it means
curve	the closed outline
holes	optional: closed curves inside it (a courtyard, a skylight)

Loft



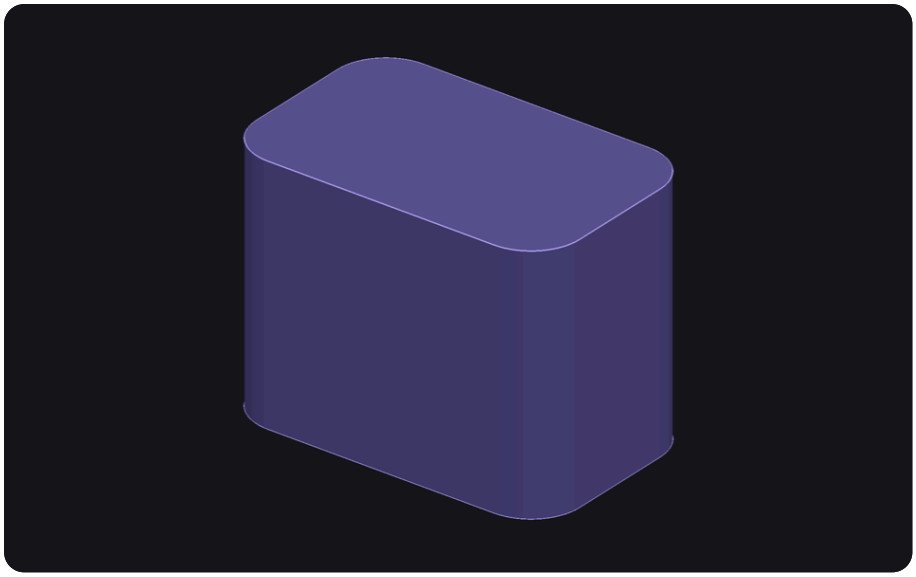
What for. A surface through several curves in order: a wavy wall through three profiles, a roof through its sections.

Input	What it means
curves	the sections, in a list
along	the level of the sections; empty: the last
smooth	yes: smooth through the sections · no: straight between them
closed	yes: the last section goes back to the first

Tips.

- If it comes out twisted, some curves go the other way: put a **Flip curve** with a direction before it.

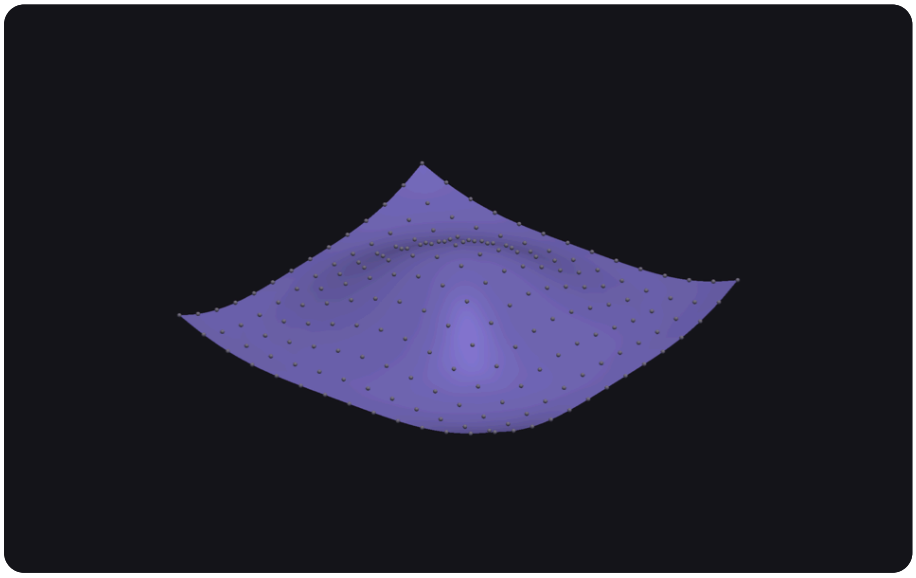
Extrude



What for. To drag curves into a sheet or a wall; with caps, a closed plan becomes a volume. Or to bring them to a point: a pyramid, a cone, a hip roof.

Input	What it means
mode	*vector*: dragged · *to a point*: meets at an apex
vector	which way and how far; 1 m up by default
point	the apex (to a point)
caps	yes: a closed curve gives a closed body

Surface from points



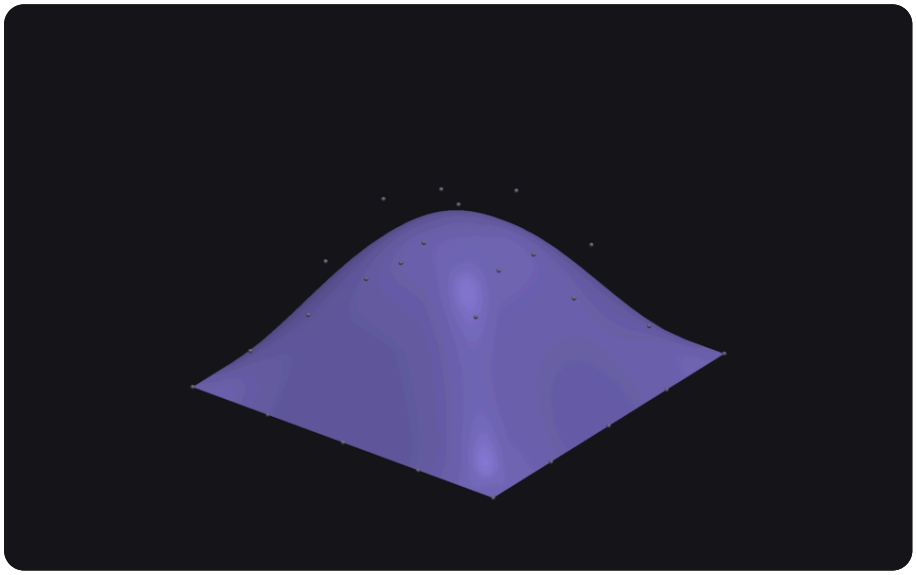
What for. A surface through points: a grid of points (rows × columns), or a loose cloud like the points of a land survey.

Input	What it means
points	in a grid (two levels) or loose
mode	*grid* or *cloud* (joined in triangles)
smooth	yes: curved between the points · no: flat facets

Tips.

- A **cloud** gives flat triangles, which is right for a survey: between two measured heights nothing more is known. With **smooth**, each triangle curves, still passing exactly through every height.
- Shapes with sharp edges show small steps when smooth: use more points, or turn smooth off.

NURBS Surface



What for. Full control of a free surface: a grid of control points that pull it, for shells and free skins.

Input	What it means
points	the control points, rows × columns
level u, **level v**	which levels are rows and columns
degree u, **degree v**	`1` : facets · `2` : curves · `3` : smooth, up to `7`
weights	how much each point pulls
closed u, **closed v**	all the way round: a tube; in both, a torus

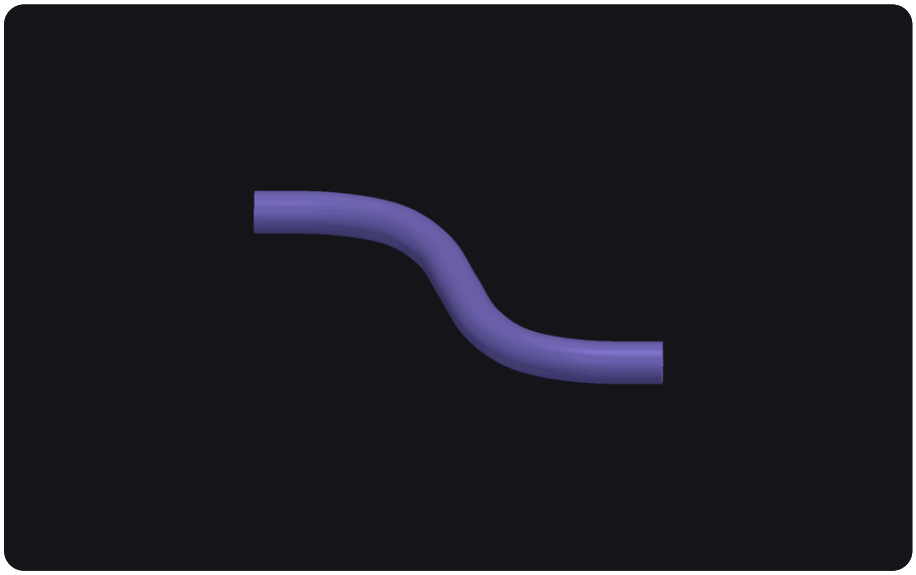
Revolve



What for. A profile turning around an axis: a column, a dome, a vase, a tower.

Input	What it means
profile	the half section that turns
origin, **axis**	a point on the axis and its direction (vertical by default)
degrees	how much it turns: `360` a full turn

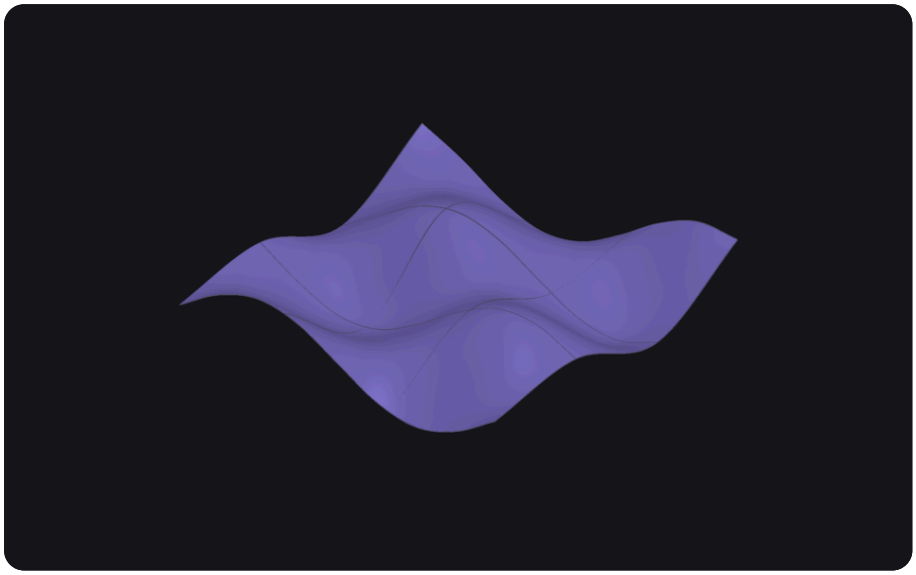
Sweep



What for. A profile running along a path without twisting: a handrail, a moulding, a curved beam, a tube.

Input	What it means
profile	the section that runs
path	the curve it runs along
path 2	optional: a second path; the profile stretches to touch both
place	*where it is* (drawn at the start of the path) or *centred on the path* (drawn anywhere)
rotation	how many degrees it turns along the whole path

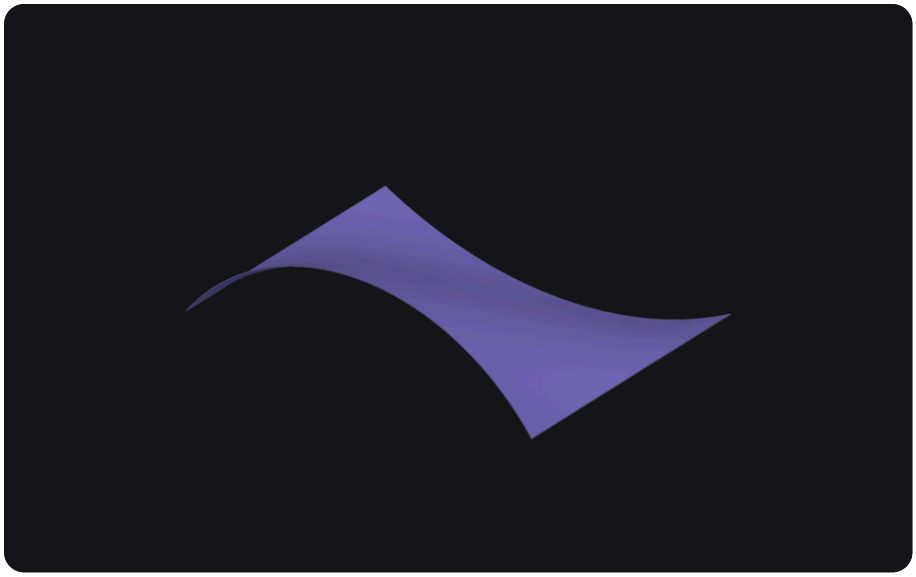
Network surface



What for. A surface through a network of curves in two directions: the ribs of a hull or of a free roof.

How. Connect one family of curves to **curves u** and the other to **curves v**; each curve of one family must cross all those of the other.

Edge surface

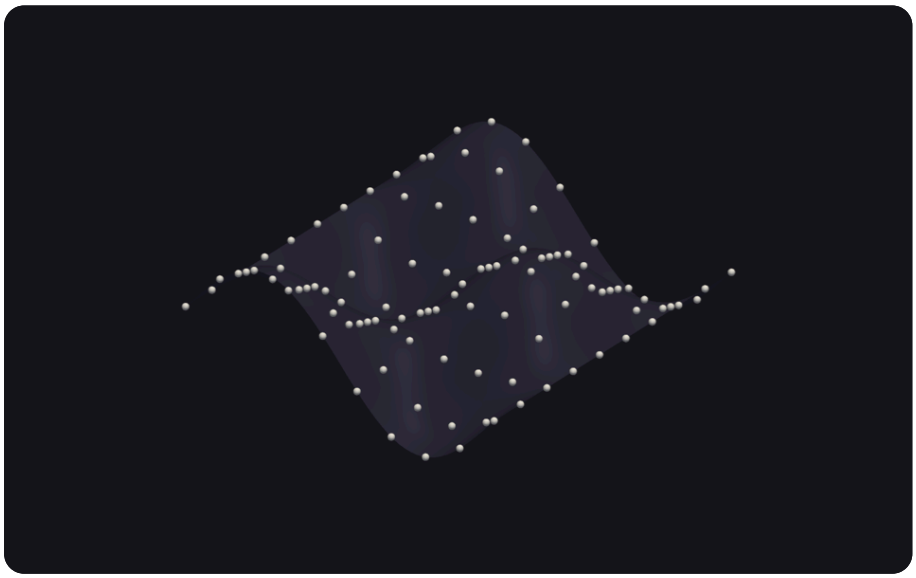


What for. A surface stretched between two, three or four edges that meet at the corners: a slab or a roof with curved edges.

How. Connect the edges to **a**, **b**, **c** and **d**, in any order. With two, a ruled surface; with three, one side closes into a point.

Work on a surface

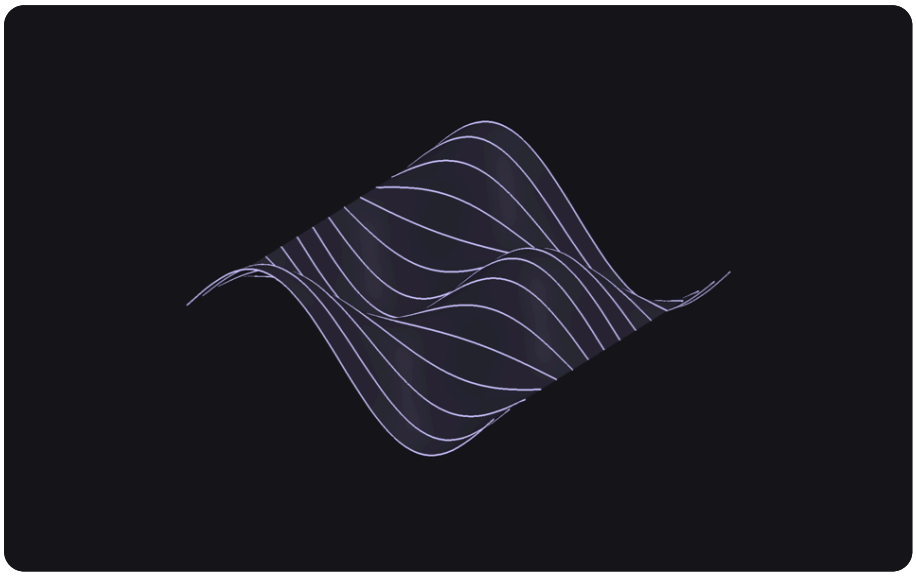
Surface points



What for. Points on a surface, or the direction it faces there: to stand columns, anchors or pieces on it.

Input	What it means
what	*points* or *normals* (where it faces)
mode	*lattice*: count u × count v from edge to edge · *centres*: the centre of each panel · *fractions*: at some u and v
count u, **count v**	how many along each direction
u, **v**	the fractions, several with `;`

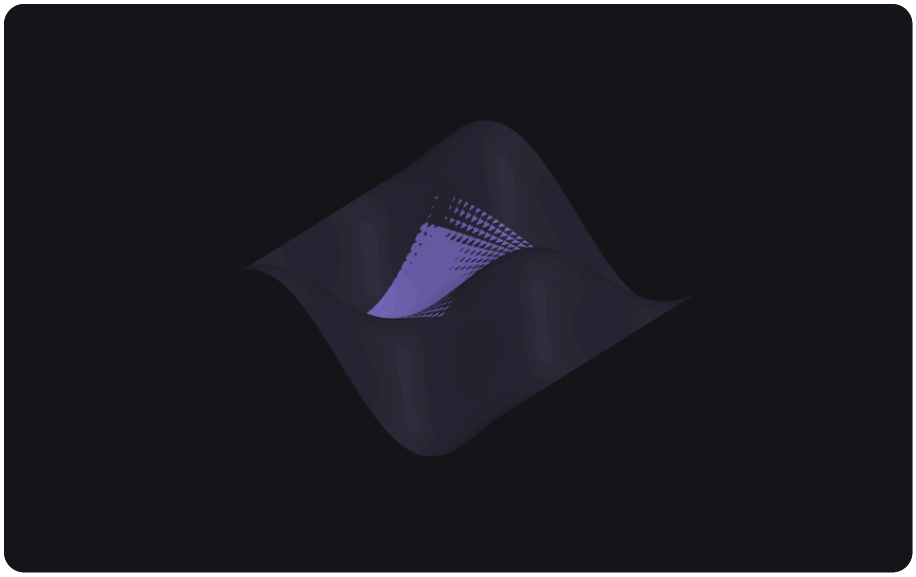
Isocurves



What for. The surface's own lines, in u or in v: ribs, mullions, a layout that follows the shape.

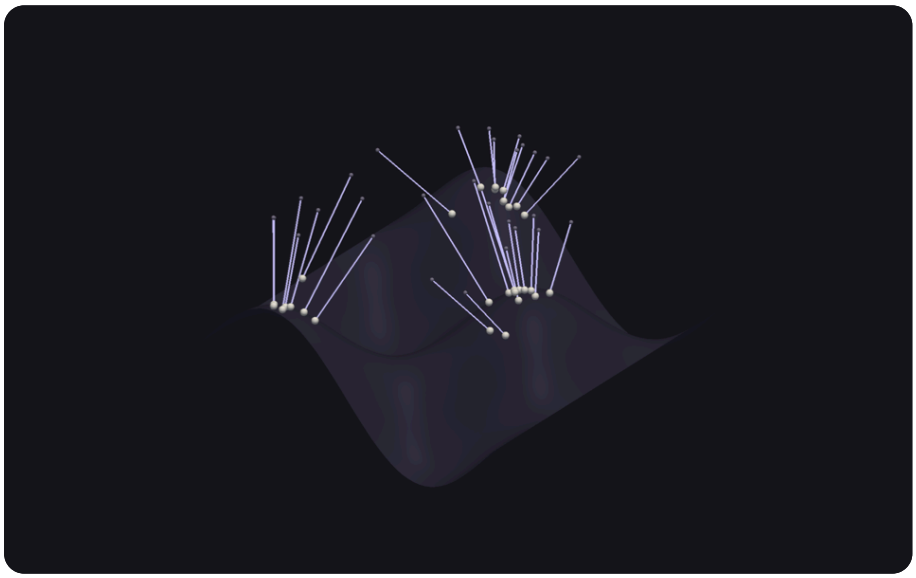
Input	What it means
direction	*u* or *v*
mode	*parts* (spread evenly) or *fraction* (at some positions)
count, **positions**	how many, or where

Sub surface



What for. The piece of a surface between two fractions of u and two of v: a bay of a roof, a segment of a dome.

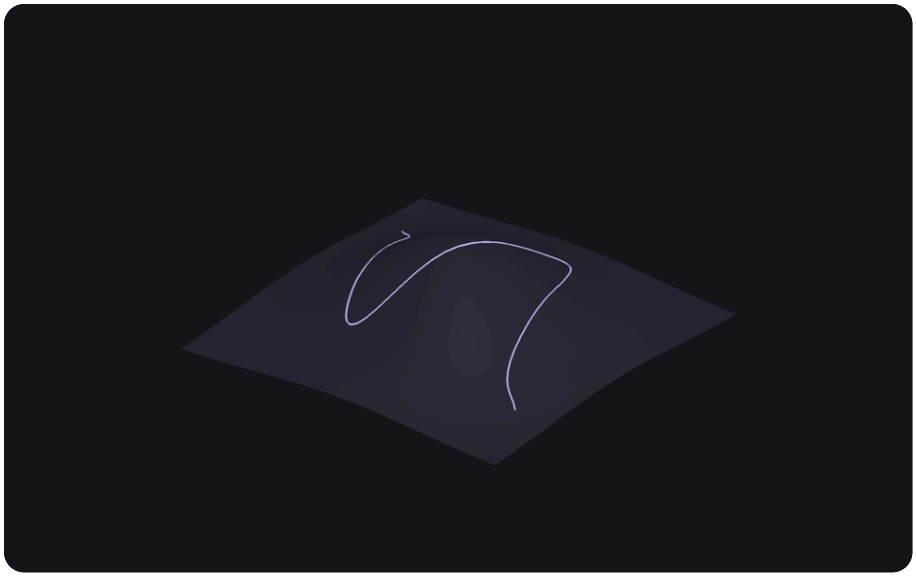
Closest point on surface



What for. The point of a surface closest to each point: to bring something onto it, or to know its u and v there, how far it is, or where the surface faces.

How. Connect the **points** and the **surface**, and choose what it **gives**: *point*, *u v*, *distance* or *normal*.

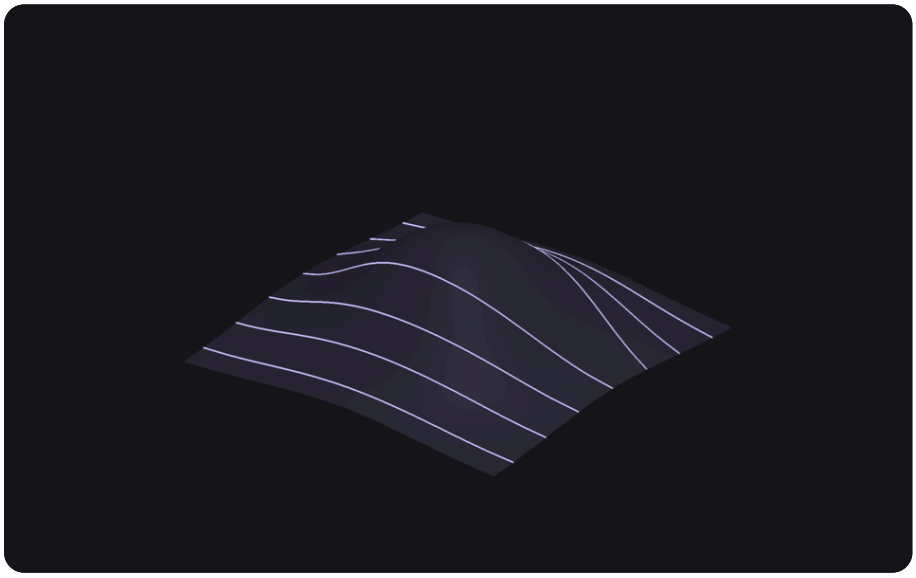
Curve on surface



What for. A curve lying on a surface and following every bump: a path on a roof, a seam on a membrane.

How. Give points by their u and v, from **Closest point on surface** (*u v*) or with a **Point** where x is u and y is v.

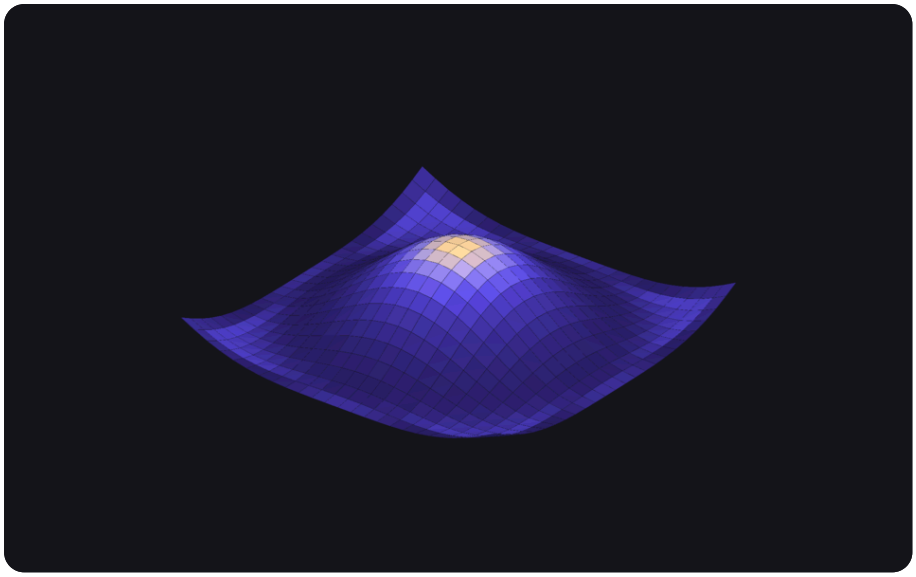
Geodesic



What for. The shortest path between two points over a surface, like a string pulled tight on it. Laths laid along geodesics stay straight in their own plane: that is why gridshells use them.

Input	What it means
from, **to**	the start and end points, one end for each start
show	*curves*, their *lengths*, the *crossings* (the knots of a gridshell) or the *bars* between crossings

Surface curvature



What for. To know whether a surface can be built from flat sheet, and how much it bends.

Input	What it means
what	*Gaussian*: zero, made from flat sheet; positive, like a dome; negative, like a saddle · *mean*: how much it bends · *developable*: yes or no
mode, **count**, **u**, **v**	where, as in Surface points

Tips.

- To see the curvature as colours on the surface, use the panel of the geometry (**Colour → by a value**).

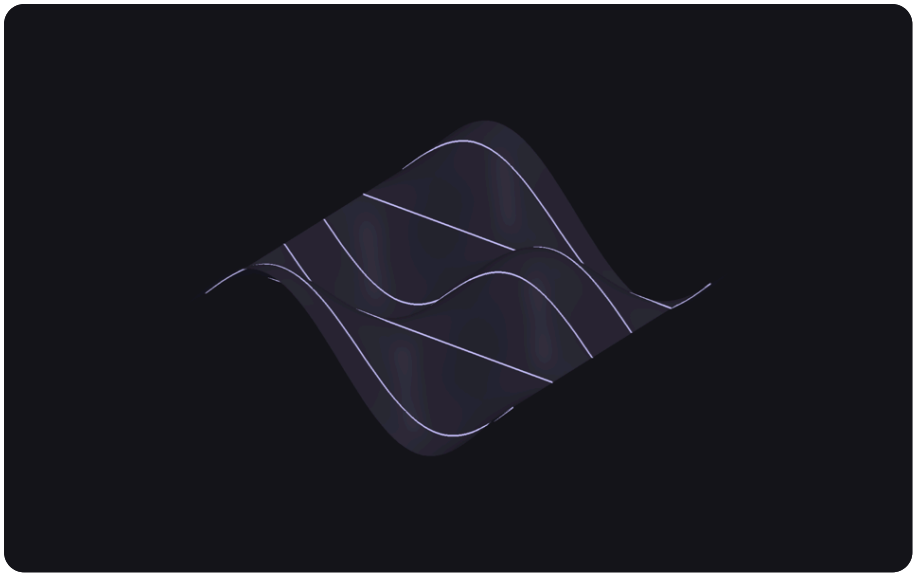
Flip surface

What for. To turn a surface the other way (its face), or to reverse or swap its u and v.

Tips.

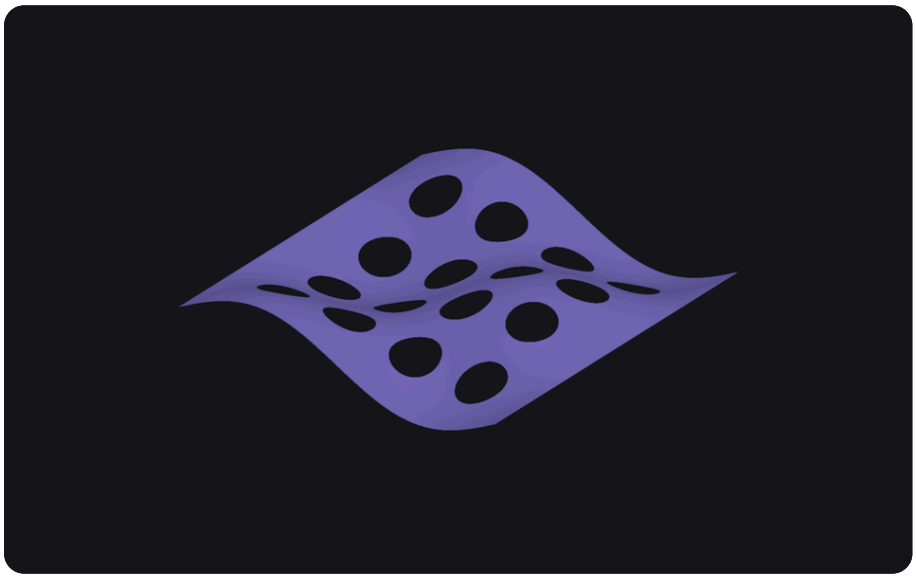
- **Thicken** grows towards the face: flip it to grow the other way.

Intersection



What for. Where two geometries cross: two surfaces give the crossing curves; a curve and a surface, the points.

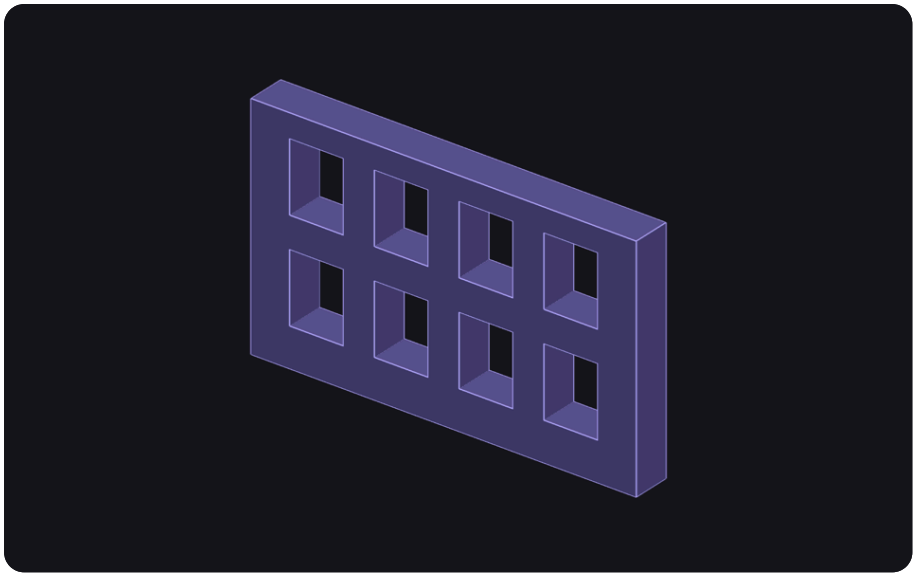
Split surface



What for. A surface split into pieces by curves projected onto it: a perforated sheet and its offcuts.

Input	What it means
curves	the curves that cut it, projected from **direction** (from above by default)
what	*all pieces* (largest first), *the largest* (the perforated sheet) or *the others* (the offcuts)

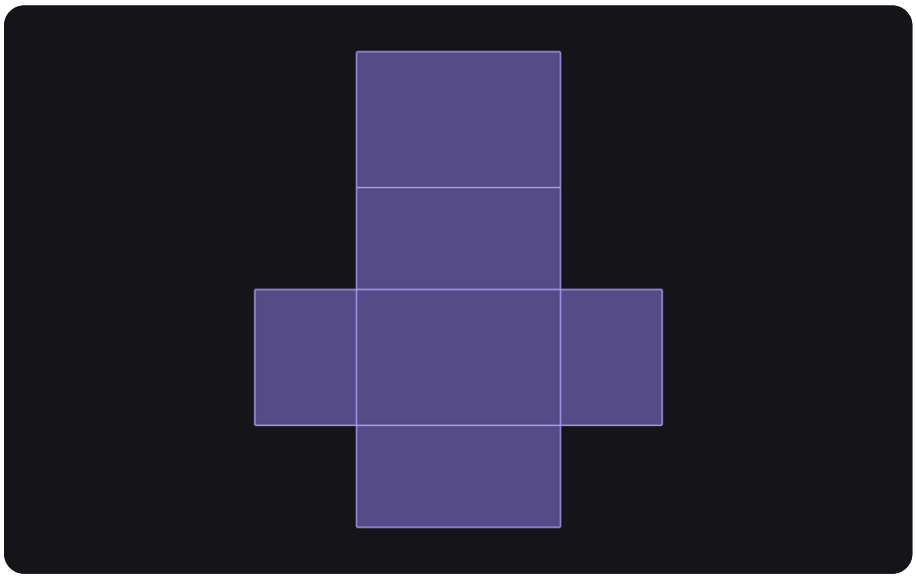
Boolean



What for. To unite, subtract or intersect closed bodies: a wall minus its openings, two volumes in one skin.

Input	What it means
a, **b**	closed bodies
operation	*union*, *difference* (a minus b) or *intersection*

Unroll



What for. To flatten surfaces to cut them from a sheet of metal, board or fabric. A body with flat faces gives its net, to cut and fold.

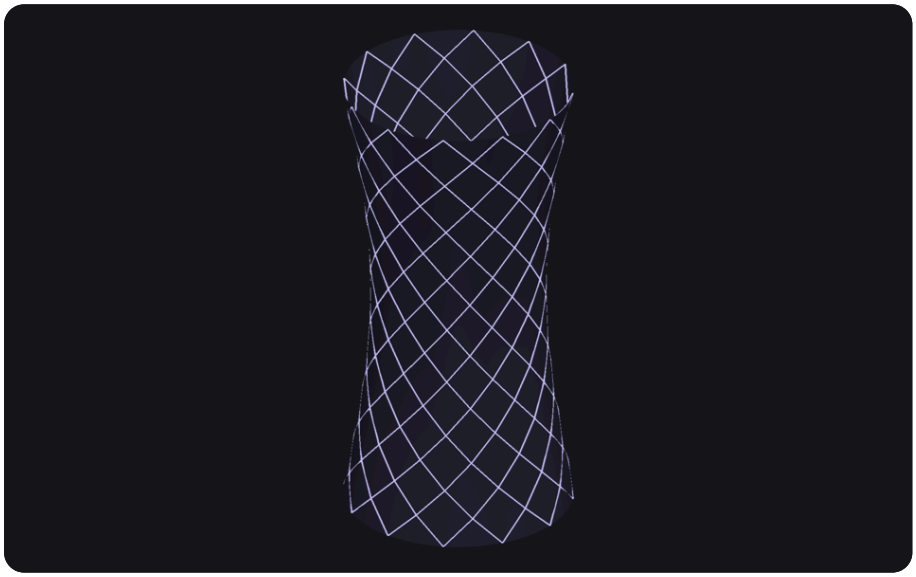
Input	What it means
what	*unrolled* pieces, their *outlines* to cut, the *folds* of a net, or the *stretch* of a surface that is not developable
place	*in a row* or *in place*
spacing	between pieces

Tips.

- A developable surface unrolls without stretching; one that is not, like a dome, unrolls approximately and says how much it stretches.

Structures on a surface

Diagrid



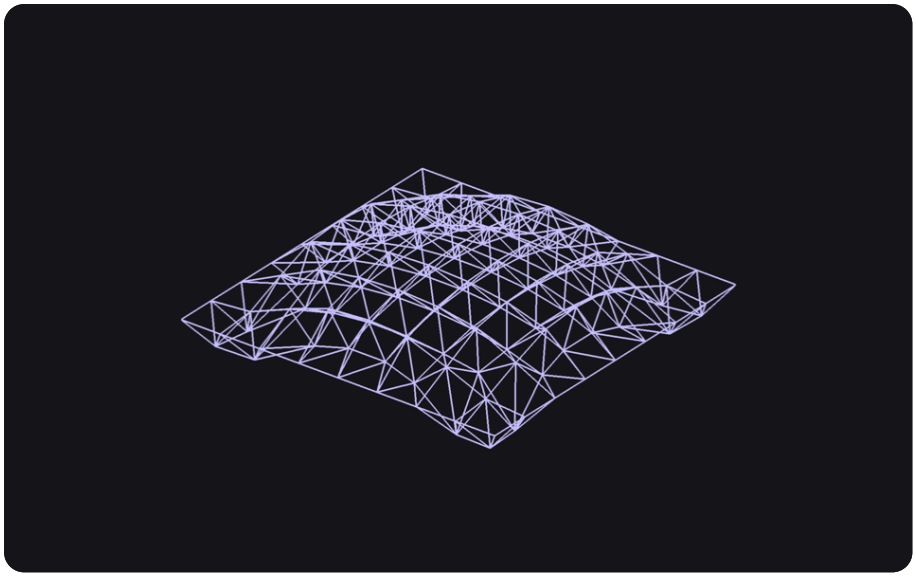
What for. A grid of diagonal members on a surface, in diamonds: a tower, a dome, a roof.

Input	What it means
divisions u, **divisions v**	diamonds along each direction
what	the *diagonals*, the *crossings* (nodes) or the *rings* through the nodes
members	*straight* or *curves* following the surface
continuous	yes: each diagonal from end to end as one member (a helix on a tower)

Tips.

- With **Pipe** on the diagonals, the structure; with **Cap** on the rings, the floor slabs.

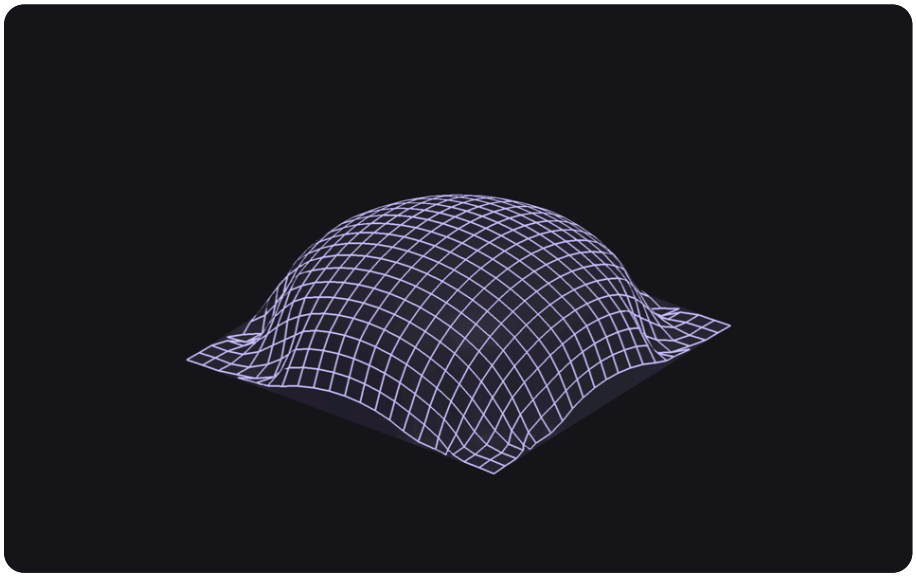
Space frame



What for. A two-layer structure on a surface: the top chord on it, the bottom one shifted half a module and lowered, and diagonals joining them in pyramids.

Input	What it means
divisions u, **divisions v**	modules along each direction
truss depth	the distance between the two chords
what	all members, one chord, the diagonals, or the crossings above or below

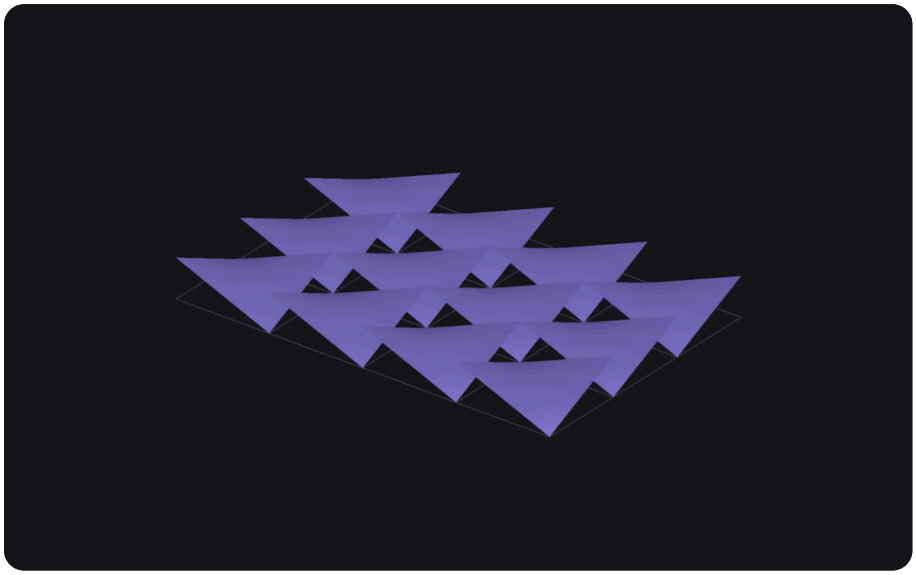
Chebyshev net



What for. The way a mesh of laths settles when bent, or a fabric when it drapes: quadrilaterals with four equal sides. For lath roofs and gridshells.

Input	What it means
u, **v**	the starting point
step	the length of each side, between nodes
count	nodes on each side of the start
angle, **rotation**	the angle between the two starting laths, and their turn
what	all laths, one family, or the knots

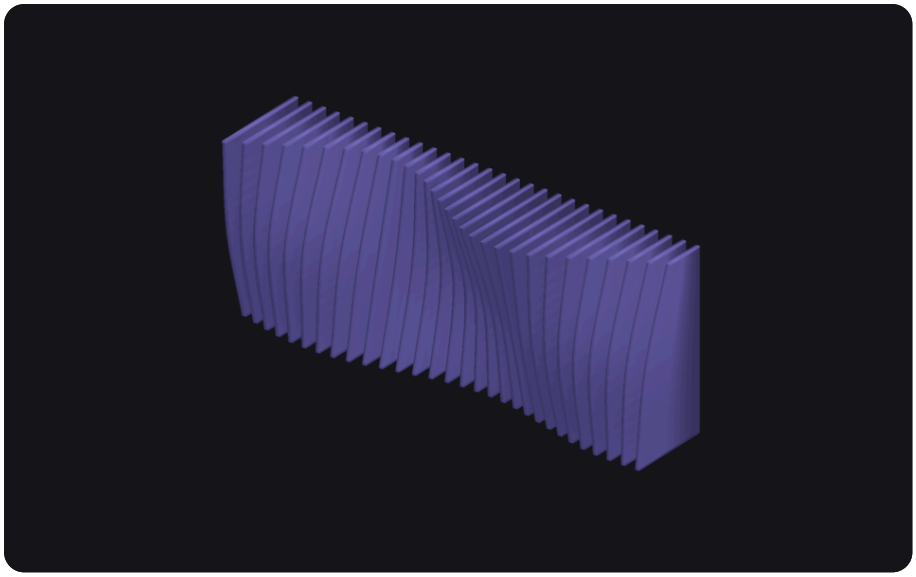
Hypar



What for. To fold each panel into a hyperbolic paraboloid: two opposite corners rise and the panel becomes a saddle of straight lines, like paper folded along a diagonal.

Input	What it means
panels	four-corner panels: a Grid, rectangles
lift	how much two opposite corners rise, in metres
corners	which two corners rise
show	the *folded piece*, the *flat piece* to cut, the *fold* or the *longest edge*

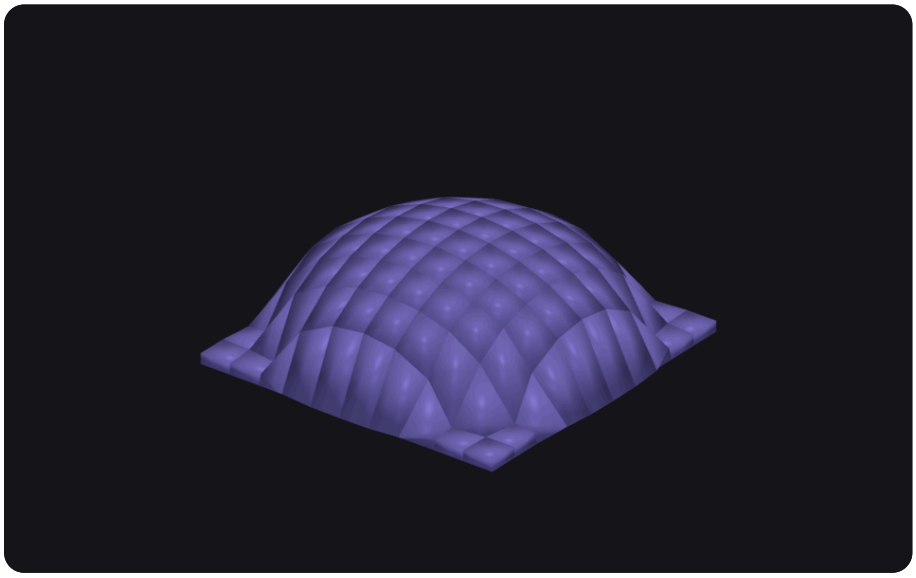
Ribs



What for. To cut a shape into parallel boards closed against a wall: a rib wall, a slatted ceiling, a sliced bench. See Draw the form, then cut it.

Input	What it means
shape	a surface, a mesh or a volume
count, **weight**	how many ribs, and their thickness
cuts	*x*, *y*: vertical ribs · *z*: horizontal slices
back	where the back wall is

Surface morph



What for. To cover a surface with a module: the surface is divided into cells that follow its curve, and the module is fitted into each one, bending with it.

Input	What it means
geometry	the module, drawn anywhere; empty: the cells themselves
count u, **count v**	cells along each direction
height	how far the top of the module rises from the surface

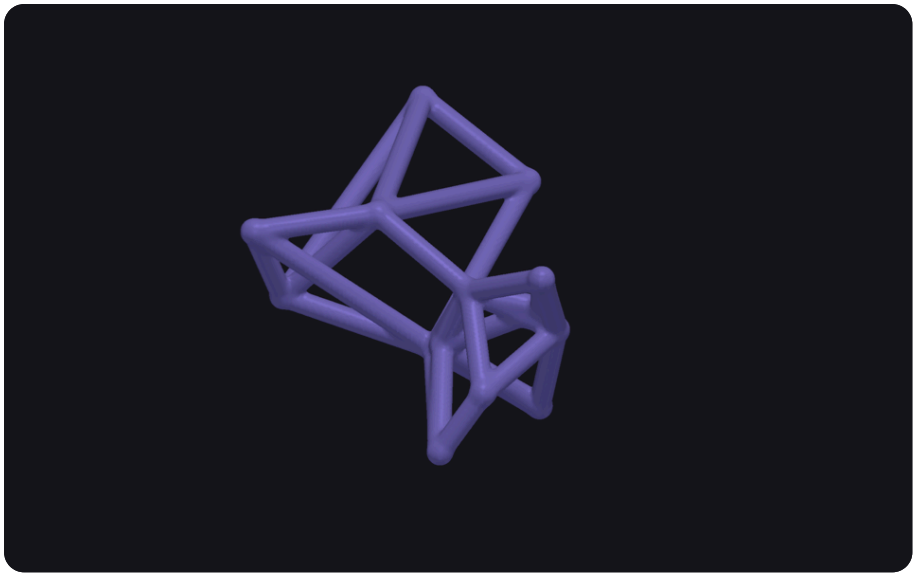
Pipe



What for. A member along each curve: round (railings, cables), rectangular (beams, mullions) or with a profile you draw (an I, an L).

Input	What it means
section	*round*, *rectangular* or *profile*
radius, **sides**	for round sections
width, **height**	for rectangular ones
profile	a curve drawn around the origin, in plan
rotation, **caps**	turn the section; close the ends

Pipes



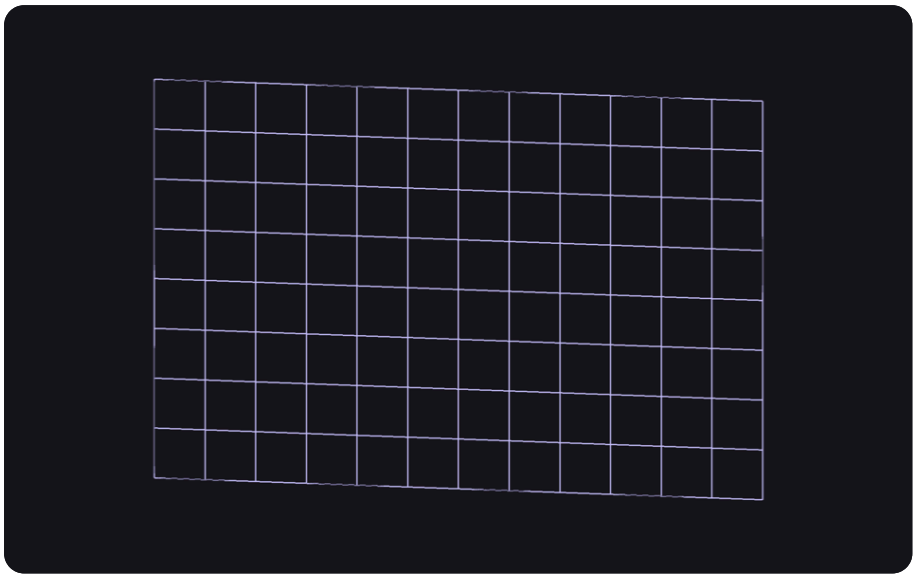
What for. A pipe around each line, all fused into one skin with smooth fillets at the nodes: a single sculptural structure.

Input	What it means
radius	of the pipes
fillet	how much the nodes are filled; `0` : none
detail	more is finer and slower

Divide

Divide is where one thing becomes many: a facade into panels, a box into its faces, a list into groups. These nodes are ochre on the canvas.

Grid



What for. To divide a rectangle or a curved surface into panels: rows and columns, diamonds, hexagons or triangles.

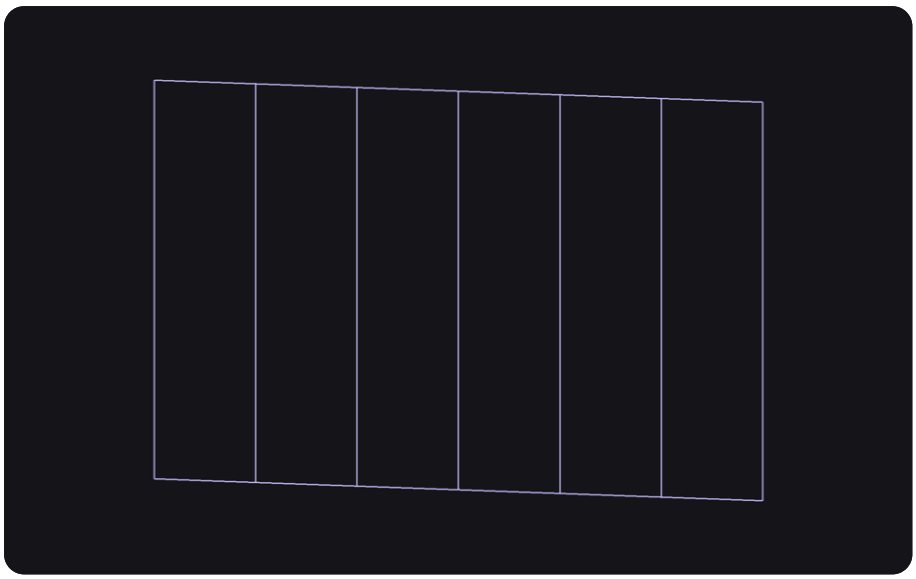
How. Connect the **rectangle** (or a surface) and say how to divide it: by number (**columns**, **rows**) or by size (**cell width**, **cell height**).

Input	What it means
rectangle	a rectangle, or a curved surface
columns, **rows**	how many, or the cuts as fractions (`0; 0.2; 0.5; 1` : uneven)
cell width, **cell height**	the size of each cell, instead of how many
rows level, **columns level**	the names of the two lists (*row* and *column* by default)
shape	*squares*, *diamonds*, *hexagons* (a honeycomb), *triangles*, or *diamond trims* (the triangles that complete the border)
panels	on a curved surface: *flat* (four corners) or *curved* (following the surface)

Tips.

- Uneven cuts can come from another node: a **Formula** gives cells that grow.
- The cable reads `row 12 × column 20 = 240`: everything after the Grid works on all the panels at once.

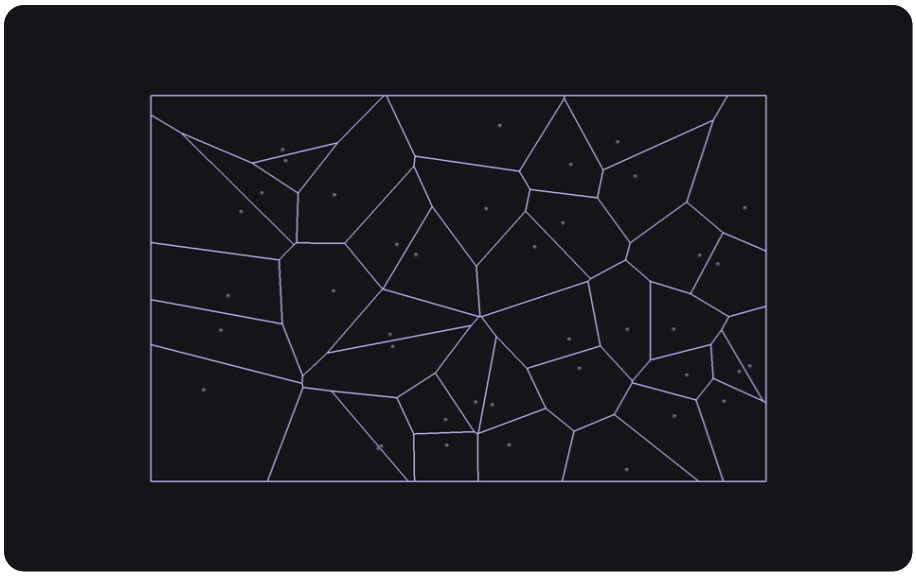
Divide



What for. To divide a rectangle into parts in one direction only: bays across a facade, strips of a floor.

Input	What it means
rectangle	what is divided
parts or **size**	how many parts, or the size of each
direction	across the *width* or the *height*
level	the name of the new list (*part* by default)

Voronoi



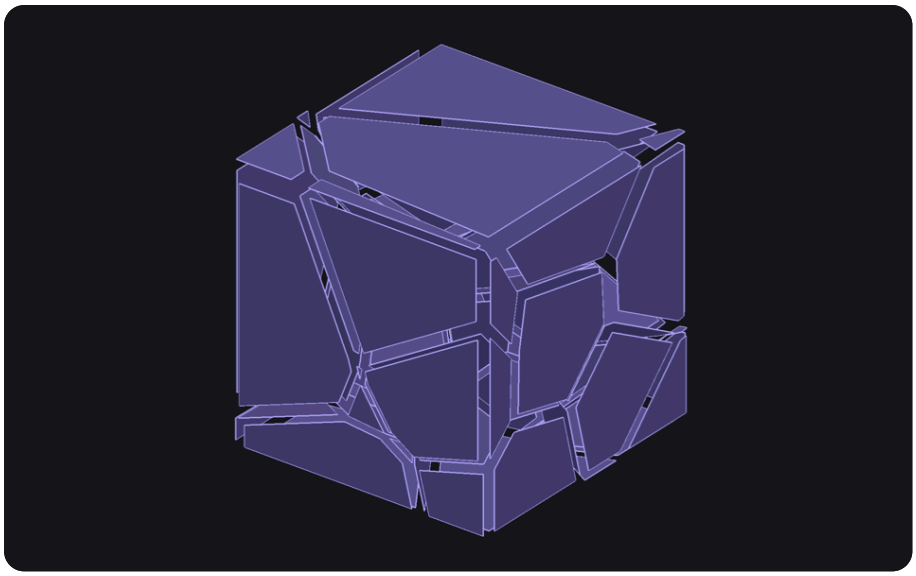
What for. Cells around seeds: each seed gets the part of the plane closer to it than to any other. Organic skins, paving, perforations, plans of a park.

Input	What it means
points	the seeds
boundary	a closed curve that holds the cells; empty: a rectangle a little larger than the seeds
radius	no cell goes farther than this from its seed: they round off and leave gaps
level	the name of the list of cells (*cell* by default)

Tips.

- Seeds from **Random points** give a natural look; change the seed until you like it.
- With **Vertices**, **Centre** and **Delaunay**, a pyramid on each cell: a Voronoi skin.

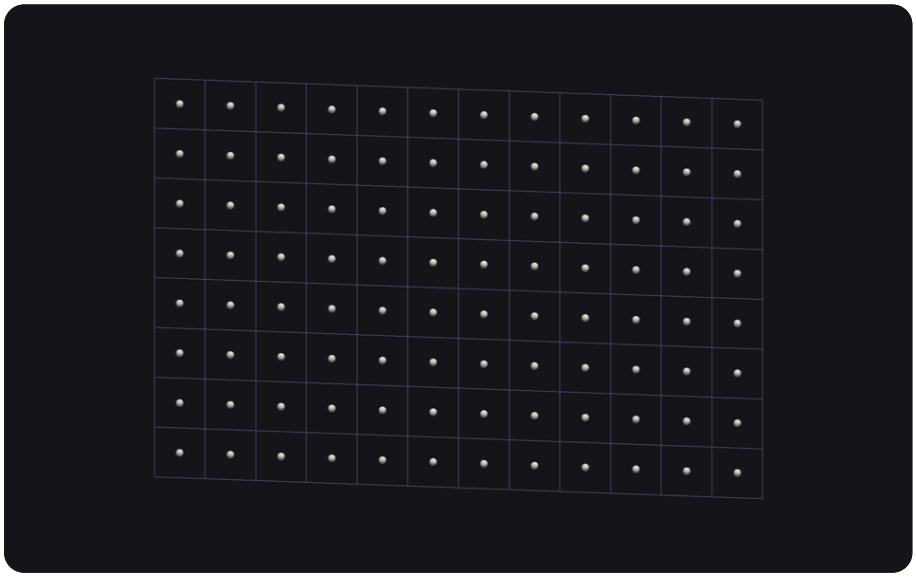
Voronoi 3D



What for. The same in a volume: each point gets the space closer to it. Each cell is a polyhedron.

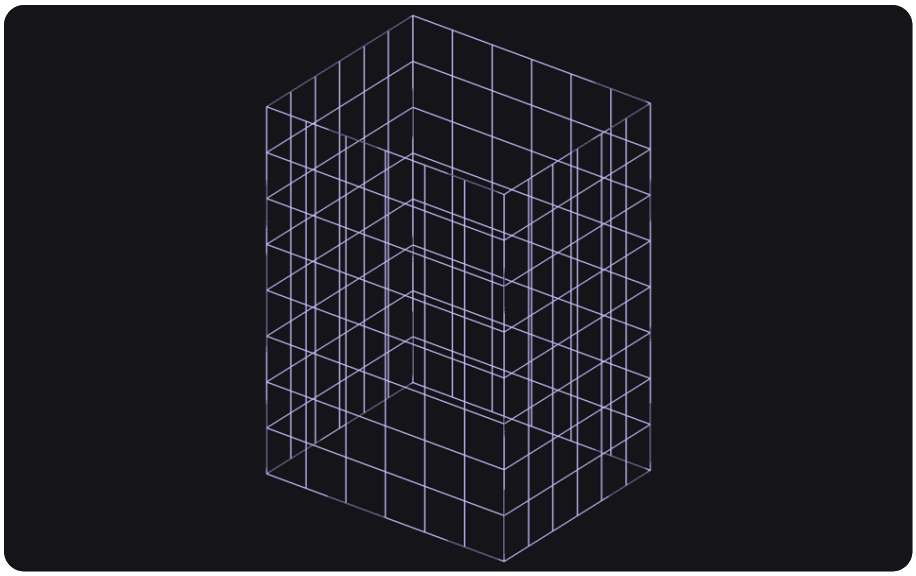
Input	What it means
points, **region**	the seeds, and the volume to divide
level, **faces level**	the names of the cells and of the faces of each cell

Centre



What for. The centre of each element: where to put a fin, a lamp, a label, a column.

Faces



What for. The faces of a prism or a box, by name, as rectangles: to put panels on the south façade, or solar panels on the roof.

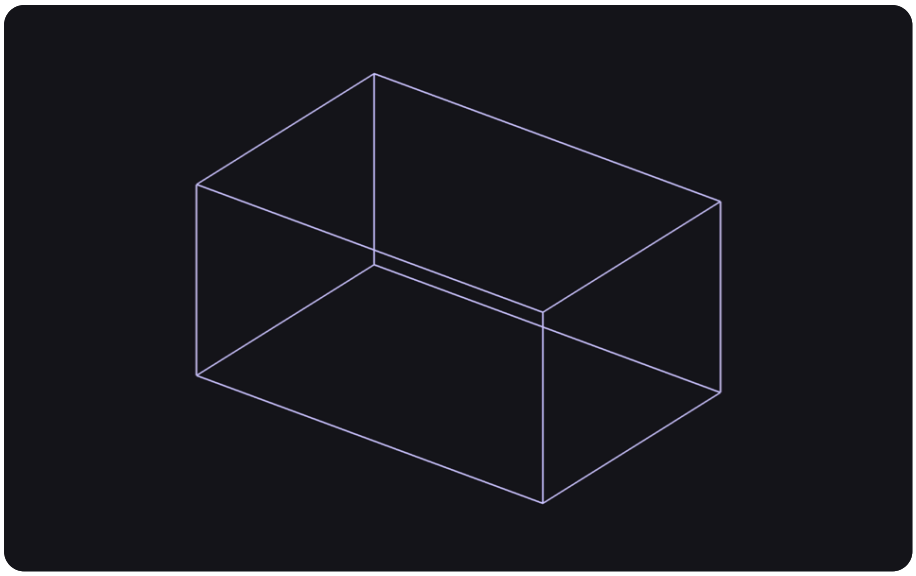
Input	What it means
volume	a Prism or a Box
which	*south*, *east*, *north*, *west*, *roof*, *base*, or *façades* (the four) or *all faces*
level	the name of the new list (*face* by default)

Vertices



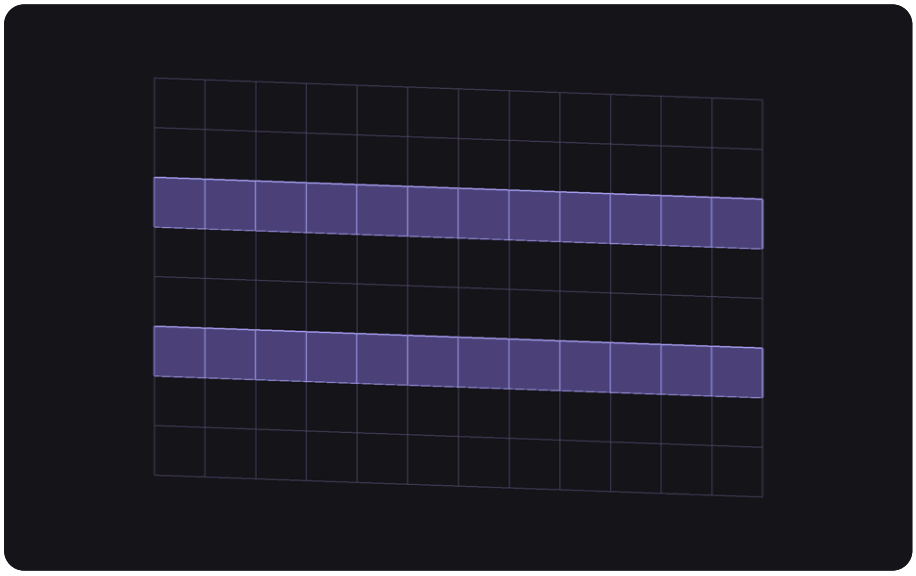
What for. The corners of each element, as points: the 8 corners of a box, the corners of each panel.

Edges



What for. The edges of each element, as lines: the 12 edges of a box, the frame of each panel.

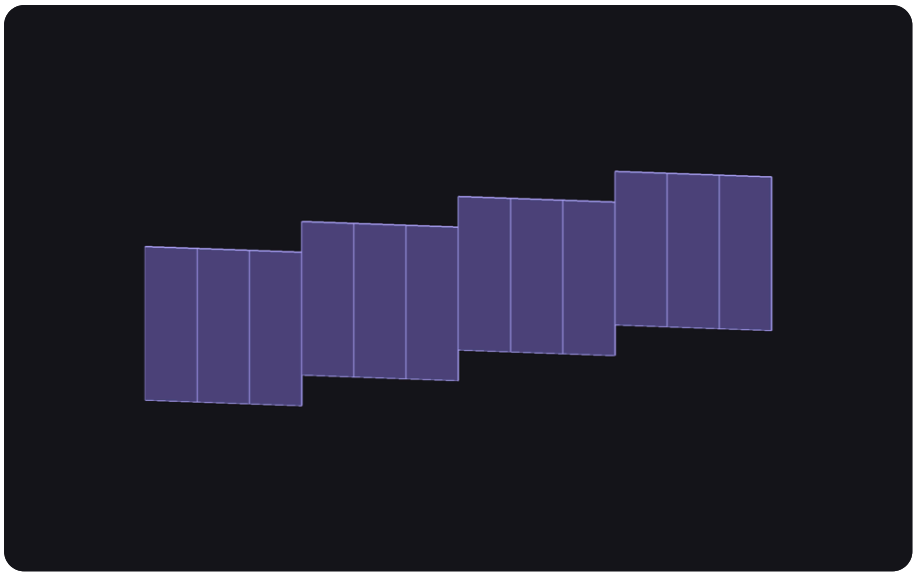
Pick



What for. To take elements of a list by their number, counting from `0`: row 4, the top floor, points 1, 5 and 3, from 2 to 5.

Input	What it means
data	the list
level	which level; empty: the only one
position	`4` · `last` · `-1` (the last) · `1; 5; 3` · `2-5` (from 2 to 5)

Split



What for. To split the elements of a list into groups: panels in groups of four, a facade into three parts.

Input	What it means
data, **level**	the list and which level
mode	*groups of*: so many at a time · *in parts*: into so many equal groups
count	how many per group, or how many groups
new level	the name of the groups: *span*, *module*, *floor*

Tips.

- To split in two at a point, use **Pick** with two ranges: `0-3` and `4-last`.

Reorder

What for. To change the order of a list: reversed, shifted a few places, or from smallest to largest by a value.

Input	What it means
data, **level**	the list and which level; empty on a curve: reverses its direction
mode	*invert*, *shift* or *sort*
count	how many places to shift (negative, the other way)
wrap	yes: what leaves at one end comes back at the other · no: it is dropped
key	the number to sort by (a distance, a height); empty: the values themselves

Tips.

- **Shift** by one, and compare each floor with the one above: the *neighbour*.

Join levels

What for. To join two levels into one, in order: *triangle 4 × vertex 3* becomes *vertex 12*. To draw one curve through all the points, or to number them in a row.

Input	What it means
data	the list
first	the outer level; empty: the first
second	the level that goes inside; empty: the last
new level	the name of the joined level; empty: that of the second

Tips.

- **Split** in groups undoes it.

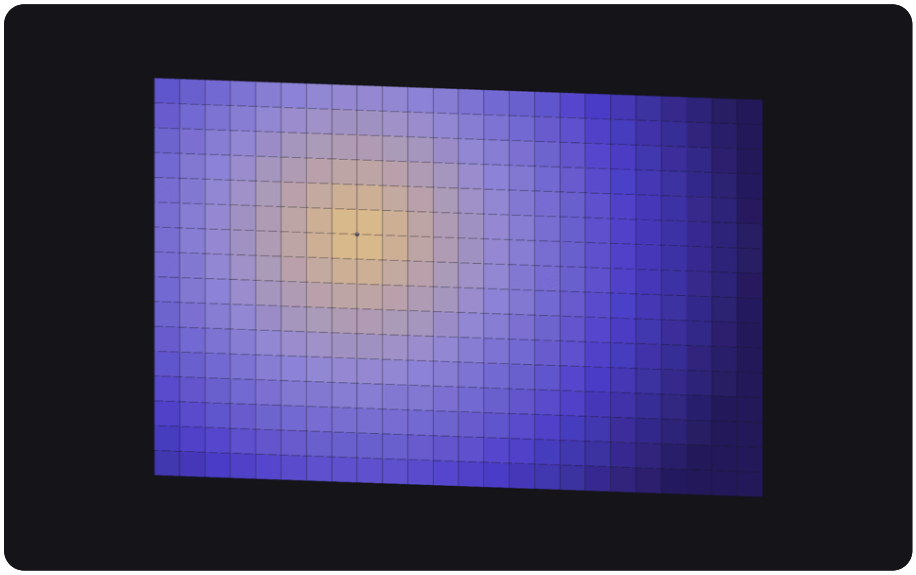
Measure

Measure is where a design starts to answer back: how far each panel is from a point, how big each cell is, which pieces pass a rule. These nodes are violet on the canvas, and most of them give **numbers**, one per element, that you connect to a size, an angle or a distance. See Conditions and attractors.

To see those numbers on the model, paint the geometry by a value in its panel (**Colour → by a value**): that is how the pictures of this page were made.

Closeness

Attractor



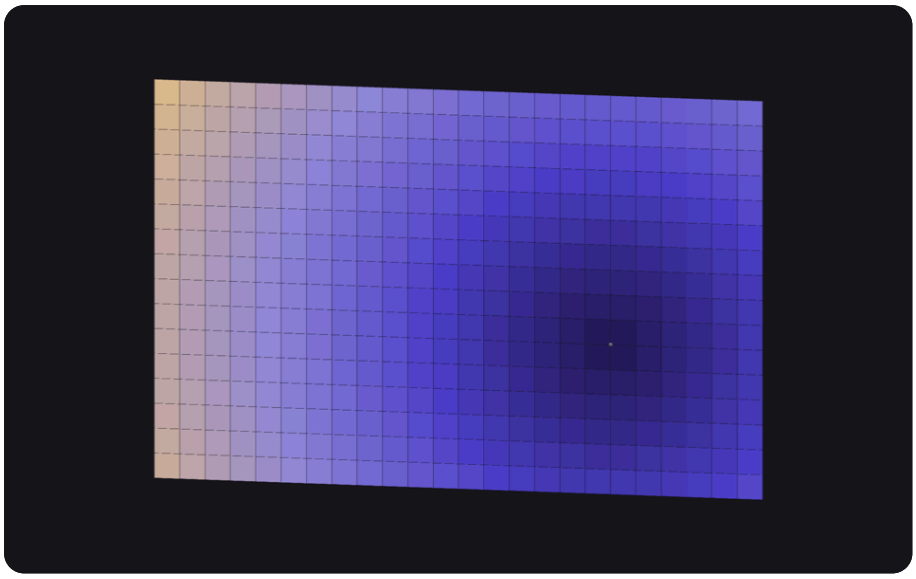
What for. A value per element by how close it is to some points or curves: fins that open near an entrance, panels that shrink towards a corner, holes that grow along a path.

Input	What it means
where	the points that feel it, or a geometry (each element by its centre)
attractors, **more**	one or more points or curves; *more* to mix points and curves
reach	how far the effect goes, in metres; empty: to the farthest one
near, **far**	the value next to the attractor, and at the reach or beyond
several	*the closest* rules, or their effects *add up*
curve	*linear*, *smooth*, *ease in* or *ease out*

Tips.

- Connect what it gives to the **rotation** of a Place, the **factor** of a Scale or the **distance** of a Move.

Distance



What for. The distance between points, in metres: each panel to a point, each column to the next.

How. Connect points to **a** and **b**. With lists of different names, every one with every other one.

Closest point

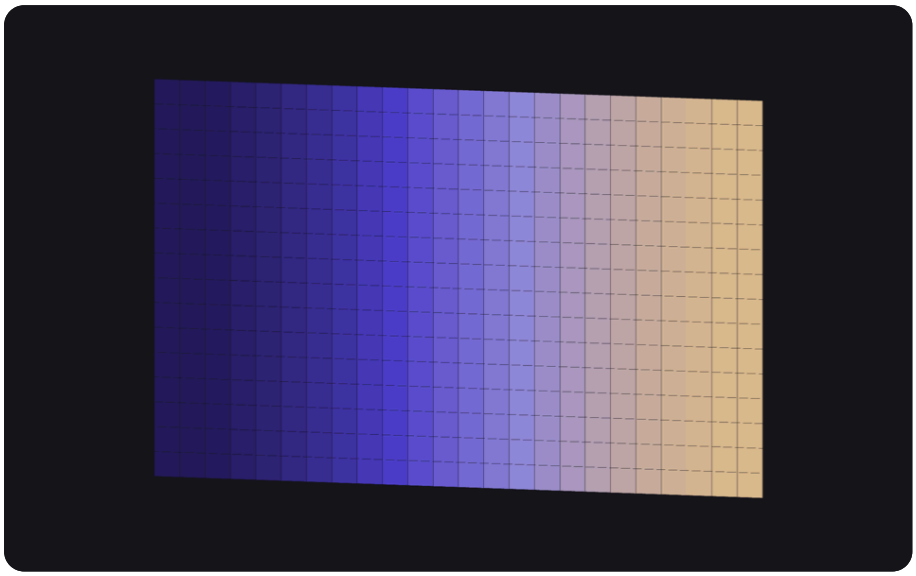


What for. The point of a curve closest to each point: to join them with lines, to orient something towards the curve, or to know where it falls along it.

Input	What it means
points	the points, or a geometry (each element by its centre)
curve	the curve to search on
gives	the *point* on the curve, its *fraction* along it, or the *distance*

From one range to another

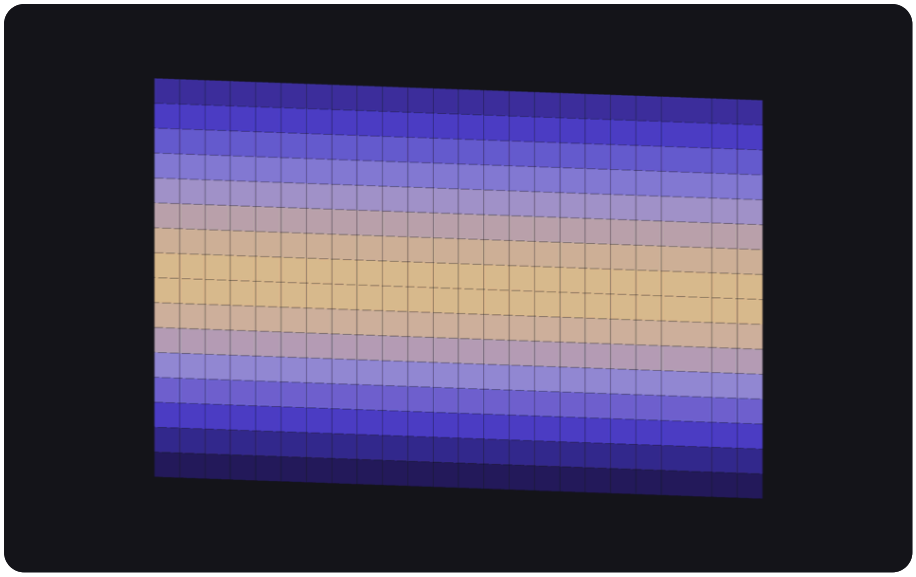
Remap



What for. To take a value from one range to another: distances from 0 to 12 m into angles from 90° to 0°.

Input	What it means
value	the numbers to remap
from min, **from max**	the source range; empty: the smallest and largest of the data
to min, **to max**	the range they go to
curve	*linear*, *smooth*, *ease in* or *ease out*
limit	yes: what goes past the source range stays at the end

Graph



What for. Like Remap, but you **draw the curve** by moving its points in the panel: you decide how a value changes from bottom to top without writing a formula.

Input	What it means
value	the numbers that go through the graph
points	the points of the curve, moved in the panel
to min, **to max**	what comes out at the bottom and the top
smooth	yes: a smooth curve · no: straight segments
from min, **from max**	the source range; empty: that of the data

Tips.

- A guide curve, a **Distance** to it and a **Graph** make reliefs without any formula.

Image sampler

What for. A value at each point read from an image: how light the picture is there. To perforate a façade from a photo, or to raise columns from a drawing.

Input	What it means
file	the image: a photo, a drawing, a gradient
where	the points that read it, or a geometry (each element by its centre)
frame	where the image is laid, stretched to fit; empty: around the points
plane	how it is laid: *plan*, *elevation* or *side elevation*
channel	*brightness*, *red*, *green*, *blue* or *alpha*
black, **white**	the values for dark and for light; swap them to invert

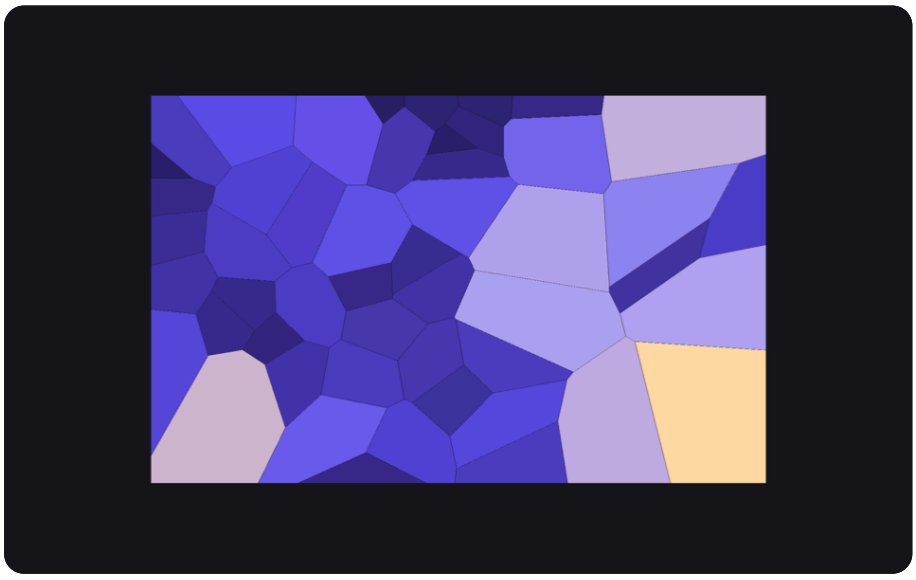
Summarise

What for. One number from many: the total area of a façade, the tallest floor, the average angle, how many.

Input	What it means
data	the numbers
operation	*maximum*, *minimum*, *sum*, *average* or *tally* (how many)
along	which level disappears; empty: all of them

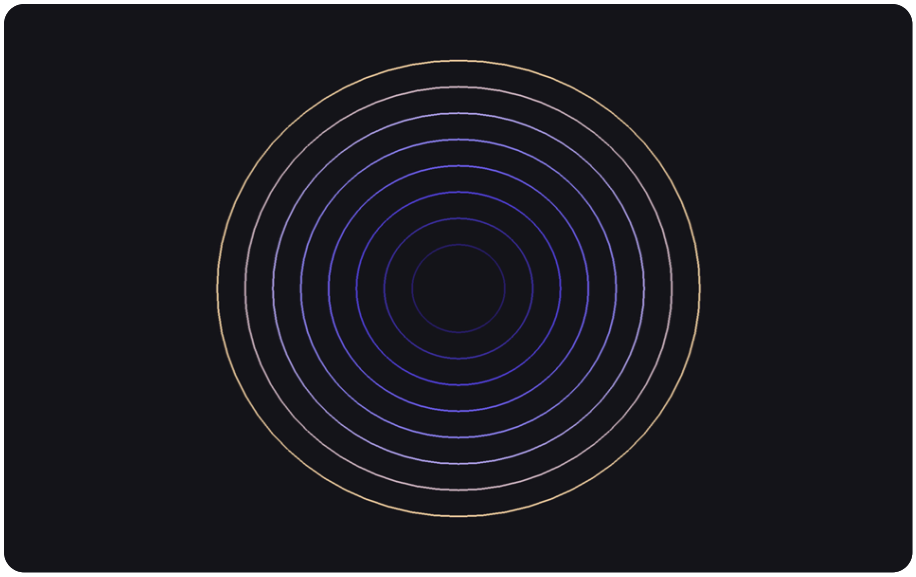
Sizes

Area



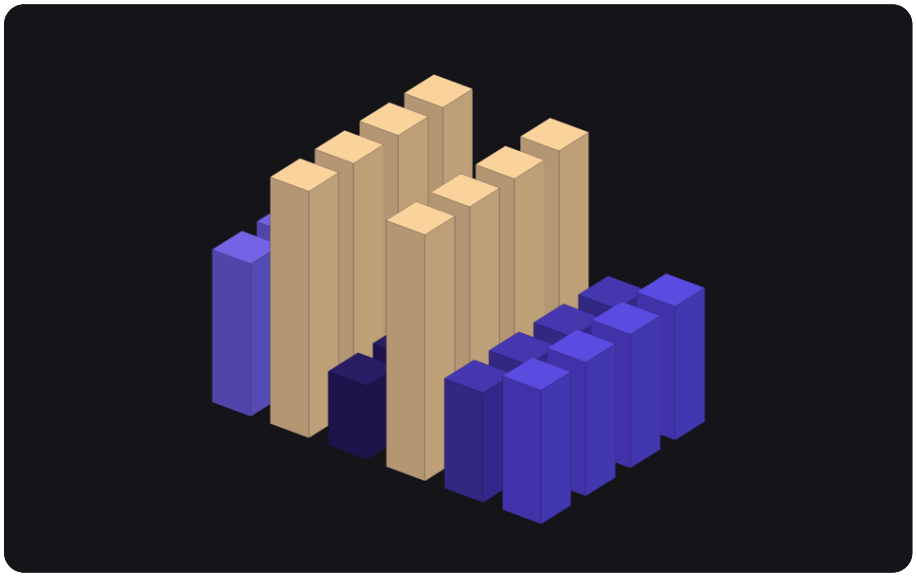
What for. The area of each element in m²: a plan, a surface, the skin of a volume.

Length



What for. The length of each curve in metres: a cable, an edge, a perimeter.

Volume



What for. The volume of each closed body in m^3: a box, a prism, some pipes.

Tips.

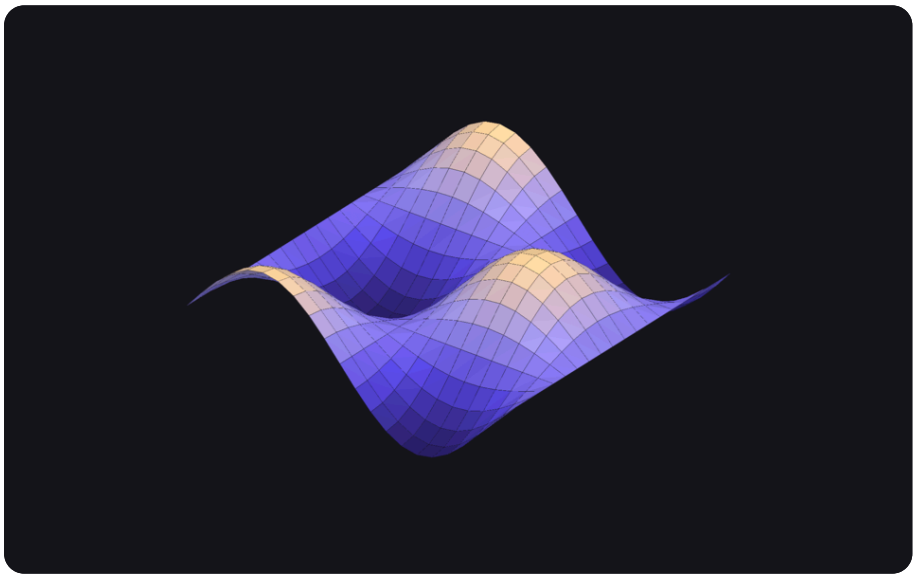
- A rectangle has no volume: it is not closed. Extrude it with caps first.

Centroid



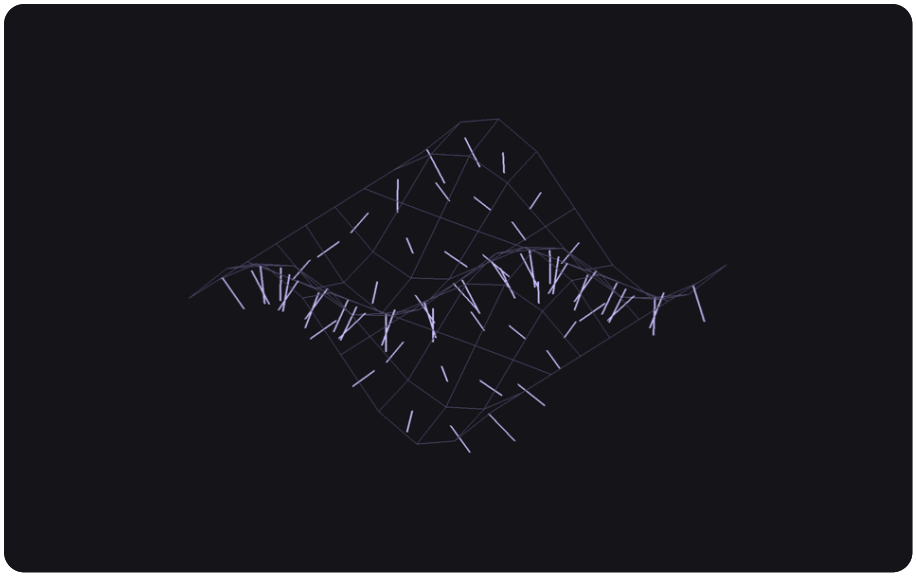
What for. The centre of gravity of each element: of its volume, area or length. Not the average of its corners, which is what **Centre** gives.

Coordinate



What for. The x, y or z of each point: to compare it (the panels on one side, the ones above a height) or to use it in a Formula.

Normal



What for. Where each rectangle faces: outwards on the faces of a prism, or on the panels of a surface. To stand things on a surface with **Orient**, or to test which panels face south.

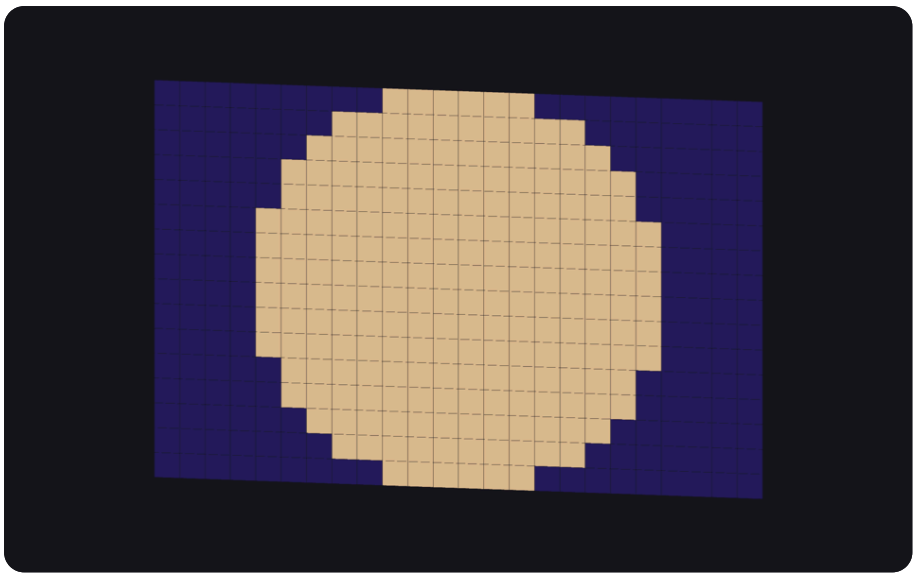
Measure vector

What for. The length of a vector, or the angle in degrees between two: `90`, perpendicular.

Input	What it means
mode	*length* or *angle*
base	the vector to measure
other	the second vector; vertical by default

Conditions

Compare

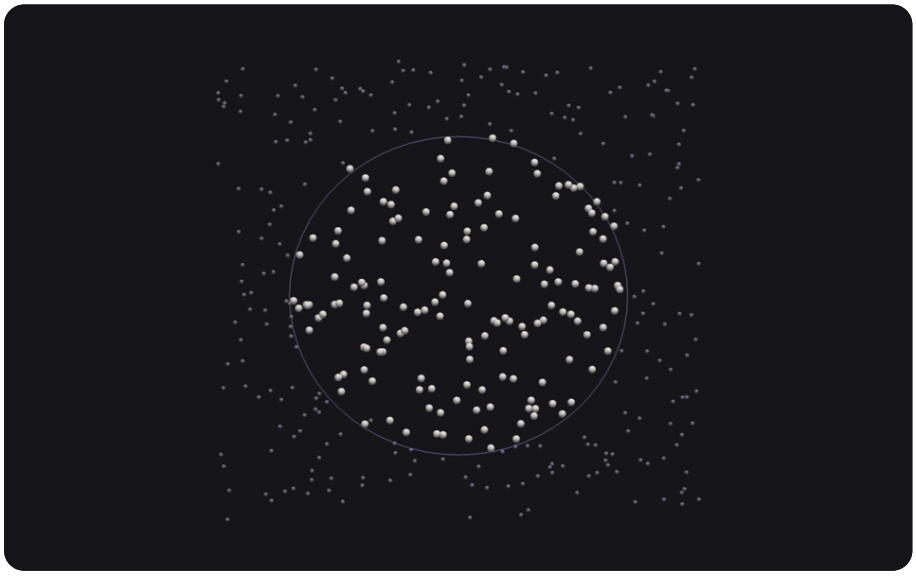


What for. A yes or no per element: *a* is *less*, *less or equal*, *greater*, *greater or equal*, *equal* or *different* from *b*.

Combine conditions

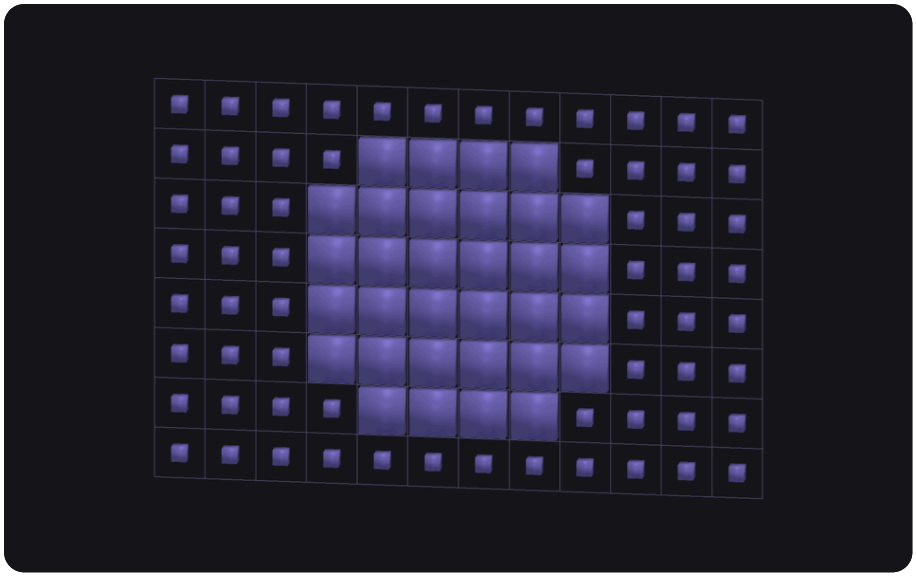
What for. To join two conditions: *and* (both), *or* (either), *only one*, or *not* (the opposite of *a*).

Inside



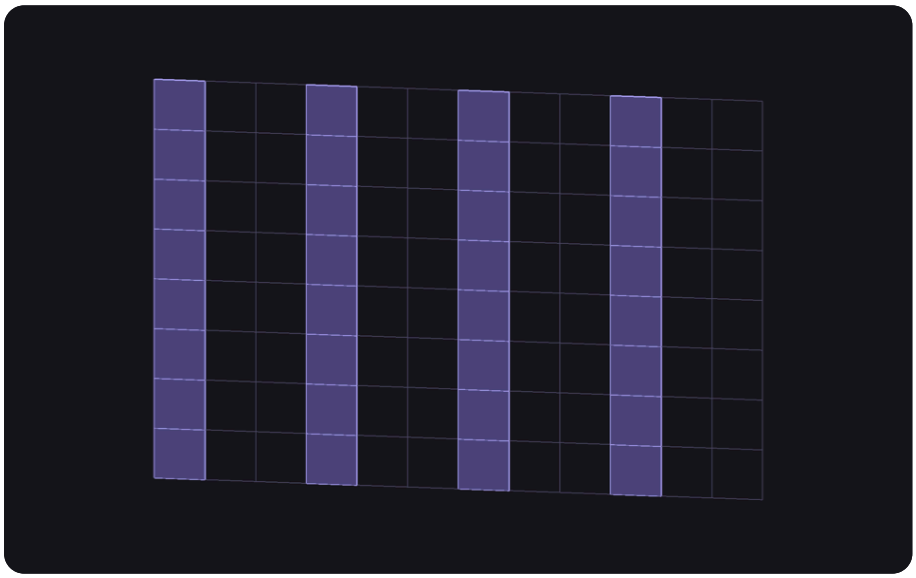
What for. Yes or no for each point: whether it falls inside a plan, seen from above, or inside a closed volume. The columns inside the courtyard, the trees inside the building.

Choose



What for. One thing or another for each element: where the condition holds, the **yes**; otherwise the **no**. An open panel or a closed one. With a single **Yes or no**, a whole variant or the other.

Filter



What for. To keep some elements: those meeting a condition, those of a pattern, or one in every few.

Input	What it means
mode	*condition*, *pattern* (`yes; no; no` keeps one in three) or *every* (one in every so many)
condition, **pattern**, **every**, **from**	the values of each mode
level	which level; empty: the only one
remove	yes: the others go and the list gets shorter · no: they stay in their place, empty
keep	*those that do* or *those that don't*: two Filters, one of each, split a list in two

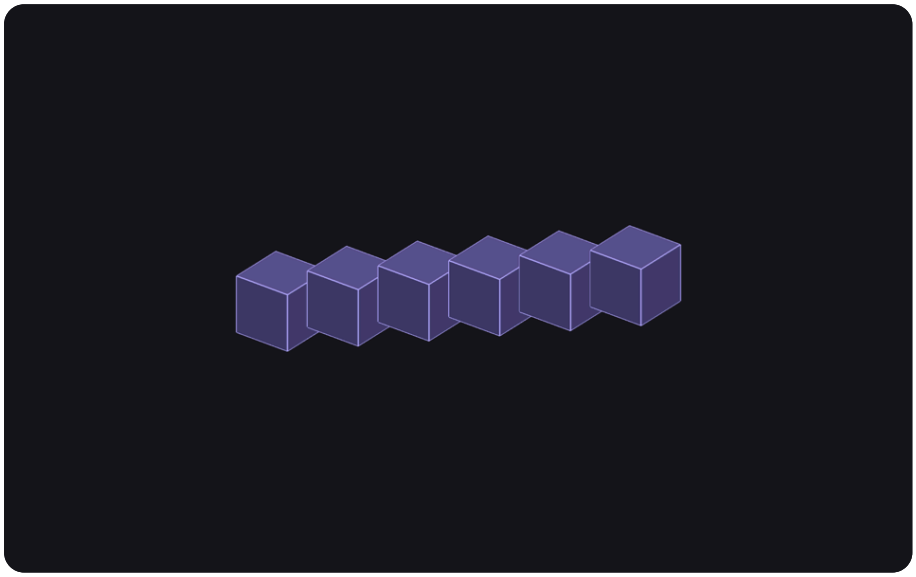
Transform

Transform takes a geometry and changes where it is or what shape it takes: copies along a facade, fins that turn, modules that bend onto a surface. These nodes are terracotta on the canvas.

Every transform works on each element. A Move with a distance from a Series of 6 values gives 6 copies; a Scale whose factor comes from an Attractor shrinks each panel by its own amount.

Take it somewhere

Move



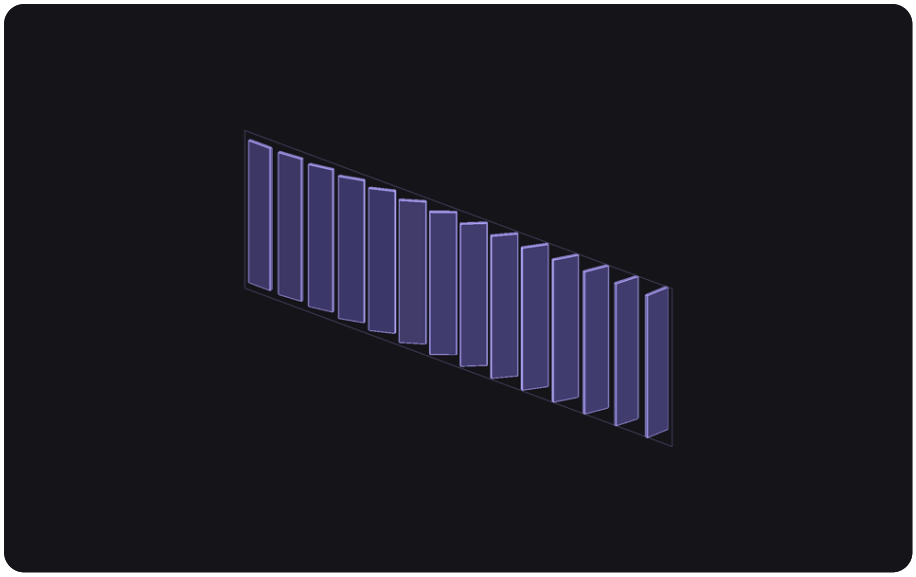
What for. To take each element somewhere else: up a floor, forward from a facade, to another point.

Input	What it means
mode	*vector* (which way and how far) or *point to point* (what is at *from* ends up at *to*)
vector	which way; up by default
distance	if given, the vector only sets the direction: `3` moves 3 m up
from, **to**	the base point and where it ends up (point to point)

Tips.

- To raise something without typing coordinates, give only a **distance**: it goes up.

Place



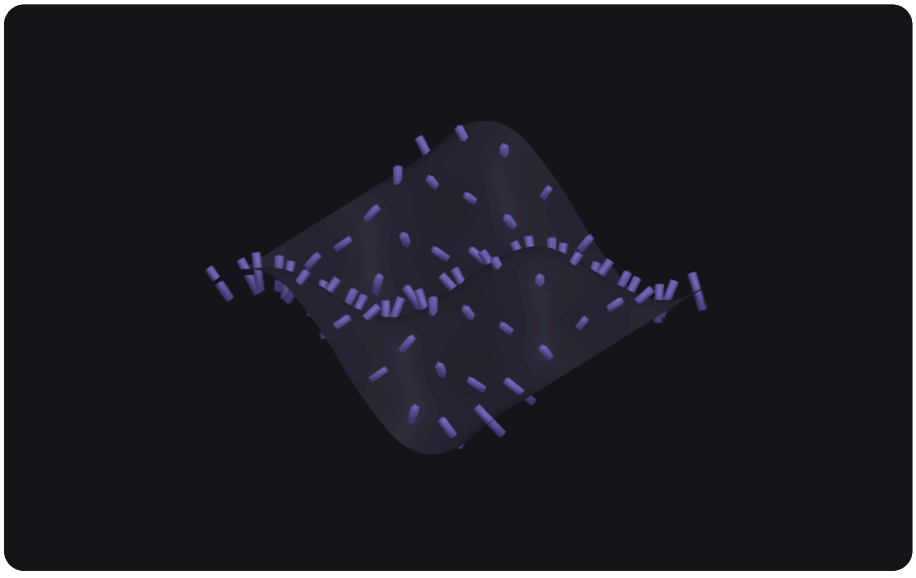
What for. To put a piece on each point, turned: a fin on every panel, a column at every crossing.

Input	What it means
geometry	the piece, drawn around the origin
at	the points where copies go
rotation	degrees, one for all or one per point
axis	what it turns around: *vertical*, *x* or *y*

Tips.

- A **Box** is centred on the origin, so it turns around its own middle. See Your first project.

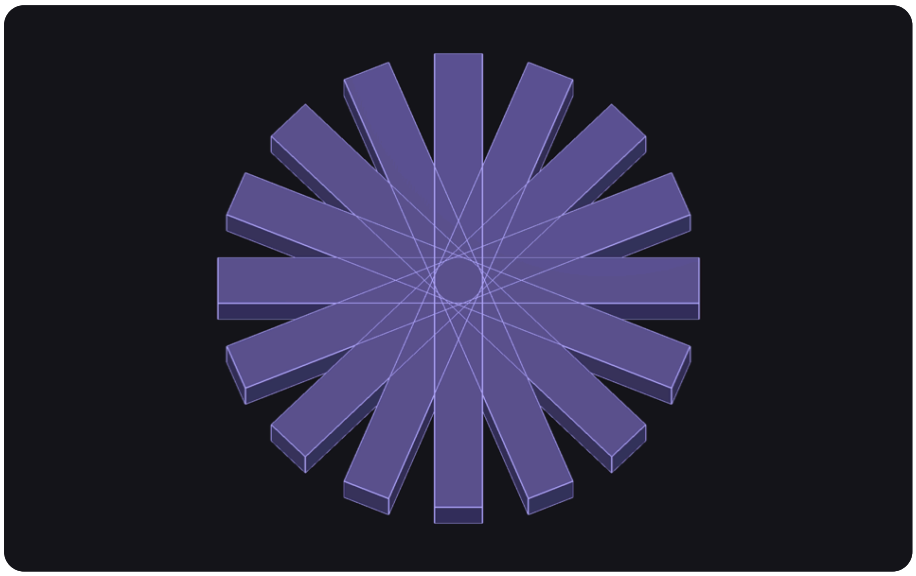
Orient



What for. To take a piece to other places turning it to their direction and their normal: a module on each point of a surface, a piece along a curve. It keeps shape and size.

Input	What it means
from	the point of the piece that goes to *to* (its base, a corner)
to	where it goes: one point, or many
to direction	where its x looks there (some tangents); empty: the world x laid on the place
to normal	what is up there (the normals of a surface)

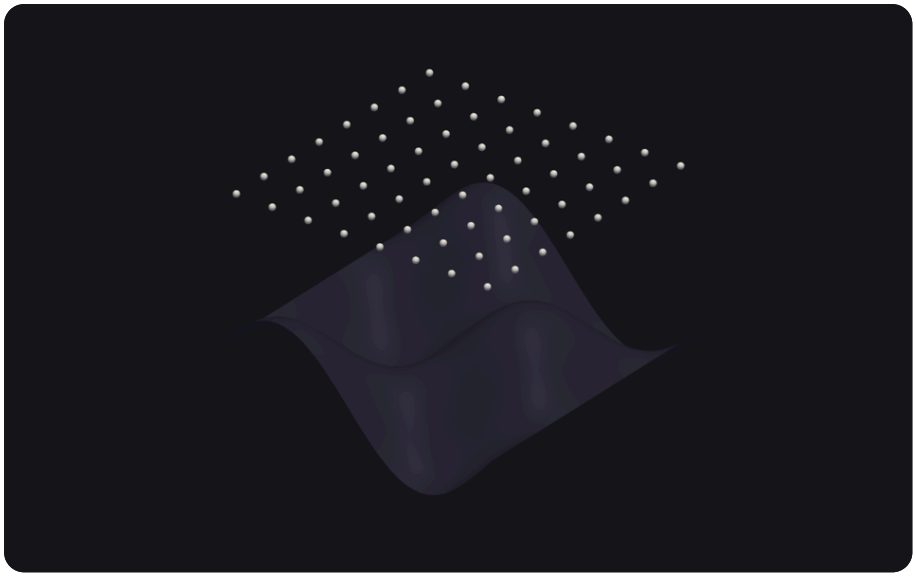
Rotate



What for. To turn each element around an axis: vertical, x, y or any direction, through a centre.

Input	What it means
degrees	how much it turns
axis	*vertical*, *x*, *y* or *direction* (a hinge, the slope of a roof)
centre	a point the axis passes through
direction	the direction of the axis, for *direction*

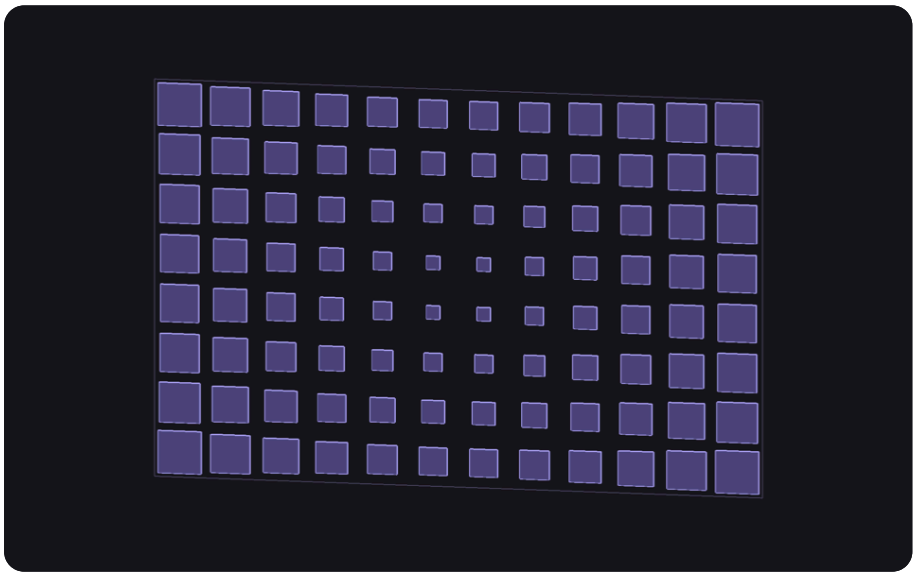
Project



What for. To bring a geometry flat to a height: its footprint on the ground, or on another level.

Change its size

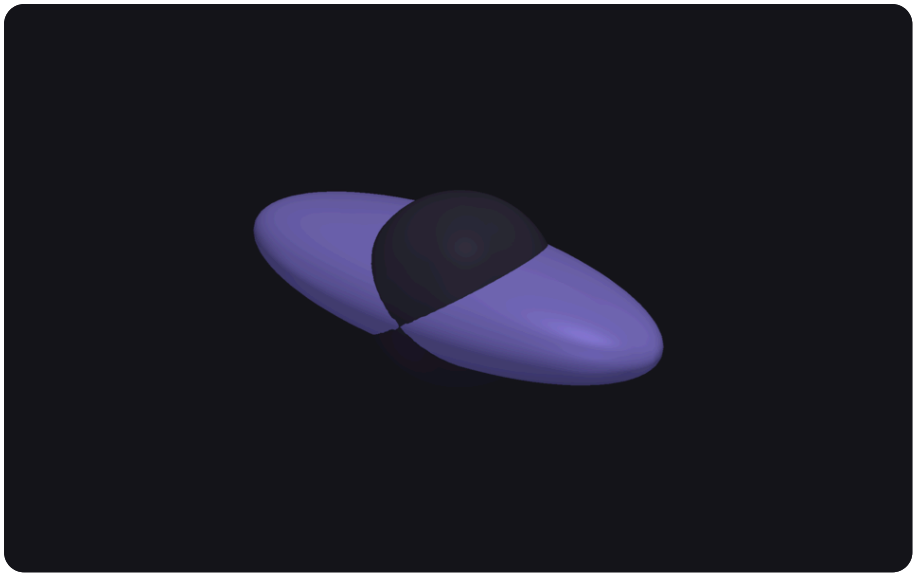
Scale



What for. To enlarge or shrink each element: panels that shrink towards a point, a smaller copy.

Input	What it means
factor	more than 1 enlarges, less than 1 shrinks
centre	empty: each element about its own centre

Scale by axis

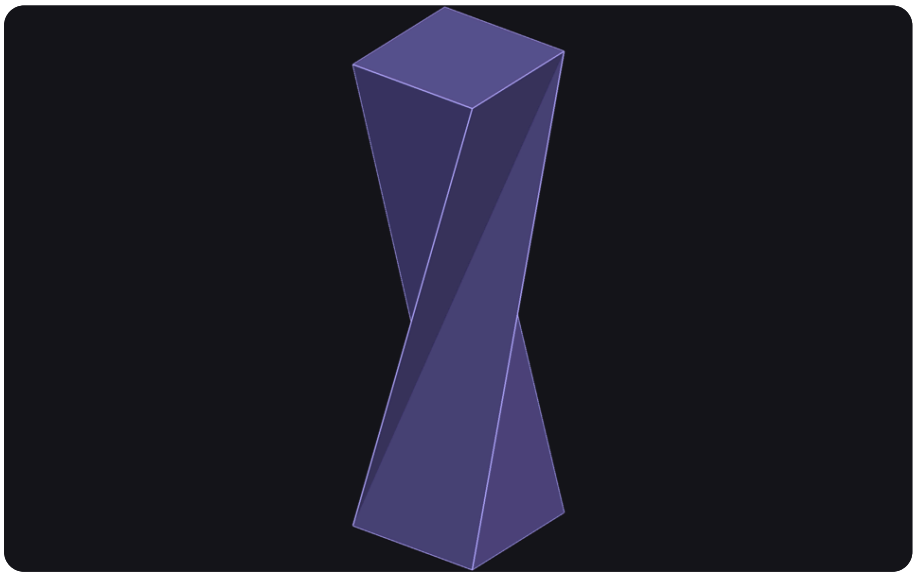


What for. To scale differently in x, y and z: a floor that stretches in width, a slab that gets thinner.

Input	What it means
x, **y**, **z**	the factor in each direction
centre	empty: each element about its own centre

Change its shape

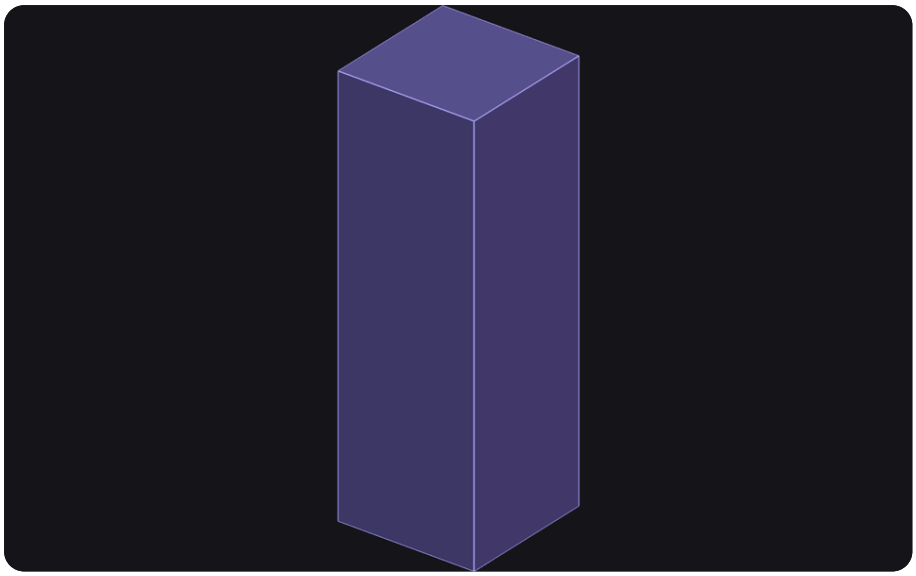
Twist



What for. To twist a geometry around a vertical axis: the bottom stays, the top turns. A twisting tower.

Input	What it means
degrees	how much the top turns
centre	where the vertical axis passes
from	at what height the twist starts, from 0 (bottom) to 1 (top)

Morph



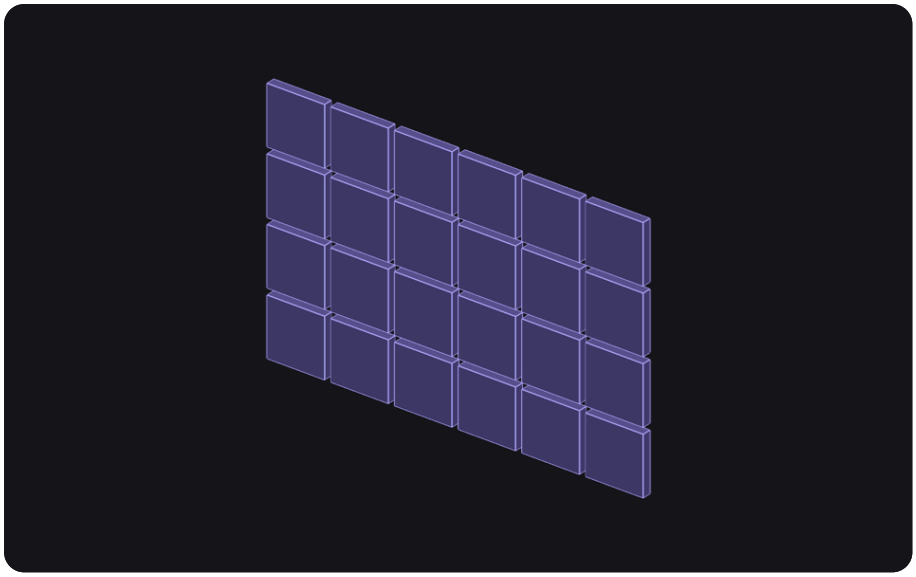
What for. To fit a module into each cell, bending with it: whatever fills the box of the module fills the cell, however twisted.

Input	What it means
geometry	the module, drawn anywhere
into	the cells: boxes, blocks (from Thicken), panels of a Grid, or eight corners moved one by one
height	for panels: how far the cell rises from the panel

Tips.

- On a curved surface, **Surface morph** divides the surface and fits the module in one step.

Thicken



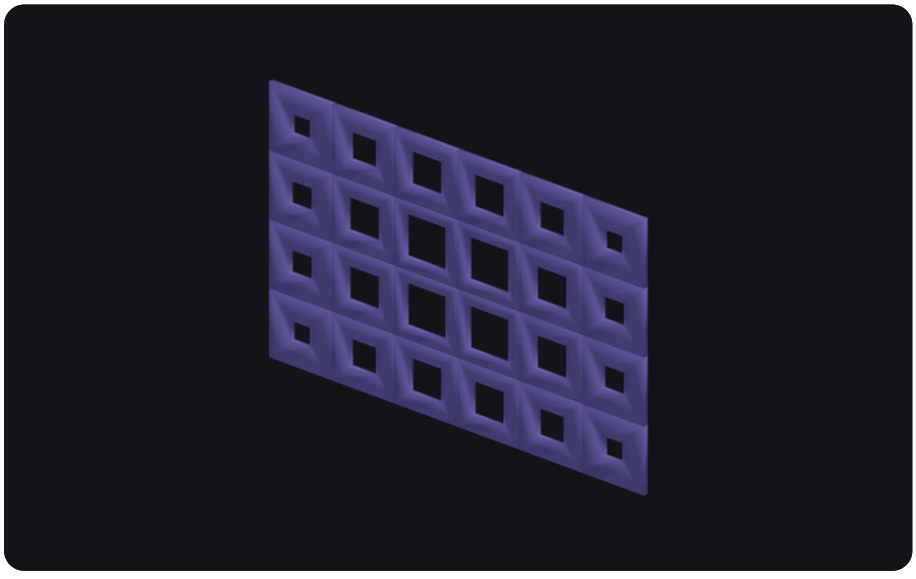
What for. To give thickness to each rectangle, a block, or to a surface: slabs, panels with depth, walls.

Input	What it means
rectangle	the rectangles or the surface
thickness	positive outwards, negative inwards

Tips.

- It grows towards the face of the surface: to grow the other way, give a negative thickness or flip the surface.

Frame



What for. A frame with an opening in each panel, or the opening alone (the glass). Openings that grow by an attractor give a facade that breathes.

Input	What it means
panel	the panels
opening	the opening relative to the panel, from 0 (closed) to 1
thickness	`0` : a flat frame
width	the frame width in metres, the same all round, instead of the opening
output	the *frame*, or the *hole* (the rectangle inside)

Directions

Vector



What for. A direction with its length, to use wherever a point or a direction goes: in Move, Line by direction, Rotate.

Mode	What it gives
x, y, z	typed
axis	along x, y or z
point to point	from **from** to **to**
unit, *to a length*	the same direction, length 1 or the **length** you give
multiply	times a **factor** (negative reverses it)
add, *invert*, *perpendicular*	two added, reversed, or perpendicular to two

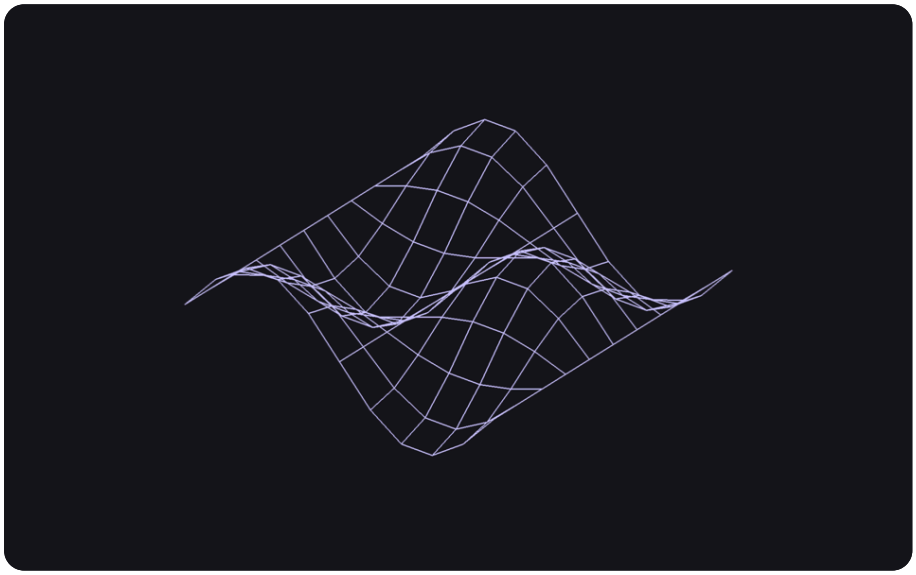
Meshes

A mesh is a surface made of flat faces that share their corners: triangles, quadrilaterals. It is how a shape is built with flat glass, panels or folded sheet, and how free shapes are modelled by subdivision. These nodes are copper on the canvas.

Geometry and topology. In a mesh, the **points** say where the corners are (its geometry) and the **faces** say how they are joined (its topology). Move a point and the faces follow; the order of the corners of a face says which side is its front.

Make a mesh

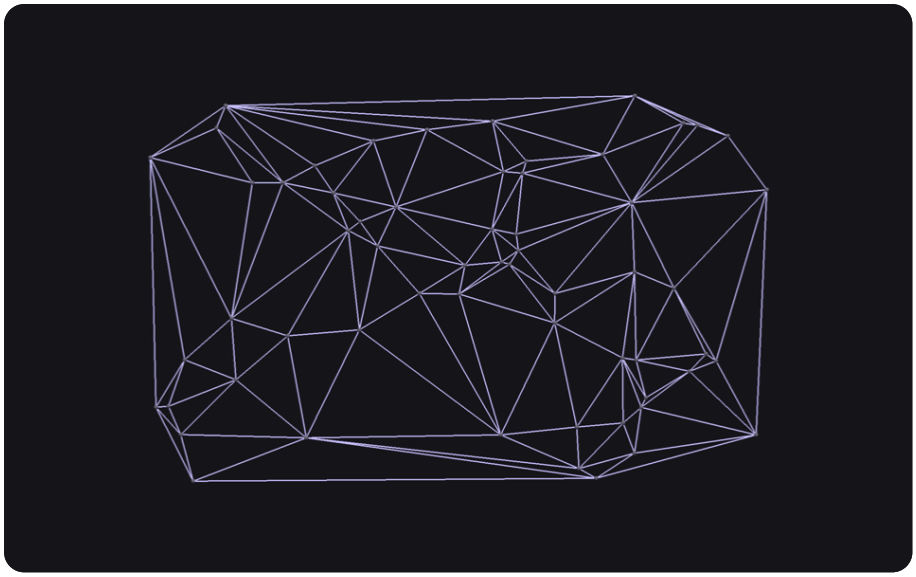
Mesh surface



What for. A surface as one mesh of quadrilaterals sharing their corners. On a surface of revolution or of translation, the quadrilaterals come out flat: panels that can be built with flat glass.

Input	What it means
surface	a surface, or a rectangle
count u, **count v**	faces across and up

Delaunay



What for. To join points in triangles as even as possible, without thin ones: a terrain from survey points, a skin of triangles, pyramids on the cells of a Voronoi.

Input	What it means
points	the points to join; with other levels, one group for each
more	optional: a point added to each group, like the raised centre of a cell to make a pyramid

Mesh

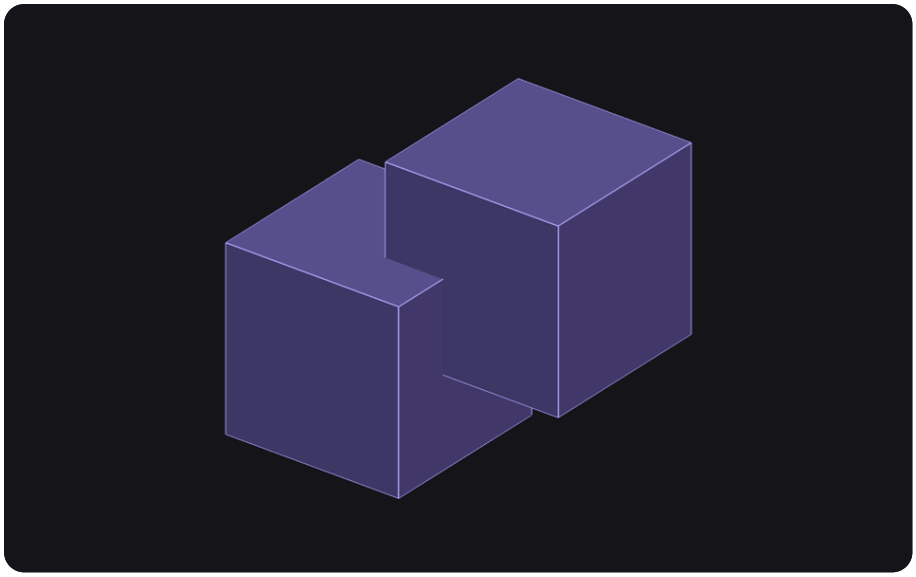
What for. A mesh written by hand, from its corners and its faces: to understand meshes, or to build a precise piece.

Input	What it means
points	the corners, numbered from 0 in order
faces	the corners of each face, `;` between faces: `0 1 2; 2 1 3` (two triangles), `0 1 2 3` (a quadrilateral)

Tips.

- Write the corners of each face counter-clockwise, seen from the front.

Join meshes



What for. Everything with faces as one mesh. Welded, the faces that touch share their corners and become one piece, which is what smoothing needs.

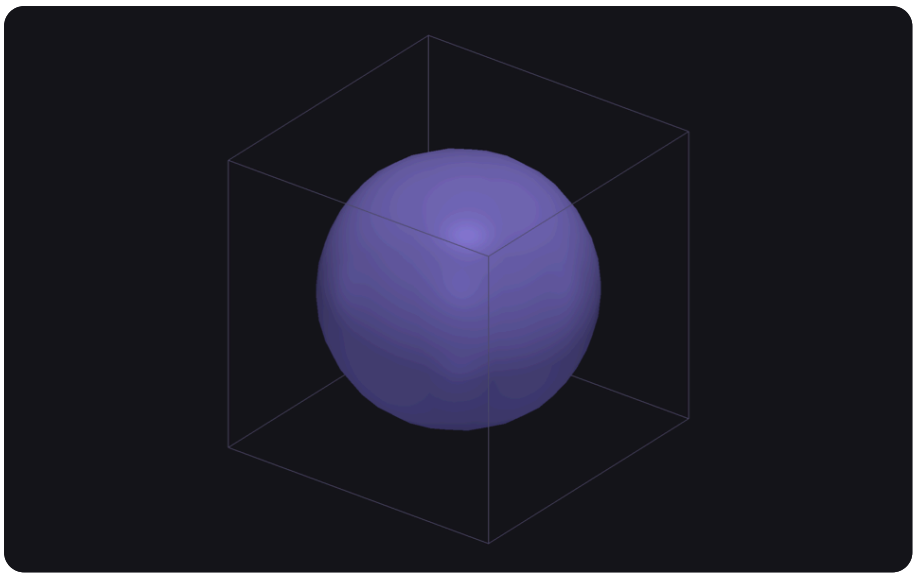
Input	What it means
a … h	meshes, or anything with faces: boxes, panels, frames, surfaces
weld	yes: corners that touch become one
doubled faces	*leave* or *remove*: removing them turns a group of boxes into its outer skin

Tips.

- Where the edges of two parts do not meet corner to corner (a T-node), they do not weld. **Mesh edges** with *naked* shows those seams.

Change it

Smooth mesh



What for. A smooth mesh from a simple one: model the shape with a few faces, and let the levels round it. A table, a pavilion, a free form.

Input	What it means
method	*Catmull-Clark* (any faces, they become quadrilaterals) or *Loop* (triangles)
levels	how many times it subdivides: each level about four times the faces
border	*smooth*, *fixed* or *corners fixed*

Tips.

- For a shape of several parts, **join and weld** them first.

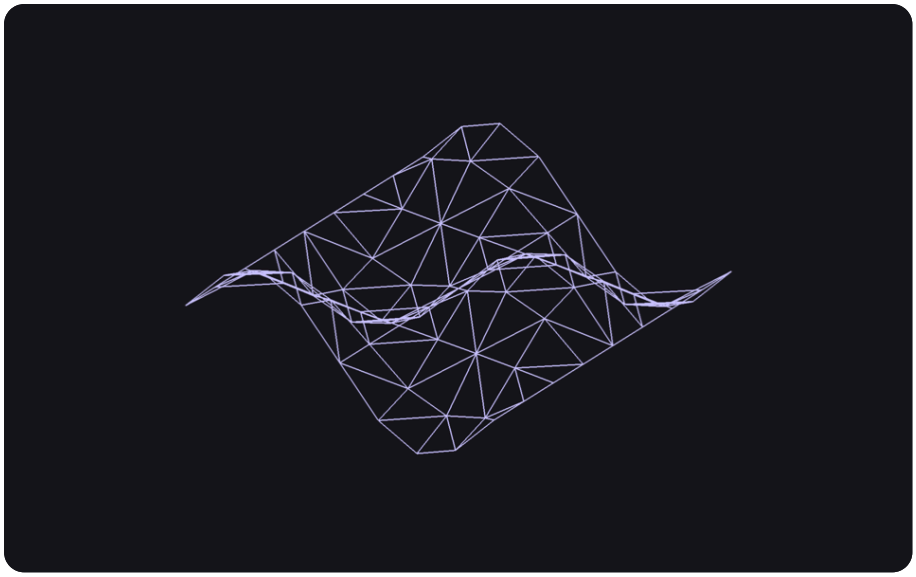
Edit mesh

What for. To shape a mesh by hand: in the 3D view, choose its corners and drag them (Alt: up or down). Only the moves are kept, so if the mesh that comes in changes, the edits stay.

Tips.

- **Edit mesh** followed by **Smooth mesh** is the way to model a free shape: a coarse cage moved by hand, and the levels round it.

Triangulate



What for. Every face as triangles, each quadrilateral split by a diagonal. Triangles are always flat.

Input	What it means
diagonal	the *shorter* one, or always the *first* (0-2) or the *second* (1-3)

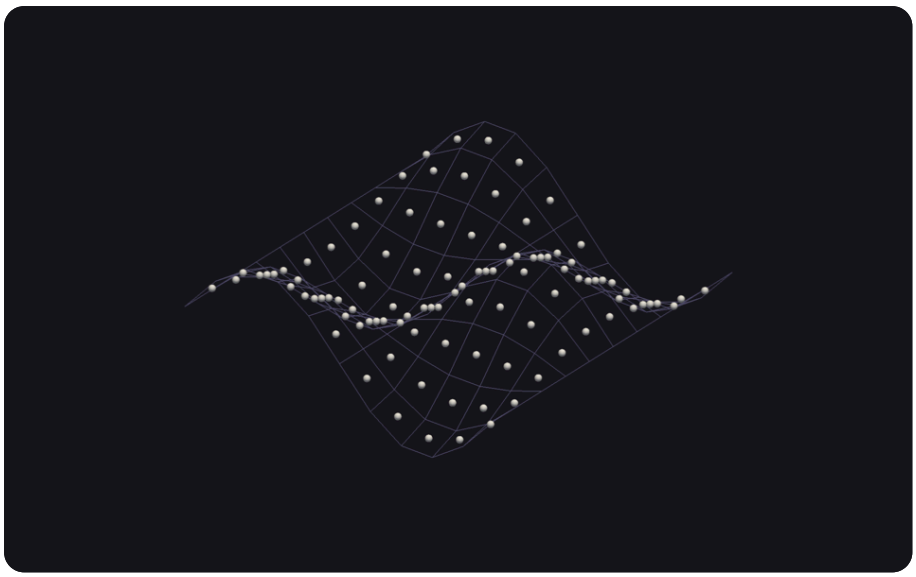
Flip faces

What for. To turn faces over, so the front becomes the back: to make all the faces of a mesh look the same way.

Input	What it means
which	the faces to turn (`1` · `0 3`); empty: all

Read it

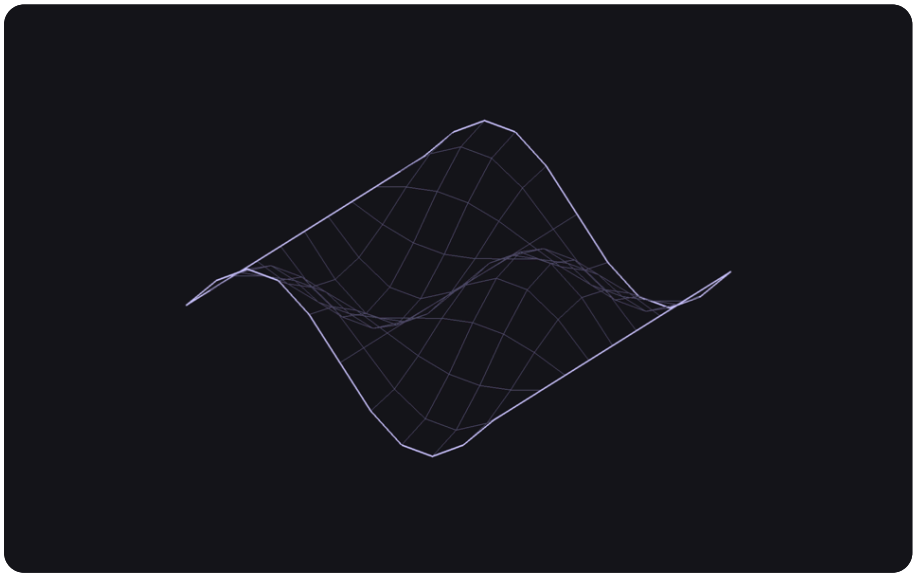
Mesh faces



What for. To read each face of a mesh: its outline, its centre, its normal, how many sides it has, whether it looks the same way as its neighbours, or how flat it is.

Input	What it means
what	*outlines*, *centres*, *normals*, *sides*, *orientation* (+1 like its neighbours, −1 turned over) or *flatness*

Mesh edges



What for. The edges of a mesh as lines: its border, or the bars of a structure along its edges.

Input	What it means
which	*naked* (used by one face: the border, or a seam that did not weld), *inner* (shared) or *all*

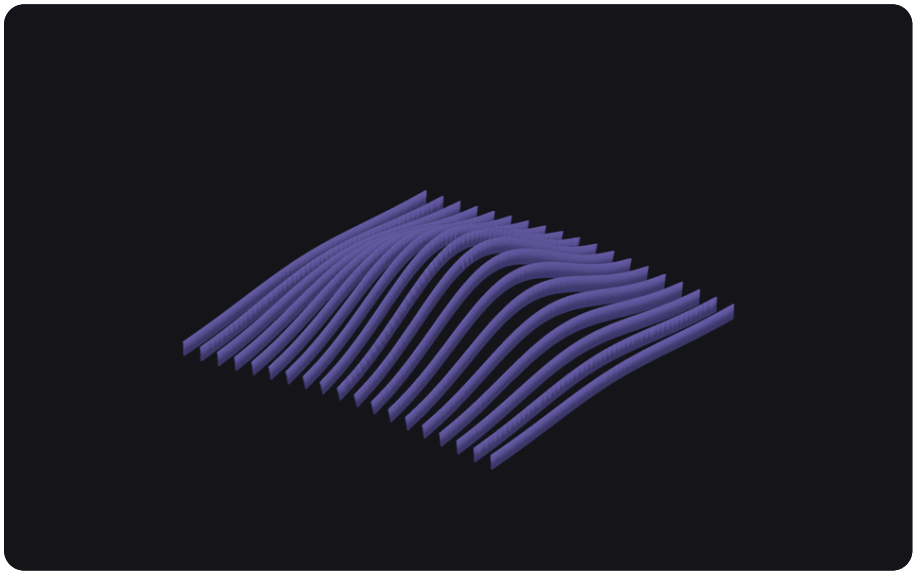
Fabrication

Fabrication takes a design to the machine: a laser cutter, a router, a 3D printer. These nodes are olive on the canvas. Measures are in metres by default, as everywhere in Tensela: `0.003` is a 3 mm sheet (or type `3 mm`; see Preferences).

The usual path to a cut file: cut the shape into boards with **Sections** (and **Slots** for a waffle), lay them flat with **Lay flat**, add their numbers with **Numbers and sheets**, and export with **File → Export → Cutting file**: red lines to cut, blue to engrave.

Cut the shape

Sections



What for. To cut a shape into boards of a sheet's thickness, each one a closed outline ready to cut: a terrain model, a sliced landscape, a pavilion of ribs. Glued in order, the boards rebuild the shape.

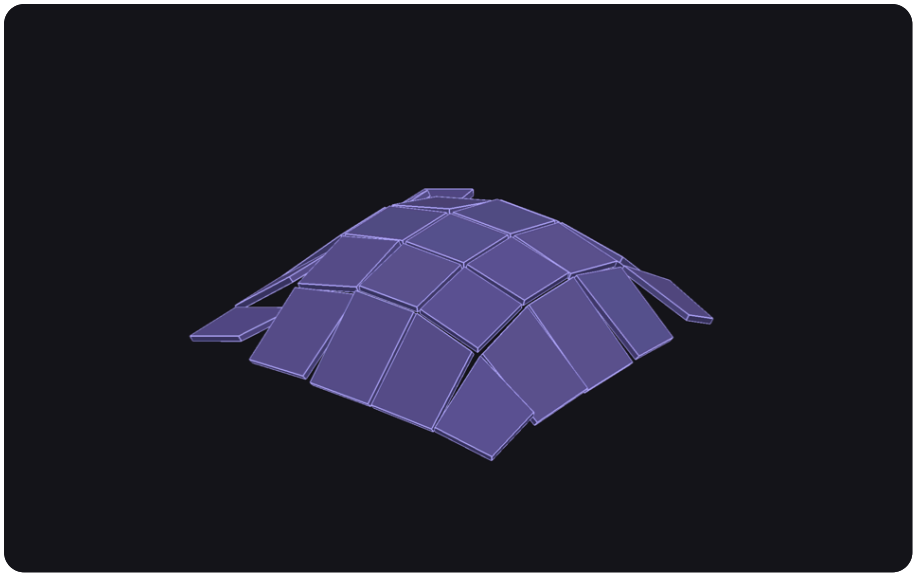
Input	What it means
shape	a surface, a mesh or a body
thickness, **gap**	of the boards, and between one and the next (`0`: glued together)
cuts	*x*, *y*, *z* (level boards, stacked), *radial* (like the segments of an orange) or *along a curve*
count	how many radial boards
guide	the curve the boards follow, each one square to it (a spine, the axis of a leaning tower)
piece	how the section of a surface becomes a piece: a *band* that follows it, or *to base*, down to a common base
depth	how deep the band is, or how far the base goes under the lowest point
show	the *outlines* to cut, or the *boards* with their thickness, to see them

Slots

What for. Where two families of boards cross (a waffle, an egg-crate), a slot halfway in each: one slotted from above, the other from below. Slotted, the boards slide into one another and the structure holds itself.

Input	What it means
first, **second**	two families of flat boards that cross (Sections across x and across y)
thickness	of the boards: the width of each slot
clearance	added to the slot so the boards go in (a laser: 0.1 to 0.2 mm; a router: more)
slotted	which family is slotted from above
show	the *first* or *second* family with its slots, ready to lay flat, or all the *boards*

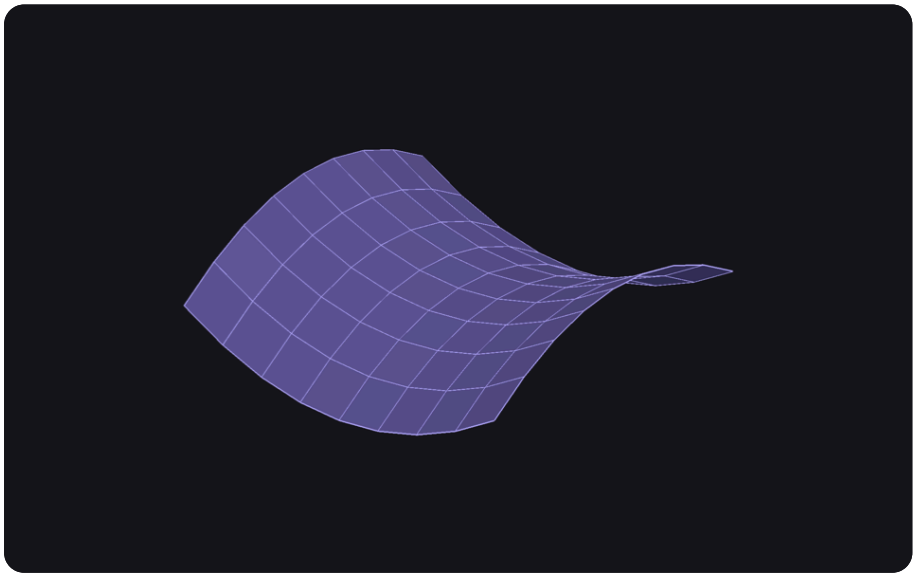
Plates



What for. To turn each face of a shape into a plate of some thickness, trimmed back from each edge so the plates keep the same gap at any angle. For faceted pavilions of boards.

Input	What it means
shape	a mesh, a box or a surface: each face, a plate
thickness, **gap**	of the board, and between two plates that meet
show	the *plates*, their *outlines* to lay flat, the *angles* of the edges, or the *setbacks*

Planarize



What for. To make the panels of a curved surface flat, so they can be cut from boards or glass, while they keep meeting corner to corner.

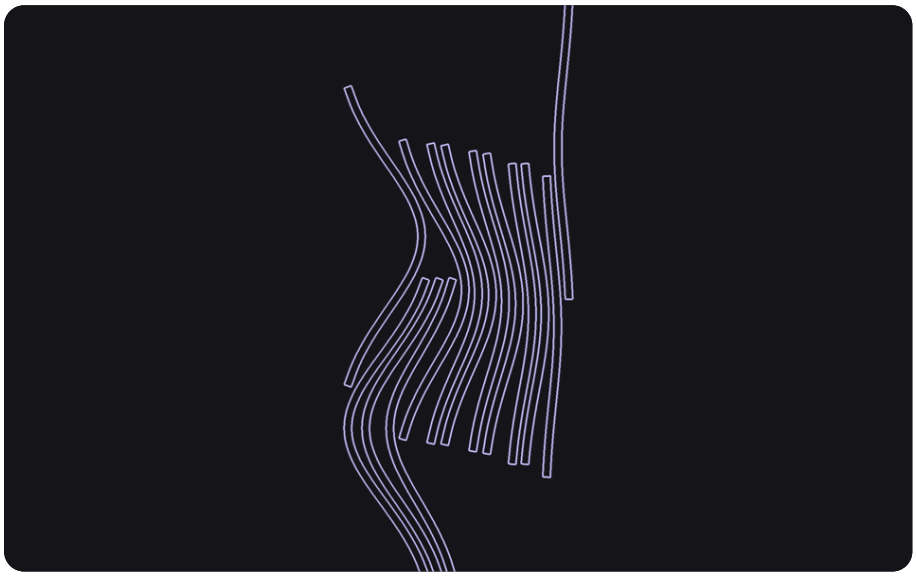
Input	What it means
panels	the panels of a surface, sharing their corners (a Grid on a surface)
show	the flat *panels*, their *flatness* now, or their *flatness before*

Tips.

- Easy on a vault or a scroll; on a saddle, the panels bend inwards a little.

To the machine

Lay flat



What for. To lay flat pieces on sheets, ready to cut: fitted into one another, the small ones in the holes of the big ones, each with a number inside, the same number the 3D view gives it.

Input	What it means
pieces	flat closed curves: from Sections, Slots, Unroll…
width, **height**	of the sheet (600 × 400 mm: a laser bed; 2440 × 1220 mm: a board for a router)
spacing, **margin**	between pieces, and from the edge
turn	turn pieces a quarter if they fit better
number	the height of the engraved numbers (`0` : none)
origin	where the first sheet goes
more	a second family on the same sheets: numbers then carry *A* and *B*

Tips.

- The panel says how many sheets and how much of them is used.
- If a piece does not fit, a notice says how big it is: use larger sheets, or split the piece.

Numbers and sheets

What for. The numbers of the pieces laid by a Lay flat, to engrave, or the outlines of its sheets. They always follow the pieces.

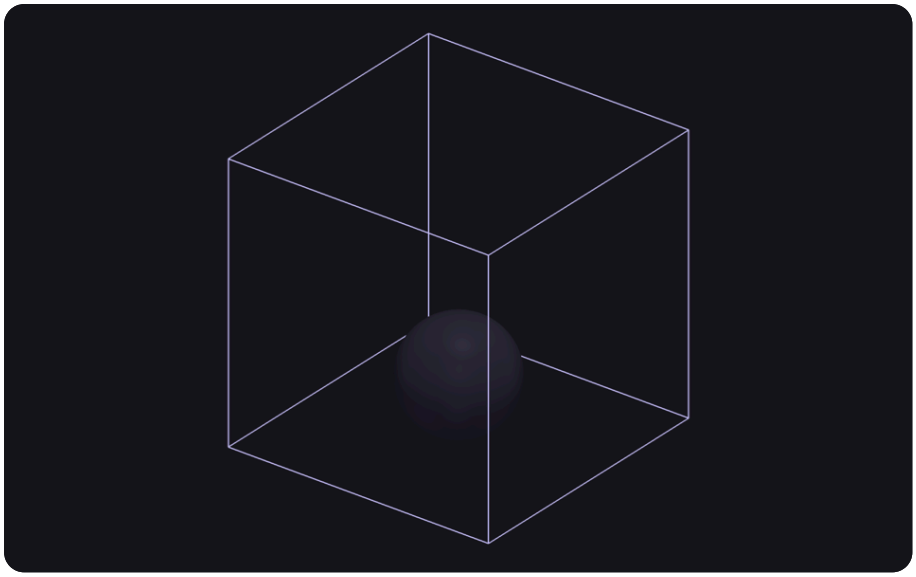
Input	What it means
laid	the pieces of a Lay flat
show	*numbers* or *sheets*
lettering, **weight**, **reading**, **font**	how the numbers are drawn: single stroke, bold or outlined; along the sheet or turned

Bevel

What for. Boards with their edges eased, to see them, print them or mill them with a bevel bit. A laser always cuts straight.

Input	What it means
pieces	flat outlines, or the boards of Sections
thickness, **size**	of the board, and how far the bevel goes
profile	*rounded*, *chamfer*, *cove*, or *custom* (drawn with a **curve**)
segments	facets of the rounded or the cove
faces	*both faces*, *top face* or *bottom face*

Print check



What for. To know whether a piece can be 3D printed, and if not, why and where: closed, every edge between two faces, faces turned outwards, not crossing itself, fits the printer, no wall too thin.

Input	What it means
geometry	the piece, or several
show	the *problems* as red lines, whether it is *ready*, or the *printer* volume around it
width, **depth**, **height**	the printer's build volume
thickness	the thinnest wall it can make (`0`: not checked)

Tips.

- A rectangle is not closed: it has no volume and cannot be printed. Give it thickness first.

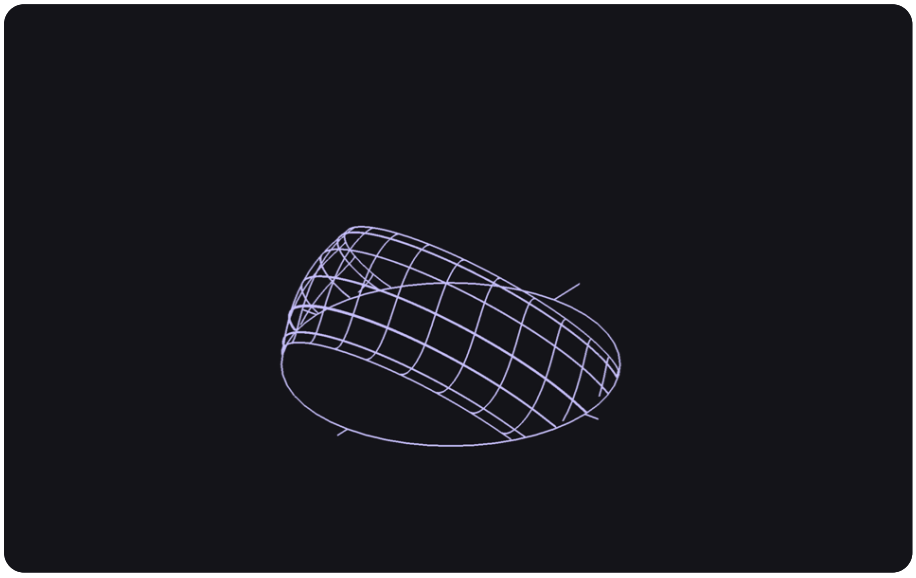
Environment

Environment brings the site into the design: where the sun is, which faces it reaches, how much light gets inside. These nodes are amber on the canvas.

Time is local standard time, without summer time. `13.5` is half past one. The **time zone** is the hours from UTC without summer time: Madrid `+1`, Montevideo `-3`.

Sun and shadow

Sun



What for. Where the sun is for a place, a day and an hour, and its path: the solar diagram, the ray of light for shadows and angles, its altitude and azimuth, sunrise and sunset.

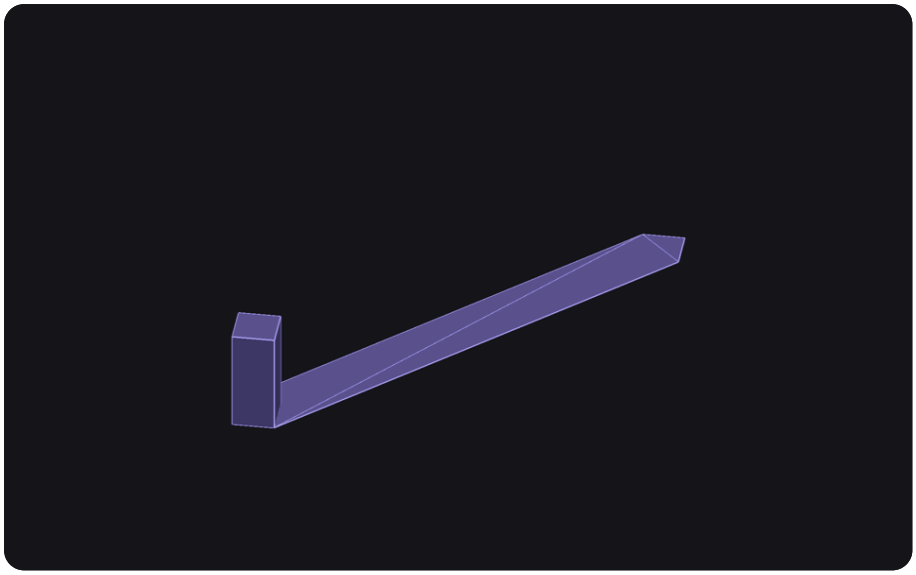
Input	What it means
location	a city of the list, or *other place* to give its **latitude**, **longitude** and **time zone**
day, **month**, **hour**	the moment; each can be a list
scale	the radius of the diagram and the distance of the sun, in metres
show	the *diagram*, the *day path*, the *position*, a *beam*, the *ray*, the *altitude*, the *azimuth*, *sunrise*…

Tips.

- The ▶ of the hour, in the panel, plays the day: shadows and angles move with it.

- For **Shadow** and **Incidence**, set **show** to *ray*.

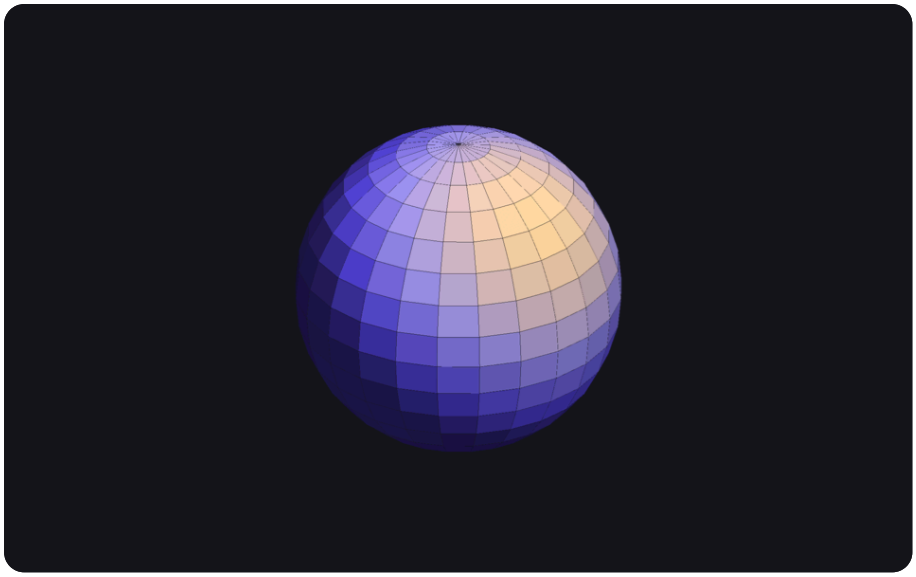
Shadow



What for. The shadow of some geometry with the ray of a Sun: on the ground, exact, with its outline and its area; or on other surfaces (a terrain, a façade), face by face.

Input	What it means
geometry	what casts the shadow
ray	the ray of a Sun; with a list of hours, the shadows of all of them
ground	the height of the ground
surfaces	optional: where it falls instead of the ground, face by face
show	the *shadow*, its *outline*, its *area*, or the faces *sunlit* or *in shadow*

Incidence

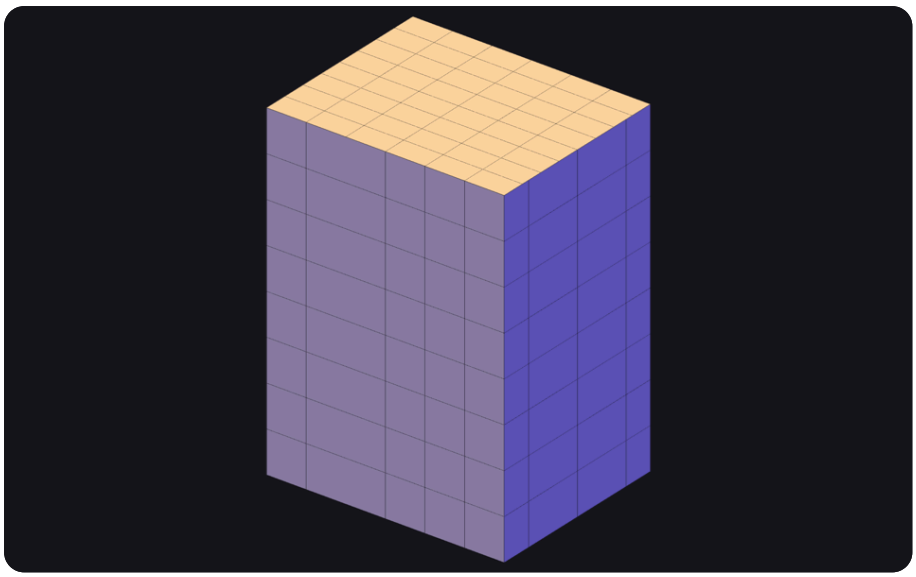


What for. How straight the sun reaches each face: 1 when it falls square, 0 when it grazes or comes from behind. For panels that open or close with the sun, or to see which side of a building gets the sun.

Input	What it means
faces	panels, a mesh, a surface or a box
ray	the ray of a Sun; with a list of hours, their average
side	*its face* (where each face looks) or *both faces*
show	*sunlight* (1 to 0) or the *angle* from the face, in degrees

Energy and light

Insolation



What for. The solar energy each face receives in a period, with what shades it: in kWh/m², the hours of sun, or how much sky it sees. For passive gain (much sun in winter, little in summer) and to place solar panels. An estimate for comparing options, not an energy or yield calculation.

Input	What it means
faces	what is analysed
obstructions	optional: what shades it (neighbours, trees, the rest of the building)
climate	optional: a **Climate** node, for the real sun of each hour; without it, a clear sky
location and its **latitude**, **longitude**, **time zone**	the place
period	*year*, *summer*, *winter*, *month* or *day*
show	*radiation*, *direct*, *diffuse*, *sun hours* or *sky view*

Tips.

- Paint the faces by its value (**Colour → by a value**) to see the result on the building.

Daylight

What for. How much natural light reaches some points inside a room under an overcast sky: the daylight factor, the lux, and the average and uniformity of the room. A simplified estimate for comparing options; for regulations or certification, a validated simulation.

Input	What it means
points	where it is measured: a grid at desk height, 0.7 to 0.85 m
room	the room: a box, or its walls, floor and ceiling
glass	the windows: rectangles on the walls
obstructions	optional: what blocks the sky outside (louvers, eaves, neighbours)
transmittance	how much light the glass lets through, in % (double glazing about 70)
reflectance	how much light the walls reflect, in % (light walls about 70; dark, 30)
show	*daylight factor*, *illuminance*, *sky component*, *average illuminance*, *average daylight factor* or *uniformity*

Tips.

- As a reference, 2 % is often taken as a well-lit room and 5 % as a very bright one.

Climate

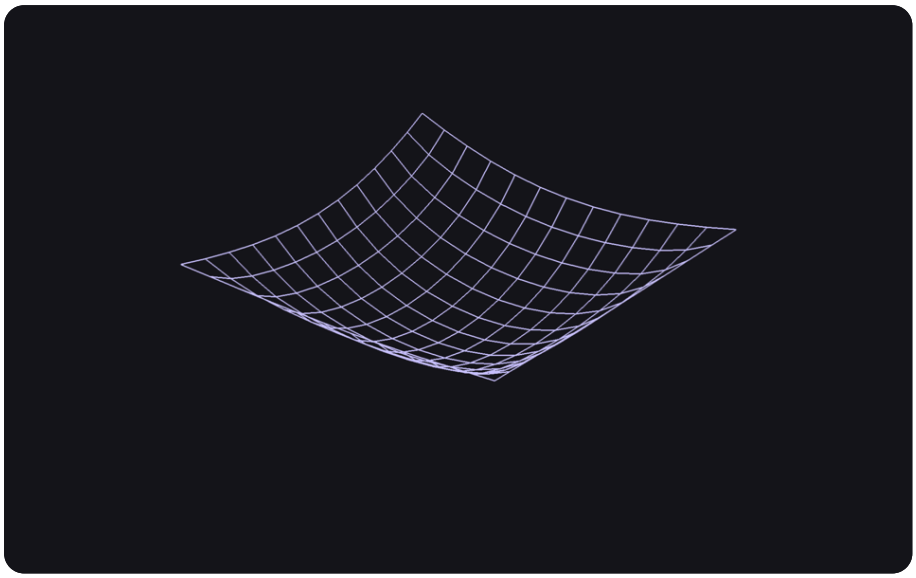
What for. The climate of a place from a weather file (`.epw`): temperature, humidity, sun and wind of each hour of a typical year. Connected to **Insolation**, the real sun instead of a clear sky.

Input	What it means
file	a weather file `.epw`, or the `.zip` it comes in (climate.onebuilding.org)
show	*temperature*, *humidity*, *radiation*, *wind speed* or *direction*, *cloud cover*
per	*hour* (8760), *day* (365) or *month* (12)

Simulation

Simulation lets the forces draw the shape, or tells you how hard a structure works. These nodes are rose on the canvas. Forces are in newtons: 1 kg weighs about 10 N, 1 kN is 1000 N.

Equilibrium



What for. The shape a cable, a net, a membrane or a shell takes at rest, hanging from its supports or lying on a floor, under its weight and its loads: the form finding of hanging models, cable nets, tensile fabrics, soap films and shells. Turned upside down, the vault that works mainly in compression.

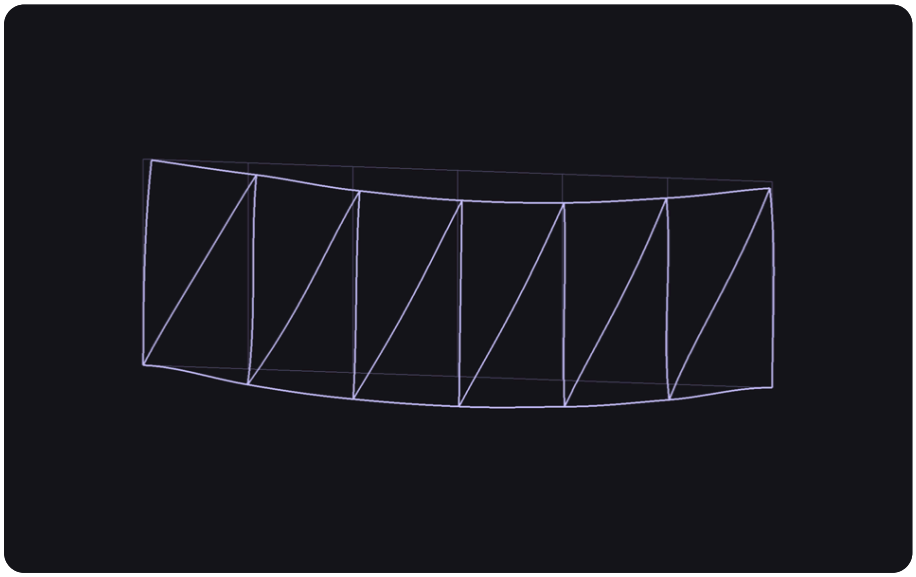
How. Connect what hangs to **shape**: curves (cables) or a mesh (a membrane, from **Mesh surface**). It always gives the shape already at rest: change a support or a load and it hangs again.

Input	What it means
shape	curves or lines (cables) or a mesh (a membrane)
spans	into how many spans each curve is divided; a mesh keeps its own
supports, **more supports**	*ends or corners*, the whole *border*, the *start*, or *none*; and points that also hold it
self weight	of all of it, in N
loads, **load**	points where a load hangs, and how much
stiffness	the force that would stretch it to twice its length: more, it stretches less
rest length	how long each span wants to be: `1` elastic, less pre-tensioned, `0` a soap film
diagonals	none, a cable net; one or both per square, a sheet that keeps its shape
bending	`0` a cable or a cloth; `0.001` rubber; `0.01` or more, a shell
floor	the height of a floor it rests on; with a floor it needs no supports
ties	lines whose ends are pulled together, as stitches (a gather, a pleat)
gravity	where the weight goes; upwards gives the shape that works in compression
moment	`1` at rest, `0` the start: the ▶ of the panel plays it
show	the *shape*, the *spans*, their *elongation* or their *forces*

Tips.

- Supports can be moved by hand in the 3D view with **Move supports**: the shape follows.
- The forces are for finding the shape, not for sizing cables or membranes.

Bar analysis



What for. How much a structure of bars bends and how hard each bar works: beams, columns, porticos, trusses, gridshells. A first sizing: it does not check buckling nor the joints, and applies no safety factors or building codes: an approximation for design, to be checked by a structural engineer.

Input	What it means
shape	the bars: lines, curves or a mesh (each edge a bar)
diagonals	of each square of a mesh: none (a frame) or braced like a truss
supports, **support**	points held, and how: *clamped*, *pinned* or *roller*
loads, **load**, **load per metre**	point loads, and a load spread along every bar (a slab, snow)
self weight	add the weight of the bars
material	steel, timber, glulam, concrete or aluminium
section	`IPE 300`, `HEA 200`, `rect 200x400`, `tube 200x100x8`, `pipe 168x8` … in mm
show	the *deformed* shape, *displacements*, *axial forces*, *moments*, *utilization*, *reactions*, the *largest* of each, or the *total weight*

Tips.

- **Utilization** above 1 (100 %) means the bar is too weak: a larger section, or more support. Below 1 is not a guarantee: it is a first check.

Topology optimization



What for. Where the material should be, inside a region, to carry some loads to some supports with the least of it. It starts full and thin, and turn after turn the material goes where it works, until a

structure is left: the arches and diagonals of a beam, the legs of a bridge. A form study: the result still has to be designed and checked as a structure.

Input	What it means
region	where the material can go: an outline (2D) or a body (3D)
voids, **solid parts**	what must stay empty (a door) or full (a slab)
supports, **rollers**	points or outlines held in every direction, or only along the weight
loads, **load**	where a force is applied, and how much
density	how much of the region stays full: `0.3` leaves 30 %
resolution	cells along the longer side: more, finer and slower
show	the *shape* (outlines to extrude or cut), the *plate*, the *cells*, the *density*, the *stress*…

Groups

How to group nodes, go inside a group, repeat it, or nest groups inside groups is told in Groups, among the key ideas. This page is about the one node of the family.

Group result

What for. Another result of a group that gives several. The card of the group gives the first one; each Group result takes one more, **without computing the group again**.

How. It appears by itself when you group nodes and two of them are used outside: next to the group card, with the name of the node it replaces. To take one more by hand, connect the group card to **group** and write in **gives** the name of the result you want.

Input	What it means
group	the card of a group with several results
gives	which result: the name of one of its *Group output* nodes

Tips.

- Why one card per result: a cable carries one thing. The group card gives the first result, and each Group result gives another.

Inside a group

Inside a group, the search offers two more kinds of node, which only exist there:

- **Group input:** what reaches the group from outside, one per input of its card.
- **Group output:** what leaves the group. Connect to it the result; add more Group outputs, each with its name, for a group that gives several.

Theory

Theory

You do not need any of this to design with Tensela. These pages are for whoever wants to know where things come from, and they come in two parts.

Designing with rules

The ideas and the mathematics of parametric design, which are the same in any tool:

- Visual algorithms: the algorithm as a recipe, graphs of nodes, and where parametric design comes from.
- Units and directions: metres, degrees, north and time.
- NURBS curves and surfaces: control points, degree, weights and knots.
- Curvature and developable surfaces: flat sheet or moulded.
- Meshes: geometry, topology, flat faces and subdivision.
- Form finding and the best value: how Equilibrium and *Find the best* work.

Inside Tensela

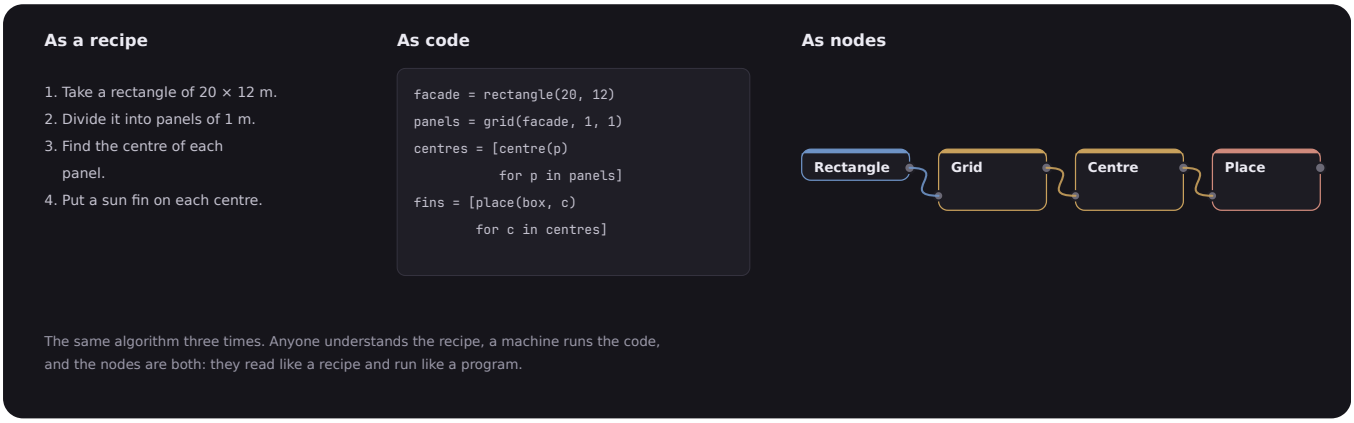
How this program in particular works, with diagrams and some code:

- How Tensela thinks: blocks of numbers with named levels.
- Named lists, inside: how a list is kept and the one rule that combines them.
- How a project is calculated: the graph, what is recalculated, and the limits.
- Lists in Tensela, Grasshopper and Dynamo: the same tasks in the three.

Visual algorithms

An algorithm is a recipe

An **algorithm** is a list of steps that, followed in order, always gives a result: a recipe. It does not need a computer. *Take a rectangle of 20 × 12 m, divide it into panels of 1 m, put a sun fin on the centre of each panel* is already an algorithm.



The same recipe can be written in three ways. In words, anyone understands it but no machine runs it. In code, a machine runs it but you need to learn a language to read it. A **visual algorithm** draws it as nodes and cables: it reads like the recipe and runs like the code.

What a visual algorithm is

A visual algorithm is a **graph**: a set of nodes joined by cables.

- **A node is a function:** it receives some inputs and gives one result. A Grid receives a rectangle and two sizes, and gives panels. It always gives the same result for the same inputs.
- **A cable is data:** it carries the result of one node to an input of another.
- **The data flow one way.** From the values you type, through the nodes, to the form. That is why the graph reads from left to right.
- **There are no loops.** A node cannot depend on its own result: Tensela refuses the cable that would close a circle, and says why. This is what makes the graph always give an answer, and what lets Tensela know exactly what to recalculate when something changes (How a project is calculated).

Repeating without loops

If there are no loops, how do you repeat a step 240 times? You do not: you give the step **a list of 240**, and it works on all of them at once. This is what lists with names are for.

When a step really has to work on its own result (a fractal, a curve that subdivides, a growth), a group can **repeat** itself a number of times: the loop is closed inside one card, with a count, so it always ends.

What makes it parametric

Parametric does not mean curved or complicated. It means that the design is described by its **parameters** (the width, the number of floors, the angle) and by the **dependencies** between its parts: the panels depend on the facade, the fins on the panels. Change a parameter and everything that depends on it follows.

The value is not in one form but in the **family of forms** the recipe can give: the same project makes a facade of 12 m or of 40 m, open or closed, and you choose among them by looking.

Where it comes from

Designing with rules is much older than computers.

- **Antoni Gaudí** found the form of his vaults by hanging chains with weights and turning the model upside down: the form came out of gravity, not of a drawing. The node Equilibrium does the same thing.
- **Frei Otto** studied soap films and hanging nets to find the forms of his tensile roofs, such as the Munich Olympic Stadium.
- **Luigi Moretti**, in Italy, spoke of *parametric architecture* in the middle of the 20th century, with studies of stadiums whose form came from parameters such as the view from each seat.
- **Ivan Sutherland's Sketchpad**, in 1963, was one of the first programs to draw with rules: lines that stay parallel, points that stay joined.
- In the late 2000s, **visual programming** reached architects, with tools such as Grasshopper and, later, Dynamo. Tensela comes from that tradition, and aims at a gentle learning curve.

The mathematics behind

Each family of nodes rests on a piece of geometry. You do not need it to design, but it helps to know where things come from:

- **Coordinates and vectors:** every point is three numbers, every direction an arrow. Units and directions.
- **Curves and surfaces:** control points that pull a smooth form. NURBS.
- **Curvature:** why some surfaces can be built with flat sheets and others cannot. Curvature.
- **Meshes:** forms made of flat faces, the language of fabrication and simulation. Meshes.
- **Forces and optimisation:** forms that find themselves, and the best value among many. Form finding.

Units and directions

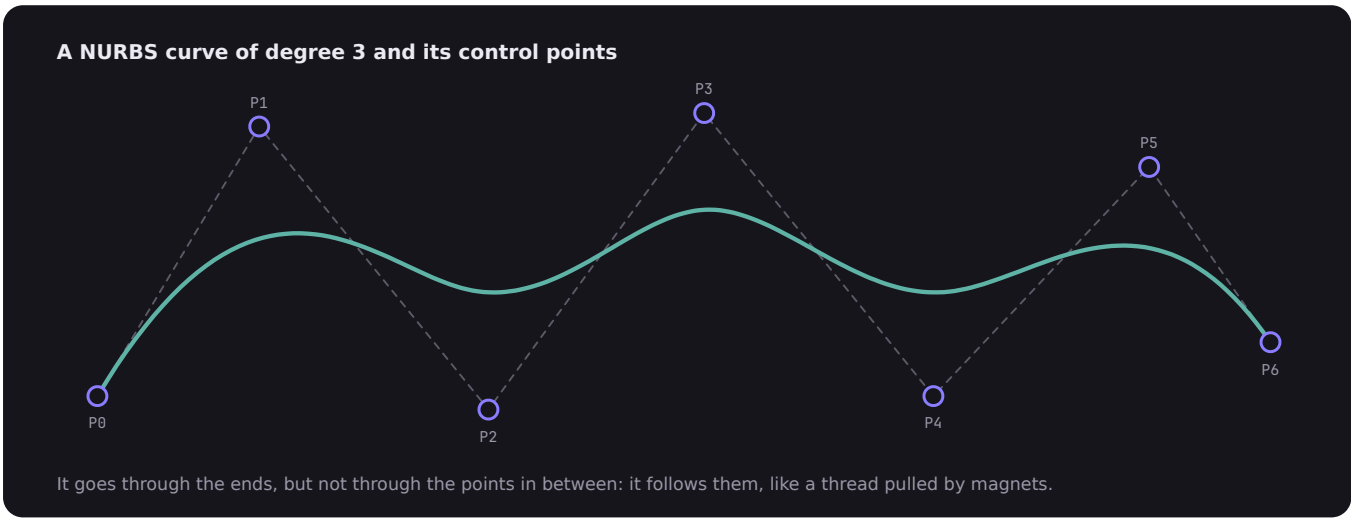
	In Tensela
Lengths	metres by default: `0.003` is a 3 mm board. You can show them in cm, mm, feet or inches (**Preferences → Units**) and type any unit (`120 cm`, `4'`); the project is always saved in metres
Areas, volumes	m², m³
Angles	degrees: `90` is a right angle; in a Formula, `sin(30)` is 0.5
Directions	X goes east, Y goes north, Z goes up
Plan, elevation, side	*plan* lies on the ground; *elevation* stands facing south; *side elevation* stands facing east
Forces	newtons: 1 kg weighs about 10 N, 1 kN is 1000 N
Time	local standard time, without summer time: `13.5` is half past one
Decimals	a point or a comma: `3.5` or `3,5`
Several values	separated by a semicolon: `1.5; 2; 3`
A point	`0 0 0`, `8 0 2.5` or `(0; 0; 1.5)`

Exports use millimetres where the machine expects them: 3D printing files (STL, 3MF) and cutting files (DXF, SVG) are written in millimetres, and the notice says so.

NURBS curves and surfaces

A **NURBS** is a smooth curve or surface defined by a few numbers. It is the usual way to describe free shapes precisely: a handrail, the edge of a slab, a shell.

Control points



The curve does **not** pass through its control points (only, usually, through the first and the last): they **pull it like magnets**. This is the difference with **Curve through points**, which passes through every point. The polygon of the control points is a good guide to the shape: the curve follows it, smoothed.

Degree

How smooth and how taut the curve is: `1` is the control polygon itself, `2` makes arcs, `3` is the usual smooth curve, and higher is tauter. A curve needs at least one more control point than its degree.

Weights

How strongly each control point pulls. What counts is the difference between them: all at `2` is the same as all at `1`. With the right weights, a NURBS draws an exact circle, which a plain polynomial curve cannot.

Knots

How the curve is spread between its control points: evenly, by distance, or by hand. Repeating a knot makes a kink. You only need them to fine-tune a curve.

Surfaces: u and v

A NURBS surface is the same in two directions: a grid of control points, rows by columns, with a degree in each direction. Its two directions are **u** (along the rows) and **v** (along the columns), each from `0` to `1`. Every point of the surface has its u and v, which is how nodes ask for *a point at the middle* or *the curves along u*.

Judging a curve

The **curvature comb** shows whether a curve is good: smooth tips mean a smooth curve; a jump, an abrupt change; a spike, a corner.

Curvature and developable surfaces

The builder’s question

Before anything else, a builder asks of a curved surface: **can it be made from flat sheet** (metal, plywood, card) **or does it have to be moulded?** Curvature answers it.

Gaussian curvature

- **Zero:** it can be made from flat sheet without stretching. Planes, cylinders, cones.
- **Positive:** it curves the same way in every direction, like a dome. It has to be moulded or cut in gores.
- **Negative:** it curves opposite ways, like a saddle. It has to be moulded too.

Mean curvature

How much the surface **bends** on average. A balanced saddle has a mean curvature of zero and still cannot be made from flat sheet: that is why both are needed. The sign of the mean depends on which way the face looks; for a pattern, use its absolute value (`abs` in a Formula).

Developable surfaces

A surface you get by bending a sheet without stretching it is **developable**: its Gaussian curvature is zero everywhere. **Surface curvature** answers yes or no, and **Unroll** flattens it exactly.

A surface that is not developable, like a dome, can still be built from flat pieces: cut it into gores with **Sub surface**, and unroll each one. **Unroll** says how much each piece stretches; with more gores, each stretches less.

Meshes

NURBS or mesh

A **NURBS** is one smooth entity defined by a formula. A **mesh** is a set of flat faces joined by their corners: it is not smooth, it only looks smoother the more faces it has. For free shapes that are hard to describe (a table, an object), a mesh smoothed by subdivision is easier than a NURBS.

Geometry and topology

A mesh has **vertices**, which are its **geometry** (where things are), and **faces**, which are its **topology** (how they are joined). The same points joined another way make another mesh; the same faces with one point moved make another geometry with the same topology.

The front of a face

The order of the corners of a face says which side is its front: counter-clockwise seen from the front. Two neighbouring faces are **compatible** when they go along their shared edge in opposite directions; a mesh whose faces are all compatible is **orientable**. **Mesh faces** with *orientation* says +1 or −1 for each face.

Flat panels

Triangles are always flat; a quadrilateral may not be. For free architecture built with flat glass or board, what matters is a mesh of **flat quadrilaterals**. A surface of revolution or of translation (one curve swept along another) gives them; a free form does not, and then **Planarize** flattens the panels, or **Triangulate** splits them into triangles at the cost of more joints.

Subdivision

Smooth mesh splits every face and moves the vertices towards the smooth surface the mesh stands for, level after level. Model a coarse cage with few faces, move its corners by hand with **Edit mesh**, and let the levels round it.

Form finding and the best value

Equilibrium: particles and springs

Equilibrium replaces a cable or a membrane by **particles** (points, each with a share of the weight) joined by **springs**: each spring pulls or pushes in proportion to how much it is stretched from its rest length. Some particles are supports and do not move. The particles move until every force balances.

The shape found this way carries its loads **only along the springs**, in tension or in compression. This is the principle of the hanging models of Gaudí and of the tensile structures of Frei Otto: hang it, and turn it upside down to get a vault that works only in compression.

Inside, it uses **dynamic relaxation**, the usual method for tensile structures: the particles move with the forces that are out of balance, and each time their motion peaks, they stop and start again from there, until they rest. The node always gives the shape already at rest.

Find the best

Find the best… moves some sliders until a number given by a node reaches its minimum, its maximum or a value: the least material, the most sun in winter, a given area.

- **Near** refines from where the sliders are: it tries a step up and a step down on each one, keeps what improves, and halves the steps when nothing improves. It finds the best value close to the start.
- **Over the whole range** keeps a population of candidates that cross and mutate, generation after generation, until the best stops improving. It explores more, and takes longer.

Both stay inside the ranges of the sliders and land on values the sliders can show.

How Tensela thinks

Blocks of numbers

Inside Tensela, every value is a **block of numbers**: one number, a list, a table, or a block with more directions. 240 panels of a facade are a block `row 12 × column 20`; the corners of each panel add one more direction, `corner 4`; their coordinates, one more, `x y z`.

In mathematics, a block of numbers with several directions is called a **tensor**. That is the *ten* of Tensela.



Directions with names

What makes Tensela different is that every direction of a block has a **name**: its **level**. *Row*, *column*, *floor*, *rib*. When a node combines two blocks, it lines them up by name:

- directions with the **same name** go one with one;
- directions with **different names** cross, every one with every other one.

The result has all the directions of both. That is the *tela*, the fabric, of Tensela: two lists are the warp and the weft, and every crossing is a piece. See Lists with names.

Why it is fast

A node does not go element by element: it computes the **whole block at once**. Moving 240 panels, or 24 000, is one operation on a block. This is why a facade of thousands of pieces can update while you drag a slider.

Why there is nothing to arrange

Because directions have names and no order, `row × column` and `column × row` are the same block. There is no need to transpose, flatten or graft lists before combining them: the names say how they go together. When two lists with the same name have different sizes, a notice says so instead of guessing.

To go further

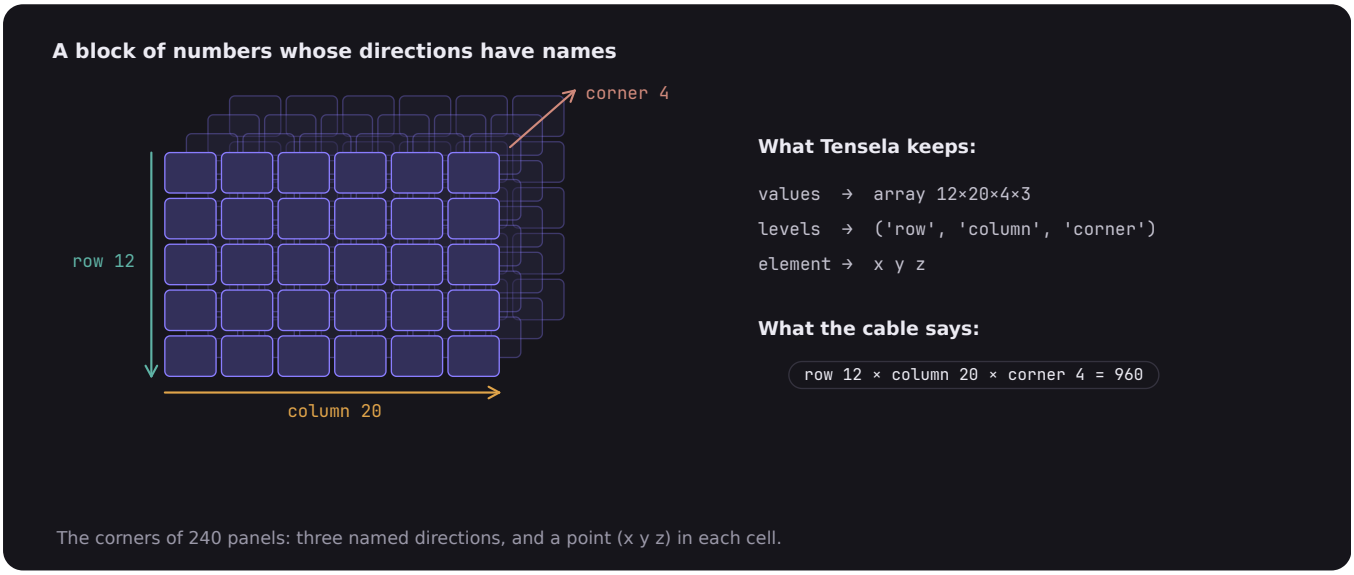
- Named lists, inside: the rule in code, levels of variable size and holes.
- How a project is calculated: what is recalculated when you change something.
- Lists in Tensela, Grasshopper and Dynamo: the same tasks in the three tools.

Named lists, inside

This page is for whoever wants to know how lists with names work inside, with some code. You do not need it to design. The code is Python, simplified from the source of Tensela (the file `data.py`).

What a value is

Every value in Tensela, from a single number to the corners of a thousand panels, is a **Data**: a block of numbers (a *numpy* array) and the names of its first directions, its **levels**.




```
centres = Data(values,                    # a numpy array of 12 × 20 × 3
               levels=("row", "column"),  # the names of the first two directions
               name="centres")

centres.sizes   # {"row": 12, "column": 20}
centres.shape   # "row 12 × column 20 = 240"   (what the cable says)
```

The directions after the levels describe **each element**: a number has none, a point has one of size 3 (x y z), a rectangle has its four corners. So the block of 240 centres is 12 × 20 × 3, but only *row* and *column* are lists.

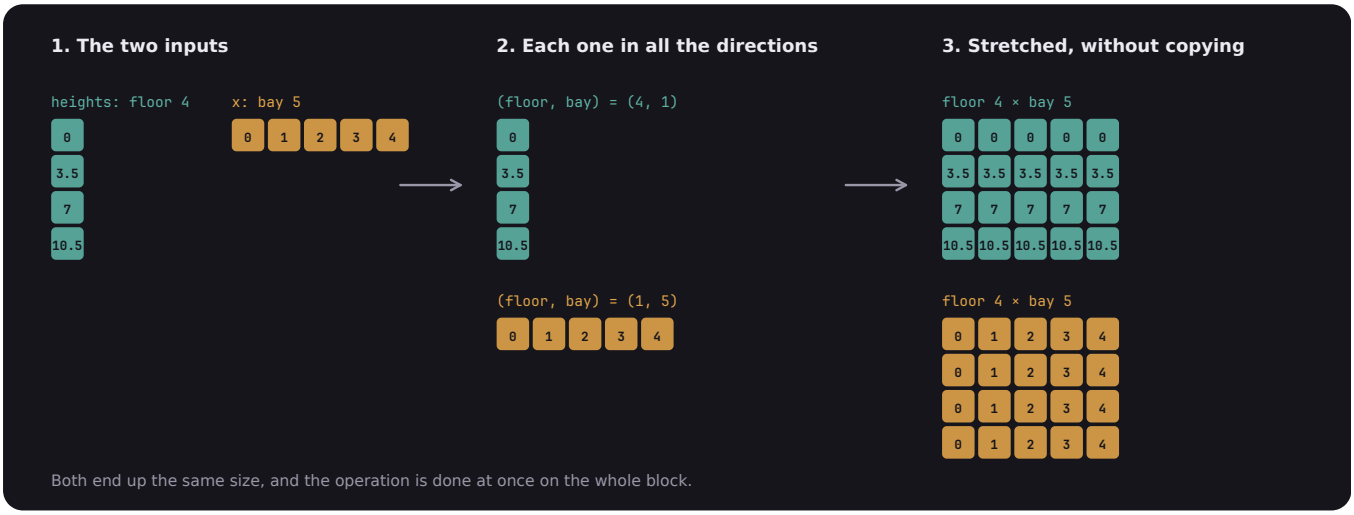
The one rule

Every node that combines inputs goes through the same function, `align`. It has three steps:

1. **The result has all the levels of the inputs**, in the order they appear.
2. **A level that is in two inputs must have the same size in both**; it pairs one with one. If the sizes differ, Tensela stops with a notice instead of guessing.
3. **A level that is in only one input crosses** with all the others.

```
def align(*inputs):
    levels, sizes = [], {}
    for data in inputs:
        for level, size in data.sizes.items():
            if level not in sizes:              # a new level: it goes into the result
                levels.append(level)
                sizes[level] = size
            elif sizes[level] != size:          # the same name, another size: a clear notice
                raise ShapeError(f"The sizes of “{level}” do not match: …")
    return levels, [extend(d, levels, sizes) for d in inputs]
```

`extend` puts each block in all the directions of the result. It turns the block to the order of the result, gives it size 1 in the levels it does not have, and **stretches** it to the full size. The stretching is numpy's *broadcasting*: no number is copied, the block is only read as if it were bigger.




```
def extend(data, levels, sizes):
    order = sorted(range(len(data.levels)), key=lambda axis: levels.index(data.levels[axis]))
    block = numpy.transpose(data.values, order + element_axes)
    block = block.reshape([data.sizes.get(n, 1) for n in levels] + element_shape)
    return numpy.broadcast_to(block, [sizes[n] for n in levels] + element_shape)
```

With the inputs aligned, a node is written **once, for whole blocks**, and works the same for one element or for 24 000:

```
>>> heights = Data([0, 3.5, 7, 10.5], levels=("floor",), name="heights")
>>> x = Data([0, 1, 2, 3, 4], levels=("bay",), name="x")
>>> points = apply(make_point, x, heights)        # different names: they cross
>>> points.shape
'floor 4 × bay 5 = 20'
>>> shift = Data([0, 0.2, 0.4, 0.6], levels=("floor",), name="shift")
>>> apply(add, heights, shift).shape              # the same name: one with one
'floor 4'
>>> apply(add, heights, Data(range(5), levels=("floor",), name="five"))
ShapeError: The sizes of “floor” do not match: 4 in “heights”, 5 in “five”.
```

Working on one level, by its name

Because levels have names, an operation can say **which** level it works on without counting positions:

```
reduce(areas, "sum", along="panel")      # the area of each floor: the panels are added up
reorder(points, "floor", mode="invert")  # the floors upside down; the bays as they were
```

Summarise, **Reorder**, **Filter**, **Pick** and **Split** all ask for a level by name. There is no *transpose*: `floor × bay` and `bay × floor` are the same block, because the order of the levels never decides anything.

Lists that are not rectangular

Real projects are not always regular blocks:

- **Levels of variable size.** A tower with more panels at the bottom than at the top: the level *panel* has a different size on each floor. Tensela keeps the block padded to the largest size, the **count** of panels of each floor, and a **mask** that says which cells really exist. The nodes only read the cells that exist.
- **Holes.** By default, **Filter** does not remove elements: it leaves them **empty in their place**, so the grid still reads the same (the panel of row 3, column 7 is still there, or is an empty place). With *remove: yes*, the ones that stay are numbered again.
- **Numbering.** The elements of a level are numbered from 0, unless the level starts at another number: with two basements, the floors go from −2.

A safety net

Before calculating, `align` multiplies the sizes of the result. Above **5 million elements** it stops with a notice instead of freezing the computer: it is almost always two levels crossing that should have been paired (the same name), or one digit too many.

Why “tensor”

A block of numbers with several directions is what mathematics calls a **tensor**. Most programs number its directions (axis 0, axis 1, axis 2). Tensela names them. That small change is what removes the need to arrange data by hand, and it is where the name *Tensela* comes from: How Tensela thinks.

How a project is calculated

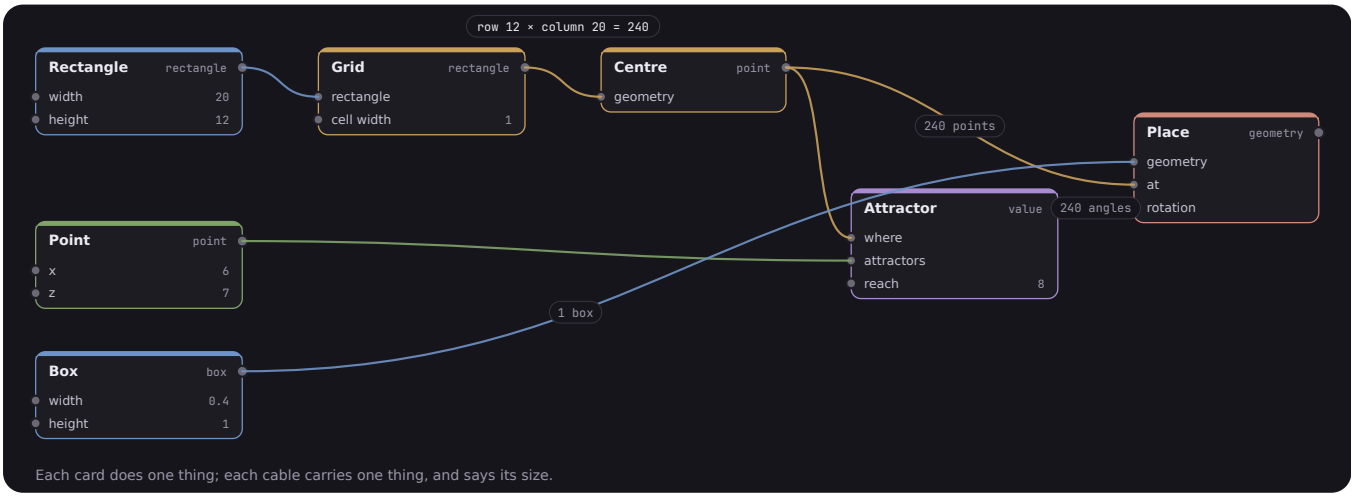
What happens between the moment you change a number and the moment the 3D view shows the new form. For whoever wants to know; you do not need it to design.

A project is a graph

A Tensela project is a list of nodes, each with its inputs: a fixed value, or a connection to another node. The `.tela` file says exactly that, in plain text (JSON). This is the whole of Your first project:

```
{
  "nodes": {
    "facade":    {"type": "Rectangle", "width": 20, "height": 12, "plane": "elevation"},
    "panels":    {"type": "Grid", "rectangle": {"node": "facade"}, "cell_width": 1, "cell_height":
1},
    "centres":   {"type": "Centre", "geometry": {"node": "panels"}},
    "point":     {"type": "Point", "x": 6, "z": 7},
    "attractor": {"type": "Attractor", "where": {"node": "centres"}, "attractors": {"node":
"point"},
                  "reach": 8, "near": 90, "far": 0},
    "fin":       {"type": "Box", "width": 0.4, "depth": 0.05, "height": 1.0},
    "fins":      {"type": "Place", "geometry": {"node": "fin"}, "at": {"node": "centres"},
                  "rotation": {"node": "attractor"}}
  }
}
```

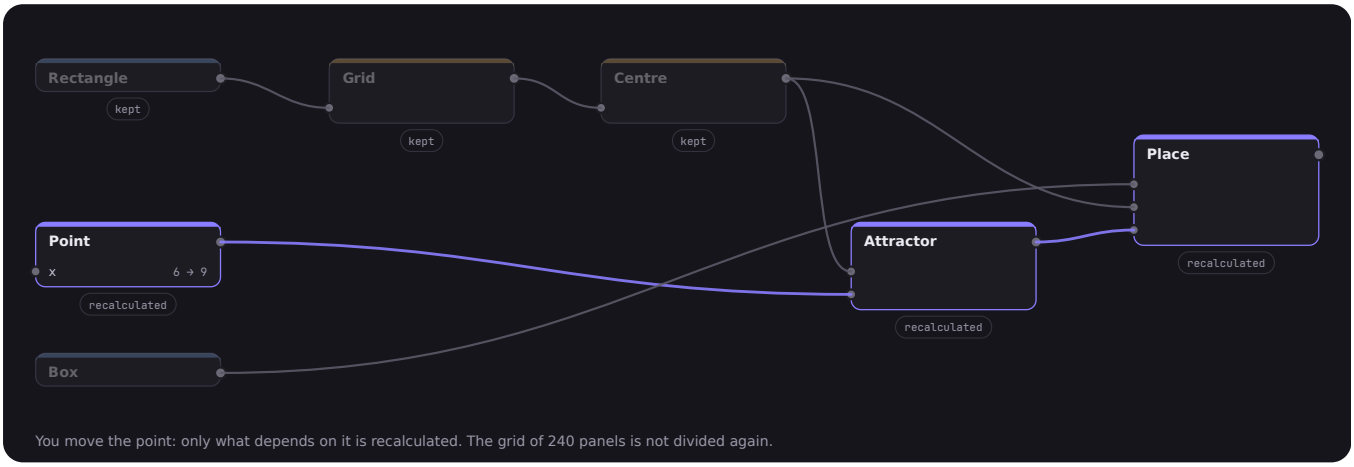
`{"node": "facade"}` is a cable. The kinds of nodes and their inputs are always written in English in the file, so a project made with Tensela in Spanish opens the same in any language. Since it is text, a project can be read, compared between two versions, or kept in a version control system like any other text file.



Calculated when asked, kept until it changes

Tensela does not calculate the whole project each time. It works the other way round:

1. When the view needs the result of a node, Tensela asks for its inputs first, and those ask for theirs, back to the values you typed.
2. Each node calculates **once**, and keeps its result.
3. When you change an input, Tensela clears the kept result of **that node and of everything that depends on it**, and nothing else.



You move the point: only what depends on it is recalculated. The grid of 240 panels is not divided again.

So when you drag the attractor point, the grid of 240 panels is not divided again: only the point, the Attractor and the Place are recalculated. That is why a large project still answers while you drag a slider.

```
def change(node, input, value):
    node.inputs[input] = value
    clear(node)                       # this node and everything after it

def clear(node):
    node.result = None
    for after in nodes_that_use(node):
        clear(after)

def result(node):
    if node.result is None:           # not kept: calculate it from its inputs
        inputs = {name: result(v) if is_node(v) else v for name, v in node.inputs.items()}
        node.result = node.kind.function(**inputs)
    return node.result
```

No loops

Because a node is calculated from its inputs, a node cannot depend on itself. If a cable would close a circle, Tensela does not connect it and says so: *Connecting “a” to “b” would make a loop: “b” already comes before “a”.* To repeat a step on its own result, use a group that repeats.

The window keeps responding

If a calculation takes more than about a tenth of a second, it goes on **in the background**: the 3D view keeps the last result, *Computing… Stop* appears, and you can keep working. If you change something in the meantime, that calculation is dropped (it no longer serves) and a new one starts with the new values. **Stop** drops it and goes back to what was shown.

Errors in words

When a node cannot calculate, its card says why, in words: what is missing, what does not fit, and usually how to fix it. The error stays in that node and in the ones that depend on it; the rest of the project keeps working.

Limits that protect your work

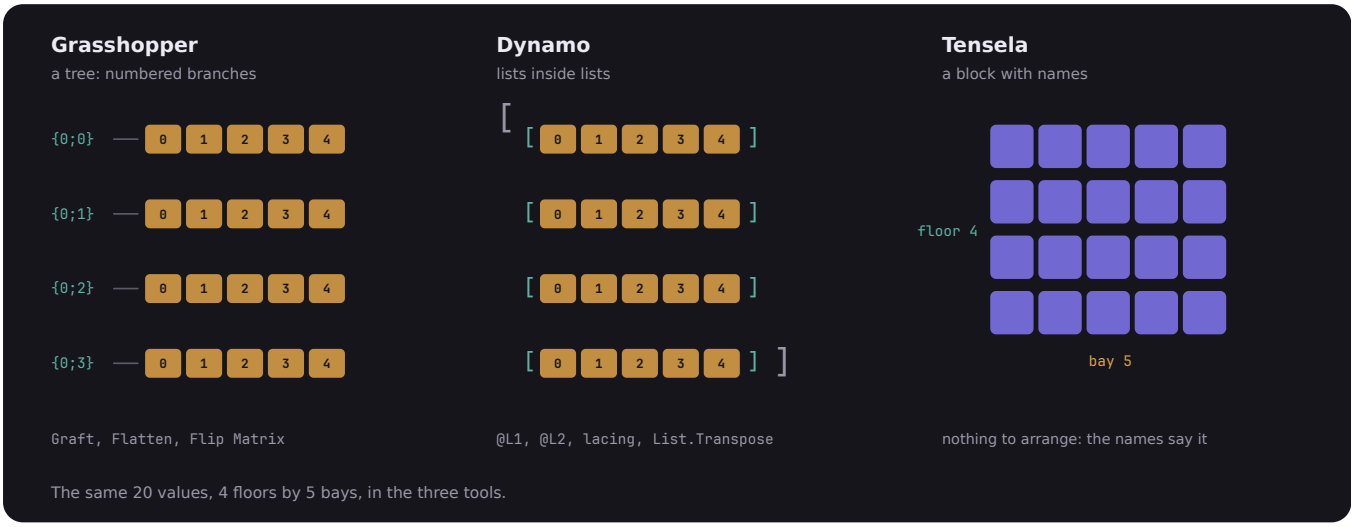
- **5 million elements** per result at most: above that, a notice before calculating. It is almost always a crossing of levels that should have been a pairing.
- **50 000 pieces** in the view: above that, the view asks before drawing (**Show them** or **Turn off its eye**).
- **The Code node** is made not to reach the network: Tensela checks the code before running it, and stops it if it tries to open files, reach the network or start other programs. It runs your Python with only a list of allowed names, and stops it after 10 seconds.

Groups

A group is a graph inside a node. Inside, it calculates like any project; outside, it is one card with its inputs and its output. A group that **repeats** runs its graph several times, passing its result back in, a fixed number of times.

Lists in Tensela, Grasshopper and Dynamo

If you come from Grasshopper or Dynamo, the biggest change is how lists are kept and combined. This page puts the three side by side, with the same 20 values: 4 floors by 5 bays.



Grasshopper: a tree of numbered branches

In Grasshopper, data are kept in a **tree**. Each branch has a **path** of numbers, such as `{0;2}`, and holds a list. The 4 floors are 4 branches, each with a list of 5 bays.

- When a component receives two inputs, it matches them item by item and, by default, repeats the last item of the shorter list (*longest list*).
- To combine **every floor with every bay**, you change the structure first: **Graft** one list (each item to its own branch), or use **Cross Reference**.
- **Flatten**, **Graft**, **Simplify**, **Flip Matrix** and **Shift Paths** exist to reshape the tree before combining.

It is very powerful and very explicit: you control the shape of the tree, step by step.

Dynamo: lists inside lists

In Dynamo, data are **nested lists**: a list of 4 floors, each a list of 5 bays.

- How two lists combine is chosen per node with its **lacing**: *shortest*, *longest* or *cross product*.
- **Replication guides** (`<1>` , `<2>`) and **list levels** (`@L1` , `@L2`) say which level of the nested lists a node works on.
- **List.Transpose** swaps the inside and the outside.

Tensela: a block with names

In Tensela, the 20 values are one block, `floor 4 × bay 5`. The names decide how it combines: the **same name** pairs, **different names** cross. There is no tree to reshape and no lacing to choose; the rule is always the same. How it works inside: Named lists, inside.

The same tasks in the three

Task	Grasshopper	Dynamo	Tensela
Every floor with every bay	Graft one input, or Cross Reference	lacing *cross product*, or `<1>` `<2>`	different names: by itself
Each floor with its own value	the same tree structure on both sides	lacing *shortest* or *longest*	the same name: by itself
Turn only the floors upside down	Flip Matrix, Reverse List, Flip Matrix	List.Reverse at the right level (`@L2`)	**Reorder**, level *floor*
Swap floors and bays	Flip Matrix	List.Transpose	nothing to do: levels have no order
The total of each floor	Mass Addition on each branch	Math.Sum at the right level	**Summarise**, sum along *bay*
Two lists that do not fit	the last item of the shorter one is repeated	depends on the lacing	a notice with the two sizes

What you gain, and what changes

- **You gain:** the cable says what it carries (`floor 4 × bay 5`), the same rule works in every node, and an operation on a level says which one by its name. A mistake in sizes gives a notice with both sizes.
- **What changes:** levels have no position, so there is no *the second level*: you name it. Irregular data, such as more panels at the bottom of a tower than at the top, are kept as **levels of variable size**, and elements taken out by a filter leave **holes**, so the grid still reads the same.

Rhino and Grasshopper are trademarks of Robert McNeel & Associates. Dynamo and Revit are trademarks of Autodesk. Tensela is not associated with them.

More

Common stumbles

Things that confused people while Tensela was being tested, so you do not have to find them out yourself.

Installing and opening

- **It does not open with a double-click.** First check that it may run (step 2 of Install). Then open it from a terminal, `./Tensela-*.AppImage`, to read what it says. If it speaks of **FUSE**, your system lacks a library that AppImages use: install it (on Ubuntu 24.04 and newer `sudo apt install libfuse2t64`, before that `libfuse2`), or open Tensela without it with `./Tensela-*.AppImage --appimage-extract-and-run`.
- **The 3D view stays black, or the window closes as it opens.** It is the graphics: Tensela needs OpenGL 3.2 or newer (see What you need). Update your system; in a virtual machine, turn on its 3D acceleration.
- **The program is in another language. Preferences → General → Language**; it applies the next time you open Tensela.

Writing numbers

- **Decimals:** `3.5` and `3,5` both work.
- **Several values:** always separated by a semicolon, `1.5; 2; 3`. With decimals, `0,0,1,5` cannot be read: write `0; 0; 1,5` or `0 0 1.5`.
- **A point:** `0 0 0`, `8 0 2.5` or `(0; 0; 1.5)`.
- **Units:** measures are in metres unless you change it in **Preferences → Units**. You can also type the unit with the number, `120 cm`, and Tensela converts it.

In the card

- **Tab** jumps to the next value of the card.
- **Delete** inside a field erases letters, not the node.
- **An empty level** in Reorder, Filter or Split means *automatic*: the card says which level it took, for example `auto: point`.

Common situations

- **Raising a point without typing coordinates:** use **Move** with a distance (it goes up by default), or **Line by direction**.
- **After Filter**, with **remove: yes** the elements that stay are numbered again from 0. With **no**, the others stay in their place, empty.

- **A curve with empty points** passes through the points that are there.
- **A rectangle is not a closed body**, so it has no volume. To measure a volume, extrude it with caps first.
- **Reliefs without formulas:** a guide curve, a **Distance** to it and a **Graph** to shape the answer.
- **A smooth surface with steps:** a smooth surface through a coarse grid of points shows small steps at sharp edges. Use more points, or turn **smooth** off for shapes with edges.

When it goes slowly

- **Turn off the eyes** of the nodes you do not need to see: drawing thousands of pieces often takes longer than calculating them.
- **Try ideas with small counts** (fewer panels, fewer sections) and raise them at the end.
- **The heavy nodes** (Topology optimization, Equilibrium, Insolation) take longer the finer they work: lower the resolution, the spans or the number of panels you give them while you look for the form.

When the view asks before drawing

If a node with its eye on makes more than 50 000 pieces (or 2 million points), a strip appears above the view: **Show them** or **Turn off its eye**. It usually means a digit too many in a count, or two lists crossing by mistake. Check the names of their levels.