Table of Contents

A gentle introduction

Python is a widely-used, open-source programming language, recognized for its simplicity, versatility, and readability. It is often embedded in applications as a scripting language, and FreeCAD is no exception. This integration allows users to automate tasks, customize workflows, and extend the software's functionality far beyond its graphical interface.

Python offers several advantages that make it particularly suitable for FreeCAD users. It is beginner-friendly, with a clear and intuitive syntax that makes it easy to learn—even for individuals with no prior programming experience. This accessibility is especially valuable in the FreeCAD community, where many users come from diverse backgrounds, such as engineering, architecture, and design, without extensive coding expertise.

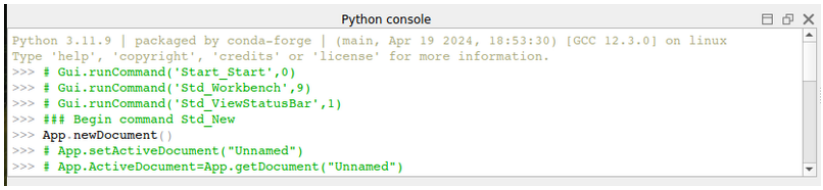
Additionally, Python's widespread adoption in other applications, such as Blender for 3D modeling, Inkscape for vector graphics, and GRASS GIS for geospatial analysis, makes it an essential skill for anyone looking to expand their scripting capabilities. Mastering Python in FreeCAD not only enhances your ability to create custom tools and macros but also provides transferable skills that can be applied across various software platforms.

In FreeCAD, Python scripting enables you to:

- Automate repetitive tasks to save time and reduce errors.
- Create custom parametric objects that dynamically adapt to changes.
- Develop personalized macros and tools tailored to specific workflows.
- Interact with FreeCAD's Application Programming Interface (API) to access and manipulate geometry, scenes, and user interface elements programmatically.

By leveraging Python, FreeCAD users can unlock the full potential of the software, transforming it into a powerful and flexible tool tailored to their unique needs.

FreeCAD includes an advanced Python console, accessible via **View** → **Panels** → **Python** Console. This tool lets users perform operations beyond the graphical interface, such as accessing advanced features, troubleshooting geometry, and automating tasks. It also logs Python commands for GUI actions if the option *Show script commands in Python console* is enabled under **Edit** → **Preferences** → **Python** → **Macro**). By keeping the console open, you can watch Python code unfold as you work, offering an intuitive way to learn the language while exploring FreeCAD's capabilities. Lastly, FreeCAD also has a macros system, which allows you to record actions to be replayed later. This system also uses the Python console, by simply recording everything that is done in it.



In this chapter, we will discover very generally the Python language. If you are interested in learning more, the FreeCAD documentation wiki has an extensive section related to Python programming.

Writing Python code[edit | edit source]

In FreeCAD, you can write Python code in two main ways: through the Python console (**View** → **Panels** → **Python Console**) or using the Macro editor (**Macro** → **Macros** → **Create**). The Python console allows you to enter commands one at a time, which are executed immediately upon pressing Return, making it perfect for quick testing or interactive exploration. The Macro editor, on the other hand, is used for writing and saving more complex scripts consisting of multiple lines of code. These macros can be executed as a whole later from the Macros window, providing a powerful way to automate repetitive tasks and extend FreeCAD's functionality. In this chapter, you will be able to use both methods, but it is highly recommended to use the Python Console since it will immediately inform you of any errors you make while typing.

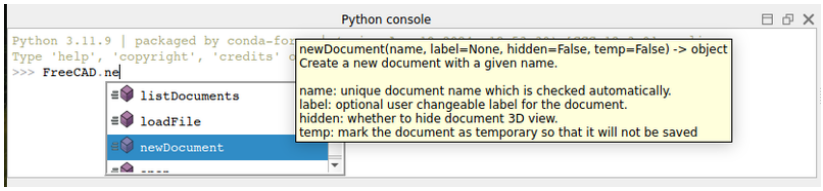
If this is your first time using Python, consider reading this short introduction to Python programming before going any further, it will make the basic concepts of Python clearer.

Manipulating FreeCAD objects[edit | edit source]

Let's start by creating a new empty document:

```
doc = FreeCAD.newDocument()
```

In the FreeCAD Python console, as soon as you type FreeCAD. (the word "FreeCAD" followed by a dot), an autocomplete window appears. This feature not only speeds up your workflow by suggesting available commands but also helps you discover new functions and features within FreeCAD. Each entry in the list includes a tooltip explaining its purpose, making it easier to understand and explore the available functionality. This autocomplete feature is especially helpful for beginners learning Python scripting and for advanced users navigating FreeCAD's extensive API efficiently. Take a moment to explore the options in the autocomplete window—you may uncover commands that simplify your workflow or open up new possibilities.



Typing **FreeCAD.newDocument()** creates a new, empty document in FreeCAD, just like clicking the **New Document** button in the toolbar. When you execute **doc = FreeCAD.newDocument()**, the new document object is assigned to the variable **doc**, allowing you to manipulate it programmatically. By using doc, you can add objects, change properties, or save the document.

In Python, the dot (.) is used to indicate that one element is contained within another. For example, **newDocument** is a function within the **FreeCAD module**, which is why we write FreeCAD.newDocument. The autocomplete window that appears shows everything available within the FreeCAD module. If you were to type a dot after newDocument (without adding the parentheses), it would display everything that belongs to the newDocument function. This illustrates how Python organizes and accesses objects and their components. It's important to note that the parentheses are mandatory when calling a Python function, as they signal the execution of the function.

Now let's get back to our document. Let's see what we can do with it. Type the following and explore the available options:

```
doc.
```

In FreeCAD, the naming conventions for Python commands can help you understand their purpose:

- Names that begin with an upper-case letter are typically attributes, which store values or data, such as properties of an object.
- Names that begin with a lower-case letter are usually functions (also called methods), which perform actions or operations, such as creating or modifying objects.
- Names that begin with an underscore (_) are generally intended for internal use within the module and can usually be ignored.

This distinction helps you navigate FreeCAD's API and select the appropriate command for your needs. For example, you might use a method to add a new object to a document. The method performs the action of creating and adding the object, while attributes store the object's properties. Understanding this structure makes it easier to explore the functionality available, especially when using the autocomplete feature or reading tooltips. Let's add a box.

```
box = doc.addObject("Part::Box", "myBox")
```

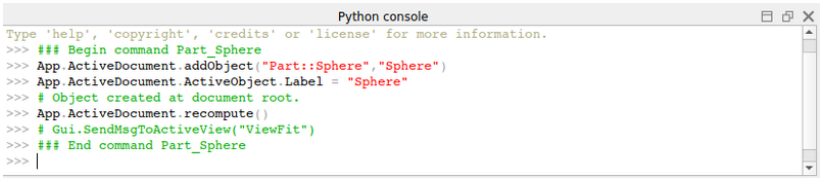
The command **box = doc.addObject("Part::Box", "myBox")** adds a new 3D box object to the document. Here's a simple explanation:

- **doc.addObject**: This tells FreeCAD to add a new object to the document.
- **Part::Box**: This specifies the type of object to create, in this case, a 3D box.
- **myBox**: This assigns a name to the new object, making it easier to identify and reference later.

The result of this command is that a box appears in the active document, and the variable box stores a reference to it so you can modify or interact with it later. Our box is added in the tree view, but nothing happens in the 3D view yet, because when working from Python, the document is never recomputed automatically. We must do that manually, whenever required:

```
doc.recompute()
```

Now our box has appeared in the 3D view. Many of the toolbar buttons that add objects in FreeCAD actually do two things: add the object, and recompute. Try now adding a sphere with the appropriate button in the Part Workbench, and you will see the two lines of Python code being executed one after the other.



You can get a list of all possible object types like Part::Box:

```
doc.supportedTypes()
```

Now let's explore the contents of our box:

```
box.
```

You'll immediately see a couple of very interesting things such as:

```
box.Height
```

This will print the current height of our box. Now let's try to change that:

```
box.Height = 5
```

If you select your box with the mouse, you will see that in the properties panel, under the **Data** tab, our **Height** property appears with the new value. All properties of a FreeCAD object that appear in the **Data** and **View** tabs are directly accessible by Python too, by their names, like we did with the Height property. Data properties are accessed directly from the object itself, for example:

```
box.Length
```

In FreeCAD, visual properties are managed by a **ViewObject**. Every FreeCAD object has an associated ViewObject, which stores information about how the object is displayed in the graphical interface, such as color, transparency, or visibility. However, the ViewObject is only accessible when FreeCAD is running with its graphical interface. If FreeCAD is launched in a non-graphical mode—such as from a terminal with the -c command line option or when used as a Python library in an external script—the ViewObject is not available. This is because there is no visual representation of the object in these modes, as the graphical interface is not loaded.

Try the following example to access the line color of our box:

```
box.ViewObject.LineColor
```

Vectors and Placements[edit | edit source]

Vectors are a fundamental concept in any 3D application. A vector is essentially a list of three numbers (x, y, and z) that describe a point, position, or direction in 3D space. Vectors are crucial for defining geometry, transformations, and interactions in the 3D environment. They serve as building blocks for operations such as translations, rotations, and scaling.

In FreeCAD, vectors are used extensively for creating and manipulating objects. They enable a wide range of mathematical operations, such as addition, subtraction, cross products, dot products, and projections. These operations allow you to calculate distances, angles, and directions or define relationships between objects in space.

Understanding vectors and how they work is essential for effective scripting and customization in FreeCAD. For example, vectors are used to position objects, define their orientation, or even calculate the paths for complex operations like sweeps or lofts.

In FreeCAD, vectors are represented using the **Vector** class, which provides various methods and properties for performing operations and accessing their components. Mastering these capabilities will significantly enhance your ability to interact with FreeCAD's 3D environment programmatically. In FreeCAD vectors work like this:

```
myvec = FreeCAD.Vector(2, 0, 0)
print(myvec)
print(myvec.x)
print(myvec.y)
```

```
othervec = FreeCAD.Vector(0, 3, 0)
sumvec = myvec.add(othervec)
```

Here is a short breakdown of the above commands:

- **myvec = FreeCAD.Vector(2,0,0)**: Creates a vector myvec with X = 2, Y = 0, Z = 0.
- **print(myvec)**: Prints the vector myvec with its components (X, Y, Z).
- **print(myvec.x)**: Prints the X-component of myvec.
- **print(myvec.y)**: Prints the Y-component of myvec.
- **othervec = FreeCAD.Vector(0,3,0)**: Creates a second vector othervec with X = 0, Y = 3, Z = 0.
- **sumvec = myvec.add(othervec)**: Adds myvec and othervec to create a new vector sumvec containing the sum of their components.

Another common feature of FreeCAD objects is their **Placement**. As we saw in earlier chapters, each object has a Placement property, which contains the position (Base) and orientation (Rotation) of the object. These properties are easy to manipulate from Python, for example, to move our object:

```
print(box.Placement)
print(box.Placement.Base)
box.Placement.Base = sumvec
otherpla = FreeCAD.Placement()
otherpla.Base = FreeCAD.Vector(5, 5, 0)
box.Placement = otherpla
```

Here is a short breakdown of the above commands:

- **print(box.Placement)**: Prints the placement of the box, which includes its position, rotation, and orientation in 3D space.
- **print(box.Placement.Base)**: Prints the position (Base) of the box, which is a vector representing its location in 3D space.
- **box.Placement.Base = sumvec**: Sets the base position (location) of the box to the vector sumvec, effectively moving the box to this new position.
- **otherpla = FreeCAD.Placement()**: Creates a new placement object named otherpla.
- **otherpla.Base = FreeCAD.Vector(5,5,0)**: Sets the base position of otherpla to a vector at coordinates (5, 5, 0).
- **box.Placement = otherpla**: Applies otherpla to the box, updating its placement to the new position and orientation defined by otherpla.

Read more

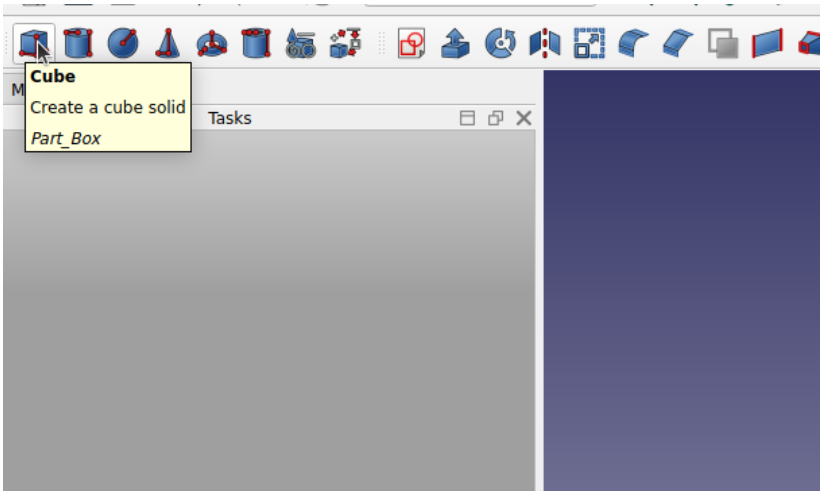
- Python
- Macros
- Introduction to Python
- Python scripting tutorial
- Power users hub

All workbenches at a glance

As previously mentioned, FreeCAD offers various workbenches, each dedicated to different applications. Although the multitude of options might seem overwhelming at first, each workbench is designed to cater to specific tasks, making the overall workflow more efficient and tailored to various project requirements. For instance, the Part Design workbench is ideal for creating and modifying solid 3D models, while the Draft workbench is perfect for 2D drafting and drawing. This modular approach allows users to customize their interface and toolset according to their specific needs and preferences.

On this page, you will find information regarding the basic set of workbenches and their functionalities. For additional information, feel free to refer to each workbench page in the FreeCAD documentation for a more complete list.

An interesting feature of FreeCAD is the ability to obtain additional information by hovering the mouse over a command. This tooltip functionality helps users understand what each command does, providing guidance and making it easier to learn and navigate the software.



Four workbenches are also designed to work in pairs, with one fully incorporated into the other: BIM contains all the Draft tools, and Part Design includes all the Sketcher tools. However, for clarity, they are separated below.

Part[edit | edit source]

The Part Workbench offers fundamental tools for working with solid parts, including primitives like cubes and spheres, as well as basic geometric and boolean operations. Serving as the primary link with OpenCasCade, the Part Workbench forms the cornerstone of FreeCAD's geometry system, with nearly all other workbenches generating Part-based geometry. This parametric modeling system enables precise control and modification of 3D models through a history-based workflow. Users can build and refine complex designs by stacking and adjusting simpler shapes and operations, ensuring a robust and flexible design process.

Tool	Description	Tool	Description
Box	Creates a box	Cylinder	Creates a cylinder
Sphere	Creates a sphere	Cone	Creates a cone

Torus	Creates a torus (ring)	Tube	Creates a tube
Create primitives	Creates various other parametric geometric primitives	Shape builder	Creates more complex shapes from primitives
Extrude	Extrudes planar faces of an object	Revolve	Creates a solid by revolving another object (not solid) around an axis
Mirror	Mirrors the selected object on a given mirror plane	Scale	Scales a selected shape
Fillet	Fillets (rounds) edges of an object	Chamfer	Chamfers edges of an object
MakeFace	Creates a face from a sketch	Ruled Surface	Create a ruled surface between selected curves
Loft	Lofts from one profile to another	Sweep	Sweeps one or more profiles along a path
Section	Creates a section by intersecting an object with a section plane	CrossSections	Creates multiple cross sections along an object
Offset	Creates a scaled copy of the original object	Thickness	Assigns a thickness to the faces of a shape
Project on Face	Projects a shape on a face	Color Face	Defines a color for each face/surface of an object
Make Compound	Makes a compound of several shapes	Explode Compound	Splits up a compound of shapes
Compound Filter	Filters compounded objects based on a parameter (e.g volume, area, other object)	Boolean	Runs a boolean operation on two selected objects
Cut	Cuts (subtracts) one object from another	Fuse	Fuses (unites) selected Part objects into one
Common	Extracts the common (intersection) part of two objects	JoinConnect	Connects interiors of walled objects
JoinEmbed	Embeds a walled object into another walled object	JoinCutout	Creates a cutout in a wall of an object for another walled object
Boolean Fragment	Computes all fragments that can result from applying Boolean operations between input shapes	Slice apart	Splits shapes by intersection with other shapes

Slice to compound	Splits shapes by intersection with other shapes	Boolean XOR	Removes geometry shared by an even number of objects
Check Geometry	Checks geometry for errors	Defeaturing	Removes features from a shape

Part Design[edit | edit source]

The PartDesign Workbench is an essential tool for creating and modifying solid 3D models. It allows users to design complex parts by sketching 2D profiles and then applying various operations such as extrusions, lofts, and revolutions to generate 3D geometry. This workbench also supports the creation of features like pockets, holes, fillets, and chamfers, providing a comprehensive set of tools for detailed part design. Additionally, the Part Design Workbench integrates seamlessly with the Sketcher Workbench, enabling users to define and constrain sketches that serve as the foundation for 3D models. This integration facilitates a parametric design approach, allowing for easy adjustments and updates to the model throughout the design process.

Tool	Description	Tool	Description
Pad	Extrudes a solid object from a selected sketch	Revolution	Creates a solid by revolving a sketch around an axis
Additive loft	Creates a solid by making a transition between two or more sketches.	Additive pipe	Creates a solid by sweeping one or more sketches along an open or closed path.
Additive helix	Creates a solid by sweeping a sketch along a helix.	Additive box	Creates an additive box
Additive cylinder	Creates an additive cylinder	Additive sphere	Creates an additive sphere
Additive cone	Creates an additive cone	Additive ellipsoid	Creates an additive ellipsoid
Additive torus	Creates an additive torus	Additive prism	Creates an additive prism
Additive wedge	Creates an additive wedge	Pocket	Creates a pocket from a selected sketch
Hole	Creates a hole feature from a selected circle sketch	Groove	Creates a groove by revolving a sketch around an axis
Subtractive loft	Creates a solid shape by making a transition between two or more sketches and subtracts it from the active body	Subtractive pipe	Creates a solid shape by sweeping one or more sketches along an open or closed path and subtracts it from the active body

Subtractive helix	Creates a solid shape by sweeping a sketch along a helix and subtracts it from the active body	Subtractive box	Adds a subtractive box to the active body
Subtractive cylinder	Adds a subtractive cylinder to the active body	Subtractive sphere	Adds a subtractive sphere to the active body
Subtractive cone	Adds a subtractive cone to the active body	Subtractive ellipsoid	Adds a subtractive ellipsoid to the active body
Subtractive torus	Adds a subtractive torus to the active body	Subtractive prism	Adds a subtractive prism to the active body
Subtractive wedge	Adds a subtractive wedge to the active body	Boolean	Performs boolean operations on the selected objects
Fillet	Fillets (rounds) the edges of the active body	Chamfer	Chamfers the edges of the active body
Draft	Applies angular draft to faces of an object	Thickness	Creates a thick shell from the active body and opens the selected face
Mirrored	Mirrors features on a plane or face	Linear pattern	Creates a linear pattern of features
Polar pattern	Creates a polar pattern of features	Multitransform	Allows creating a pattern with any combination of the other transformations

Draft[edit | edit source]

The Draft Workbench is designed for creating and editing 2D drawings. It offers a range of tools for drafting, including lines, arcs, circles, and text. Users can also perform operations like trimming, extending, and offsetting. The Draft Workbench is particularly useful for architectural drawings and can be used to create complex 2D geometries that can be later transformed into 3D models. It integrates seamlessly with other workbenches, providing a versatile foundation for both 2D and 3D design projects.

Tool	Description	Tool	Description
Line	Draws a line segment between 2 points	Polyline	Draws a line made of multiple line segments (polyline)
Fillet	Creates a fillet (rounded corner) or a chamfer	Arc	Draws an arc segment from the center, radius,

	(straight edge) between two lines		start angle and end angle
Arc by 3 points	Creates a circular arc from three points that define its circumference.	Circle	Draws a circle from the center and radius
Ellipse	Draws an ellipse from two corner points	Rectangle	Draws a rectangle from 2 opposite points
Polygon	Draws a regular polygon from a center and a radius	BSpline	Draws a B-Spline from a series of points
Bézier Curve	Draws a Bézier curve from a series of points	Cubic Bézier curve	Creates a Bézier curve comprised of cubic segments
Point	Inserts a single point	Facebinder	Creates a new object from selected faces on existing objects
Shapestring	Inserts a compound shape representing a text string at a given point in the current document	Hatch	Creates hatches on the planar faces of a selected object
Text	Creates a multi-line annotation	Dimension	Creates a dimension
Label	Creates a label	Annotation styles	Allows you to define styles that affect the visual properties of annotation-like objects
Move	Moves or copies objects from one location to another	Rotate	Rotates objects by a certain angle around a point
Scale	Scales objects in relation to a point	Mirror	Mirrors objects across a line
Offset	Offsets an object to a certain distance	Trimex	Trims, extends or extrudes an object
Stretch	Stretches objects by moving selected points.	Clone	Creates linked copies of objects
Array	Creates an orthogonal array from a selected object	Polar array	Creates an array from a selected object by placing copies along a circumference
Circular array	Creates an array from a selected object by placing copies along concentric circumferences	Path array	Creates an array from a selected object by placing copies along a path.

Path link array	Works similar to a path array but creates a Link array instead of a regular array.	Point array	Creates an array from a selected object by placing copies at the points from a point compound.
Point link array	Works similar to a path array but create a Link array instead of a regular array.	Path twisted array	Creates twisted copies along a path
Path linked twisted array	Creates twisted linked copies along a path	Edit	Edits the active model
Subelement highlight	Temporarily highlights selected objects or the base objects of selected objects	Join	Joins Draft Lines and Draft Wires into a single object
Split	Splits a Draft Line or Draft Wire at a specified point or edge.	Upgrade	Turns or joins objects into a higher-level object
Downgrade	Turns or separates objects into lower-level objects	Shape 2D View	Creates a 2D object which is a flattened view of another object
Wire to B-spline	Converts Draft Wires to Draft BSplines and vice versa.	Draft to Sketch	Converts a Draft object to a Sketch and vice-versa
Set slope	Slopes selected Draft Lines or Draft Wires	Flip dimension	Rotates the dimension text of selected Draft Dimensions 180° around the dimension line
Shape 2D view	Creates 2D projections from selected objects	Snap lock	Enables or disables snapping globally
Snap endpoint	Snaps to the endpoints of edges.	Snap midpoint	Snaps to the midpoint of edges
Snap center	Snaps to the center point of faces and circular edges	Snap angle	Snaps to the special cardinal points on circular edges, at multiples of 30° and 45°
Snap intersection	Snaps to the intersection of two edges	Snap perpendicular	Snaps to the perpendicular points on faces and edges.
Snap extension	Snaps to an imaginary line that extends beyond the endpoints of straight edges	Snap parallel	Snaps to an imaginary line parallel to straight edges
		Snap near	

Snap special	Snaps to special points defined by the object.		Snaps to the nearest point on faces and edges
Snap ortho	Snaps to imaginary lines that cross the previous point at multiples of 45°.	Snap grid	Snaps to the intersections of grid lines.
Snap working plane	Projects snap points onto the current working plane	Snap dimensions	Shows temporary X and Y dimensions

Sketcher[edit | edit source]

The Sketcher Workbench contains tools to build and edit complex 2D objects, called sketches. The geometry inside these sketches can be precisely positioned and related by the use of constraints. They are primarily meant to be the building blocks of PartDesign geometry but are useful everywhere in FreeCAD.

Tool	Description	Tool	Description
Point	Draws a point	Polyline	Draws a line made of multiple line segments
Line	Draws a line segment from 2 points	Arc	Draws an arc segment from the center, radius, start angle and end angle
Arc 3 points	Draws an arc segment from two endpoints and another point on the circumference	Arc of ellipse	Creates an arc of ellipse
Arc of hyperbola	Creates an arc of hyperbola	Arc of parabola	Creates an arc of parabola
Circle	Draws a circle from the center and radius	Circle 3 points	Draws a circle from three points on the circumference
Ellipse	Draws an ellipse by a center point, major radius point and minor radius point	Ellipse 3 points	Draws an ellipse by major diameter (2 points) and minor radius point
Rectangle	Draws a rectangle from 2 opposite points	Centered rectangle	Creates a centered rectangle
Rounded rectangle	Creates a rounded rectangle	Triangle	Draws a regular triangle inscribed in a construction geometry circle
Square	Draws a regular square inscribed in a construction geometry circle	Pentagon	Draws a regular pentagon inscribed in a construction geometry circle

Hexagon	Draws a regular hexagon inscribed in a construction geometry circle	Heptagon	Draws a regular heptagon inscribed in a construction geometry circle
Octagon	Draws a regular octagon inscribed in a construction geometry circle	Regular polygon	Creates a regular polygon. The number of sides can be specified
Slot	Creates a slot	Arc slot	Creates an arc slot
B-spline by control points	Creates a B-spline curve by control points	Periodic B-spline by control points	Creates a periodic (closed) B-spline curve by control points.
B-spline by knots	Creates a B-spline curve by knot points	Periodic B-spline by knots	Creates a periodic (closed) B-spline curve by knot points
Construction mode	Toggles an element to/from construction mode	Dimension	Based on the current selection, it offers appropriate dimensional/geometric constraints
Horizontal distance	Fixes the horizontal distance between two points or the endpoints of a line	Vertical distance	Fixes the vertical distance between two points or the endpoints of a line
Distance	Fixes the length of a line, the distance between two points, the perpendicular distance from a point to a line, the distance between edges of two circles/arcs or between a circle/arc and a line, or the length of an arc	Auto radius/ diameter	Fixes the radius of arcs and B-spline weight circles, and the diameter of circles
Radius	Fixes the radius of circles, arcs and B-spline weight circles	Diameter	Fixes the diameter of circles and arcs.
Angle	Fixes the angle between two edges, the angle of a line with the horizontal axis of the sketch, or the aperture angle of a circular arc	Lock	Constrains the selected item by setting vertical and horizontal distances relative to the origin, thereby locking the location of that item
Coincident (unified)	Creates a coincident constraint between points, fixes points on edges or	Horizontal/ vertical	Constrains lines or pairs of points to be horizontal or vertical,

	axes or creates a concentric constraint		whichever is closest to the current alignment
Horizontal	Constrains the selected lines or polyline elements to a true horizontal orientation. More than one object can be selected before applying this constraint	Vertical	Constrains the selected lines or polyline elements to a true vertical orientation. More than one object can be selected before applying this constraint.
Parallel	Constrains lines to be parallel.	Perpendicular	Constrains two lines to be perpendicular
Tangent or collinear	Creates a tangent constraint between two selected entities or a co-linear constraint between two line segments	Equal length	Constrains two selected entities equal to one another. If used on circles or arcs their radii will be set equal
Symmetric	Constrains two points to be symmetrical around a line or axis, or around a third point	Block	Blocks edges in place with a single constraint. It is mainly intended for B-splines
Fillet	Creates a fillet between two non-parallel edges	Chamfer	Creates a chamfer between two non-parallel edges
Trim	Trims an edge at the nearest intersections with other edges	Split	Splits an edge while transferring most constraints
Extend	Extends or shortens a line or an arc to an arbitrary location, or to a target edge or point	External geometry	Projects edges and/or vertices belonging to objects outside the sketch onto the sketch plane
Projection	Projects edges and/or vertices belonging to objects outside the sketch onto the sketch plane.	Intersection	Intersects faces and/or edges belonging to objects outside the sketch with the sketch plane.
Carbon copy	Copies all geometry and constraints from another sketch into the active sketch	Select origin	Selects the origin of the sketch
Select horizontal axis	Selects the horizontal axis of the sketch	Select vertical axis	Selects the vertical axis of the sketch
Array transform	Moves or optionally creates copies of selected elements	Polar transform	

			Rotates or optionally creates rotated copies of selected elements
Scale transform	Scales or optionally creates scaled copies of selected elements	Offset geometry	Creates equidistant edges around selected edges
Symmetry	Creates mirrored copies of selected elements	Remove axes alignment	Removes the axes alignment of selected edges by replacing Horizontal and Vertical constraints with Parallel and Perpendicular constraints

BIM[edit | edit source]

The BIM Workbench (a combination of the former BIM, Native-IFC and Arch Workbenches) workbench is tailored for architectural design and construction planning, offering tools to create and manipulate parametric building elements like walls, slabs, columns, beams, roofs, and windows. It supports IFC (Industry Foundation Classes) for seamless data exchange with other BIM software, facilitating collaboration. Users can generate detailed 2D drawings, annotations, and construction documentation from 3D models, integrating closely with the Draft workbench for tasks like creating floor plans, sections, and elevations.

Tool	Description	Tool	Description
Site	Creates a site including selected objects	Building	Creates a building including selected objects
Space	Creates a space object in the document	Level	Creates a floor including selected objects
Wall	Creates a wall from scratch or using a selected object as a base	Curtain Wall	Creates a curtain wall from scratch or using a selected object as a base
Column	Creates a column at a specified location	Beam	Creates a beam between two points
Slab	Creates a slab from a planar shape	Door	Places a door at a given location
Window	Creates a window using a selected object as a base	Pipe	Creates a pipe.
Connector	Creates a corner or T-connection between 2 or 3 selected pipes	Stairs	Creates a stairs object in the document
Roof	Creates a sloped roof from a selected face	Panel	Creates a panel object from a selected 2D object

Frame	Creates a frame object from a selected layout	Fence	Creates a fence from the selected section, post and path
Truss	Creates a truss from a selected line or from scratch	Equipment	Creates an equipment or furniture object
Rebar	Creates a reinforcement bar based on a sketch or a selected face	Profile	Creates a parametric 2D profile
Box	Creates a generic box	Library	Opens the object library
Component	Adds un undefined architectural component	Reference	Creates an external reference object
Aligned Dimension	Creates an aligned dimension	Horizontal Dimension	Creates a horizontal dimension
Vertical Dimension	Creates a vertical dimension	Leader	Creates a polyline with an arrow at it's endpoint
Axis	Adds a 1-direction array of axes	Axis System	Adds an axis system composed of several axes
Grid	Adds a grid-like object	Section Plane	Adds a section plane object
Hatch	Creates hatched at the selected face	View	Creates a tech draw view from a section plane or 2D object
Material	Sets or creates a material for the selected object	Schedule	Creates different types of schedules to collect data from the model

Other built-in workbenches[edit | edit source]

Although the above summarizes the most important tools of FreeCAD, many more workbenches are available, among them:

- The TechDraw Workbench for producing technical drawings from 3D models.
- The Mesh Workbench allows to work with polygon meshes. Although meshes are not the preferred type of geometry to work with in FreeCAD, because of their lack of precision and support for curves, meshes still have a lot of uses and are fully supported in FreeCAD. The Mesh Workbench also offers a number of Part-to-Mesh and Mesh-to-Part tools.
- The Spreadsheet Workbench permits the creation and manipulation of spreadsheet data, that can be extracted from FreeCAD models. Spreadsheet cells can also be referenced in many areas of FreeCAD, making it possible to use them as master data structures.
- The FEM Workbench deals with Finite Elements Analysis and makes it possible to perform pre- and post-processing of FEM analyses and to display the results graphically.

External workbenches[edit | edit source]

A number of other very useful workbenches produced by FreeCAD community members also exist. Although they are not included in a standard FreeCAD installation, they are easy to install as plug-ins. They are all referenced in the FreeCAD-addons repository. Among the most developed are:

- The Drawing Dimensioning Workbench offers many new tools to work directly on Drawing Sheets and allow you to add dimensions, annotations and other technical symbols with great control over their aspect. **The Drawing Dimensioning Workbench is no longer maintained.**
- The Fasteners Workbench offers a wide range of ready-to-insert fasteners objects like screws, bolts, rods, washers and nuts. Many options and settings are available.
- The A2plus workbench offers a series of tools to mount and work with assemblies.

Read more

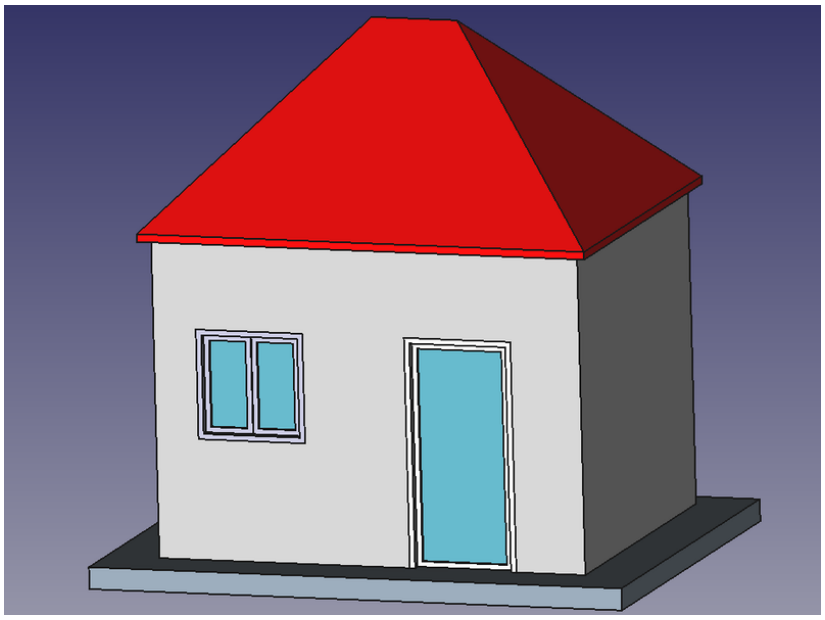
- The complete list of workbenches
- The Part Workbench
- The Draft Workbench
- The Sketcher and Part Design Workbench
- The BIM Workbench
- The TechDraw Workbench
- The FEM Workbench
- The FreeCAD-addons repository

Building Information Modeling is a process used in architecture, engineering, and construction to create and manage digital representations of physical structures. It integrates not only 3D geometry but also critical data such as materials, costs, and schedules, allowing for advanced analysis and collaboration throughout the entire lifecycle of a project.

In FreeCAD, BIM functionality has evolved significantly, especially with the release of version 1.0, where the previously separate Arch and BIM workbenches were merged into an integrated BIM Workbench. This consolidation streamlines workflows, allowing users to model, document, and manage building projects more efficiently within a single environment.

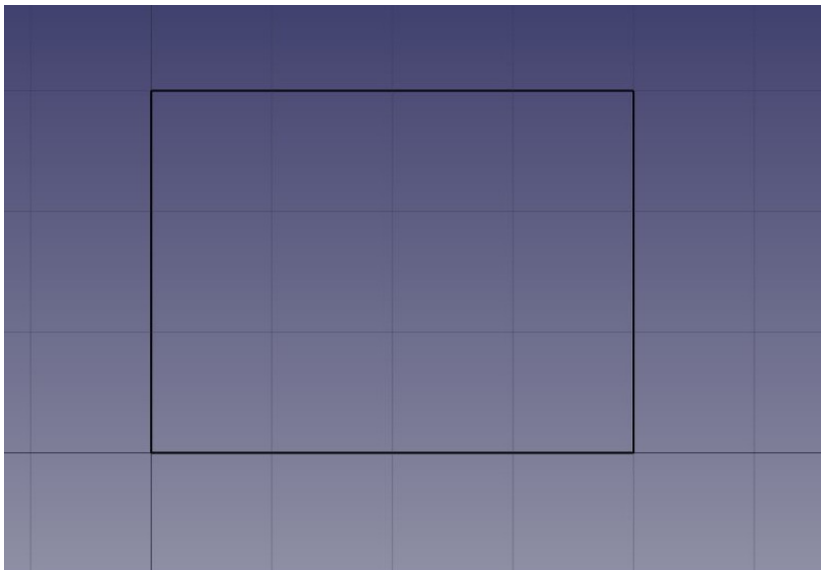
A major advancement introduced in FreeCAD v1.0 is the adoption of the Native IFC concept. Previously, like most BIM applications, FreeCAD translated data back and forth between its internal data model and the IFC (Industry Foundation Classes) file format, leading to potential data loss during the opening and saving processes. With Native IFC, FreeCAD users can now open, manipulate, and save IFC files directly, where the IFC file itself serves as the data structure. This approach eliminates unnecessary data translation and ensures that modifications are saved without rewriting the entire file, making it compatible with version control systems like Git and providing a more transparent, precise workflow for handling IFC files.

In this chapter, we will see how to model this small building:



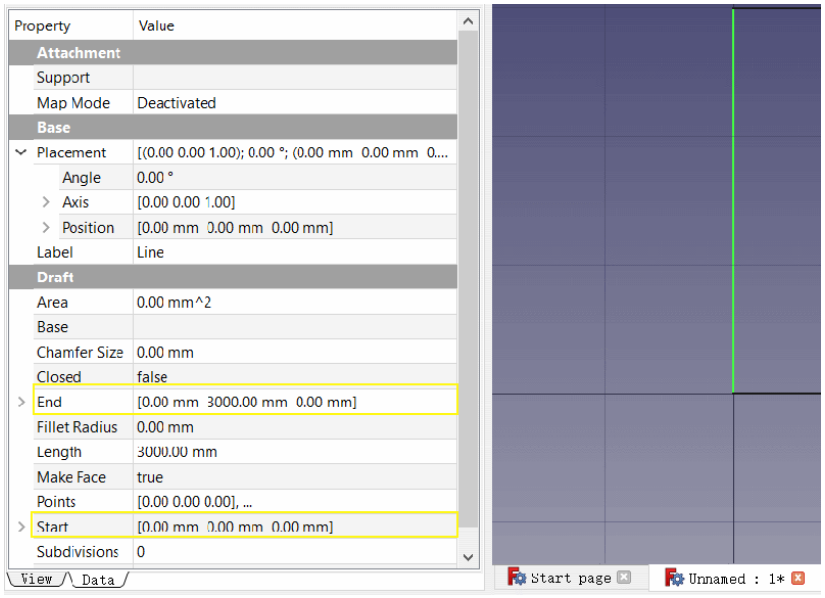
- Create a new document, and switch to the BIM Workbench.
- Open menu **Edit** → **Preferences** → **Draft** → **Grid and Snapping** and set:
  - **Main lines every** `10`.
  - **Grid spacing** `1000mm` to have a one meter-based grid, which is convenient for the size of our building.
  - **Grid size** `100 lines`.
- On the **snapping toolbar** make sure the Snap grid button is enabled, so we can use the grid as much as possible.
- If you do not see the axes then click the Toggle grid button.
- Set the Working Plane to **XY** plane
- Draw four lines with the Draft Line tool. You can enter coordinates manually, or simply pick the points on the grid with the mouse. We will be using meters for our dimensions:
  - From point (0,0) to point (0,3)

- From point (0,3) to point (4,3)
- From point (4,3) to point (4,0)
- From point (4,0) to point (0,0)



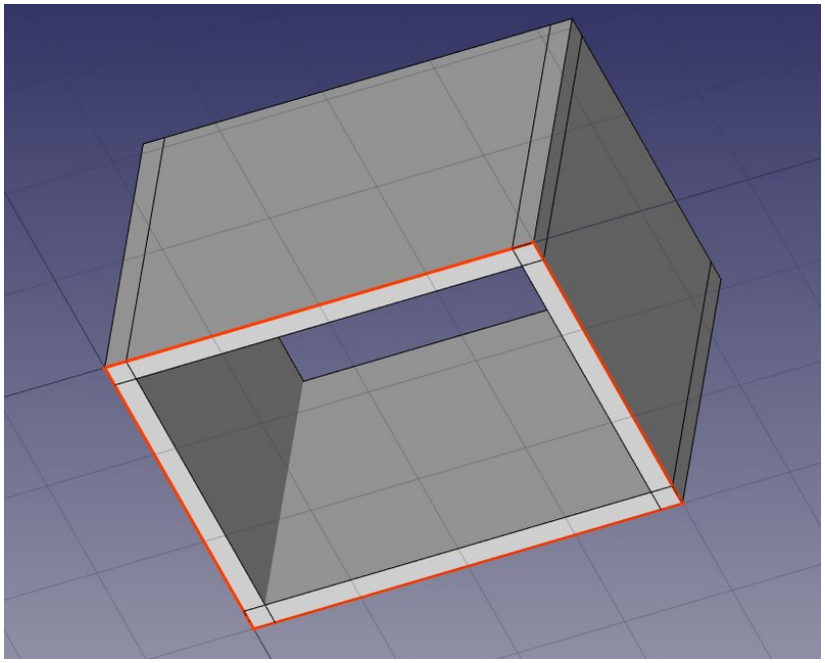
Notice that we consistently drew the lines in the same direction (clockwise). While this isn't required, it helps ensure that the walls we will build next have consistent left and right orientations. You might wonder why we didn't just draw a rectangle, which would have been simpler. However, using four separate lines gives us the opportunity to showcase additional BIM functionalities, such as how to combine multiple objects into one, which is an essential part of the workflow.

- Once you have created the lines check their start and end points and adjust if necessary to get them exactly correct.



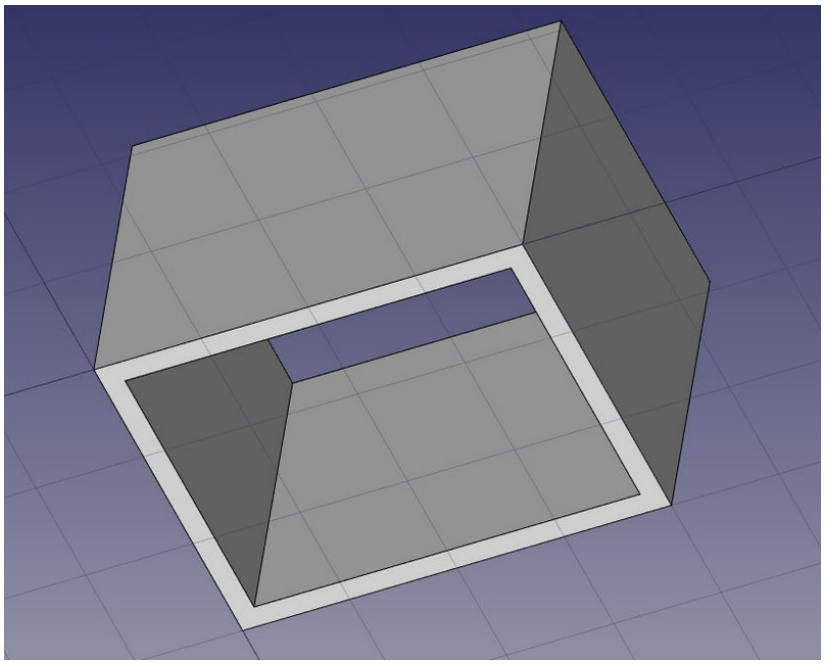
- Select all four lines, then press the Wall button.
- Set the walls' **Height** to 3m (default).
- Set **Alignment** property to **left**. Setting the Alignment property to the left ensures that the walls you create will be positioned to the left side of the lines you drew. In FreeCAD's BIM Workbench, walls are typically generated based on a reference line, and the left or right alignment dictates which side of the line the wall will be placed.

If you didn’t draw the lines in the same order as instructed (clockwise), the orientation of some walls may be flipped, meaning they could be positioned on the opposite side of the line (to the right instead of the left). In that case, you would need to adjust the alignment to the right for those specific walls to ensure they all align consistently. Once this is set correctly, you'll have four walls that intersect at the corners, positioned on the inside of the baseline, forming the desired layout.



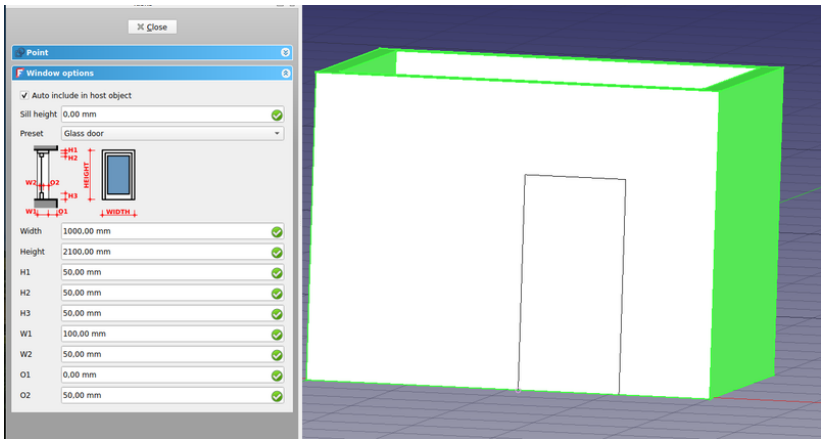
After creating walls, the next step is to join them so they intersect properly. This is necessary when walls don't connect cleanly at their intersections. To do this, you select one wall as the "host" and add the other walls as "additions", merging their geometry with the host. All objects in the BIM Workbench can have multiple additions (which add geometry) or subtractions (which remove geometry). These relationships can be managed anytime by double-clicking the object in the tree, allowing for flexible adjustments to ensure the walls and other architectural elements integrate smoothly.

- Select the four walls with Ctrl pressed, the last one being the wall that you chose to become the host
- Press the Add button. The four walls have now been turned into one:

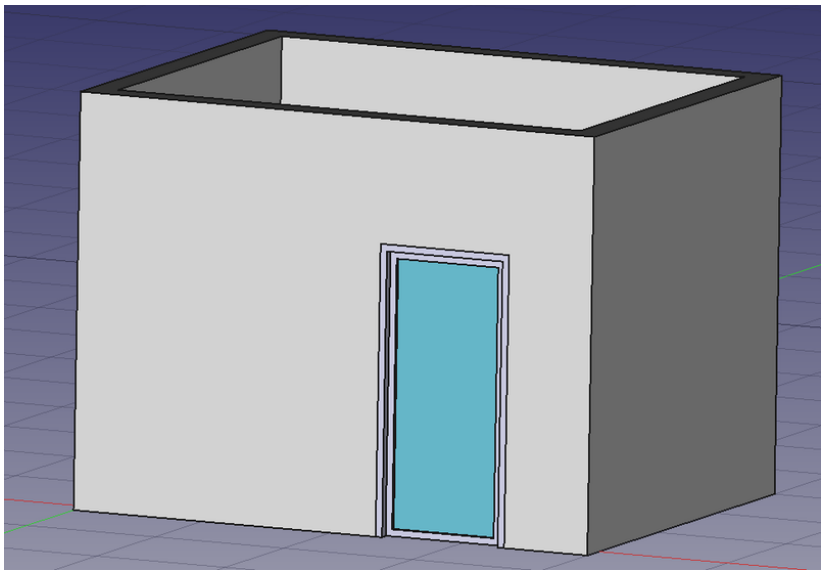


The individual walls are however still accessible, by expanding the wall in the tree view.

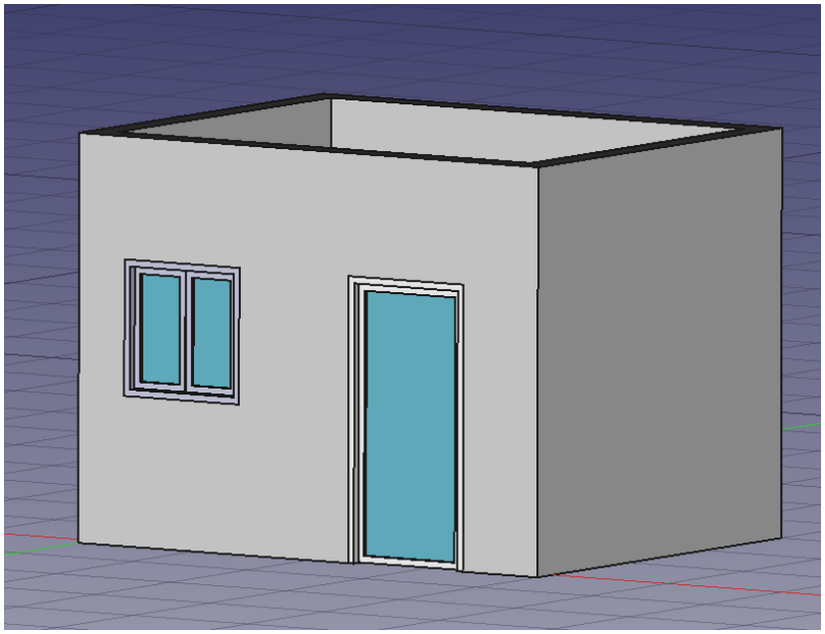
- Let's now place a door by pressing on the Door tool.
- Begin by selecting the wall. While this step isn't required, it's a useful habit to develop. If an object is selected before starting an operation, the operation will automatically apply to that entity by default.
- Set the Working Plane to **auto** so we are not restricted to the ground plane
- Press the Door button.
- In the door creation panel, select the **Glass door** preset, and set its **Width** to 1 m and its **Height** to 2.1m. You will notice that you can choose between various door types and set up their parameters as you wish. In FreeCAD a door is derived by an window operation.
- Make sure the Snap near option is turned on, so we can snap on faces
- Place your door roughly on the middle of the front face of the wall:



- We can now set the precise location by expanding the wall and the window objects in the tree view and changing the **Placement** property of the base sketch of our door. Set its position to **x = 0.5 m, y = 0, z = 0**. Our door is now exactly where we want it:

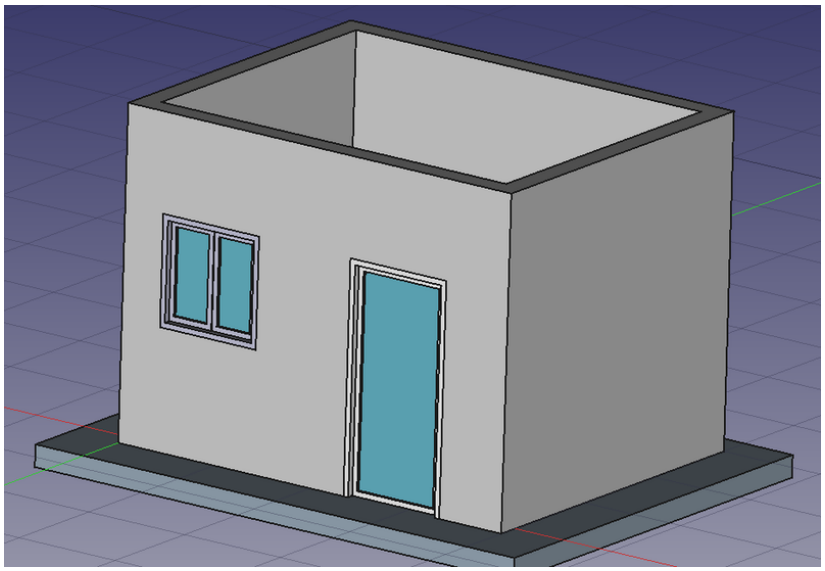


- Let's place a window next to our door. Select the wall, press the Window tool, select the **Open 2-pane** preset, and place a **1m x 1m** window in the same face as the door. Set the placement of the underlying sketch to position **x = 0, y = 0, z = 1.1m**, so the upper line of the window is aligned to the top of the door.



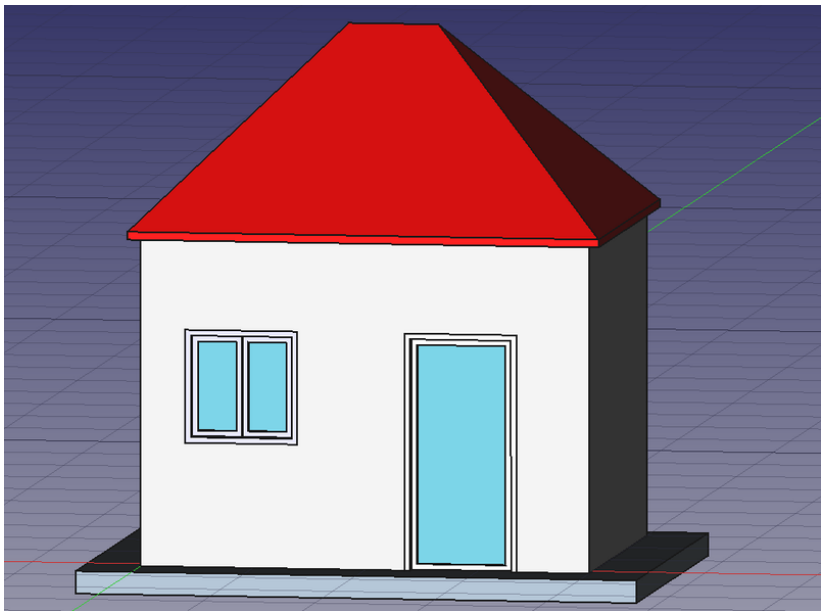
Windows are always based on sketches. You can easily create custom windows by first drawing a sketch on a face, then turning that sketch into a window by selecting it and clicking the window button. Afterward, you can define the window's parameters—such as which parts of the sketch should be extruded and by how much—by double-clicking the window in the tree view. Now, let's move on to creating a slab:

- Set the Working Plane to **XY** plane
- Create a rectangle with a **length** of 5m, a height of **4m**, and place it at position x:-0.5m, y:-0.7m, z:0.
- Select the rectangle
- Click the Slab tool to create a slab from the rectangle
- Keep the default values of 0.2m for the **height** property and set the normal **direction** to (0,0,-1), so the extrusion goes downward. While we could have moved the object 0.2m downward instead, it's a good practice to keep extruded objects aligned with their base profile to maintain consistency and accuracy in the model.
- Set the **Ifc Type** property of the slab to **slab**. This is not necessary in FreeCAD, but is important for IFC export, as it will ensure that the object is exported with the correct IFC type.



- Let's put a roof over our heads now. We can easily do it by using the Roof tool.
- Press the Snap working plane option to enable the drafting on all planes.

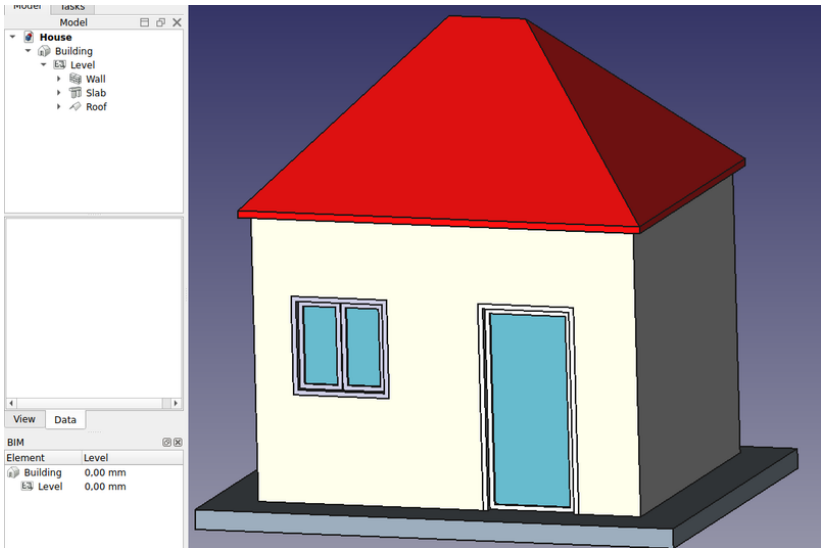
- By choosing one of the top faces of our house press the Select plane button. The working plane is now set to that face.
- Create a rectangle, snapping to two opposite points of the walls:
- On the **data** tab of the roof set the **Runs** parameter to 1600.
- If you wish to change the color of the roof you can do so on the view tab



With that, our model is now complete. The next step is to organize it properly to ensure it exports correctly to the IFC format. IFC files require all building elements to be grouped within a **building** object, and optionally, within a specific **story**. Additionally, all buildings must be located on a **site**. However, FreeCAD's IFC exporter will automatically generate a default site if one isn't present, so we don't need to add it manually. It's important to properly structure the model to comply with IFC standards, ensuring smooth collaboration and compatibility with other BIM software. Proper organization will also help avoid any data loss or errors during the export process.

- Select the walls, the slab, and the roof.
- Press the Floor button
- Select the floor we just created
- Press the Building button

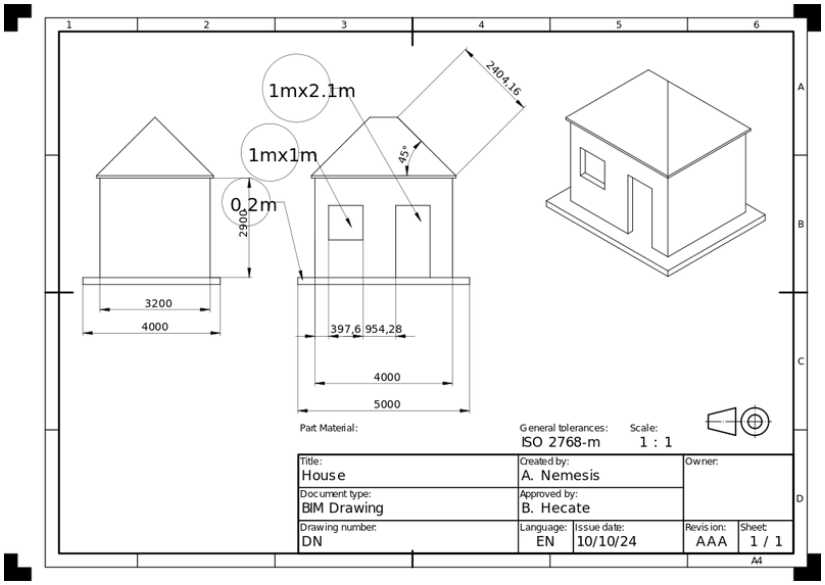
Our model is now ready to export:



The IFC format is one of the most precious assets in a free BIM world, because it allows the exchange of data between any application and actor of the construction world, in an open manner (the format is open, free and maintained by an independent consortium). Exporting your BIM models as IFC ensures that anyone can see and analyze them, no matter the application used.

- Select the top object you want to export, that is, the Building object.
- Select menu **File -> Export -> Industry Foundation Classes** and save your file.
- The resulting IFC file can now be opened in a wide range of applications and viewers (the image below shows the file opened in the free IfcPlusPlus viewer). Checking the exported file in such a viewer application before distributing it to other people is important to check that all the data contained in the file is correct. FreeCAD itself can also be used to re-open the resulting IFC file.

We can use the TechDraw Workbench to create a drawing of our building. The process is similar to what was shown in the previous section, so we won't go into too much detail here. Simply create a new view by using the insert Default Page option, then select the view you want to display in the drawing and add dimensions where necessary. This will allow us to create a professional 2D representation of the 3D model for documentation or presentation purposes.



Our page is now ready, and we can export it to SVG or DXF formats, or print it. The SVG format allows you to open the file using illustration applications such as Inkscape, with which you can quickly enhance technical drawings and turn them into much nicer presentation drawings. It offers many more possibilities than the DXF format.

Downloads[edit | edit source]

- The file produced during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/house.FCStd
- The IFC file exported from the above file: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/house.ifc
- The SVG file exported from the above file: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/house.svg

Related[edit | edit source]

- BIM Workbench
- The Arch Workbench
- The Draft working plane
- The Draft snapping settings
- The expressions system
- The IFC format
- IfcOpenShell
- IfcPlusPlus
- Inkscape

Creating FEM analyses

Finite Element Method (FEM) is a powerful computational technique used to solve complex problems in engineering, physics, and applied mathematics. It works by breaking down a large, complex object or structure into smaller, simpler parts called finite elements. These elements are analyzed individually, and their behavior is combined to predict how the entire structure will respond to external influences, such as forces, heat, or vibrations.

FEM is widely used in fields like structural engineering, mechanical design, aerodynamics, and electromagnetism to simulate how objects deform under stress, how heat flows through materials, and how electromagnetic fields interact with different objects. By providing detailed insights into these interactions, FEM allows engineers and designers to optimize their products for performance, safety, and efficiency without needing physical prototypes.

Obtaining such simulations in FreeCAD is done using the FEM Workbench, which is specifically designed for performing finite element analysis (FEA). It provides a comprehensive set of tools for preparing the model, assigning material properties, meshing, and running simulations. The FEM Workbench is versatile, supporting a wide range of simulations such as structural, thermal, and dynamic analyses, with solvers like CalculiX and others available.

This workbench allows for the integration of other FreeCAD workbenches, enabling seamless model preparation and analysis. It also provides powerful post-processing tools to visualize and interpret simulation results, such as stress, deformation, and thermal distributions. The workflow follows these steps:

- **Preparing the Geometry**: The model must be simplified or optimized for FEM analysis. This often includes removing unnecessary details or features that don't contribute to the simulation but could make it computationally expensive. You can use tools from other workbenches, like PartDesign or Part, to prepare your 3D geometry. The FEM Geometry Preparation and Meshing page describes how to properly prepare the geometry for use in the FEM Workbench.

- **Assigning Material Properties**: Material definitions are critical for accurate simulations. Properties such as Young's modulus, Poisson's ratio, and density are assigned for structural simulations, or thermal conductivity and specific heat capacity for thermal analysis. Materials can be selected from FreeCAD's material library or customized as needed.

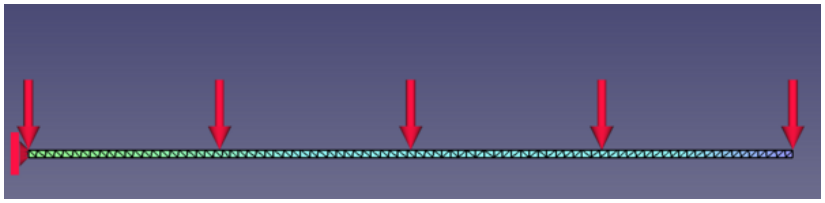
- **Meshing**: Meshing divides the geometry into finite elements, allowing the solver to analyze the object. Mesh quality is crucial, as finer meshes result in more accurate simulations but require more computational power. Tools are available to refine the mesh locally, focusing on areas where stress or deformation is expected to be higher. Check Meshing basics for more information.

- **Applying Loads and Constraints**: In this step, physical conditions such as forces, pressures, moments, or thermal loads are applied to the model. Boundary conditions are also defined, such as fixing points, applying symmetry constraints, or restricting movement, depending on the scenario being simulated.

- **Running the Solver**: Once the setup is complete, the solver calculates the model's response to the applied conditions. Solvers like CalculiX compute displacements, stresses, and other quantities, depending on the type of analysis performed. The

process can take varying amounts of time depending on the mesh density and model complexity.

- **Post-Processing**: After the simulation, results are visualized using tools in the FEM Workbench. Stress, strain, and displacement fields are represented as color maps and deformation plots can be generated. These visualizations allow for a thorough analysis of the model's performance, highlighting areas of high stress or deformation.



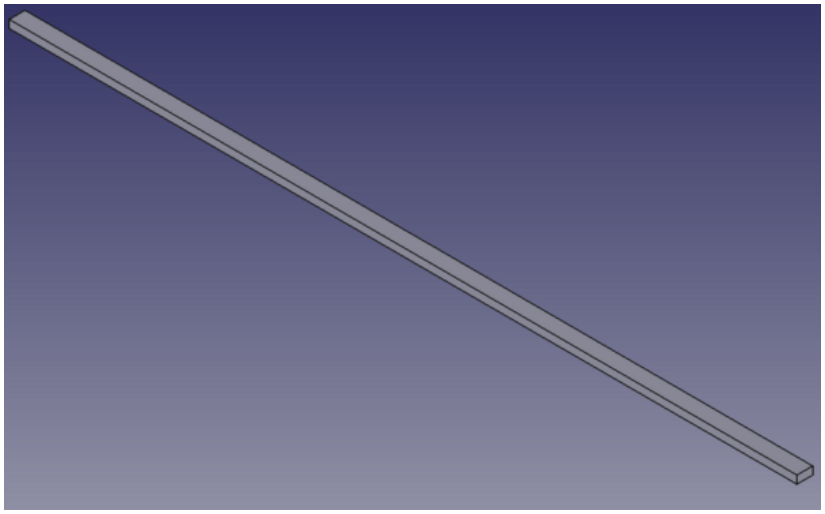
Preparing FreeCAD[edit | edit source]

In this section, we will demonstrate the general FEM analysis procedure through a simple example. While the topic of FEM is vast, we will focus on a straightforward geometry: a cantilever beam. Our objective is to determine the maximum vertical displacement of this beam under an applied load, and we will compare the numerical results with the analytical solution. In computational mechanics, verifying numerical results against experimental data or analytical solutions is essential to ensure the accuracy and reliability of the simulation. Additionally, we will be using packages that are already included in the FreeCAD installation, so no additional installations will be required for this analysis.

Preparing the geometry[edit | edit source]

First, we will create our simple geometry. For this, we will be using the PartDesign workbench.

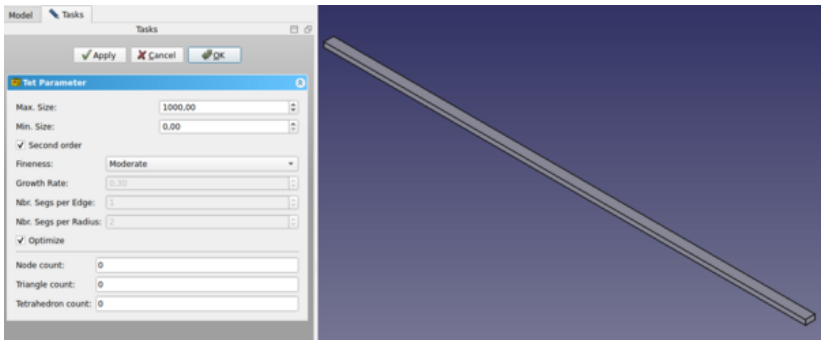
- Create a new document and go to the PartDesign workbench.
- Press on New Sketch to create a new sketch on the YZ plane.
- Create a centered rectangle around the origin point.
  - Using the Sketcher Dimension set the vertical dimension to **20 mm** and the horizontal to **10 mm**
- Exit the sketch mode.
- By having selected our newly created sketch apply a padding operation with a length of **1000 mm**.
- Our geometry is now ready. In this example, we've made the height (h) and width (b) of the beam much smaller than the length (L) to focus on how it bends. By doing this, we can make sure the beam behaves like a typical long, thin object where bending is the main effect when a force is applied. This setup also makes it easier to compare our results with simple formulas we can calculate by hand.



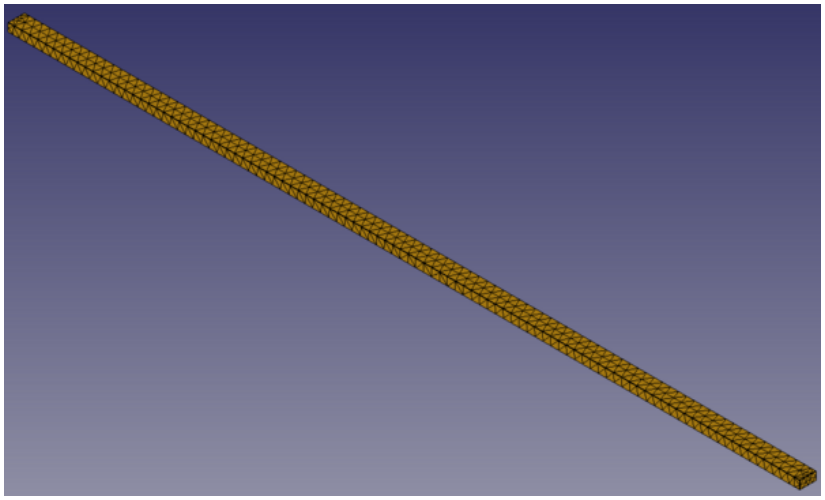
Creating the analysis[edit | edit source]

- We are now ready to start a FEM analysis. Let's switch to the FEM workbench
- Press the New Analysis
- A new analysis will be created and a settings panel will be opened. The **Create Analysis** button sets up the framework for running a finite element analysis. It creates an analysis container that organizes key elements such as the mesh, material properties, constraints (e.g., fixed points), applied loads, and the solver. This button essentially prepares everything needed for the simulation, allowing for further steps like meshing and running the solver to analyze how the object behaves under the defined conditions.

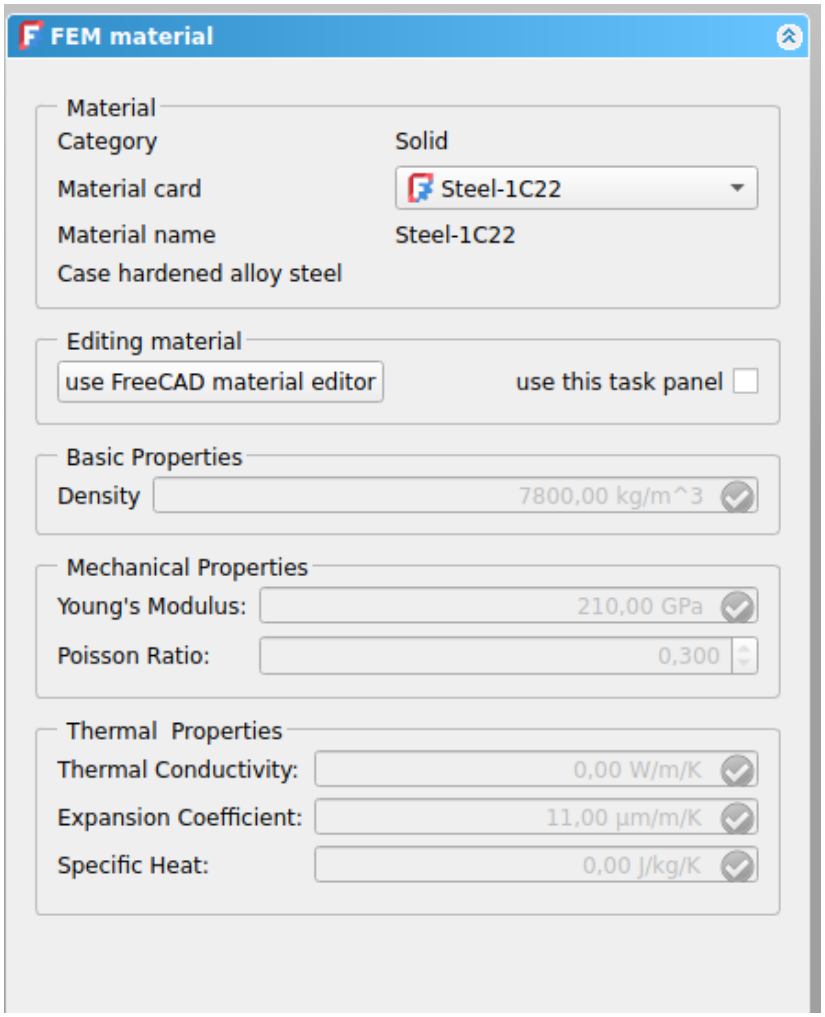
- We will start by creating the **Mesh**. For this reason, having selected our body, press the FEM mesh from shape by Netgen button. This option uses the Netgen mesher, an open-source tool used for creating high-quality tetrahedral meshes, particularly suited for complex geometries in finite element analysis.
- In the mesher parameters window we will keep things simple and only change the maximum cell size. The **Max Size** option defines the largest allowable size for the individual mesh elements. It controls how coarse or fine the mesh will be. A larger Max Size will result in a coarser mesh with fewer elements, which can speed up computations but may reduce accuracy. A smaller Max Size creates a finer mesh with more elements, increasing accuracy but also requiring more computational resources. Set this value to **10** and press **apply**.



- Our mesh is ready.



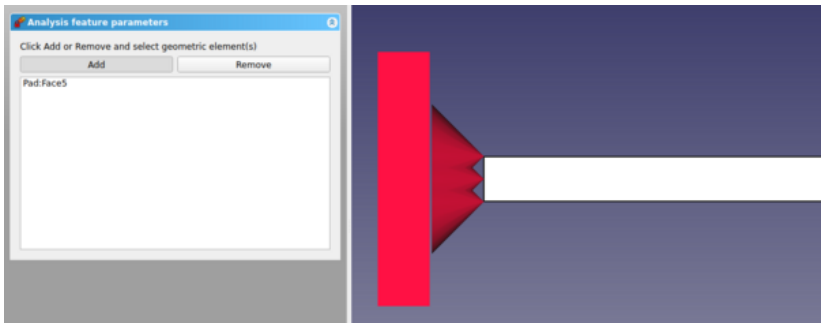
- We can now define the material to be applied to our mesh by pressing on the New Material option. The choice of material is crucial in any analysis, as different materials with varying properties will behave differently under the same conditions. Factors like strength, elasticity, and density play a significant role in how a material responds to forces, pressures, or temperatures. Selecting the appropriate material ensures accurate simulation results, reflecting how the object would react in real-world scenarios.
- A task panel will open to allow us to choose a material. In the Material drop-down list, choose the **Steel-1C22** material, and press the **OK**.



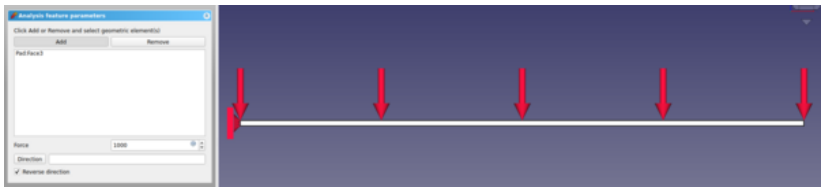
- The final step is to apply forces and constraints, translating the physical conditions into the FEM analysis. In this simple case, we have a beam that is fixed on one side

(representing attachment to a wall), while the other side is free to move. A distributed force is applied along the entire length of the beam, simulating the load it experiences in real-world conditions. Let's start by specifying the face fixed into the wall and can therefore not move. Press the Constraint fixed button.

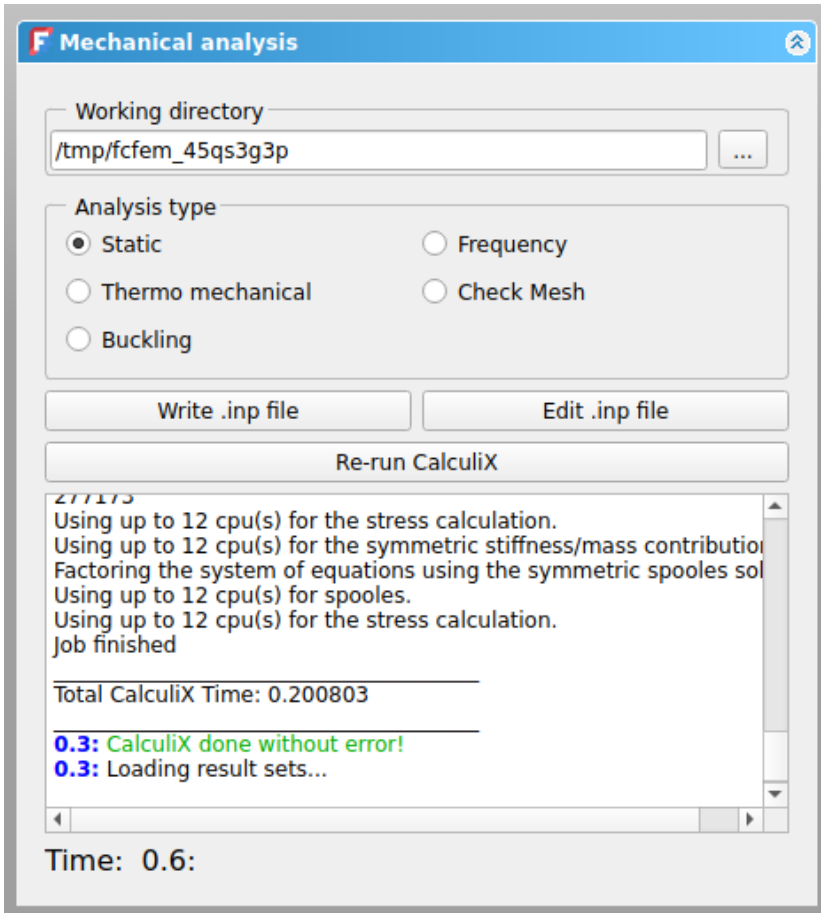
- Press on the **add** button and select the left face of our beam (the one at the origin). Click **Apply**. This face is now designated as unmovable:



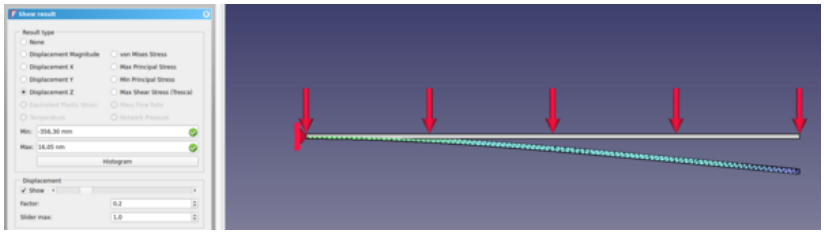
- We will now add a distributed load on the top face, that could represent, for example, a massive weight being placed on the beam. For this, we will use a press on the Force load option.
- Click the top face of the beam, set the force to **1000 N** and click on the **reverse direction** option. Then press the **OK**. Our force is now applied:



- We are now ready to start the calculation. Select the Calculix solver.
- Select **Static** analysis and press on the **Write .inp file**to create the input file for CalculiX. Then press on the **Run CalculiX**. The simulation will now run.



- We can now look at the results. Click on the show results option.
- Tick the **Displacement Z** option, which is the vertical coordinate for our case.
- You can see the minimum and maximum values for the vertical displacement. Based on the analysis the maximum vertical displacement is **-356.30 mm**. This matches our analytical solution of **-357.14 mm** well.
- You can move the slider next to it. You will be able to see the deformation growing as you apply more force:



The results displayed by the FEM workbench are of course currently not enough to perform real-life decisions about structures dimensioning and materials. However, they can already give precious information about how the forces flow through a structure, and which are the weak areas that will feel the most stress.

Read more

- The FEM Workbench
- Installing required FEM components
- CalculiX
- NetGen

Creating and manipulating geometry

In earlier chapters, we explored the various workbenches in FreeCAD and how each introduces its own set of tools and geometry types. The same principles apply when working with FreeCAD through Python scripting.

We also observed that most FreeCAD workbenches rely on one foundational workbench: the Part Workbench. Many other workbenches, such as the Draft Workbench, utilize the Part Workbench's tools and geometry, which is exactly what we will do in this chapter —use Python to create and manipulate Part geometry.

To begin working with Part geometry in Python, we need to perform the scripting equivalent of switching to the Part Workbench: importing the Part module.

```
import Part
```

Take a moment to explore the Part module by typing **Part.** in the Python console and browsing through the available methods and attributes in the autocomplete window. This is a great way to familiarize yourself with the functionality the module offers. You'll find a variety of convenient functions, such as makeBox and makeCircle, which allow you to quickly create geometric shapes and objects with just a single command. Many of these functions also offer optional parameters, giving you precise control over dimensions and placement.

Spending some time browsing the module's contents not only helps you understand what tools are at your disposal but also gives you insight into how the Part Workbench operates under the hood. This foundational knowledge will prove invaluable as we move forward and begin creating and manipulating geometry programmatically. Type the following command

```
Part.makeBox(3,5,7)
```

This command creates a 3D box, also known as a rectangular prism, with specific dimensions. The first parameter, 3, defines the length of the box along the X-axis. The second parameter, 5, sets the width along the Y-axis, and the third parameter, 7, specifies the height along the Z-axis. While this function generates the geometry of the box, it does not automatically add it to the active FreeCAD document. In the Python console you will see the following:

```
<Solid object at 0x5f43600>
```

The output **<Solid object at 0x5f43600>** indicates that a Part Shape has been created in memory. This is a geometric object stored at a specific memory address, as shown by the hexadecimal value (0x5f43600). However, it is important to understand that what we created here is not yet a FreeCAD document object—it exists only as a raw geometric shape in memory.

This distinction highlights a fundamental concept in FreeCAD: objects and their geometry are independent. A FreeCAD document object serves as a container that hosts a shape. These document objects can have additional properties, such as Length, Width, and Height, and they can be parametric. Parametric objects recalculate their geometry (or shape) dynamically whenever one of their properties changes. For instance, modifying

the length of a parametric box will automatically regenerate its shape with the updated value.

In this case, we manually created a shape using the **Part.makeBox()** function. This shape is a non-parametric object, meaning it won't update automatically based on any properties—it is static unless we manipulate it programmatically. To make this shape part of the active FreeCAD document, it would need to be assigned to a document object (like a **Part::Feature**), which would link it to the graphical interface and make it visible and manageable within the FreeCAD environment.

This separation between shapes and document objects is what makes FreeCAD highly versatile, allowing users to manipulate shapes programmatically and integrate them into a parametric modeling workflow as needed.

We can now easily create a "generic" document object in the current document (make sure you have at least one new document open), and give it a box shape like the one we just made:

```
boxShape = Part.makeBox(3,5,7)
myObj = FreeCAD.ActiveDocument.addObject("Part::Feature","MyNewBox")
myObj.Shape = boxShape
FreeCAD.ActiveDocument.recompute()
```

Here is a breakdown of the previous commands:

- **boxShape = Part.makeBox(3,5,7)**: Creates a 3D box with dimensions 3x5x7 (length, width, and height) and stores it as a Part Shape in the variable boxShape. This shape exists only in memory and is not yet part of the FreeCAD document.
- **myObj = FreeCAD.ActiveDocument.addObject("Part::Feature", "MyNewBox")**: Adds a new Part::Feature object named "MyNewBox" to the active FreeCAD document and assigns it to the variable myObj. The new object will appear in the FreeCAD document tree.
- **myObj.Shape = boxShape**: Links the boxShape geometry to the Shape property of myObj, integrating the geometry into the FreeCAD document.
- **FreeCAD.ActiveDocument.recompute()**: Updates the document to reflect the changes, ensuring that the new object and its geometry appear in the graphical interface.

Note how we handled **myObj.Shape**. It was done in the same way as in the previous chapter, where we changed other properties of an object, such as **box.Height = 5**. In fact, **Shape** is also a property, just like **Height**. However, instead of taking a number, **Shape** requires a Part Shape. In the next chapter, we will take a closer look at how these parametric objects are constructed.

For now, let's explore Part Shapes in more detail. In the chapter on traditional modeling with the Part Workbench, we introduced a table explaining how Part Shapes are constructed and the different components they consist of, such as **vertexes**, **edges**, and **faces**. These same components are available when working with Part Shapes in Python, allowing for detailed exploration and manipulation of geometry. Part Shapes in FreeCAD always have the following attributes:

- **Vertexes**: Points in 3D space that define the corners or endpoints of geometry.
- **Edges**: Straight or curved lines connecting two vertexes.
- **Wires**: Closed or open loops formed by one or more connected edges.
- **Faces**: Surfaces enclosed by one or more wires.

- **Shells**: Groups of connected faces, forming a continuous surface.
- **Solids**: 3D volumes enclosed by one or more shells.

All of these attributes are represented as lists in Python. Each list can contain any number of elements or be empty, depending on the shape being explored. For example, a box will have eight **Vertexes**, twelve **Edges**, six **Faces**, one **Shell**, and one **Solid**, while a line will only have two **Vertexes** and one **Edge**, with all other attributes being empty. These components are fundamental building blocks of Part geometry and can be accessed and manipulated programmatically. Understanding how they interact provides powerful control over the creation and modification of 3D models. We can access those lists as follows:

```
print(boxShape.Vertexes)
print(boxShape.Edges)
print(boxShape.Wires)
print(boxShape.Faces)
print(boxShape.Shells)
print(boxShape.Solids)
```

Let's find the area of each face of our box shape above: (Make sure to indent the second line, as it appears below. Press Enter twice after the last line to run the Python command.)

```
for f in boxShape.Faces:
   print(f.Area)
```

Or, for each edge, its start point and end point:

```
for e in boxShape.Edges:
   print("New edge")
   print("Start point:")
   print(e.Vertexes[0].Point)
   print("End point:")
   print(e.Vertexes[1].Point)
```

As you see, if our boxShape has a "Vertexes" attribute, each Edge of the boxShape also has a "Vertexes" attribute. As we can expect, the boxShape will have 8 vertices, while the edge will only have 2, which are both parts of the list of 8.

We can always check what is the type of a shape:

```
print(boxShape.ShapeType)
print(boxShape.Faces[0].ShapeType)
print(boxShape.Vertexes[2].ShapeType)
```

Here is a short explanation of the above commands:

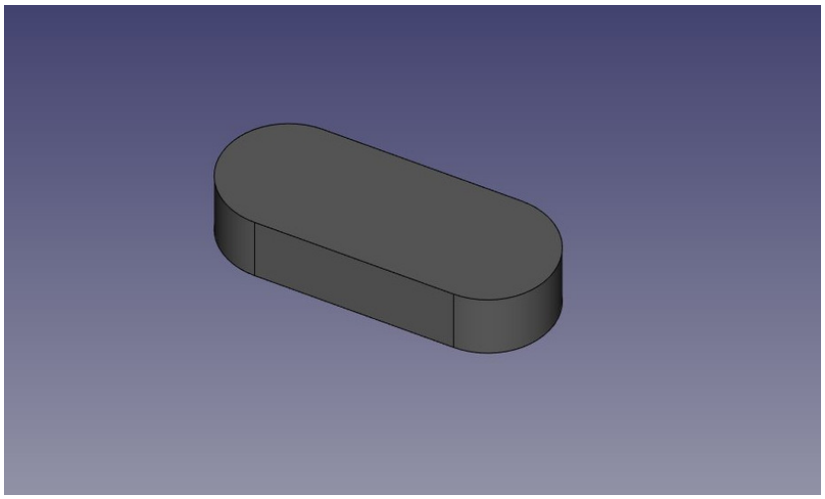
- **print(boxShape.ShapeType)**: Displays the type of the top-level shape represented by **boxShape**. In this case, since **boxShape** was created as a box

using **Part.makeBox**, the output will be "Solid", indicating that the shape is a 3D solid object.

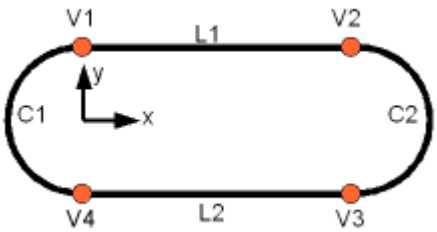
- **print(boxShape.Faces[0].ShapeType)**: Accesses the first face in the **Faces** list of **boxShape** (index 0) and prints its shape type. For a box, each face is a flat surface, so the output will be "Face".
- **print(boxShape.Vertexes[2].ShapeType)**: Accesses the third vertex in the **Vertexes** list of **boxShape** (index 2) and prints its shape type. Since this is a specific point in 3D space, the output will be "Vertex".

To summarize the concept of Part Shapes: Everything begins with **Vertexes**, the most basic elements of geometry. Using one or two **Vertexes**, you can create an **Edge** (note that full circles require only one **Vertex**). One or more **Edges** can then form a **Wire**, which can be either open or closed. When you have one or more closed **Wires**, you can create a **Face**. Additional **Wires** inside the main **Wire** will act as "holes" in the **Face**. Combining one or more **Faces** allows you to construct a **Shell**, which is essentially a collection of connected surfaces. If a **Shell** is fully closed and watertight, it can then be used to form a **Solid**—a 3D object with volume. Finally, any number of shapes of any type, including **Vertexes**, **Edges**, **Wires**, **Faces**, **Shells**, or **Solids**, can be grouped together into a **Compound**, which acts as a container for multiple shapes.

We can now try creating complex shapes from scratch, by constructing all their components one by one. For example, let's try to create a volume like this:



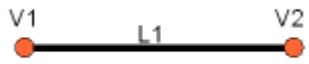
We will start by creating a planar shape like this:



First, let's create the four base points:

```
V1 = FreeCAD.Vector(0,10,0)
V2 = FreeCAD.Vector(30,10,0)
V3 = FreeCAD.Vector(30,-10,0)
V4 = FreeCAD.Vector(0,-10,0)
```

Then we can create the two linear segments:

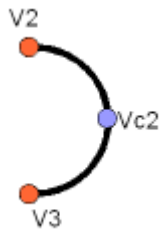



```
L1 = Part.LineSegment(V1,V2)
L2 = Part.LineSegment(V4,V3)
```

Note that we didn't need to create **Vertexes** explicitly. Instead, we could directly create **Part.LineSegments** using **FreeCAD Vectors**. This is because at this stage, we are working with base geometry, not actual **Edges**. A **Part.LineSegment**, as well as **Part.Circle**, **Part.Arc**, **Part.Ellipse**, or **Part.BSpline**, defines the underlying geometry but does not generate an edge by itself. In FreeCAD, edges are always constructed from such base geometry, which is stored in the **Curve** attribute of the **Edge**. This means that an edge is essentially a wrapper around the base geometry, inheriting its properties. If you have an edge, you can access its underlying geometry by referring to the curve attribute. The following command:

```
print(Edge.Curve)
```

allows you to understand the underlying structure of the edge and how it was constructed. Now, let's return to our exercise and proceed with building the arc segments. To create an arc, we need three points: a start point, an end point, and a midpoint that determines the curvature. For this purpose, we can use the convenient **Part.Arc** function, which takes these three points as input and generates the base geometry for an arc.




```
VC1 = FreeCAD.Vector(-10,0,0)
C1 = Part.Arc(V1,VC1,V4)
VC2 = FreeCAD.Vector(40,0,0)
C2 = Part.Arc(V2,VC2,V3)
```

Now we have 2 lines (L1 and L2) and 2 arcs (C1 and C2). We need to turn them into edges:

```
E1 = Part.Edge(L1)
E2 = Part.Edge(L2)
E3 = Part.Edge(C1)
E4 = Part.Edge(C2)
```

Alternatively, base geometries also have a **toShape()** function that does exactly the same thing:

```
E1 = L1.toShape()
E2 = L2.toShape()
 ...
```

Once we have a series of edges, we can now form a **Wire**, by giving it a list of edges. We do need to pay attention to the order. Also, note the brackets.

```
W = Part.Wire([E1,E4,E2,E3])
```

And we can check if our wire was correctly understood, and that it is correctly closed:

```
print( W.isClosed() )
```

Which will print "True" or "False". In order to make a **Face**, we need **closed Wires**, so it is always a good idea to check that before creating the face. Now we can create a face, by giving it a single wire (or a list of wires if we want holes):

```
F = Part.Face(W)
```

Then we extrude it:

```
P = F.extrude(FreeCAD.Vector(0,0,10))
```

Note that P is already a **Solid**:

```
print(P.ShapeType)
```

This is because when we extrude a single face, we always get a solid. This wouldn't be the case, for example, if we had extruded the wire instead:

```
S = W.extrude(FreeCAD.Vector(0,0,10))
print(S.ShapeType)
```

Which will of course give us a hollow shell, with the top and bottom faces missing.

Now that we have our final Shape, we are anxious to see it on screen! So let's create a generic object, and assign our new Solid to it:

```
myObj2 =
FreeCAD.ActiveDocument.addObject("Part::Feature","My_Strange_Solid")
myObj2.Shape = P
FreeCAD.ActiveDocument.recompute()
```

Alternatively, the Part module also provides a shortcut that does the above operation quicker (but you cannot choose the name of the object):

```
Part.show(P)
```

All of the above, and much more, is explained in detail on the Part Scripting page, together with examples.

Read more:

- The Part Workbench
- Part scripting

In the last two chapters, we saw how to create Part geometry and create parametric objects. One last piece is missing to gain full control over FreeCAD: Create tools that will interact with the user.

In many situations, it is not very user-friendly to construct an object with zero values, like we did with the rectangle in the previous chapter, and then ask the user to fill in the Height and Width values in the Properties panel. This works for a very small number of objects but will become very tedious if you have a lot of rectangles to make. A better way would be to be able to already give the Height and Width when creating the rectangle.

Python offers a basic tool to have the user enter text on the screen:

```
text = raw_input("Height of the rectangle?")
print("The entered height is ",text)
```

However, this requires a running Python console, and when running our code from a macro, we are not always sure that the Python console will be turned on on the user's machine.

The Graphical User Interface (GUI)—encompassing the menus, toolbars, 3D view, and other visual components of FreeCAD—is designed to make the software intuitive and accessible. It serves as a bridge between the user and FreeCAD's underlying functionality, allowing both casual users and experts to interact with the program effectively.

FreeCAD’s GUI is built using Qt, a powerful and open-source GUI toolkit that provides a wide range of features. Qt offers essential building blocks for interface design, such as dialog boxes, buttons, labels, text input fields, and pull-down menus, collectively known as "widgets." These widgets form the foundation of FreeCAD's user experience.

One of the key advantages of Qt is its cross-platform compatibility, enabling FreeCAD to run seamlessly on different operating systems like Windows, macOS, and Linux. Additionally, Qt’s flexibility makes it easy for developers to extend or customize FreeCAD’s interface, either by creating new toolbars and menus or by building entirely new modules that integrate into the software. This adaptability ensures that FreeCAD remains both user-friendly and highly extensible.

For users interested in scripting or developing new tools, FreeCAD's Python API also provides access to many Qt features. This means you can not only automate tasks but also create custom widgets or dialogs that integrate directly into the FreeCAD environment.

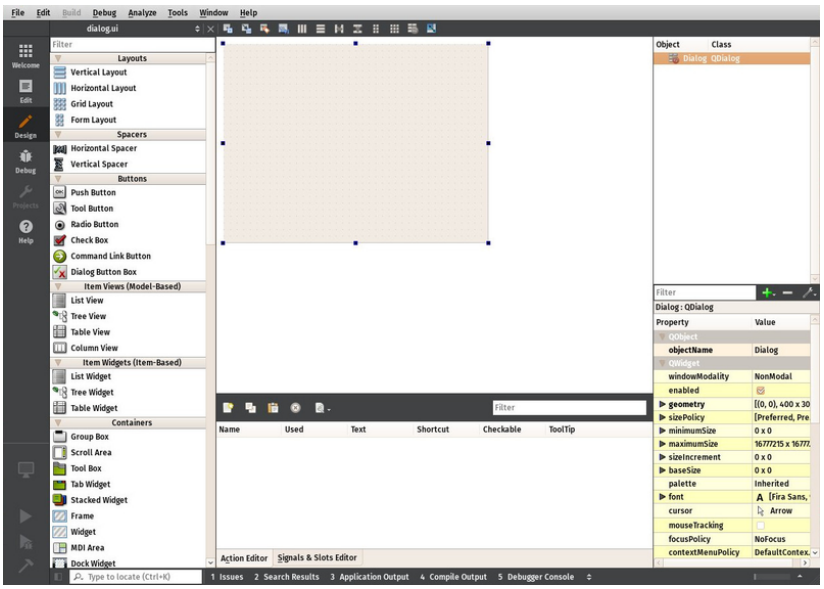
The Qt tools are very easy to use from Python, thanks to a Python module called PySide. PySide is the official Python binding for the Qt library, providing a seamless way to create and interact with widgets programmatically. It allows developers to design interfaces, manage user inputs (such as reading text from input boxes), and define actions based on user interactions, such as responding to a button press. Using PySide, you can build custom dialog boxes, menus, and toolbars directly within FreeCAD, extending its functionality in a way that integrates smoothly with its existing interface.

PySide makes it easy to connect user actions to specific functions in your code. For example, you can set up a button so that when it’s pressed, it triggers a script to execute a command or modify an object in the 3D view. This interactive capability opens up endless possibilities for customizing workflows and automating repetitive tasks.

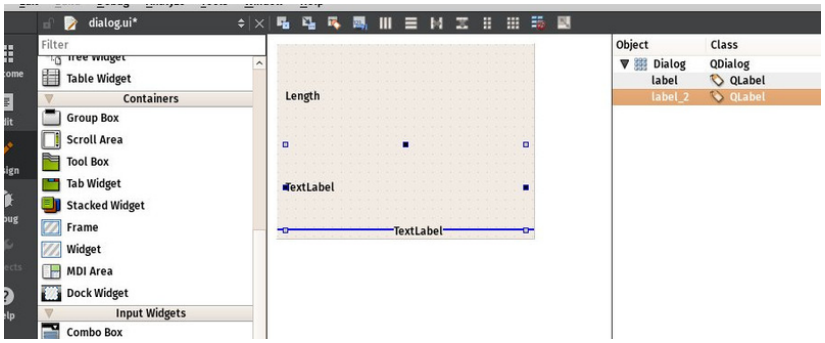
Qt also provides another interesting tool called Qt Designer, which is today embedded inside a bigger application called Qt Creator. It makes it possible to design dialog boxes and interface panels graphically, instead of having to code them manually. In this chapter, we will use Qt Creator to design a panel widget that we will use in the **Task** panel of FreeCAD. So you will need to download and install Qt Creator from its official page if you are on Windows or Mac (on Linux it will usually be available from your software manager application).

In the following exercise, we will first create a panel with Qt Creator that asks for length, width and height values, then we will create a Python class around it, that will read the values entered by the user from the panel, and create a box with the given dimensions. This Python class will then be used by FreeCAD to display and control the task panel:

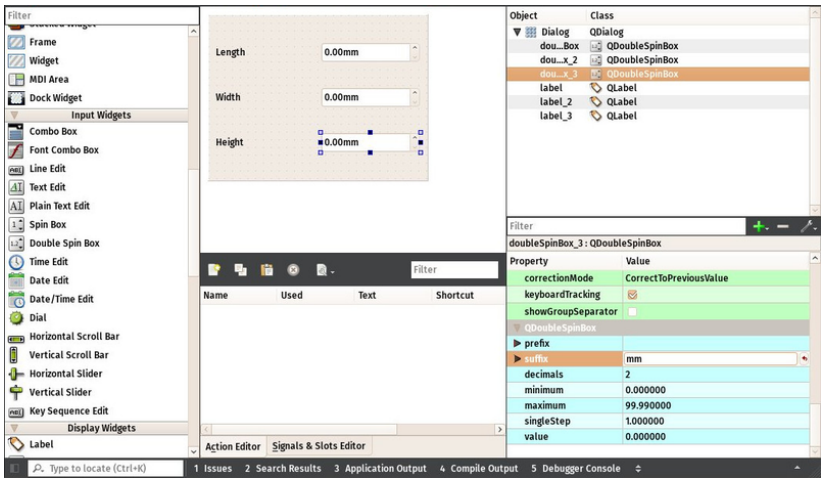
Let's start by creating the widget. Start Qt Creator, then menu **File** → **New File or Project** → **Qt** → **Qt Designer Form** → **Dialog without buttons**. Click **Next**, give it a filename to save, click **Next**, leave all project fields to their default value ("<none>"), and **Create**. FreeCAD's Task system will automatically add OK/Cancel buttons, that's why we chose here a dialog without buttons.



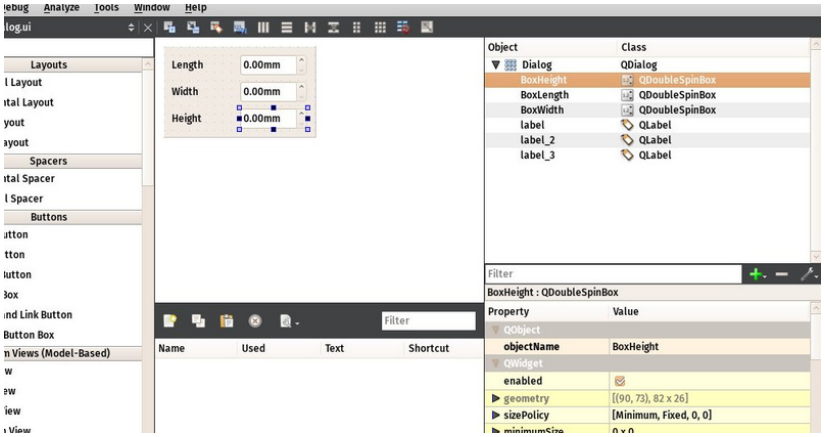
- Find the **Label** in the list in the left panel (under the Display Widgets section), and drag it onto the canvas of our widget. Double-click the recently placed Label, and change its text to **Length**.
- Right-click the widget canvas, and choose **Lay out** → **Lay out in a Grid**. This will turn our widget into a grid with currently only one cell, occupied by our first label. We can now add the next items at the left, right, top or bottom of our first label, and the grid will expand automatically.
- Add two more labels below the first one, and change their text to Width and Height:



- Now place 3 **Double Spin Box** widgets (under the Input Widgets section) next to our Length, Width and Height labels. For each of them, in the lower right panel, which shows all the available settings for the selected widget, locate **Suffix** and set their suffix to **mm**. FreeCAD has a more advanced widget, that can handle different units, but that is not available in Qt Creator by default (but can be compiled), so for now we will use a standard Double Spin Box, and we add the "mm" suffix to make sure the user knows in which units they work:



- Now our widget is done, we just need to make sure of one last thing. Since FreeCAD will need to access that widget and read the Length, Width and Height values, we need to give proper names to those widgets, so we can easily retrieve them from within FreeCAD. Click each of the Double Spin Boxes, and in the upper right window, double-click their Object Name, and change them to something easy to remember, for example: BoxLength, BoxWidth and BoxHeight:



- Save the file, you can now close Qt Creator, the rest will be done in Python.
- Open FreeCAD and create a new macro from menu **Macro** → **Macros** → **Create**
- Paste the following code. Make sure you change the file path to match where you saved the .ui file created in QtCreator:

```
import FreeCAD,FreeCADGui,Part

# CHANGE THE LINE BELOW
path_to_ui = "C:\Users\yorik\Documents\dialog.ui"

class BoxTaskPanel:
   def __init__(self):
       # this will create a Qt widget from our ui file
       self.form = FreeCADGui.PySideUic.loadUi(path_to_ui)

   def accept(self):
       length = self.form.BoxLength.value()
       width = self.form.BoxWidth.value()
       height = self.form.BoxHeight.value()
       if (length == 0) or (width == 0) or (height == 0):
           print("Error! None of the values can be 0!")
           # we bail out without doing anything
           return
       box = Part.makeBox(length,width,height)
       Part.show(box)
       FreeCADGui.Control.closeDialog()

panel = BoxTaskPanel()
FreeCADGui.Control.showDialog(panel)
```

In the code above, we used a convenience function **PySideUic.loadUi** from the **FreeCADGui** module. That function loads a .ui file, creates a Qt Widget from it, and maps names, so we can easily access the subwidget by their names (ex: self.form.BoxLength).

The "accept" function is also a convenience offered by Qt. When there is an "OK" button in a dialog (which is the case by default when using the FreeCAD Tasks panel), any function named "accept" will automatically be executed when the "OK" button is pressed. Similarly, you can also add a "reject" function which gets executed when the "Cancel" button is pressed. In our case, we omitted that function, so pressing "Cancel" will do the default behaviour (do nothing and close the dialog).

If we implement any of the accept or reject functions, their default behaviour (do nothing and close) will not occur anymore. So we need to close the Task panel ourselves. This is done with:

```
FreeCADGui.Control.closeDialog()
```

Once we have our BoxTaskPanel that has 1- a widget called "self.form" and 2- if needed, accept and reject functions, we can open the task panel with it, which is done with these two last lines:

```
panel = BoxTaskPanel()
 FreeCADGui.Control.showDialog(panel)
```

Note that the widget created by **PySideUic.loadUi** is not specific to FreeCAD, it is a standard Qt widget that can be used with other Qt tools. For example, we could have shown a separate dialog box with it. Try this in the Python Console of FreeCAD (using the correct path to your .ui file of course):

```
from PySide import QtGui
 w = FreeCADGui.PySideUic.loadUi("C:\Users\yorik\Documents\dialog.ui")
 w.show()
```

Of course, we didn't add any "OK" or "Cancel" buttons to our dialog, since it was made to be used from the FreeCAD Task panel, which already provides such buttons. So there is no way to close the dialog (other than pressing its Window Close button). But the function show() creates a non-modal dialog, which means it doesn't block the rest of the interface. So, while our dialog is still open, we can read the values of the fields:

```
w.BoxHeight.value()
```

This is very useful for testing.

Finally, don't forget there is much more documentation about using Qt widgets on the FreeCAD Wiki, in the Python Scripting section, which contains a dialog creation tutorial, a special 3-part PySide tutorial that covers the subject extensively.

Relevant Links[edit | edit source]

- Qt Creator Documentation
- Installing Qt Creator
- FreeCAD Python scripting documentation
- FreeCAD Dialog creation tutorial
- FreeCAD PySide tutorials
- PySide documentation

Creating parametric objects

In the previous chapter, we explored how to create **Part** geometry and display it on screen by attaching it to a "dumb" (non-parametric) document object. While effective, this approach becomes cumbersome when you need to modify the shape. Each change requires creating a new shape and reassigning it to the object, resulting in repetitive and inefficient workflows.

Throughout this manual, we've seen how parametric objects address this issue by allowing dynamic updates. By modifying a single property, the shape recalculates automatically, eliminating the need for manual updates. This recalculation process enables more efficient modeling and introduces adaptability to designs. Internally, parametric objects operate similarly to what we've already done: they recalculate the contents of their **Shape** property every time one of their properties changes. This iterative process is seamless and ensures that the object remains consistent with its defined parameters.

FreeCAD offers a user-friendly system for creating parametric objects entirely in Python. These objects are defined using a Python class, which:

- Declares the necessary properties for the object.
- Defines the behavior when any of these properties are modified.

The structure of such a parametric object is as simple as this:

```
class myParametricObject:

    def __init__(self, obj):
        obj.Proxy = self
        obj.addProperty("App::PropertyFloat", "MyLength")
        ...

    def execute(self,obj):
        print ("Recalculating the shape...")
        print ("The value of MyLength is:")
        print (obj.MyLength)
        ...
```

All Python classes typically have an **__init__** method. This method is executed when a class is instantiated—that is, when a Python object is created from the class. You can think of a class as a "template" or blueprint used to create live instances of itself. Each instance of the class becomes an independent object with its own attributes and methods. In our **__init__** method, we perform two crucial tasks:

- Store the class itself in the **Proxy** attribute of the FreeCAD document object:

By assigning **self** to the **Proxy** attribute of the FreeCAD document object, we link the logic of our Python class to the FreeCAD object. This means the document object will "carry" the Python class code inside itself, allowing it to behave according to the logic defined in the class. This connection enables FreeCAD to know how to interact with and recalculate the parametric object.

- Create all the properties the object needs:

Using the **addProperty** method, we define the custom properties required by the object. Properties act as parameters or variables for the object and can be accessed, modified,

and displayed in FreeCAD's property editor. In the example, we add a floating-point property named **MyLength**. This property will later influence the shape or behavior of the object.

There are many types of properties available in FreeCAD, ranging from floating-point numbers to strings and even specialized types for geometry and materials. To explore the full list of available property types, you can type the following code in the Python console:

```
FreeCAD.ActiveDocument.addObject("Part::FeaturePython",
"dummy").supportedProperties()
```

The second key part of our class is the **execute** method. This method is automatically triggered whenever the object is marked for recomputation, which happens when one of its properties is modified. The **execute** method is where all the recalculations for the object take place, ensuring that its shape and behavior are updated to reflect any changes. Within the **execute** method, you perform all the necessary operations to generate the new geometry for your object. Typically, this involves recalculating the shape based on the current values of its properties and then assigning the updated shape to the object's **Shape** attribute using a statement like **obj.Shape = myNewShape**. The **execute** method takes a single argument, **obj**, which represents the FreeCAD document object associated with your parametric object. This allows you to directly manipulate the object within the method, such as accessing its properties, updating its geometry, or performing other operations.

In summary:

- The **execute** method is called whenever the object needs to be updated.
- It is responsible for recalculating the shape and assigning it to the object's **Shape** attribute.
- The **obj** argument gives access to the FreeCAD document object, enabling you to make changes programmatically.

With this system, FreeCAD handles the rest—ensuring that the object is properly updated in the document and displayed correctly in the graphical interface.

One crucial thing to remember is that when you create parametric objects in a FreeCAD document, the Python code used to define them is not saved within the file. This is an intentional security measure. If FreeCAD files were allowed to store Python code, it could open the door for malicious actors to distribute files containing harmful scripts that might damage someone’s computer. As a result, when you share a FreeCAD file containing parametric objects created with custom Python code, the recipient must also have access to the code used to define those objects. Without it, FreeCAD won’t know how to recalculate or interact with the objects properly.

The simplest way to ensure this is to save the Python code in a macro file. When distributing your FreeCAD file, you can include the macro alongside it. Alternatively, you can share the macro on the FreeCAD macros repository, allowing others to easily download and use it. This approach ensures that your custom parametric objects remain functional on other systems while maintaining security best practices.

Below, we will do a small exercise, building a parametric object that is a simple parametric rectangular face. More complex examples are available on the parametric object example and in the FreeCAD source code itself.

We will give our object two properties: Length and Width, which we will use to construct a rectangle. Then, since our object will already have a pre-built Placement property (all

geometric objects have one by default, no need to add it ourselves), we will displace our rectangle to the location/rotation set in the Placement, so the user will be able to move the rectangle anywhere by editing the Placement property.

```
class ParametricRectangle:

    def __init__(self,obj):
        obj.Proxy = self
        obj.addProperty("App::PropertyFloat", "Length")
        obj.addProperty("App::PropertyFloat", "Width")

    def execute(self, obj):
        # We need to import the FreeCAD module here too, because we might
be running out of the Console
        # (in a macro, for example) where the FreeCAD module has not been
imported automatically:
        import FreeCAD
        import Part

        # First we need to make sure the values of Length and Width are
not 0
        # otherwise Part.LineSegment will complain that both points are
equal:
        if (obj.Length == 0) or (obj.Width == 0):
            # If yes, exit this method without doing anything:
            return

        # We create 4 points for the 4 corners:
        v1 = FreeCAD.Vector(0, 0, 0)
        v2 = FreeCAD.Vector(obj.Length, 0, 0)
        v3 = FreeCAD.Vector(obj.Length,obj.Width, 0)
        v4 = FreeCAD.Vector(0, obj.Width, 0)

        # We create 4 edges:
        e1 = Part.LineSegment(v1, v2).toShape()
        e2 = Part.LineSegment(v2, v3).toShape()
        e3 = Part.LineSegment(v3, v4).toShape()
        e4 = Part.LineSegment(v4, v1).toShape()

        # We create a wire:
        w = Part.Wire([e1, e2, e3, e4])

        # We create a face:
        f = Part.Face(w)

        # All shapes have a Placement too. We give our shape the value of
the placement
        # set by the user. This will move/rotate the face automatically.
        f.Placement = obj.Placement

        # All done, we can attribute our shape to the object!
        obj.Shape = f
```

Instead of pasting the above code in the Python console, we'd better save it somewhere, so we can reuse and modify it later. For example in a new macro (menu **Macro -> Macros -> Create**). Name it, for example, "ParamRectangle". However, FreeCAD macros are saved with a .FCMacro extension, which Python doesn't recognize when using import. So, before using the above code, we will need to rename the ParamRectangle.FCMacro

file to ParamRectangle.py. This can be done simply from your file explorer, by navigating to the Macros folder indicated in menu Tools -> Macros.

Once that is done, we can now do this in the Python Console:

```
import ParamRectangle
```

By exploring the contents of ParamRectangle, we can verify that it contains our ParametricRectangle class.

To create a new parametric object using our ParametricRectangle class, we will use the following code. Observe that we use **Part::FeaturePython** instead of **Part::Feature** that we have been using in the previous chapters (The Python version allows to define our own parametric behaviour):

```
myObj = FreeCAD.ActiveDocument.addObject("Part::FeaturePython",
"Rectangle")
ParamRectangle.ParametricRectangle(myObj)
myObj.ViewObject.Proxy = 0 # This is mandatory unless we code the
ViewProvider too.
FreeCAD.ActiveDocument.recompute()
```

Here is a quick breakdown of the previous commands:

- **myObj = FreeCAD.ActiveDocument.addObject("Part::FeaturePython", "Rectangle")**: Creates a new **Part::FeaturePython** object named **Rectangle** in the active FreeCAD document. This object is specifically designed for custom parametric behavior, allowing Python-defined logic to manage its properties and behavior.

- **ParamRectangle.ParametricRectangle(myObj)**: Initializes the object by associating it with the **ParametricRectangle** class from the **ParamRectangle** module or script. This links the custom Python-defined logic to the object, enabling it to act as a parametric object.

- **myObj.ViewObject.Proxy = 0**: Resets the **ViewObject.Proxy** attribute to 0, ensuring that the object uses FreeCAD's default view handling. This step is required unless you define a custom ViewProvider to manage the visual representation of the object.

- **FreeCAD.ActiveDocument.recompute()**: Recomputes the document to update the geometry and reflect changes in the FreeCAD graphical interface, making the new object fully visible and functional

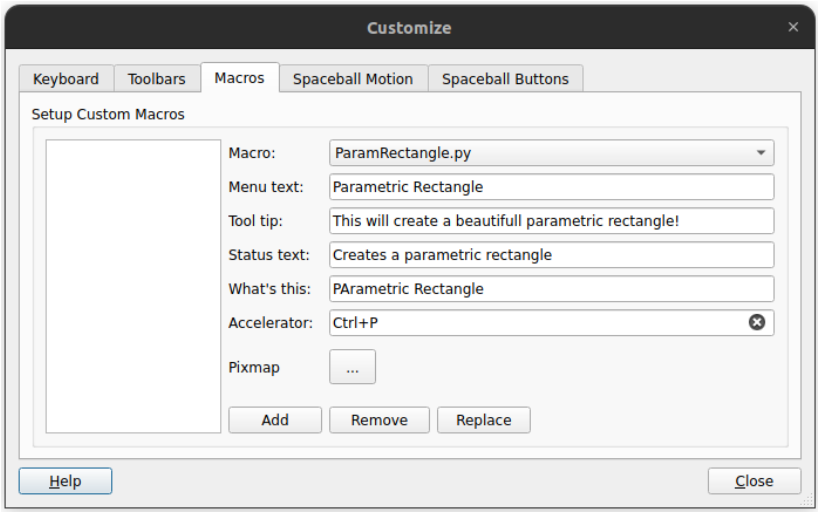
Nothing will appear on screen just yet, because the **Length** and **Width** properties are 0, which will trigger our "do-nothing" condition inside execute. We just need to change the values of Length and Width, and our object will magically appear and be recalculated on the fly.

Of course, it would be tedious to have to type these 4 lines of Python code each time we want to create a new parametric rectangle. A very simple way to solve this is placing the 4 lines above inside our **ParamRectangle.py** file, at the end, after the end of the **ParametricRectange** class (We can do this from the Macro editor).

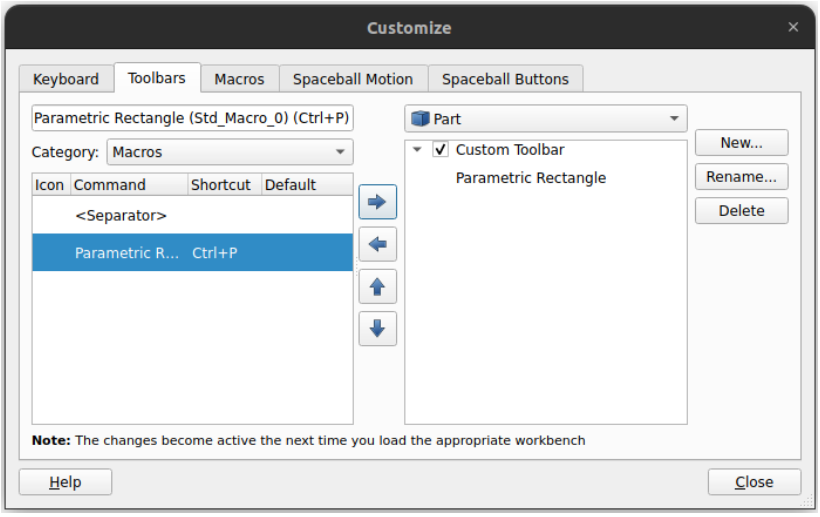
Now, when we type import ParamRectangle, a new parametric rectangle will automatically be created. Even better, we can add a toolbar button that will do just that:

- Open menu **Tools -> Customize**

- Under the "Macros" tab, select our ParamRectangle.py macro, fill in the details as you wish, and press "Add":



- Under the Toolbars tab, create a new custom toolbar in the workbench of your choice (or globally), select your macro and add it to the toolbar:



- That's it, we now have a new toolbar button which, when clicked, will create a parametric rectangle.

Remember, if you want to distribute files created with this new tool to other people, they must have the **ParamRectangle.py** macro installed on their computers too.

Read more

- The FreeCAD macros repository
- Parametric object example
- More examples in the FreeCAD code

Creating renderings

Rendering is the process of creating highly realistic images from 3D models by simulating lighting, materials, and textures. It’s commonly used in industries like film, video games, and product design, where photorealistic visualizations are necessary to showcase designs or concepts. While rendering can create images that closely resemble real-life photographs, it requires specialized tools to control things like lighting, reflections, and shadows.

FreeCAD, however, is focused primarily on technical modeling rather than artistic or visual effects. Its primary purpose is to create accurate 3D models for engineering, design, and manufacturing. As a result, FreeCAD does not feature advanced built-in rendering tools for photorealism.

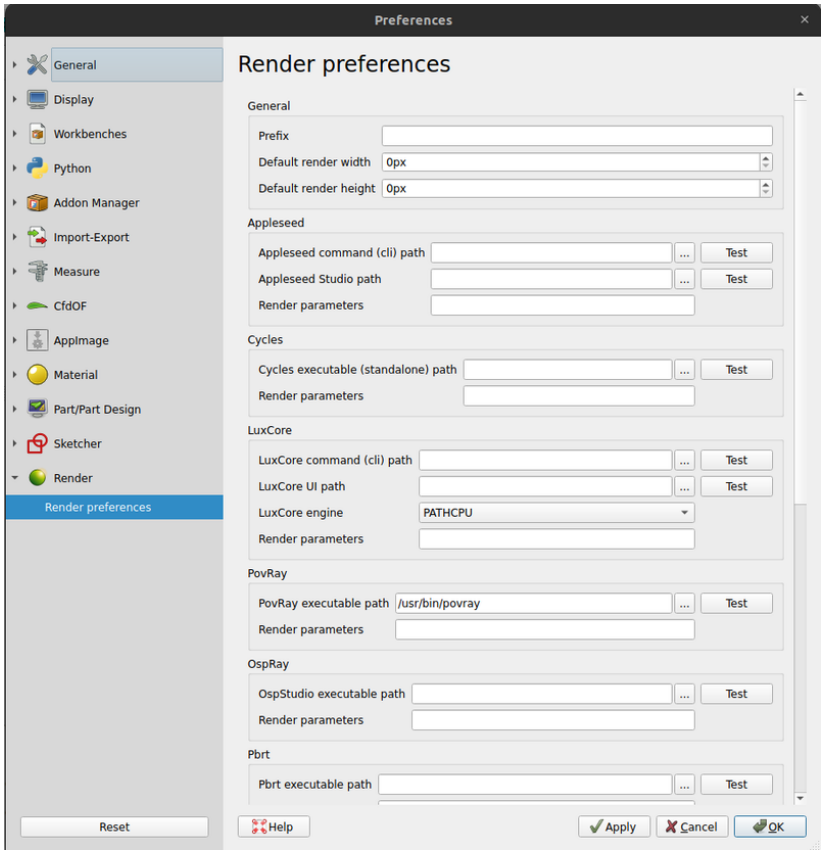
However, FreeCAD does offer the Render workbench, which can be installed as an add-on (it’s not one of the default workbenches). This workbench allows users to connect FreeCAD models with external rendering engines like Blender Cycles, LuxCoreRender, or POV-Ray. Through the Render Workbench, users can use their models and leverage these powerful external tools to render their designs with realistic lighting and textures. This approach keeps FreeCAD lightweight and focused while still providing the flexibility for photorealistic rendering when needed.

The Render Workbench in FreeCAD integrates with several external renderers, including LuxCorerender, POV-Ray, and Blender Cycles. LuxCoreRender is a modern, physically-based renderer that delivers photorealistic images, but it requires significant computational power, especially for large scenes. POV-Ray, while older, remains a reliable raytracing engine and is less resource-intensive, though it lacks the realism of newer technologies. Blender Cycles, accessible through FreeCAD when Blender is installed, offers a powerful rendering solution with GPU and CPU support, producing high-quality images efficiently. However, it requires installing Blender separately and exporting models to Blender for rendering. Each renderer offers strengths depending on the user's balance of realism, performance, and system capabilities. Every option has its strengths and weaknesses, depending on the type of image one wants to render. The best way to know is to look at examples on each engine's website.

Installation[edit | edit source]

Before using the Render Workbench in FreeCAD, you'll need to install both the workbench itself (as shown in this section and one of the external rendering applications such as LuxCoreRender, POV-Ray, or Blender Cycles (with Blender installed). Installing these applications is typically straightforward, as they provide installers for various platforms and are often included in software repositories on Linux distributions. Once these tools are installed, you'll be able to connect FreeCAD to these renderers to produce high-quality images.

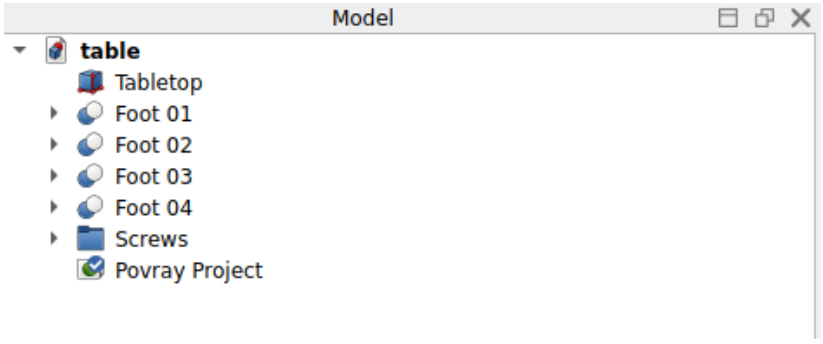
Once POV-Ray or LuxCorerender is installed, we need to set the path to their main executable in the FreeCAD preferences. This is usually only required on Windows and Mac. On Linux, FreeCAD will pick it from the standard locations. The location of the povray or luxrender executables can be found by searching your system for files named povray (or povray.exe on Windows) and luxrender (or luxrender.exe on Windows). In the **Preferences** tab you can designate the path as well as set up some parameters.



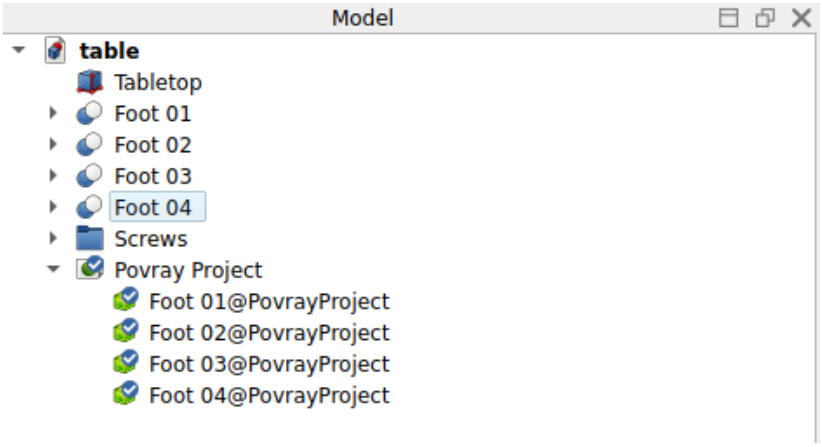
Rendering with PovRay[edit | edit source]

We will use the table we have been modelling in the traditional modeling chapter to produce renderings with PovRay.

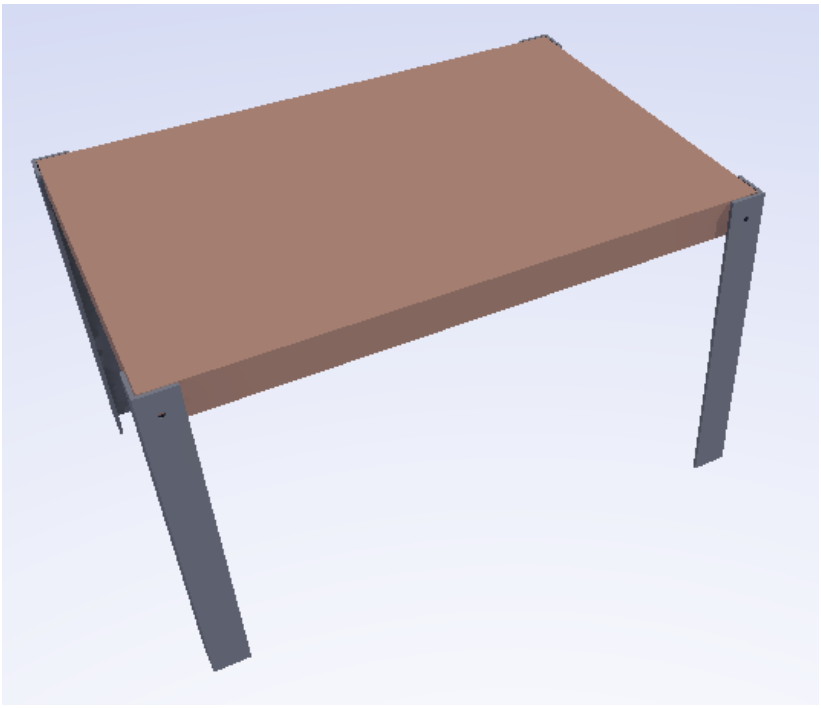
- Start by loading the table.FCStd file that we modelled earlier or from the link at the bottom of this chapter and entering the workbench.
- Create a rendering project by pressing the button in the toolbar corresponding to your renderer. For our example, we will choose the povray renderer.
- Select a template suitable for your project. We will be going with the **povray_sunlight.pov** one.
- You can also try other templates after you create a new project, simply by editing its **Template** property.
- A new project has now been created:



- You can add the desired objects to the project by selecting them and pressing on the rendering view option.



- If we wish we can apply a material to our bodies by pressing on the Material option. For our case, we will choose the matte option.
- We can now press on the button and our rendered result will appear in a separate window.



Truth be told, the end result is not very impressive. The rendering process is iterative and takes time and patience to achieve high-quality outcomes. Additionally, as mentioned above, POV-Ray is somewhat limited in terms of realism. Feel free to experiment with different renderers. The procedure remains largely the same, with the only difference being the selection of a different renderer at the start of the process.

Downloads

- The table model: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/table.FCStd
- The file produced during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/render.FCStd

Read more

- Blender
- POV-Ray

Generating 2D drawings

When your model cannot be printed or milled directly by a machine, for example, it is too big (a building) or it requires manual assembly after the pieces are ready, you will usually need to explain to another person how to do it. In technical fields (engineering, architecture, etc), this is usually done with drawings. The drawings are handed over to the person responsible for assembling the final product and will explain how to do it.

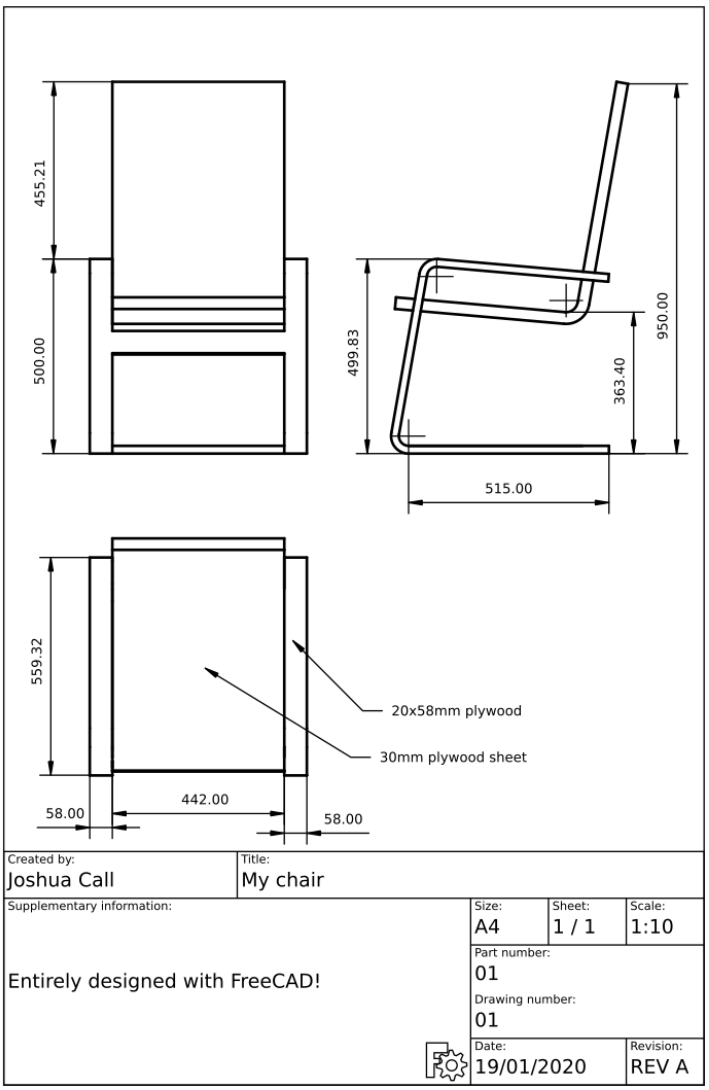
Typical examples are Ikea instructions, architectural drawings, and blueprints. These drawings usually contain not only the drawing itself, but also many annotations, such as text, dimensions, numbers, and symbols that will help other people to understand what needs to be done and how.

In FreeCAD, the workbench responsible for making such drawings is the TechDraw Workbench.

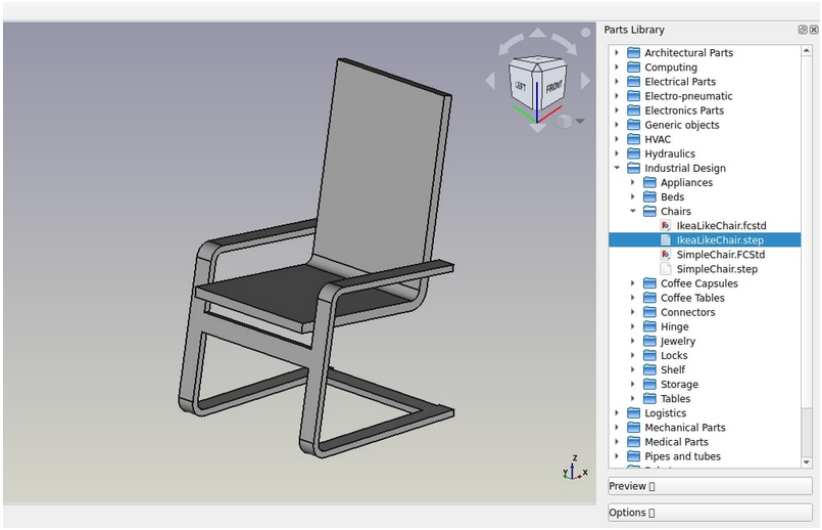
The TechDraw Workbench allows you to create sheets, which can be blank or use a pre-made template to already have a series of items on the sheet, such as borders and a title. On these sheets, you can then place views of the 3D objects you modeled previously, and configure how these views will appear on the sheet. You can also place all kinds of annotations on the sheet, such as dimensions, texts, and other symbols commonly used in technical drawings.

Drawing sheets, once complete, can be printed or exported as SVG, PDF or DXF files.

In the following exercise, we will see how to create a simple drawing of a chair model found in the FreeCAD library (Industrial Design → Chairs → IkeaLikeChair). The FreeCAD library can easily be added to your FreeCAD installation (refer to the installing chapter of this manual), or you can simply download the model from the library webpage, or via the direct link provided at the bottom of this chapter.

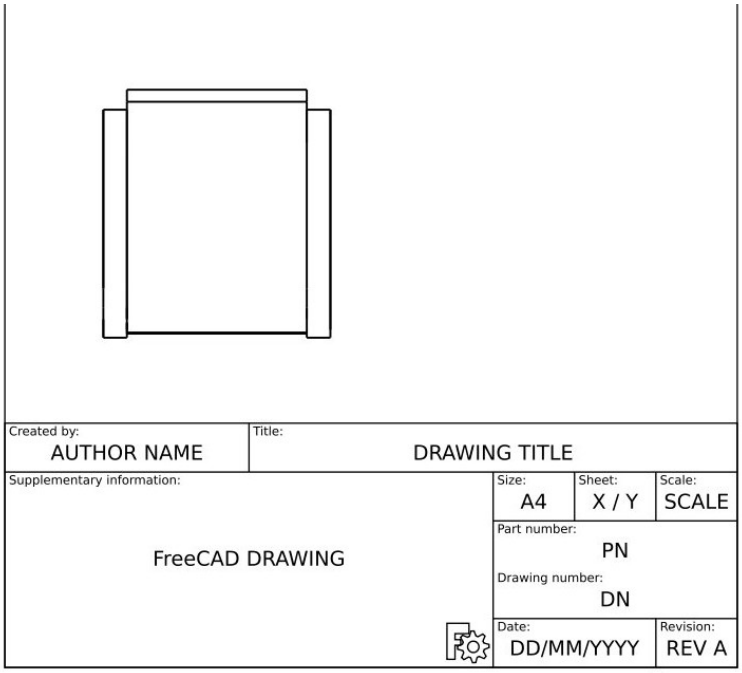


- Load the IkeaLikeChair file from the library. You can choose between the .FCStd version, which will load the full modeling history or the .step version, which will create only one object, without the history. Since we won't need to model any further now, it is best to choose the .step version, as it will be easier to manipulate.

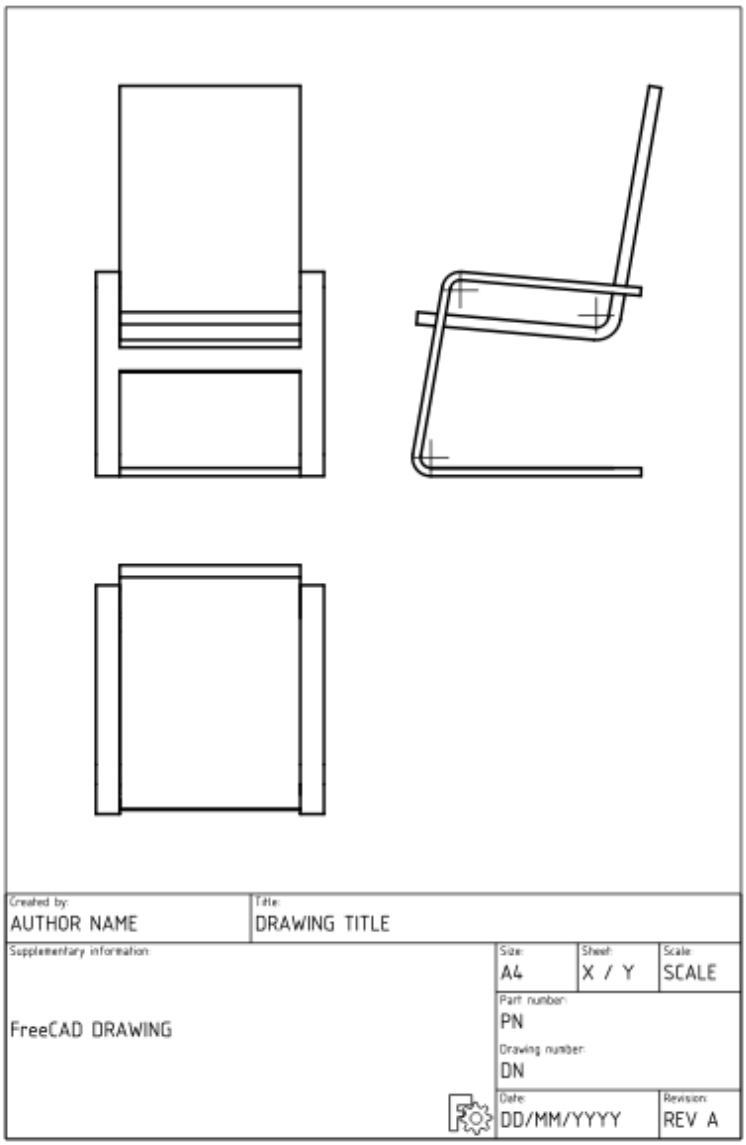


- Switch to the TechDraw Workbench
- Press the TechDraw PageTemplate button.

- Select the **A4_Portrait_ISO7200TD** template. A new tab will open in your FreeCAD window, showing the new page.
- In the tree view (or in the model tab), select the chair model. It will most likely be named something like "Open CASCADE STEP translator."
- Press the TechDraw View button.
- A View object will be created on our page. Select the view object in the tree view, and then give the view the following properties in the data tab of the combo view:
    - Under the Base category:
        - X: 70 mm
        - Y: 120 mm
        - Rotation: 0
        - Scale: 0.1
    - Under the Projection category (hit the drop-down arrow to modify the x, y, and z components of these properties individually):
        - Direction: [0 0 1]
        - XDirection: [0 -1 0] (Change the y field first, then the x field)
- We now have a nice top view of our chair. Hit the TechDraw ToggleFrame button to turn the View frames, labels, and vertices off.



- Let's repeat the operation twice, to create two more views. We will set their X and Y values, which indicate the position of the view on the page, in order to show them apart from the top view, and their direction, to create different view orientations. Give each new view the following properties:
    - View001 (front view): X: 70, Y: 220, Scale: 0.1, Rotation: 0, Direction: (-1,0,0), XDirection: (0,-1,0)
    - View002 (side view): X: 150, Y: 220, Scale: 0.1, Rotation: 0, Direction: (0,-1,0), XDirection: (1,0,0)
- After that, we obtain the following page:

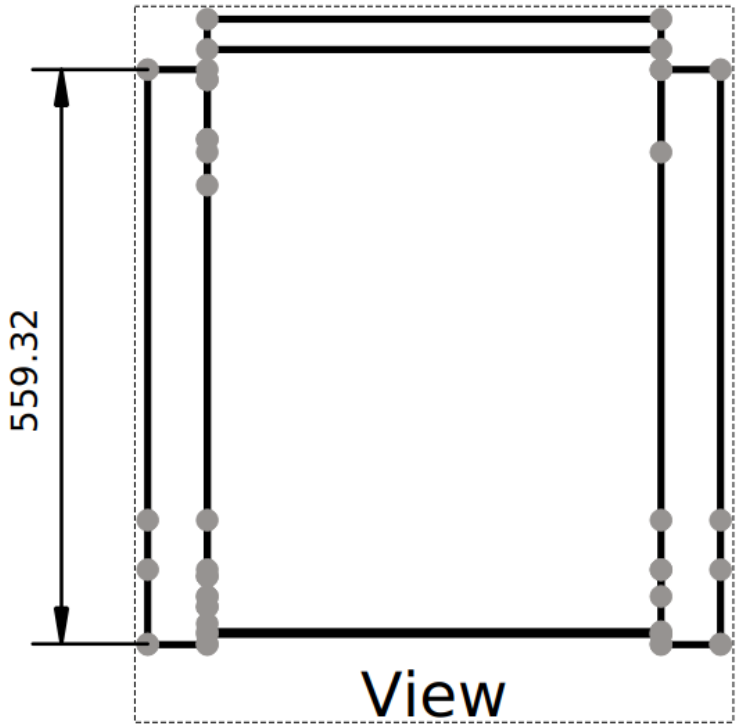


- Note that there may be easier ways to get the views that you want. You can simply rotate the 3D view of your model, and once you have the view you want, select the model in the tree view and hit New View. This will automatically insert a view with the desired rotation and direction properties. You can also use the TechDraw ProjectionGroup tool.

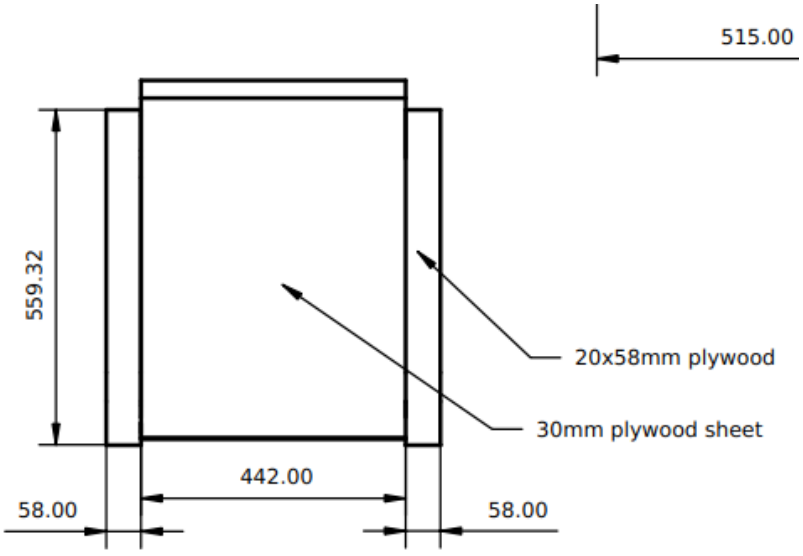
- We can tweak the aspect of our views if we want, for example, we can change their **Line Width** property (under the View tab in the Combo View) to 0.5.

We will now place dimensions and indications on our drawing. There are two ways to add dimensions to a model: one is placing the dimensions inside the 3D model, using the Dimension tool of the Draft Workbench, and then placing a view of these dimensions on our sheet with the TechDraw DraftView tool. The other is to do things directly on the TechDraw sheet. We'll use the latter method.

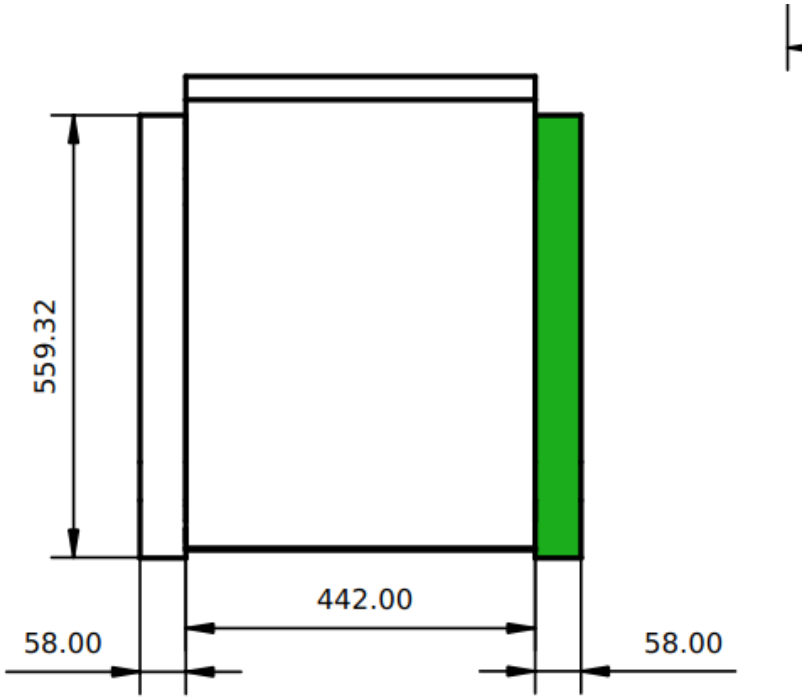
- Hit the Toggle button to turn the vertices on.
- Use Ctrl + Left Mouse Click to select the two vertices you want to measure the distance between.
- Hit the TechDraw LengthDimension button.



- Repeat the operation, until all the dimensions you wish to indicate are placed. Use the TechDraw VerticalDimension and TechDraw HorizontalDimension tools as necessary.
- Take a minute to look at the properties of the Dimension object in the Combo View.
- Please note that if you are dimensioning an axonometric view (e.g., isometric view) instead of a multiview view (e.g., front view) as we have done here, you will need to use the TechDraw LinkDimension tool to get an accurate dimension.



- We will now place the two callouts shown in the image above, using the TechDraw Balloon tool.



1. Looking at the Page in the 3D view window, select the View to which the Balloon will be attached, as shown in the image above.
2. Press the Balloon button.
3. The cursor is now displayed as a balloon icon. Click on the page to place the balloon origin at the desired position.
4. The balloon bubble may be dragged to the desired position.
5. Change the balloon properties by double-clicking the balloon label or the balloon object in the tree view. This will open the Balloon Task dialog. Set the Value field to the desired text and change the Symbol drop-down menu selection to **None**
6. Press OK
7. Repeat the operation for the second callout.

- We will now fill in the sheet title block.
  - Make sure that the View frames, labels, and vertices are visible. If not, hit the Toggle button.
  - Edit the text in each section of the sheet title block by clicking on the small green square on the left side of the text.

Our page can now be exported to SVG for further work in graphical applications like Inkscape or to DXF. Select the page in the tree view and then select menu **File** → **Export**. The DXF format is importable in almost all existing 2D CAD applications. TechDraw pages can also be directly printed or exported to PDF.

Downloads

- The file created during this exercise: drawing.FCStd
- The SVG sheet produced from that file: drawing.svg

Read more

- The TechDraw workbench
- Create custom templates
- Another TechDraw tutorial
- The FreeCAD library
- Inkscape

Watch tutorials

- Sliptonic's TechDraw playlist
- Symbols and Views

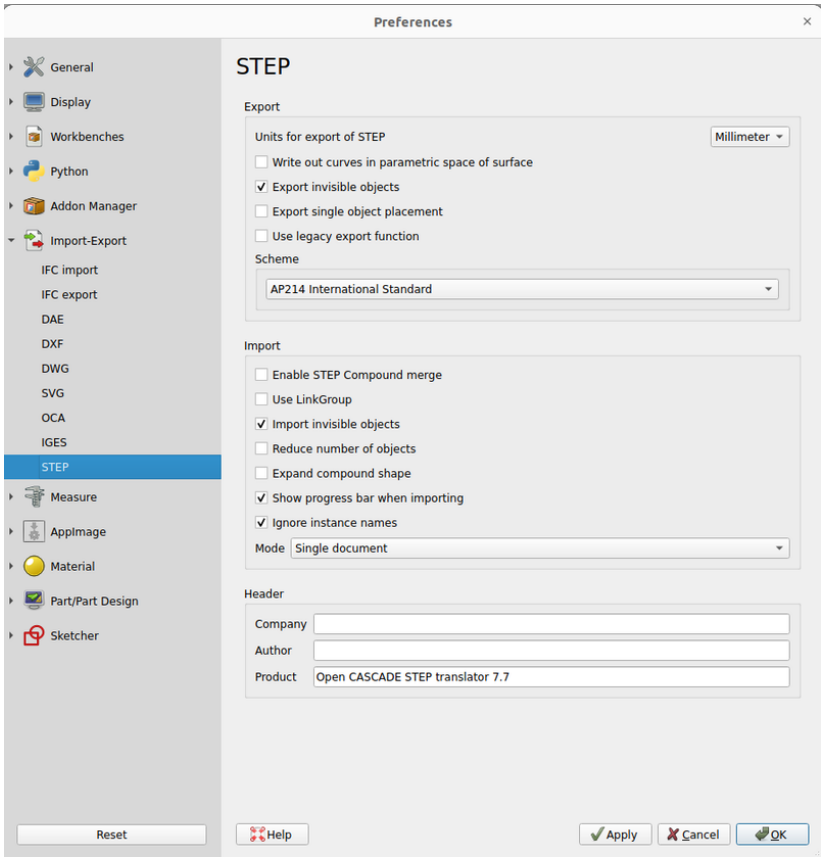
Import and export to other filetypes

FreeCAD can import and export to many file types. Here is a list of the most important ones with a short description of the available features:

Format	Import	Export	Notes
STEP	Yes	Yes	This is the most faithful import/export format available since it supports solid geometry and NURBS. Use it whenever it is possible.
IGES	Yes	Yes	An older solid format, also very well supported. Some older applications don't support STEP but have IGES.
BREP	Yes	Yes	The native format of OpenCasCade, FreeCAD's geometry kernel.
DXF	Yes	Yes	An open format maintained by Autodesk. Since the 3D data inside a DXF file is encoded in a proprietary format, FreeCAD can only import/ export 2D data to/from this format.
DWG	Yes	Yes	The proprietary version of DXF. Requires the installation of the Teigha File Converter utility. This format suffers from the same limitations as DXF.
OBJ	Yes	Yes	A mesh-based format. Can only contain triangulated meshes. All solid and NURBS-based objects of FreeCAD will be converted to mesh on export. An alternative exporter is provided by the Arch workbench, more suited to the export of architectural models.
DAE	Yes	Yes	The main import/export format of Sketchup. Can only contain triangulated

			meshes. All solid and NURBS-based objects of FreeCAD will be converted to mesh on export.
STL	Yes	Yes	A mesh-based format, commonly used for 3D printing. Can only contain triangulated meshes. All solid and NURBS-based objects of FreeCAD will be converted to mesh on export.
PLY	Yes	Yes	An older mesh-based format. Can only contain triangulated meshes. All solid and NURBS-based objects of FreeCAD will be converted to mesh on export.
IFC	Yes	Yes	Industry Foundation Classes. Requires the installation of IfcOpenShell-python. The IFC format and its compatibility with other applications is a complex affair, use with care.
SVG	Yes	Yes	An excellent, widespread 2D graphics format
VRML	Yes	Yes	A rather old mesh-based web format.
GCODE	Yes	Yes	FreeCAD can already import and export to/from several flavors of GCode, but only a small number of machines are supported at the moment.
CSG	Yes	No	OpenSCAD's CSG (Constructive Solid Geometry) format.

Some of these file formats have options. These can be configured from menu **Edit** → **Preferences** → **Import/export:**



Read more

- All file formats supported by FreeCAD
- Working with DXF files in FreeCAD:
- Working with SVG files in FreeCAD
- Importing and exporting to IFC
- OpenCasCade
- Teigha File Converter
- IFC Specifications Database
- IfcOpenShell

Installing

FreeCAD is licensed under the LGPL license, which allows you to download, install, redistribute, and use it for any purpose, commercial or non-commercial, without any restrictions. You retain full ownership of the files you create.

FreeCAD operates consistently across Windows, macOS, and Linux, although the installation process varies by platform. For Windows and Mac users, the FreeCAD community offers ready-to-use precompiled installers. On Linux, the source code is provided to distribution maintainers who package the software for their specific systems. Typically, Linux users can install FreeCAD directly through their system's software manager.

The official FreeCAD download page can be found at the FreeCAD download page. Additional information about the installation process can be found at the dedicated download wiki.

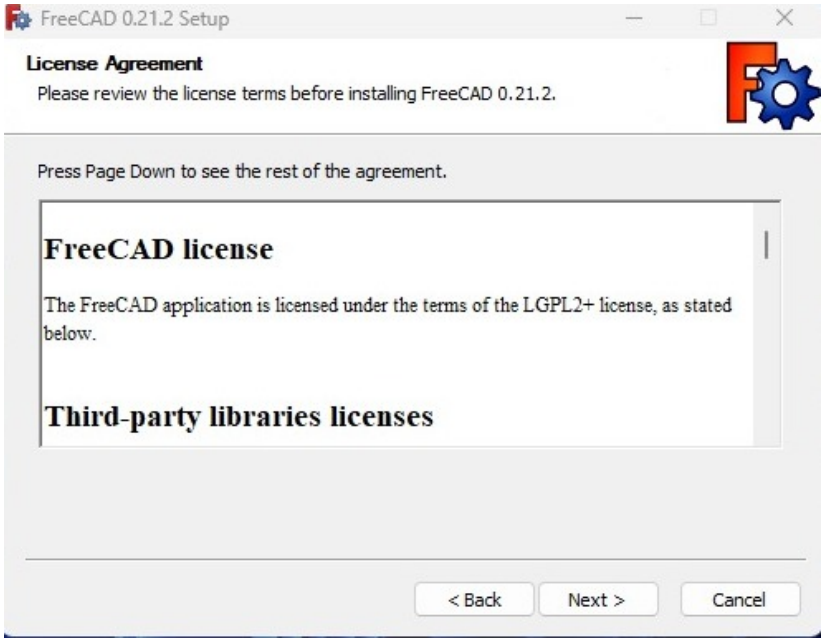
FreeCAD versions

The official stable releases of FreeCAD are available on the referenced download page and within your distribution's software manager. However, FreeCAD's development pace is brisk, with new features and bug fixes incorporated almost daily. Due to the extended periods between stable releases, you may want to experiment with more current, bleeding-edge versions of FreeCAD. These development versions, or pre-releases, can be found at the same download page. For users on Ubuntu or Fedora, the FreeCAD community also provides PPA (Personal Package Archives) and copr 'daily builds', which are regularly updated with the latest developments.

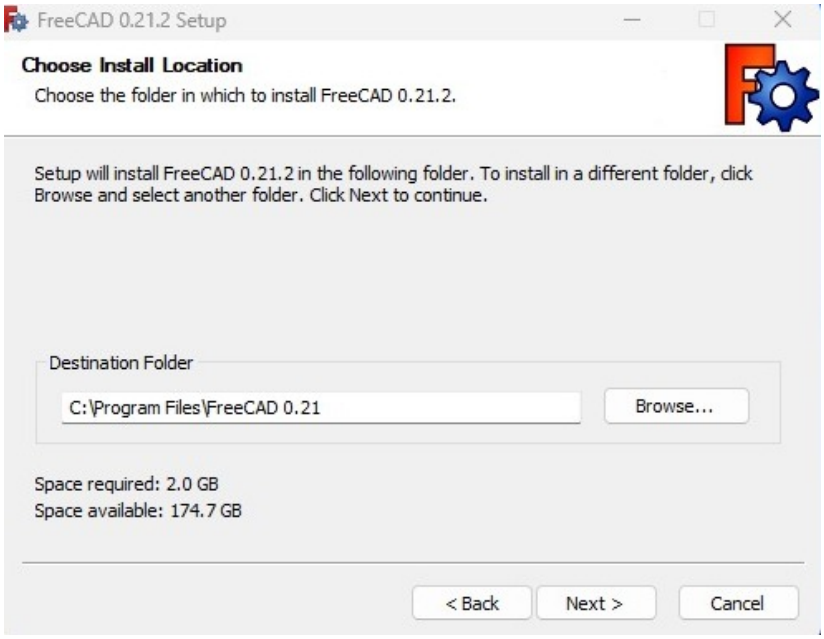
If you plan to install FreeCAD on a virtual machine, be aware that its performance might be significantly impaired, and perhaps unusable, due to limited OpenGL support in many virtual machines.

Installing on Windows[edit | edit source]

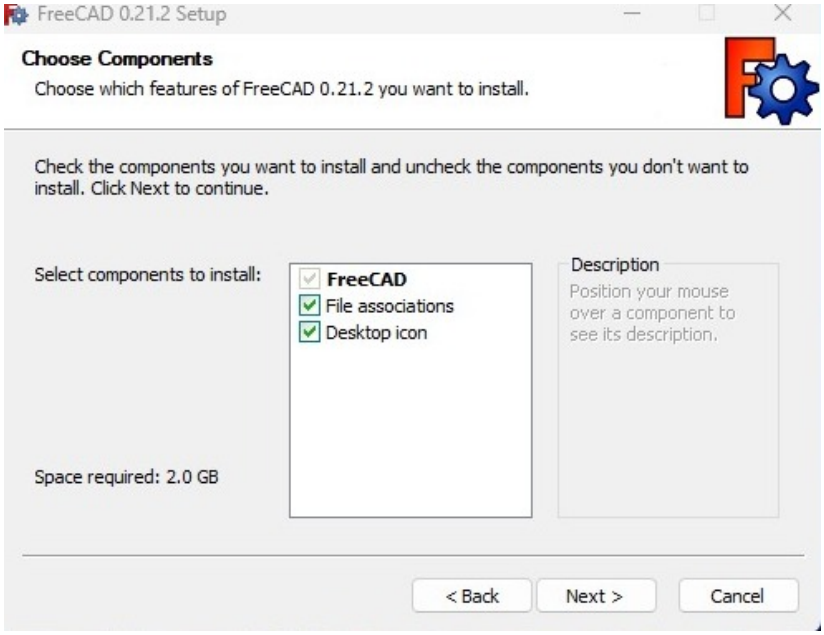
1. Download an installer (.exe) from the download page. The FreeCAD installers should work on any Windows version starting from Windows 7.
2. Accept the terms of the LGPL license; this will be one of the few cases where you can really, safely click the "accept" button without reading the text. No hidden clauses:



3. You can leave the default path here, or change it if you wish:



4. Make sure to check all the components to install:



5. That’s it. The installation is now complete and you can start exploring the capabilities of FreeCAD.

Installing a development version

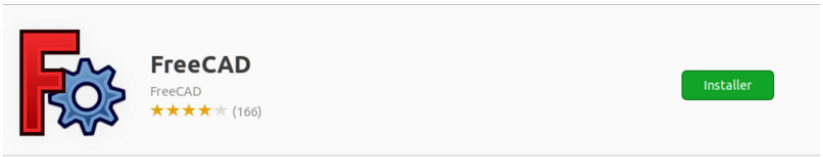
Packaging FreeCAD and developing an installer involves a considerable investment of time and effort. As a result, development versions (also referred to as pre-release versions) are typically delivered in the form of .zip or .7z archives located at the FreeCAD download page. There's no need for a formal installation process with these files; simply extract the contents and start FreeCAD by double-clicking the FreeCAD.exe file located inside. This approach also enables you to maintain both the stable and the "unstable" versions on the same computer. It’s like having both a dependable daily car and an experimental jet pack in your garage!

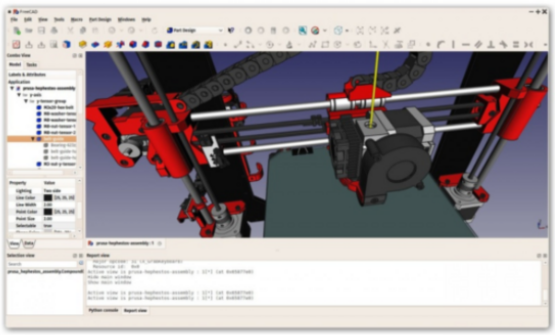
Installing on Linux[edit | edit source]

For users of modern Linux distributions such as Ubuntu, Fedora, openSUSE, Debian, Mint, and Elementary, installing FreeCAD is as simple as a single click. You can seamlessly install it through the software management tool provided by your distribution, though the appearance of these tools may differ from any illustrative images since each distribution employs its own distinct application.

1. Open the software manager and search for "freecad":

2. Click the "install" button and that's it, FreeCAD gets installed. Don't forget to rate it afterwards!





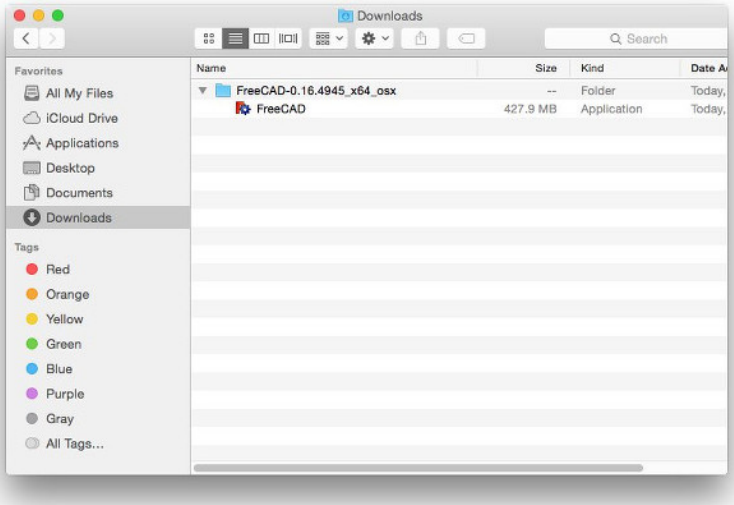
Alternative ways

One of the great pleasures of using Linux is the vast array of options available for customizing your software experience, so don't hold back. For users of Ubuntu and its derivatives, FreeCAD can be installed from a PPA maintained by the FreeCAD community, which includes both stable and development versions. On Fedora, you can access the latest development versions of FreeCAD via copr. Additionally, since FreeCAD is open source, you also have the freedom to compile FreeCAD yourself.

Installing on Mac OS[edit | edit source]

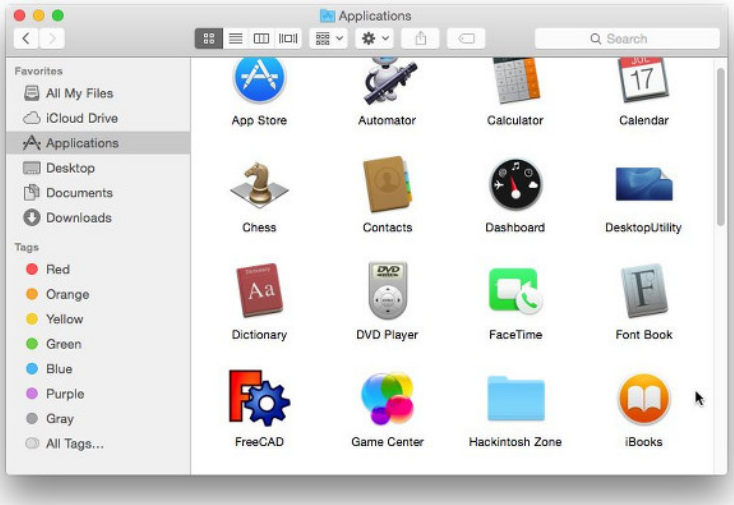
Installing FreeCAD on Mac OSX is nowadays as easy as on other platforms. However, since there are fewer people in the community who own a Mac, the available packages sometimes lag a few versions behind the other platforms.

1. Download a zipped package corresponding to your version.
2. Open the Downloads folder, and expand the downloaded zip file:

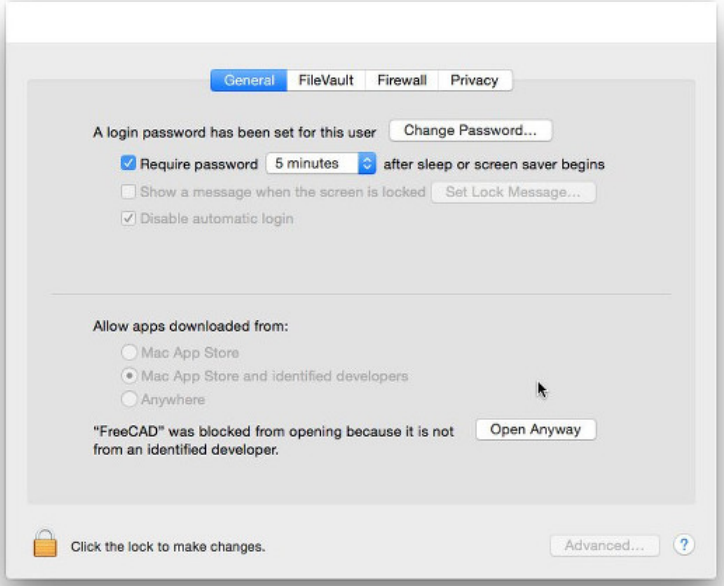


3. Drag the FreeCAD application from inside the zip to the Applications folder:

4. That's it, FreeCAD is installed!



5. If the system prevents FreeCAD from launching due to restricted permissions for applications not coming from the App store, you will need to enable it in the system settings:



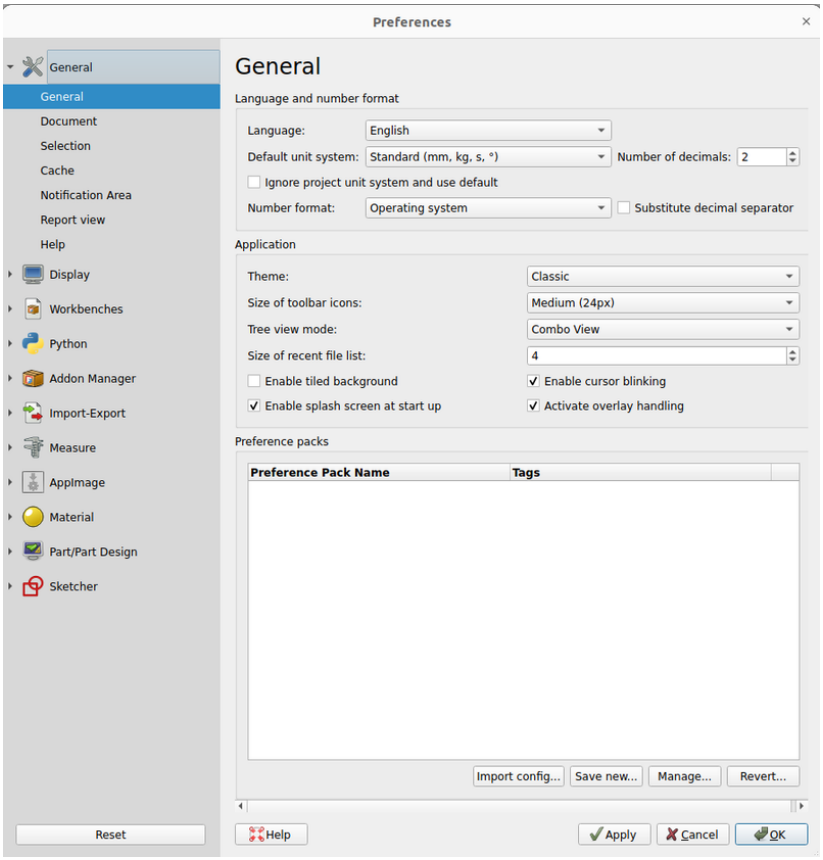
Uninstalling[edit | edit source]

Ideally, you'll never want to part ways with FreeCAD, but should you ever need to uninstall it, rest assured the process is simple. On Windows, use the familiar "remove software" option from the control panel. For Linux users, uninstall it using the same software manager you employed to install it. Mac users have it easiest—just drag FreeCAD from the Applications folder to the trash.

Setting basic preferences[edit | edit source]

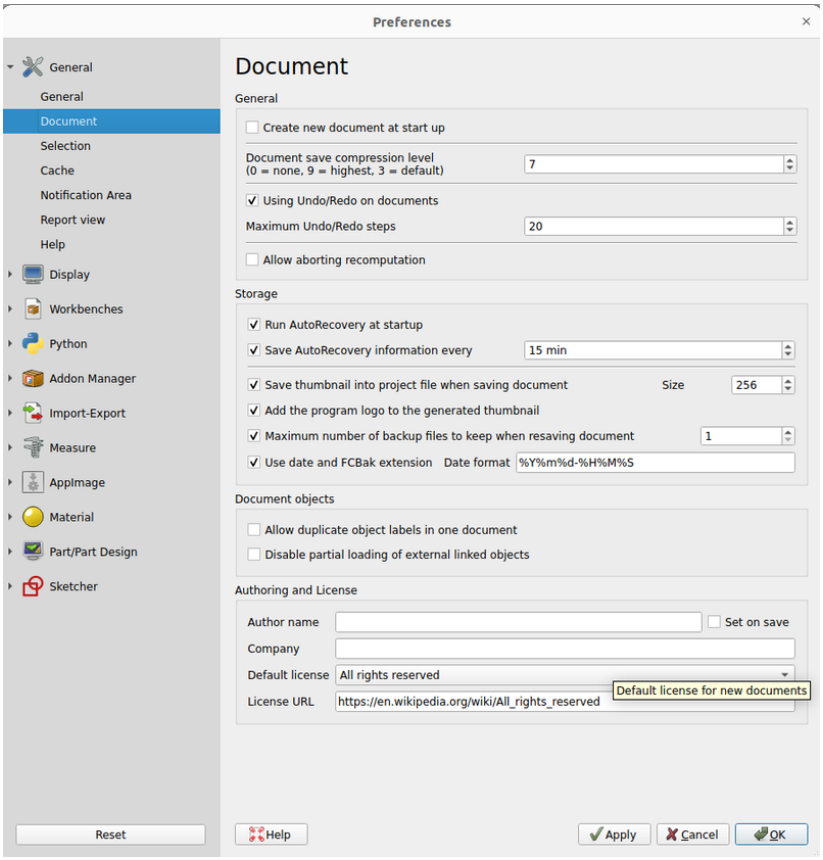
After installing FreeCAD, you'll likely want to personalize it by adjusting some settings. You can find the preference settings in FreeCAD by navigating to **Edit** → **Preferences** in the menu. Below are a few basic settings you might consider modifying right away, but feel free to explore the various pages of preferences to tailor the software to your needs even further.

General category, General tab[edit | edit source]



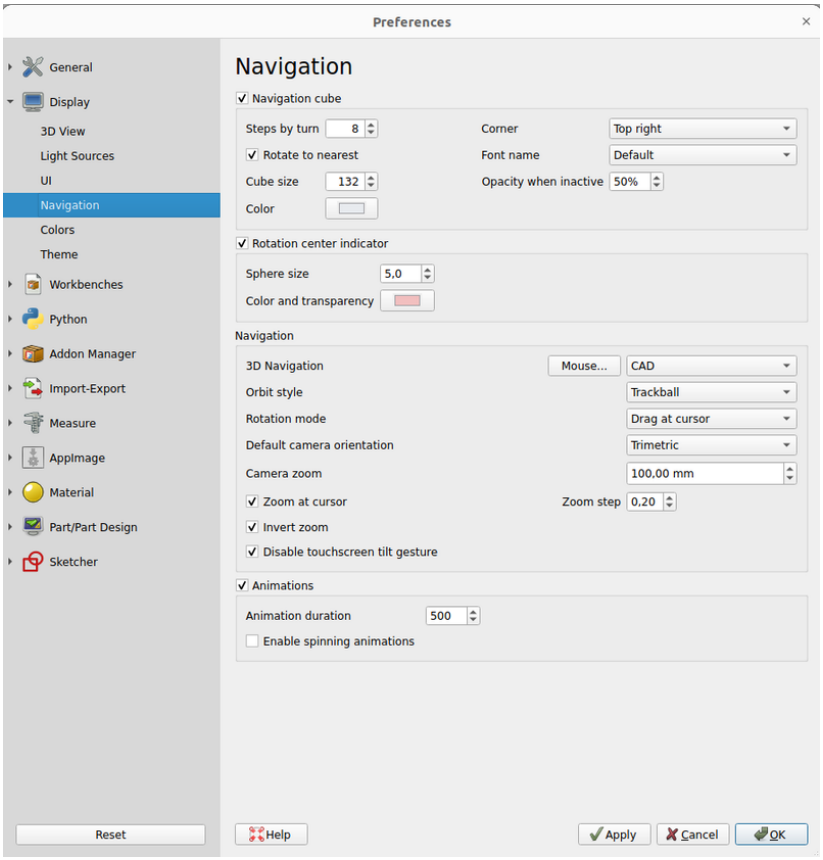
1. **Language**: By default, FreeCAD will select your operating system's language, but you have the option to change it. Thanks to the dedication of many contributors, FreeCAD is available in a wide array of languages.
2. **Units**: This setting allows you to choose the default units system for your projects.

General category, Document tab[edit | edit source]



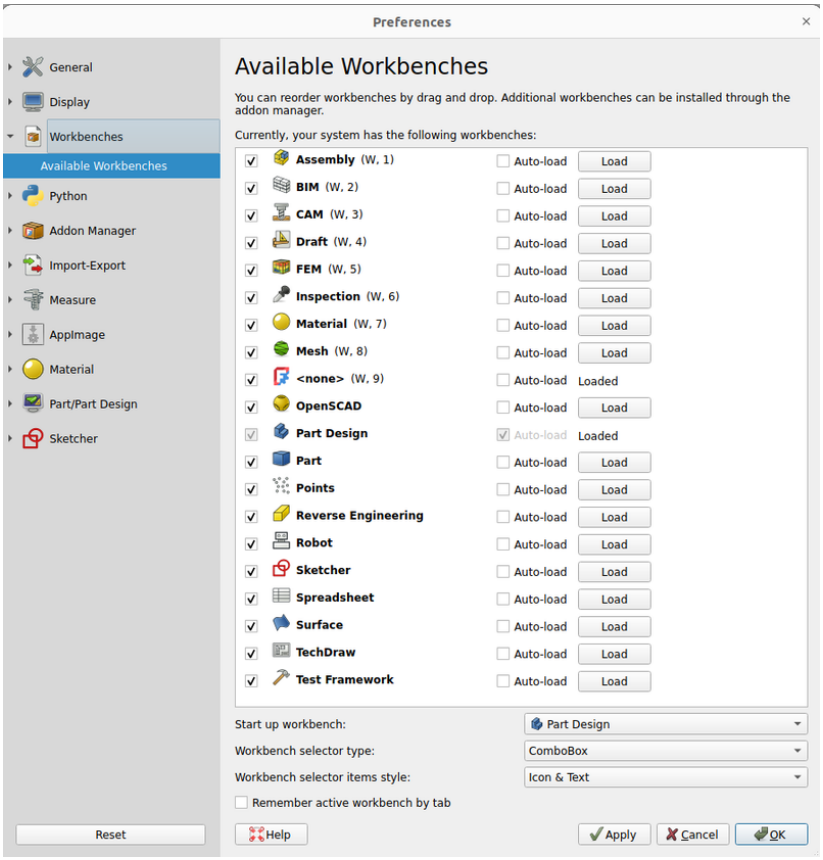
1. **Create a new document at startup**: FreeCAD will automatically open a new document each time the program starts.
2. **Storage options**: Configure settings here to help you recover your work in the event of a crash.
3. '**Authoring and license**': In this area, you can determine the settings for new files. To facilitate sharing, consider starting with a more permissive, copyleft license like Creative Commons.

Display category, Navigation tab[edit | edit source]



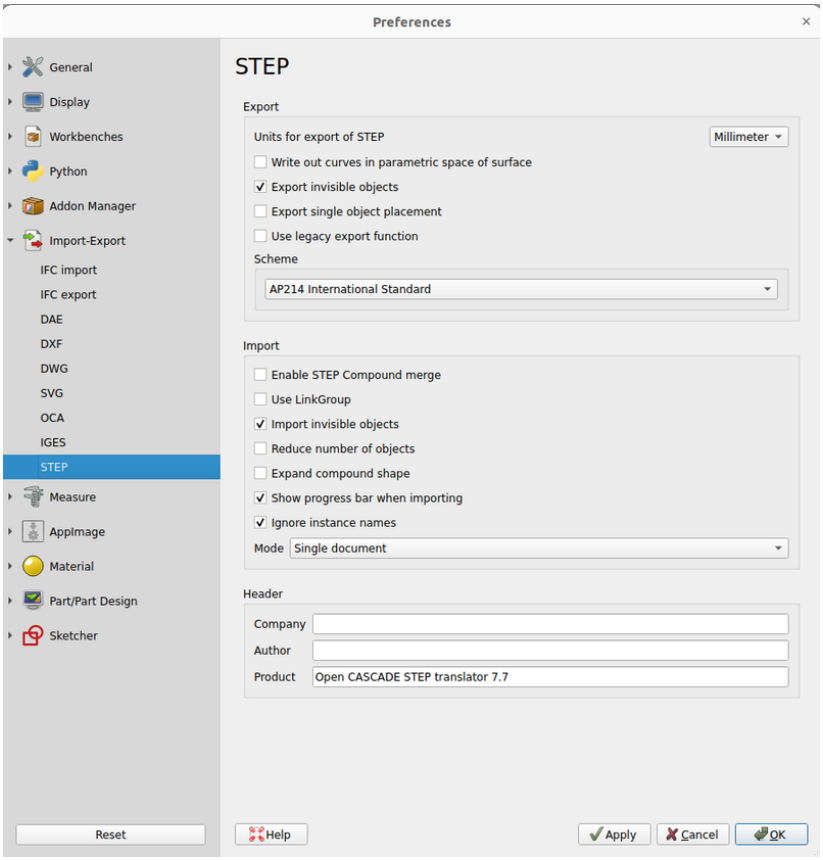
1. **Zoom at cursor**: When enabled, zoom actions center on the mouse cursor. If disabled, zoom focuses on the center of the view.
2. **Invert zoom**: This option reverses the zoom direction in relation to mouse movement.

Workbenches tab[edit | edit source]



Although FreeCAD typically opens to the start page, this setting lets you bypass it. You can start directly in your preferred workbench. Additionally, you can customize which workbenches are displayed in the selector menu.

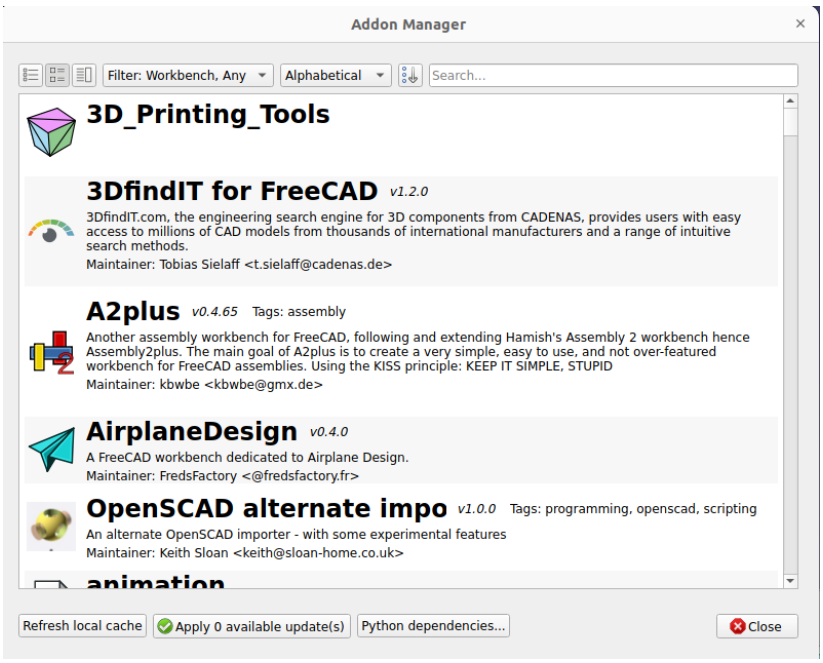
Import-Export tab[edit | edit source]



Here, define basic parameters for importing and exporting in various formats.

Installing additional content[edit | edit source]

As the FreeCAD community expands and the ease of customization grows, numerous external contributions and side projects by community members and enthusiasts start popping up all over the internet. Many of these projects take the form of workbenches or macros, and you can easily add them to your toolbox via the Addon Manager, which is accessible from the Tools menu. The Addon Manager opens up a world of possibilities, allowing you to install various interesting components, such as:



1. A Parts library. This library is a treasure trove of useful models or model fragments crafted by FreeCAD users. All items in this library are freely available for use in your projects and can be accessed directly within your FreeCAD setup.
2. Additional workbenches. These are specialized extensions that enhance FreeCAD's functionality for specific tasks. Example applications include animating model parts or managing specific construction processes like sheet metal folding or BIM (Building Information Modeling). Detailed information about each workbench, including an overview of the tools it contains, is provided on each addon's page, accessible through the corresponding link in the addon manager.
3. A wide range of macros is also available for download. These can significantly streamline your workflow, and detailed documentation on how to use them can be found on the FreeCAD wiki.

As of FreeCAD v0.17.9940, the recommended installation method of any of the above tools is the built-in Addon Manager. However, if for any reason this option is not available, then manual installation is always possible. More information can be found at the FreeCAD addons page

Read more

- More download options
- FreeCAD PPA for Ubuntu
- FreeCAD addons PPA for Ubuntu
- Compile FreeCAD yourself
- FreeCAD translations
- FreeCAD github page
- The FreeCAD addons manager

Introduction

FreeCAD is an open-source, free parametric 3D computer-aided design (CAD) modelling tool used to create objects ranging from simple designs to complex projects consisting of extensive assemblies of numerous parts. Due to its open-source nature, FreeCAD can be freely downloaded, distributed, modified, and used for both personal and commercial purposes. The source code is openly published under the LGPL license. This makes FreeCAD an appealing option not only for relatively inexperienced users, such as hobbyists in the 3D printing world, but also for experienced users looking to experiment with a different parametric tool.

The software was initially released in its infancy in 2002. Since then, thanks to the continuous contributions from a devoted community of developers and users, FreeCAD has grown slowly but steadily. The passionate efforts of numerous contributors have not only improved FreeCAD's capabilities as a CAD tool but have also led to the development of additional functionalities in separate domains closely tied to the base FreeCAD suite. Some notable examples include the FEM workbench, which enables the execution of simple Finite Element Analysis, as well as the BIM workbench which offers Building Information Modeling (BIM) capabilities.

This user manual is being written and updated for version 1.0 of FreeCAD. Its purpose is to serve as a guide for individuals who wish to start using FreeCAD for their projects. No extensive prior knowledge of any CAD software is required, and the contents of this manual are organized in a straightforward and comprehensive manner, divided into various application areas. The guide includes instructions regarding the installation of FreeCAD, the use of its most prominent workbenches, and extends to more complex topics such as Python scripting. Efforts have been made to present the information in an organized, step-by-step fashion, complete with images. This structure ensures that both novice CAD users and more experienced individuals can benefit from studying this manual. The contents of this manual are published under the Creative Commons 4.0 license, and can be freely used, downloaded, copied, and modified. The FreeCAD Manual is continuously improved through user experiences and contributions. A Python script is available to convert the latest version of this manual into PDF or EPUB formats for offline reading.

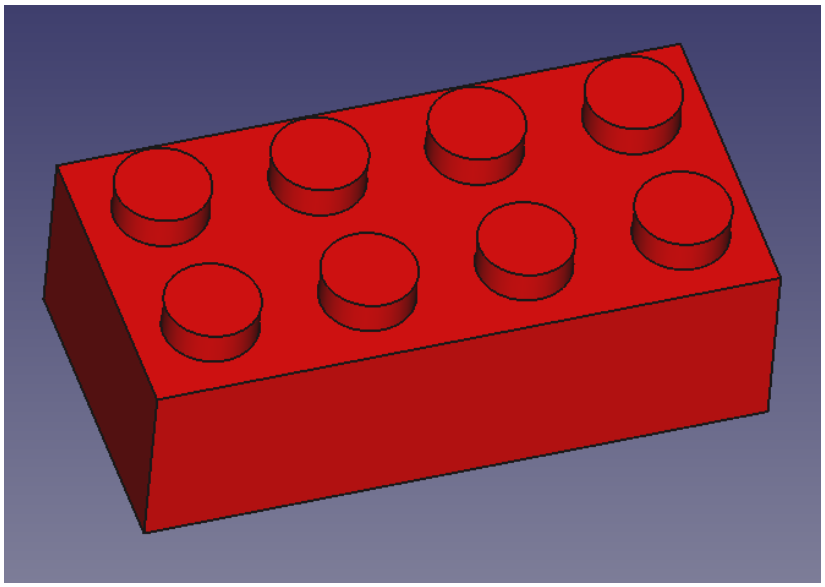
Another valuable resource is the FreeCAD wiki, where numerous contributors have added useful information covering every aspect of the software. Due to the wealth of information available, navigating and using the wiki can be somewhat challenging and overwhelming, especially for new users. For this reason, it is advised to first study the user manual, which offers a more streamlined and smooth introduction, before delving into the more extensive details available in the wiki. Finally, the FreeCAD Forums are also very helpful and provide additional support and insights.

Modeling for product design

The PartDesign Workbench in FreeCAD is a versatile tool for creating parametric 3D models, particularly useful for solid body designs. It allows you to start with 2D sketches, which can then be transformed into 3D objects using operations like padding, revolving, and pocketing. This workbench is essential for designing parts that require precision and parametric control, as changes to sketches or features automatically update the entire model.

One of the key advantages of the PartDesign Workbench is its suitability for creating parts for 3D printing. Since 3D printers require solid, watertight models, the PartDesign Workbench ensures that all features remain within a single coherent body. This eliminates common issues like gaps or overlapping faces, which can cause problems during slicing for 3D printing. Once your design is complete, you can easily export the model as an STL file—a format widely supported by 3D printers. This makes the PartDesign Workbench a go-to option for creating high-quality, printable objects, whether you're prototyping, designing functional parts, or creating intricate models for 3D printing.

To illustrate how the PartDesign Workbench works, let's model this well-known piece of Lego. You can also refer to Basic Part Design Tutorial 019 if you wish to try a different object.



We will now use exclusively the Sketcher and PartDesign tools. Since all the tools from the Sketcher Workbench are also included in the Part Design Workbench, we can stay in Part Design and we will not need to switch back and forth between the two.

In FreeCAD's PartDesign Workbench, objects are primarily built from Sketches, which are 2D profiles made up of linear segments such as lines, arcs, or ellipses, along with a series of constraints. These constraints dictate specific geometric rules for the sketch and can be applied to both the segments themselves and their key points, like endpoints or center points. For example, you can use a vertical constraint on a line to keep it perfectly vertical, or a position (lock) constraint to fix an endpoint in place, preventing it from moving.

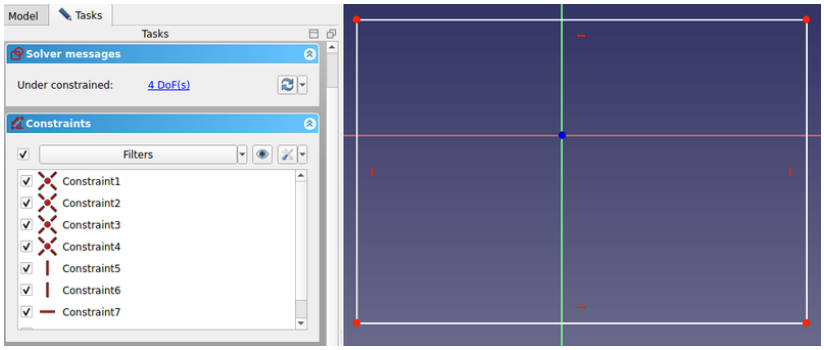
A sketch is considered fully constrained when every point is locked into position by the appropriate number of constraints, meaning no part of the sketch can be moved freely. Achieving a fully constrained sketch is ideal because it ensures the design is well-defined

and stable, allowing for predictable changes later in the design process. On the other hand, if more constraints are added than necessary—referred to as an over-constrained sketch—this can cause conflicts in the geometry. FreeCAD will alert you to any redundant or conflicting constraints, as over-constraining can cause issues in further operations like extrusions or cuts.

Adding the right constraints is key to creating a stable, parametric model. By carefully balancing constraints, you can easily modify or adjust sketches without breaking the geometry. This control makes the Part Design Workbench a powerful tool for precise, parametric modeling, especially for tasks like 3D printing, where maintaining proper geometric relationships is crucial to producing accurate, functional parts.

Sketches have an edit mode, where their geometry and constraints can be changed. When you are done with editing, and leave edit mode, sketches behave like any other FreeCAD object, and can be used as building blocks for all the Part Design tools, but also in other workbenches, such as Part or Arch. The Draft workbench also has a tool that converts Draft objects to Sketches, and vice-versa.

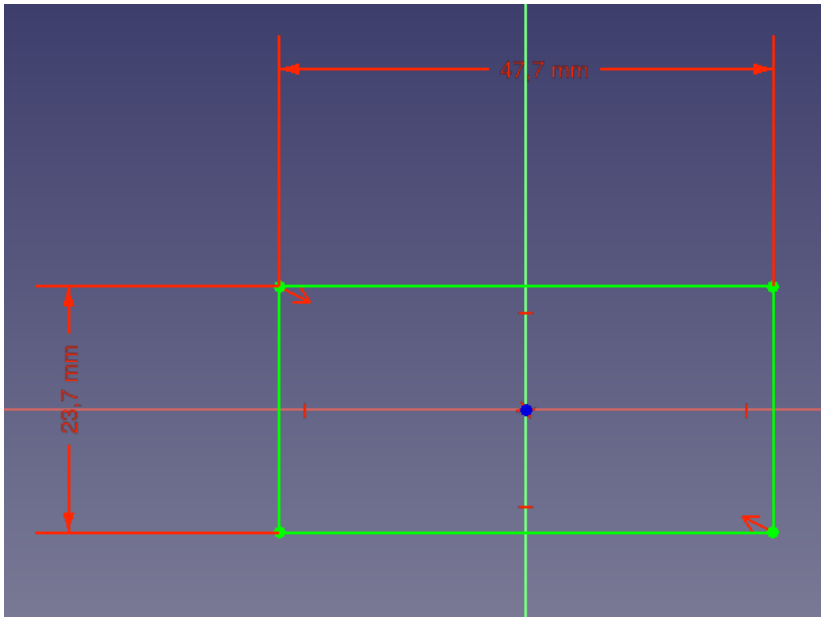
- Switch to the PartDesign Workbench.
- Click on the New Sketch button and choose the **XY** plane, which is the "ground" plane. The sketch will be created and will immediately be switched to edit mode, and the view will be rotated to look at your sketch orthogonally.
- Draw a rectangle by selecting the Rectangle tool and clicking two corner points. You can place the two points anywhere, but do not click on either axes. The correct location of the points will be set in one of the next steps.
- You will notice that a couple of constraints have automatically been added to our rectangle: the vertical segments have received a vertical constraint, the horizontal ones a horizontal constraint, and each corner a point-on-point constraint that glues the segments together. You can experiment with moving the rectangle around by dragging its lines with the mouse, all the geometry will keep obeying the constraints.
- Right now our sketch is under-constrained, as it's missing four constraints: its length, width, and its X and Y positioning. This lack of constraints allows you to freely move the sketch along the X and Y axes. Until these constraints are defined, the geometry is not fully locked in place, meaning both the size and position of the sketch remain adjustable. To fully define the sketch, we need to apply constraints that specify these values and lock the sketch in position.



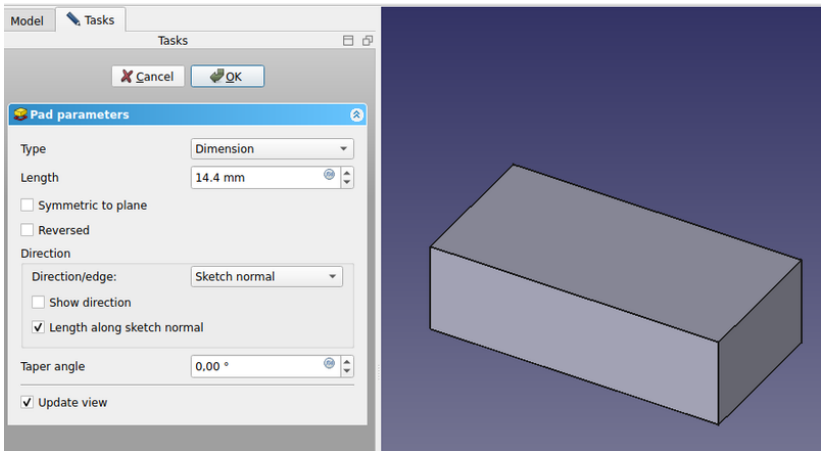
- Now, let's add three more constraints:
  - Activate the automatic dimension tool, select one of the vertical segments, and set its length to 23.7 mm.
  - With the tool still active, select one the horizontal segments, and set its length to 47.7 mm.
  - Activate the symmetry constraint tool, select the upper left corner point of the rectangle, then the lower right corner point, and finally the origin point (the dot where the red and green axes intersect). This ensures the rectangle

stays centered on the origin, limiting its range of movement and providing symmetry across the two axes.

- You will now notice that our rectangle has turned green, indicating that it is fully constrained. This means every aspect of the sketch, including its position, size, and shape, is now fully defined and locked in place. It is generally a good practice to ensure that sketches are fully constrained, as this helps maintain control over your design and prevents unintended changes during further operations.



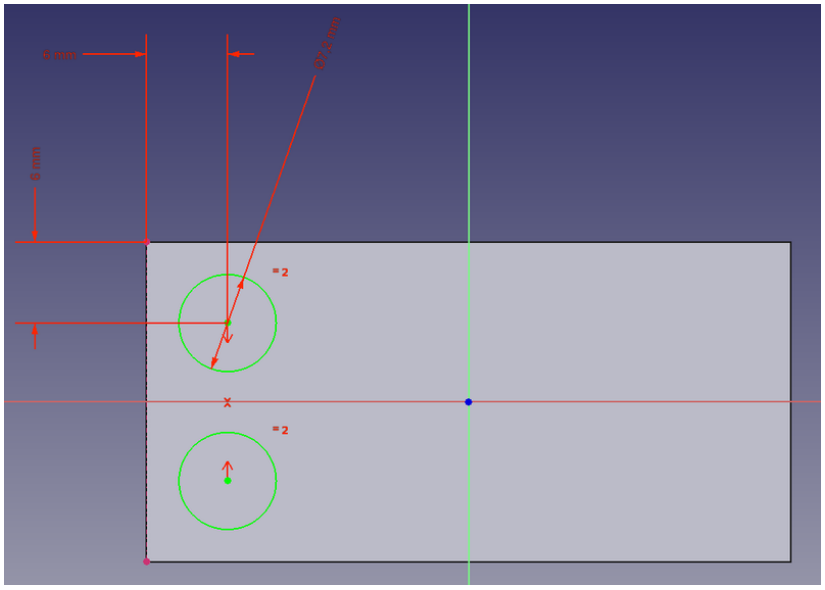
- Our base sketch is now ready, we can leave edit mode by pressing the **Close** button on top of its task panel, or simply by pressing the **Escape** key. If needed later on, we can reenter edit mode anytime by double-clicking the sketch in the tree view, or by right-clicking and pressing on the edit sketch option.
- Let's extrude it by using the Pad tool, and giving it a distance of 14.4 mm. The other options can be left at their default values:



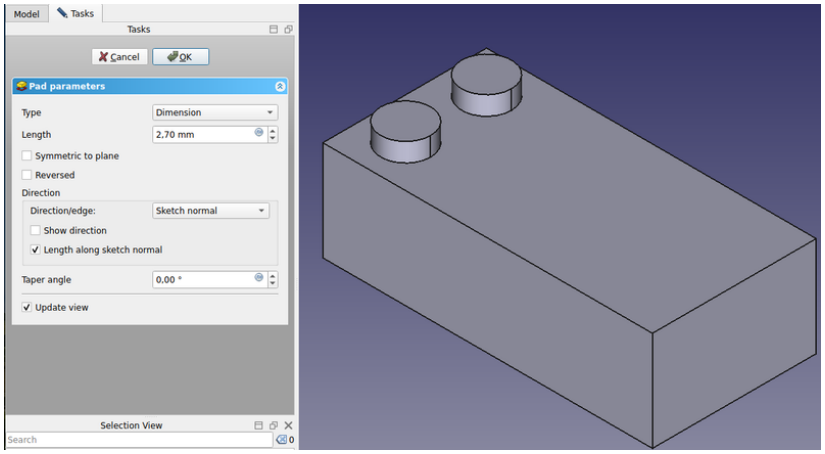
- The **Pad** tool is similar to the Part Extrude from the Part Workbench, but with a key difference: a pad is always linked to its sketch and cannot be moved independently. To reposition the pad, you must move the base sketch, ensuring that the pad remains securely attached. The pad will always remain part of the same body, maintaining the continuity of your design, which is especially useful for complex parts where features need to be built progressively and in alignment with

one another. This adds stability to your design, especially when you want to ensure everything stays properly aligned and fixed in place.

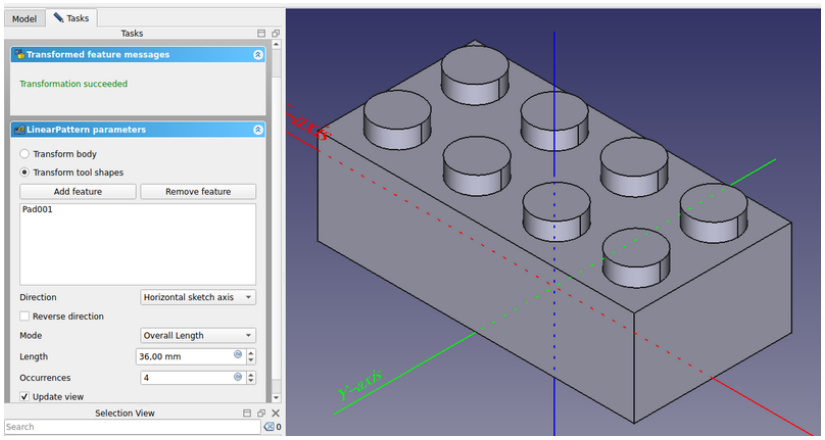
- Let's create the eight cylinders on the top face of the block. First, select the top face of the block and then click on the Align to selection option to align the view to this face. This will provide a clear and direct view, making it easier to accurately place the cylinders.
- Click on the New Sketch button. The new sketch will be created directly on the top face.
- Create two circles anywhere you want.
- Choose the center of both circles and the x axis (red line). Then press on the symmetry constraint option.
- Select the edge of each circle and apply an equal constraint.
- Using the automatic dimension tool, set the diameter of one circle to 7.2 mm. Since we've already constrained both circles to have the same diameter, there's no need to set the diameter for the second circle—it will automatically adjust to match the first one.
- We now need to position the circles relative to the edges of the face. However, you may notice that we are unable to select any points or edges directly. To resolve this, we can use the External geometry tool to reference the face's edges, allowing us to accurately constrain the circles in relation to the face. Press on the button and select the left edge of the face. This edge will now be highlighted in red and you will be able to create reference points from it, enabling you to apply constraints to precisely position the circles relative to the face's boundaries.
- You can now set the X and Y center distances for one of the circles to 6 mm. Since the circles are constrained to each other, the second circle will adjust accordingly.



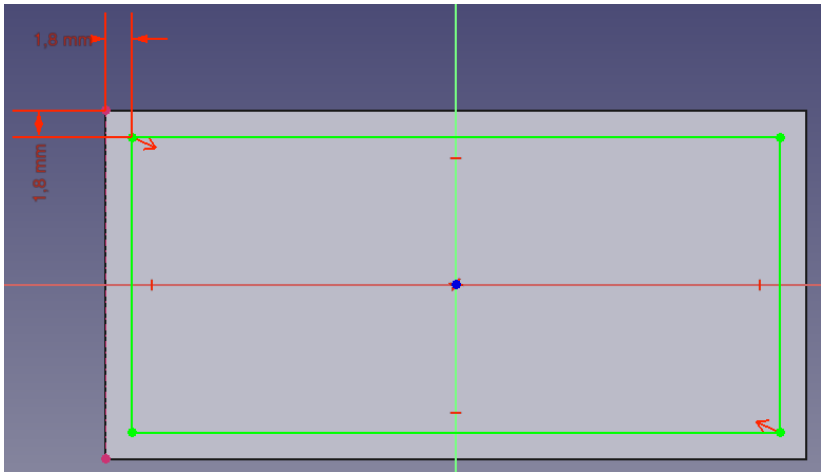
- Notice how, once again, when you lock the position and dimension of everything in your sketch, it becomes fully constrained. This always keeps you on the safe side. You could change the first sketch now, everything we did afterwards would keep tight.
- Leave edit mode, select this new sketch, and create a Pad of 2.7 mm:



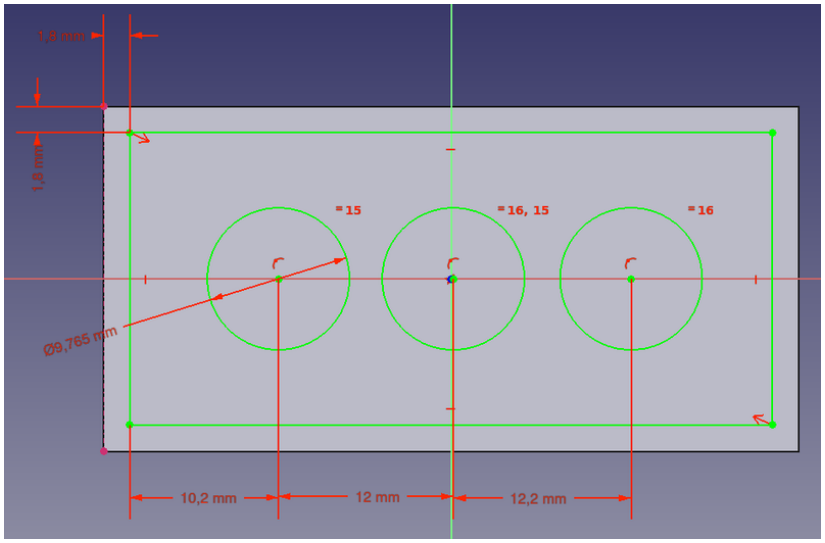
- Since we used the top face of our base block as the foundation for this new sketch, any Part Design operation applied to it will be correctly built on top of the base shape. The two circles are not independent objects; they will be extruded directly from the existing block. This is the key advantage of working in the Part Design Workbench—as long as you ensure each step is built upon the previous one, you are effectively constructing a single, cohesive solid object.
- We can now duplicate our two dots four times. Select the latest Pad we just created.
- Press the Linear pattern button.
- Give it a length of 36 mm (which is the total "span" we want our copies to fit in), in the "horizontal sketch axis" direction, and make it 4 occurrences:



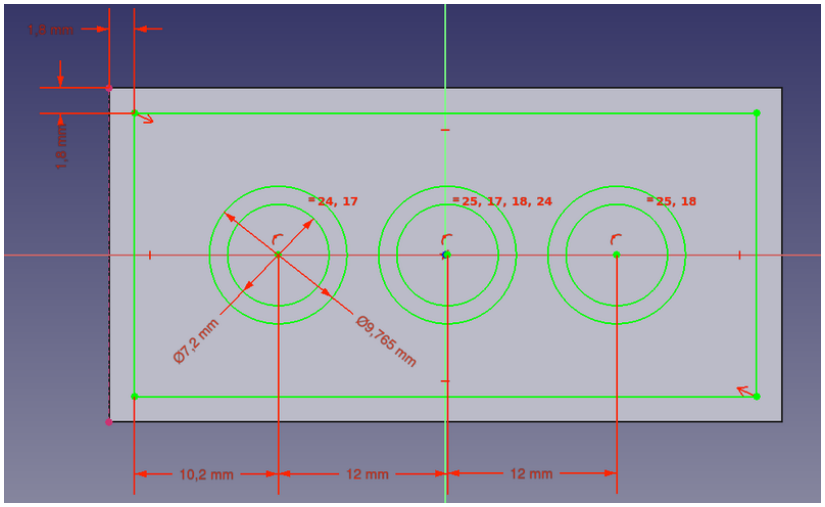
- We will now carve the inside of the block, using the Pocket tool, which is the PartDesign version of Part Cut. To make a pocket, we will create a sketch on the bottom face of our block, which will be used to remove a part of the block.
- With the bottom face selected, press the New Sketch button.
- Draw a Rectangle on the face.
- Apply a symmetry constraint by selecting the upper left corner point of the rectangle, then the lower right corner point, and finally the origin point (the dot where the red and green axes intersect).
- By using the External geometry tool choose the left edge of the bottom face. Notice again that it will be highlighted in red.
- Select the upper endpoint of the left edge of the bottom face, and the upper left corner of the rectangle. Set the horizontal and vertical distances between these points to 1.8 mm by using the automatic dimension constraint.



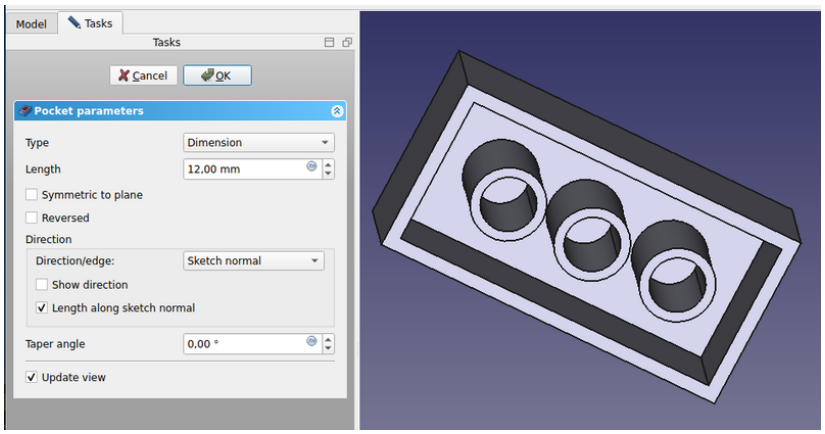
- Create a circle by clicking on the X-axis (red line) and then clicking at another point away from the X axis. This will automatically constrain the center of the circle to lie on the X axis. Repeat this step to create two more circles with centers on the X axis.
- Select the edges of all three circles, and apply an = equal constraint.
- Set the diameter of one circle to 9.765 mm.
- Set the distance between the center of the left circle and the left edge of the rectangle we created before to 10.2 mm.
- Set the distance between the left and middle circles to 12 mm. Repeat this step to set the same 12mm distance between the middle and right circles



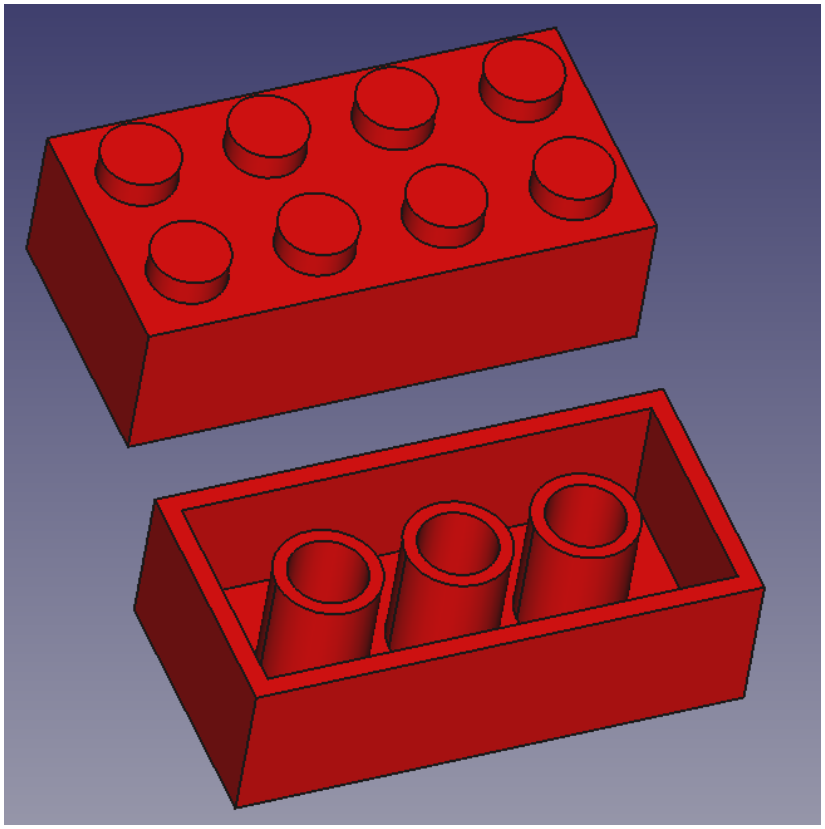
- We are almost done.
- Create three additional circles, ensuring that each new circle is concentric with one of the previously drawn circles. Alternatively, you can place the new circles anywhere in the sketch and use the coincident constraint tool to align their centers with the centers of the existing circles.
- By choosing all three circles apply an = equal constraint.
- Set the diameter of one circle to 7.2 mm.
- You can now exit the sketch.



- Select the completed sketch, and use the Pocket tool with a length of 12 mm.



- This is it. Our brick is ready. If you wish to change its color, you can do so by going to the **View tab**.



You may notice that the modeling history in the tree view has become quite extensive. This is incredibly valuable, as it allows us to revisit and modify any step in the design process at any time. For instance, adapting this model to create a 2x2 brick instead of a 2x4 would be a breeze—just a few adjustments to the dimensions and pattern occurrences would do the trick. This same flexibility allows us to design larger, custom pieces that aren't part of the original Lego set. The parametric nature of FreeCAD makes it easy to modify existing models, giving us full control to adapt or expand the design as needed.

But we could also want to get rid of the history, for example, if we are going to model a castle with this brick, and we don't want to have this whole history repeated 500 times in our file.

There are two simple ways to get rid of the history, one is using the Create simple copy tool from the Part Workbench, which will create a copy of our piece that doesn't depend anymore on the history (you can delete the whole history afterwards), the other way is exporting the piece as a STEP file and re-importing it.

Downloads

- The model produced during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/lego.FCStd

Read more

- The Sketcher
- The Part Design Workbench
- The Assembly2 Workbench

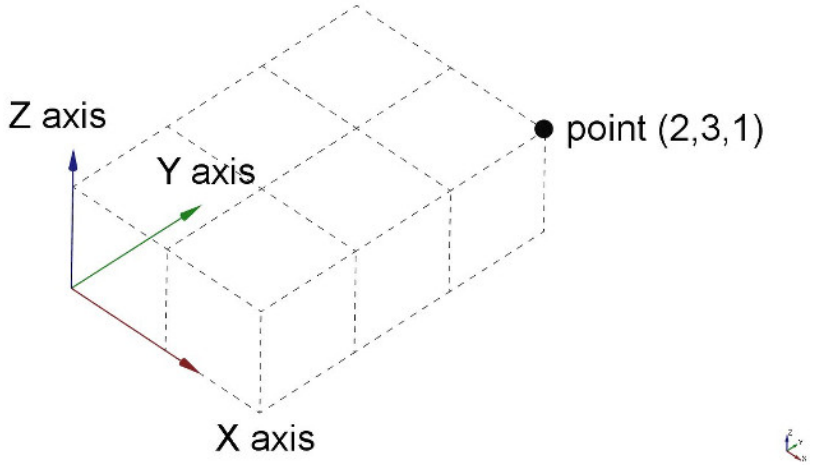
Navigating in the 3D View

A word about the 3D space[edit | edit source]

If you are new to 3D modeling applications, it's essential to understand some fundamental concepts before proceeding. If you already have experience with such software, you may skip this introduction.

FreeCAD operates within a three-dimensional Euclidean space which centers around an origin point defined by three axes: X, Y, and Z. Typically, when viewing from above in FreeCAD, the X axis extends horizontally to the right, the Y axis extends towards the back, and the Z axis moves vertically upwards. The intersection point of these axes, where each coordinate is zero, is known as the origin.

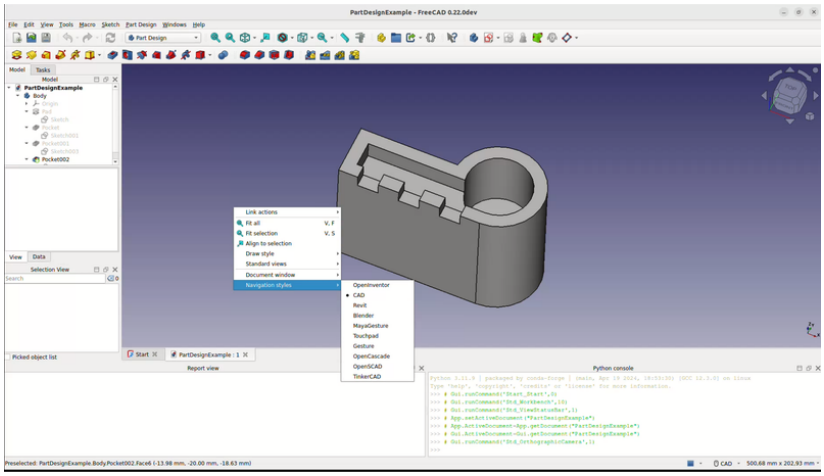
Every location in FreeCAD's space is determined by (x, y, z) coordinates. For instance, a point located at coordinates (2,3,1) is positioned 2 units along the X axis, 3 units along the Y axis, and 1 unit along the Z axis. Navigating this space is akin to manipulating a camera. You can move the camera left, right, up, or down (panning), swivel it around the focal point (rotating), or move it closer to or away from objects (zooming), allowing you to explore your design from various perspectives.



The FreeCAD 3D View[edit | edit source]

Mouse navigation[edit | edit source]

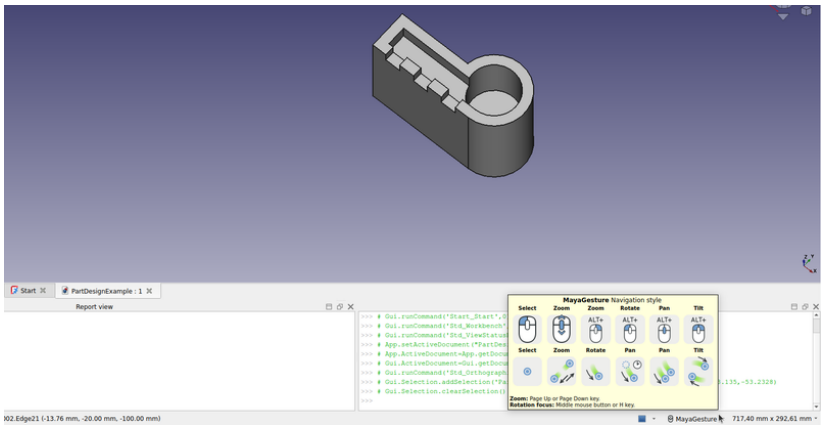
Navigating the 3D View in FreeCAD can be achieved using various inputs including a mouse, a Space Navigator device, keyboard shortcuts, a touchpad, or any combination thereof. FreeCAD offers a range of navigation styles that define how the three fundamental viewing operations—pan, rotate, and zoom—are executed. Additionally, these styles govern how objects are selected within the workspace. To access and switch between these navigation styles, you can navigate to the Preferences Editor or simply right-click in the 3D View. Additionally, there is a third, more immediate option for altering the navigation style directly from the user interface located in the lower-right part of the screen.



Each of these styles allocates different mouse buttons, or mouse + keyboard combinations, or mouse gestures, to these four operations. The following table shows the principal available styles. Keyboard keys and mouse buttons indicated in magenta must be held down.

Style	Select	Zoom	Rotate	Pan
Blender				or Shift +
CAD **(default)**		or Ctrl + Shift +	or or Shift +	or Ctrl +
Gesture				
Maya-Gesture		or Alt +	Alt +	Alt +
OpenCascade		or Ctr +	or Ctr +	or Ctr +
OpenInventor	Shift +	or		
OpenSCAD		or or Shift +	or	
Revit			or Shift +	or
Siemens NX introduced in 1.1		or		or Shift +
SolidWorks introduced in 1.1		or Shift +		Ctrl +
TinkerCAD				
Touchpad		Ctrl + Shift +	Alt + or Shift +	Shift +

It's worth noting that when a user hovers over the navigation styles menu located at the bottom right of the screen, a tooltip will appear. This tooltip provides a brief description of the highlighted navigation style, offering immediate guidance on its use.



Keyboard navigation[edit | edit source]

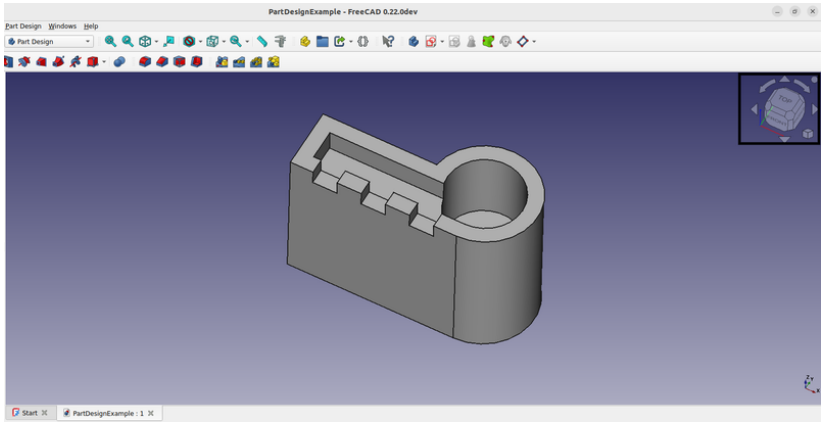
Alternatively, some keyboard controls are always available, no matter the navigation style:

- Ctrl + + and Ctrl + − or PgUp and PgDn to zoom in and out, respectively.
- The arrow keys, ◄ ► ▲ ▼, to pan the view left/right and up/down.
- Shift + ◄ and Shift + ► to rotate the view by 90 degrees.
- The numeric keys, 0 1 2 3 4 5 6, for the seven standard views: Isometric, Front, Top, Right, Rear, Bottom, and Left
- V O will set the camera in Orthographic View.
- While V P sets it in Perspective View.
- Ctrl will allow you to select more than one object or element

These controls are also available from the View menu and some from the View toolbar.

Using the Navigation Cube[edit | edit source]

In the default setup, there is a Navigation Cube in the upper right corner of the 3D View. This may be used to change the view.



Clicking on a face of the cube will switch the view to that face. Clicking one of the four triangular arrows rotates the view 45 degrees in the indicated direction. Clicking one of the two curved arrows rotates the view 45 degrees in the indicated direction around a line pointing towards you. Clicking the round button in the top right corner of the cube rotates the view 180 degrees around the vertical axis of the view.

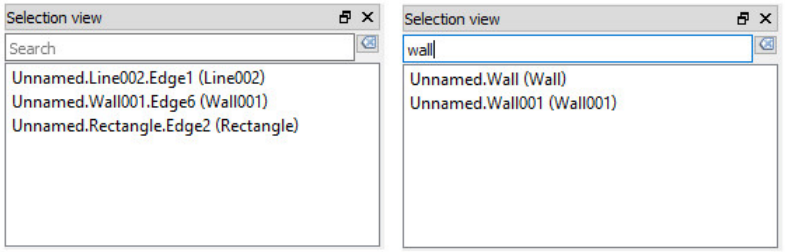
There is a smaller mini-cube in the lower right of the Navigation Cube which activates a drop-down menu with several options, including an option to make the cube movable, which allows to temporarily move the cube to a different position by dragging.

Selecting objects[edit | edit source]

Objects in the 3D View can be selected by clicking them with the corresponding mouse button, depending on the navigation style (described above).

- A single click will select the object and one of its subelements (edge, face, vertex).
- Double-clicking will select the object and all its subelements.
- You can select more than one object and more than one subelement, from the same or different objects, by pressing the CTRL key.
- Clicking with the selection button on an empty portion of the 3D View will deselect everything.

A panel called Selection View, available from the View menu, can also be turned on, which shows you what is currently selected:



You can also use the Selection View to select objects by searching for a particular object.

Read more

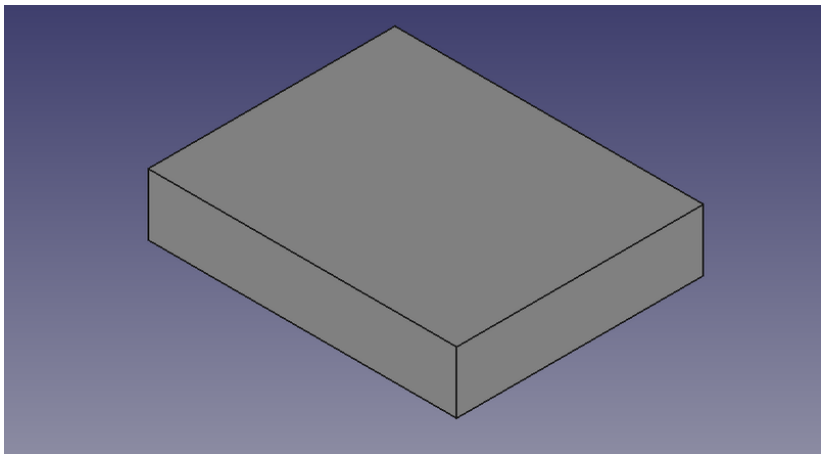
- FreeCAD navigation styles
- Navigation Cube

Parametric objects

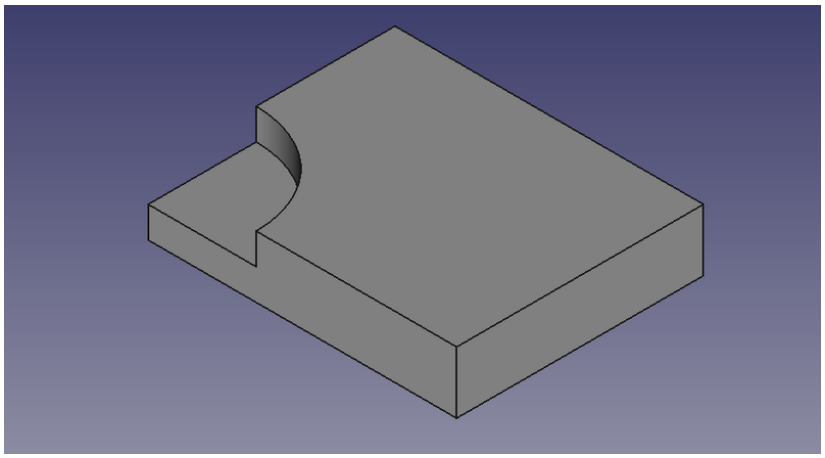
FreeCAD employs a parametric modeling approach, where the geometry of objects is governed by underlying rules and parameters rather than being freely sculpted. This means each component's dimensions and characteristics are defined by parameters, which instruct the program on how to generate the geometry. For instance, to create a cylinder, parameters like radius and height are specified. With these values, FreeCAD generates the precise geometrical shape.

In FreeCAD, parametric objects are essentially small, programmable scripts that execute when any parameter is altered. These parameters can vary widely, including integers and floating-point numbers, real-world dimensional values such as millimeters, meters, or feet, coordinates (expressed as x, y, z), text strings, or even references to other objects. Such versatility in parameters allows the creation of complex models through a series of chained operations where each new object derives its characteristics from a previous one, while also introducing additional attributes.

For example, consider creating a solid cubic object through parametric modeling. You start with a basic 2D rectangular shape labeled as 'plate.' of length l and width w. This sketch defines the base of your cubic object. Next, you define an 'Extrude' operation, or 'Pad,' specifying the distance to push or pull the sketch into a 3D object. This results in a solid cubic form based on the Sketch's shape and the extrusion distance specified.



On the top face of the plate you sketch a circle of a given diameter d. You then use this circle to create a pocket (remove material) from the original plate.



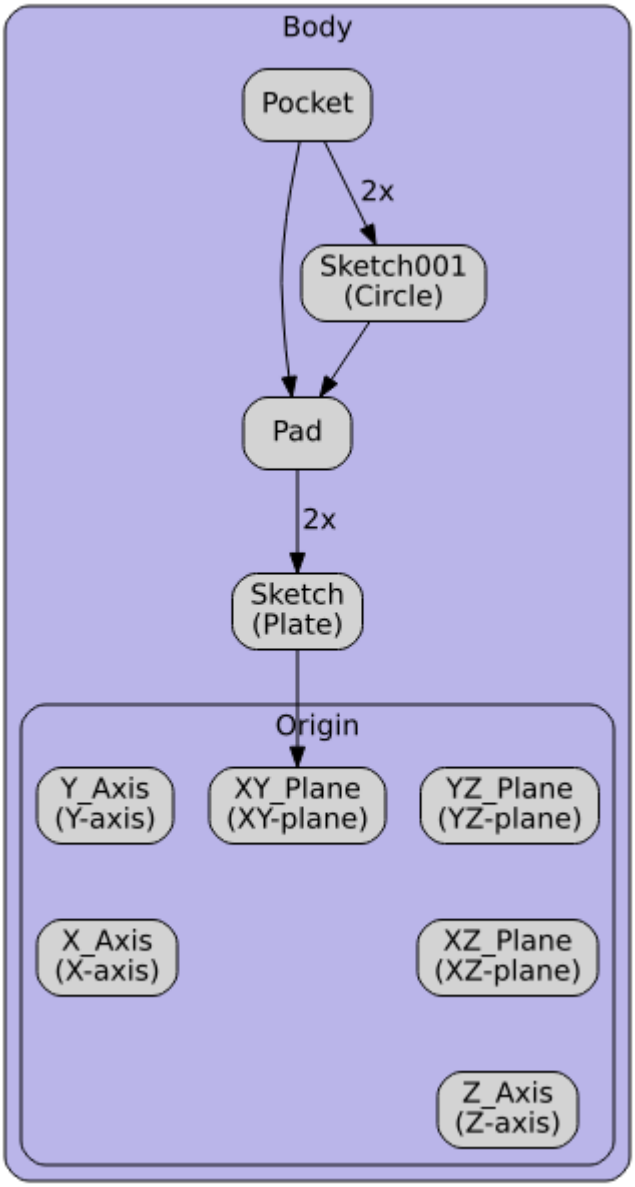
If you decide to change either of the dimensions of the plate, or of the circle, the final object would be also modified. Thanks to the use of a parametric design approach, there is no need to redo the object from the beginning.

1. Recomputation is not always automatic. Heavy operations, that might modify a big portion of your document, and therefore take some time, are not performed automatically. Instead, the object (and all the objects that depend on it) will be marked for recomputation (a small blue icon appears on them in the tree view). You must then press the recompute button (or **Edit->Refresh**) to have all the marked objects recomputed.
2. The dependency tree must always flow in the same direction. Loops are forbidden. (See DAG, and DAG view) You can have object A which depends on object B which depends on object C. But you cannot have object A which depends on object B which depends on object A. That would be a circular dependency. However, you can have many objects that depend on the same object, for example, objects B and C both depend on A. Menu **Tools -> Dependency graph** shows you a dependency diagram like on the image above. It can be useful to detect problems.

In FreeCAD's parametric modeling process, examining the dependency tree of an object provides a clear insight into the sequential build and relationships within a model. At the foundation of the structure in the above example is the 'Plate Sketch,' which serves as the base for the initial form of the model. A 'Pad' operation is then applied, which adds material to this foundational sketch, effectively creating a three-dimensional structure from the two-dimensional base.

Following this, a 'Circle Sketch' is drafted on the newly formed surface. This circle forms the basis for the subsequent 'Pocket' operation. The pocket operation strategically removes material from the structure, essentially carving out a portion based on the circle sketch. This process of adding and then subtracting material allows for complex geometries and features to be integrated into the basic model seamlessly.

Through this sequence of operations—starting from the base sketch, adding volume with a pad, and creating detailed features with additional sketches and pockets—the final object takes shape. Each step in this chain depends on its predecessor, illustrating the interconnected nature of parametric design in FreeCAD.



Not all objects are parametric in FreeCAD. Often, the geometry that you import from other files won't contain any parameters and will be simple, non-parametric objects. However, these can often be used as a base, or starting point for newly created parametric objects, depending, of course, on what the parametric object requires and the quality of the imported geometry.

All objects, however, parametric or not, will have a couple of basic parameters, such as a Name, which is unique in the document and cannot be edited, a Label, which is a user-defined name that can be edited, and a placement, which holds its position in the 3D space.

Finally, it is worth noting that custom parametric objects are easy to program in Python.

Read more

- The properties editor
- How to program parametric objects
- Positioning objects in FreeCAD
- Enabling the dependency graph

Preparing models for 3D printing

One of the primary purposes of FreeCAD is to design objects that can be turned into real-world, physical products. These designs can be shared with others for manufacturing or, increasingly, exported directly to 3D printers or a CNC mill &CNC machines for automated fabrication. With FreeCAD, you can create precise, detailed models that are ready for various production methods. This chapter will guide you through the process of preparing your models for these machines, ensuring they meet the necessary specifications for successful manufacturing, whether you're working with a team or handling the entire process yourself.

If you've been careful while modeling, most of the challenges associated with 3D printing your model should already be minimized. The key aspects to focus on include:

- **Ensuring Your Objects Are Solid**: Just like real-world objects, your 3D models must be solid. FreeCAD, especially within the PartDesign Workbench, helps you ensure that your models remain solid throughout the design process. The software will notify you if an operation compromises the solid nature of the object. Additionally, the Part Workbench offers a  Check Geometry tool, which allows you to identify potential defects or issues that might interfere with the 3D printing process.

- **Confirming the Accuracy of Dimensions**: Precision is critical—what you design in FreeCAD will translate directly to real-world measurements. A millimeter in FreeCAD is a millimeter in the physical object, so each dimension must be carefully considered and verified to ensure accuracy.

- **Managing Degradation**: It's important to remember that no 3D printer or CNC mill can directly process FreeCAD files. These machines use G-Code, a machine language with various dialects depending on the machine or vendor. The process of converting your model into G-Code can often be done automatically through slicer software, but you also have the option to do it manually for greater control. However, during this conversion, some loss of detail or quality is inevitable, particularly when converting the model to a mesh format for printing. You must ensure that this degradation remains within acceptable limits and doesn’t affect the functionality or appearance of your final object.

- **Export Format Compatibility**: For 3D printing, STL is the most commonly used format, but it inherently converts your model into a mesh of triangles, which can result in some loss of detail. It’s essential to choose the right resolution when exporting to STL, balancing between detail retention and file size. Similarly, for CNC machining, formats like STEP or IGES are preferable as they maintain the original geometric integrity of the design better than STL. Choosing the right format ensures that the conversion to G-Code remains accurate.

- **Mesh Analysis and Calibration**: Before exporting your model to a slicer or CNC toolpath generator, it’s advisable to run a mesh analysis using FreeCAD’s Mesh Workbench to detect irregularities, non-manifold edges, or other mesh issues that might complicate the manufacturing process. Additionally, even with a perfect model, make sure your 3D printer or CNC machine is properly calibrated (e.g., for bed leveling, stepper motor settings, or extruder configuration) to avoid quality problems in the final product.

In the following sections, we'll assume that you've already taken care of creating solid models with the correct dimensions. Our focus will now shift to managing the conversion

process to G-Code, ensuring that your model maintains the necessary quality for 3D printing or CNC machining. By addressing these considerations, you'll be better equipped to produce successful physical objects directly from your FreeCAD models.

Exporting to slicers[edit | edit source]

The most common technique for preparing a 3D model for printing involves exporting the 3D object from FreeCAD to a specialized software known as a slicer. The slicer generates G-code by slicing the model into thin layers, which the 3D printer will follow to build the object layer by layer. Because many 3D printers—especially home-built or hobbyist models—have unique configurations, slicer programs provide a wide range of advanced settings. These settings allow you to customize key parameters, such as layer height, print speed, infill density, and support structures, ensuring the G-code is tailored to the specific features and capabilities of your printer.

Many slicers also offer simulation and print validation features that are invaluable for previewing the print process. You can visualize the toolpath for each layer, which helps detect potential issues like overhangs that may require supports or areas where cooling might be insufficient. This pre-print validation ensures your model is properly prepared before the print begins, avoiding failed prints or wasted material.

Slicers often include additional insights, such as estimating print time, material usage, and cost based on the filament or resin being used. This allows you to make informed decisions about the printing process and tweak settings for efficiency or material conservation. Although the deeper intricacies of 3D printing—such as machine calibration, material selection, and post-processing—are beyond the scope of this guide, we will focus on how to properly export your FreeCAD model and use slicer software to ensure the output is correct and optimized for your specific printer

Converting objects to meshes[edit | edit source]

None of the slicers currently available can directly accept the solid geometry produced in FreeCAD. Slicers like Cura and PrusaSlicer work with mesh-based formats such as STL, OBJ, or 3MF, which represent the object's surface geometry using a network of triangles. Therefore, to use a model created in FreeCAD, it must first be converted into a mesh format that these slicers can interpret.

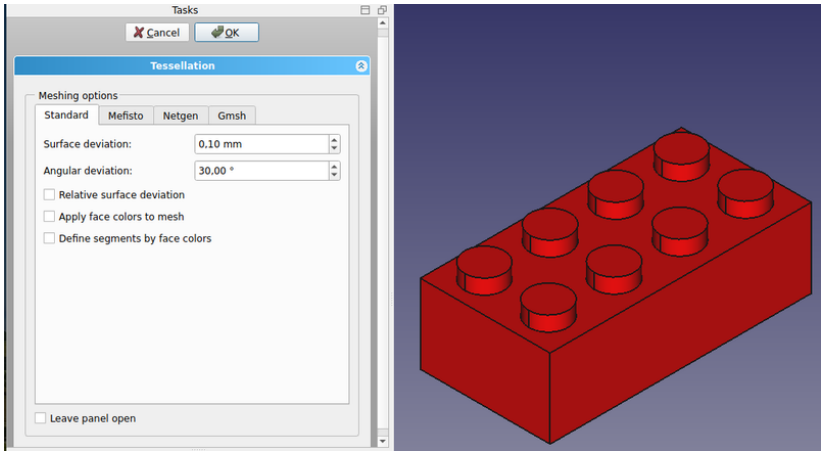
The most commonly used format for 3D printing is STL. One reason STL is preferred is its simplicity—it represents the 3D geometry as a mesh of triangles without including complex details like colors, materials, or textures. This minimalistic approach ensures that STL files are lightweight and compatible with virtually all slicers and 3D printers, making it the industry standard. While OBJ and 3MF are also supported, they can carry additional information like textures and materials, which is unnecessary for most 3D printing tasks and can complicate the slicing process.

Fortunately, converting a solid object to a mesh in FreeCAD is straightforward, even though converting a mesh back into a solid is a more complicated operation. During the conversion process, it is crucial to keep in mind that some degradation of the model's quality may occur, especially when reducing complex geometry to a simple triangular mesh. You must ensure that this degradation remains within acceptable limits to maintain the accuracy of your printed object.

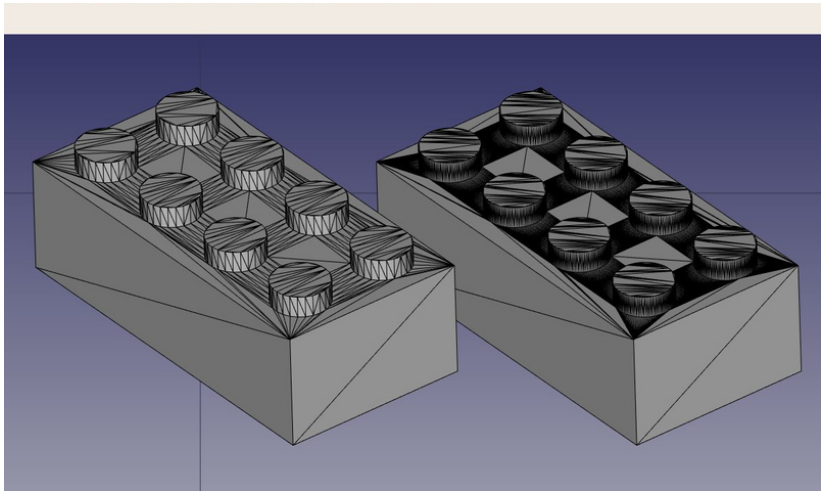
In FreeCAD, the Mesh Workbench handles all mesh-related tasks. This workbench contains tools not only for converting between Part and Mesh objects but also for analyzing and repairing meshes. While mesh manipulation isn't the primary focus of FreeCAD, it becomes essential when preparing models for 3D printing. Mesh objects are widely used in other applications, and the Mesh Workbench allows you to fully manage

and adjust these objects, ensuring they are ready for the next step in the printing process.

- Let's convert the Lego piece we created in the last chapter into an STL mesh. The geometry can be downloaded at the end of said chapter.
- Open the FreeCAD file containing the Lego piece.
- Switch to the Mesh Workbench
- Select the Lego brick
- Select menu **Meshes → Create Mesh from Shape**
- A task panel will open with several options. Some additional meshing algorithms (Mefisto or Netgen) might not be available, depending on how your version of FreeCAD was compiled. The Standard meshing algorithm will always be present. It offers fewer possibilities than the two others, but is totally sufficient for small objects that fit into the maximum print size of a 3D printer.



- Select the **Standard** mesher, and leave the deviation value to the default value of **0.10**. Press **Ok**.
- A mesh object will be created, exactly on top of our solid object. Either hide the solid or move one of the objects aside, so you can compare both.
- Change the **View → Display Mode** property of the new mesh object to **Flat Lines**, in order to see how the triangulation occurred.
- If you are not happy, and think that the result is too coarse, you can repeat the operation, lowering the deviation value. In the example below, the left mesh used the default value of **0.10**, while the right one uses **0.01**:



In most cases, though, the default values will give a satisfying result.

- We can now export our mesh to a mesh format, such as STL, which is currently the most widely used format in 3D printing, by using menu **File** → **Export** and choosing the STL file format.

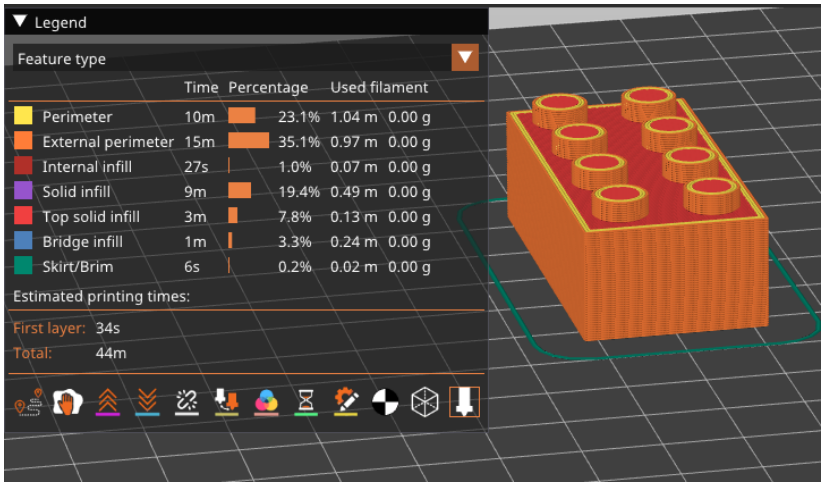
In FreeCAD, the Mesh Workbench provides several algorithms for converting a solid model into a mesh, including Standard, Mefisto, Netgen, and Gmsh. The Standard algorithm is commonly used for small to medium-sized objects as it provides a balance between speed and mesh quality. When creating a mesh, two critical parameters are the surface deviation and angular deviation. Surface deviation controls how closely the mesh follows the original geometry, with smaller values providing a finer, more accurate mesh but potentially leading to larger file sizes. Angular deviation defines how much deviation is allowed based on changes in the model's angles, particularly for curves and sharp edges. Other options like relative surface deviation allow you to adjust the precision dynamically based on the model's scale, and features like applying face colors or defining mesh segments by color are useful for advanced rendering or grouping different regions of the model. Once the mesh is generated, it can be exported in formats like STL, OBJ, or 3MF, which are essential for preparing models for 3D printing. Mesh quality is crucial for ensuring that 3D printers interpret the model correctly, so selecting the right meshing algorithm and deviation settings can significantly affect the final print outcome.

Using PrusaSlicer[edit | edit source]

PrusaSlicer is an application that converts STL, OBJ, and 3MF objects into G-code that can be sent directly to 3D printers. Like FreeCAD, it is free, open-source, and available on Windows, Mac OS, and Linux. Although it is developed by Prusa Research and optimized for Prusa 3D printers, PrusaSlicer can be used with almost any 3D printer, making it versatile for a wide range of machines. PrusaSlicer is based on Slic3r, the original slicer software, but with significant improvements and more frequent updates. Slic3r is no longer actively updated, while PrusaSlicer continues to evolve, adding new features such as adaptive layer heights, tree supports, and improved print strategies.

Correctly configuring a slicer for 3D printing is a complex process that requires a good understanding of your 3D printer's capabilities. While generating G-code without this knowledge might result in a file that doesn't work well on other printers, PrusaSlicer still provides an excellent way to verify that your STL file is correctly formatted and printable. The slicer's simulation features allow you to preview the G-code paths and check for any potential printing issues before you begin the actual print.

This is our exported STL file opened in PrusaSlicer. By just pressing on the **slice** button, the software divides your model into layers, generates the toolpaths for the 3D printer, and applies the necessary speed and temperature settings. It calculates the infill, support structures, and perimeters, then creates the G-code, which contains detailed instructions for the printer. You can preview the sliced model layer by layer, check estimated print time and filament usage, and finally save or send the G-code to your printer for the actual printing process.



Apart from PrusaSlicer, there are several other slicer software options available for 3D printing. Cura, developed by Ultimaker, is one of the most popular open-source slicers and supports a wide range of printers with extensive customization. Simplify3D is a paid slicer known for its advanced features and efficient toolpath generation.MatterControl is an open-source slicer that also includes basic CAD tools, while IdeaMaker offers a user-friendly interface with adaptive layer heights, developed by Raise3D. Finally, OrcaSlicer, a newer open-source option based on PrusaSlicer and Bambu Studio, provides additional features for various printers. Each slicer has unique strengths, making the best choice dependent on specific printer models and print requirements.

Generating G-code[edit | edit source]

The CAM Workbench in FreeCAD provides advanced options for generating G-code directly for CNC machines, offering greater flexibility and control compared to automatic slicing tools like those used for 3D printing. While 3D printing slicers can automatically convert a model into G-code with minimal input, CNC milling requires much more user involvement to ensure precise control over the toolpaths, speeds, depths, and other machining parameters. This makes the CAM Workbench essential for tasks that demand fine-tuned G-code, particularly for CNC milling, where machine complexity and the variety of operations (like cutting, drilling, and contouring) require careful planning.

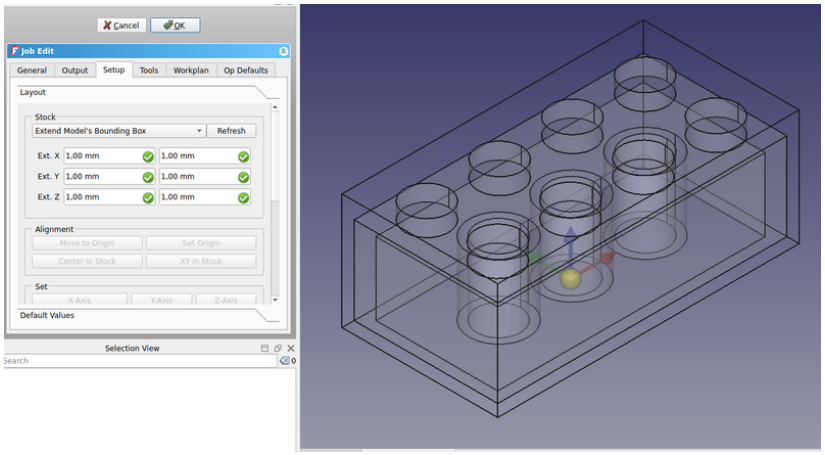
In the CAM Workbench, G-code path generation is highly customizable. It features tools to generate complete machine paths for various operations, or, alternatively, you can build partial G-code segments and assemble them into a full milling operation. This modular approach allows you to tailor each step of the machining process, optimizing the toolpaths for efficiency, material type, and specific machine capabilities.

The CAM process is indeed much more intricate than 3D printing because CNC machines use different tools and must account for material removal, tool geometry, and safety margins, all of which are configured manually. In FreeCAD, building a simple CAM project requires defining toolpaths, adjusting cutting depths, selecting appropriate tools, and configuring work offsets, feeds, and speeds. Unlike slicer software, which handles most of this automatically, the CAM Workbench places the control in your hands, making it highly customizable but also more complex.

Though generating CNC milling paths is a topic too broad to cover in detail here, we'll demonstrate how to create a simple CAM project in FreeCAD. While we won't focus on every detail of real-world CNC machining, this guide will introduce you to the essential steps, emphasizing the level of input required to ensure accurate and efficient results.

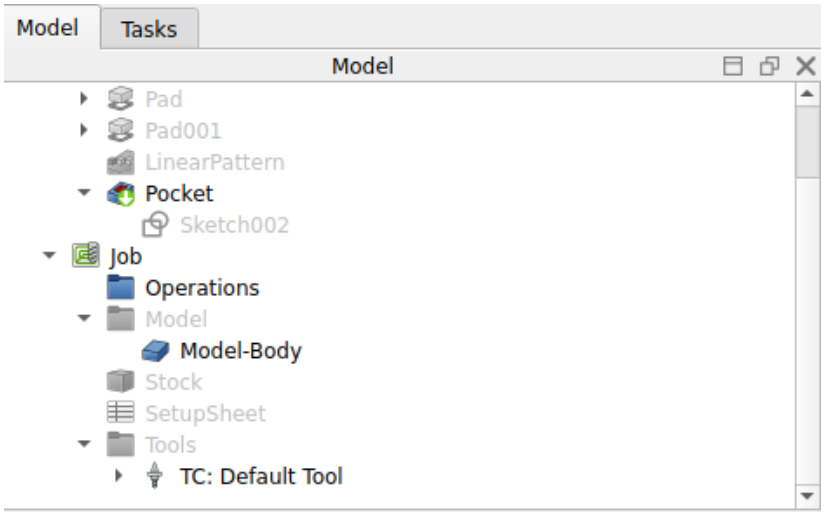
This added complexity is essential for CNC projects, where precision and customizability are critical to achieving desired machining outcomes.

- Load the file containing our Lego piece, and switch to the CAM Workbench.
- Press on the Job button and select our lego piece.
- Since this section doesn't aim to provide an in-depth tutorial of the CAM Workbench, we will be using the default values. If you would like a more detailed tutorial, please refer to CAM walk-through. Keep in mind that in the CAM Workbench, a stock body is automatically created around your object, representing the raw material that will be machined. Right now, this stock body extends 1 mm in all directions from the object.



- The first step is to remove the unnecessary material from around our object. At this stage, we're starting with a solid block of raw material, and we need to carve out the Lego brick from this block. This process involves defining the toolpaths that will gradually cut away the excess material, leaving behind the desired shape of the Lego.

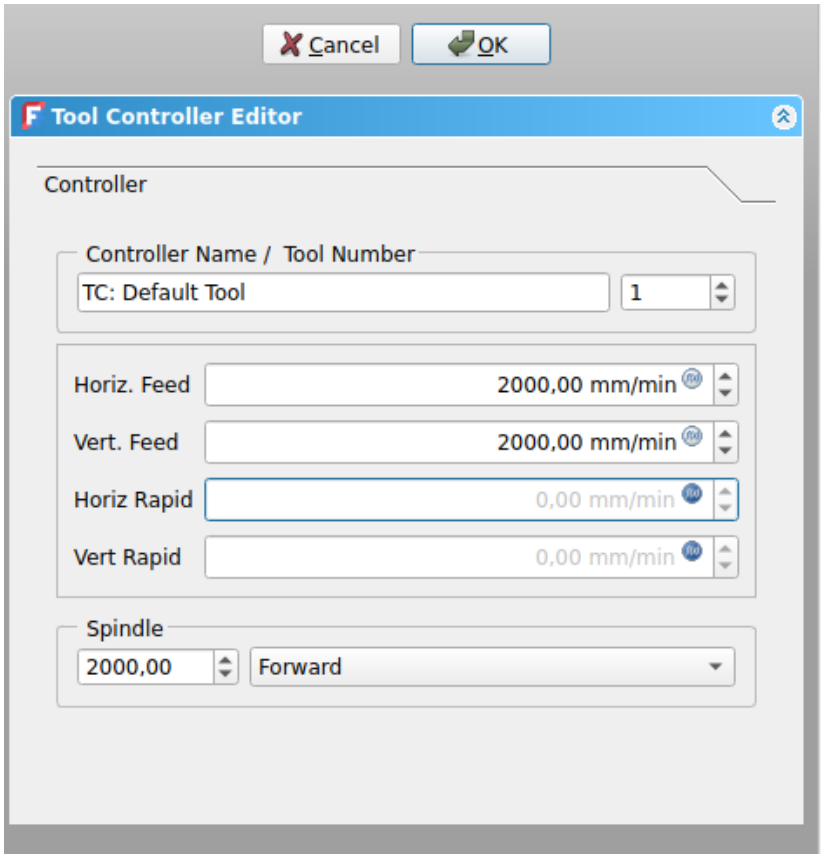
- The following image shows the FreeCAD CAM Workbench setup for machining a Lego block. The model tree includes solid modeling operations like Pad, Pocket, and LinearPattern, which were used to shape the part. A Job is created, containing toolpaths under Operations that define how the material will be removed from the Stock. The Default Tool is selected for machining, and the Model-Body represents the 3D part being worked on. This setup prepares the object for generating G-code to control the CNC machine.



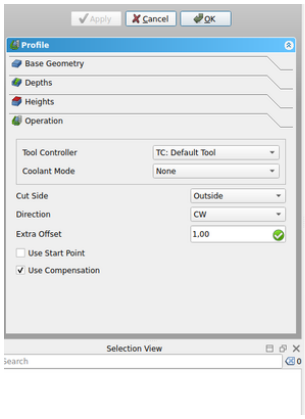
- Before we begin cutting away the excess material, let's make some adjustments to the milling tool that will be used. Although the CAM Workbench allows you to define

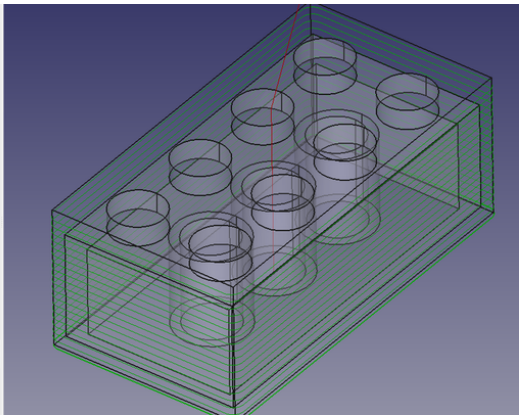
custom tools, for simplicity, we will modify the default tool. This will ensure the settings are optimized for our project without needing to create a new tool from scratch.

- Click on the **TC:Default Tool** text. This will open up the **Tool Controller Editor**. Change the feed rates and spindle velocities as shown in the picture. The feed rates for horizontal and vertical cutting are set to 2000 mm/min, while the spindle speed is set to 2000 RPM with forward rotation. These settings control the movement and cutting speed of the tool during the machining process.



- Double-click on the tool itself and change its diameter to 1mm.
- Now we are ready to begin removing the excess material from the block, gradually carving out the Lego geometry. This process will involve the toolpaths we set, ensuring the final shape matches the intended design.
- Click on the Profile. This option is used to carve out the unnecessary material around the perimeter of the part, effectively shaping the outer boundaries to achieve the general dimensions of the Lego piece.
- Normally you will not have to change any of the default values, except the **Extra Offset** located in the Operation tab. Set this option to 1 mm to ensure that the remaining object corresponds correctly to the Lego's boundaries.
- Once you press **apply** you should be able to see those green lines around the object. Those lines visualize the path our cutting object will follow when cutting the initial block.





- Our next step is to create the 6 extruding cylinders on the top of the Lego block.
- Choose the top face and click on the Pocket Shape button. On the **Extensions** tab, enable Extensions and click on the edge of the top face (it should normally be automatically added in the default length box.
- Finally, on the **Operation** tab input -1.5 mm in the **Pass Extension box** and change the pattern option to a ZigZagOffset.
- Press **apply** and then close the tab.
- In a similar manner we can create the three cylinders on the bottom of the Lego piece.
- We can easily visualize the steps followed during the milling of the object by using the SimulatorGL option.

Downloads

- The STL file generated in this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/lego.stl
- The file generated during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/path.FCStd
- The G-code file generated in this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/lego.gcode

Read more

- The Mesh Workbench
- The STL file format
- Slic3r
- Cura
- The Cura Workbench
- The CAM Workbench
- Camotics

Videos[edit | edit source]

- How To Use FreeCAD For 3D Printing | Using The Realthunder Branch A video playlist by Maker Tales about how to use FreeCAD for 3D printing.

The Community

No manual dealing with free and open-source software would be complete without a chapter about the community. Like the vast majority of free and open-source software projects, FreeCAD is made by a community, and maintained by that community. Instead of some opaque, unknown, impersonal and inaccessible company that is more often than not found behind commercial software, free and open-source software communities are open spaces, where you as a user are welcome, and where you can get answers very fast, and even have your say in the development of the software itself. You are also more than welcome to help, there are tasks for everybody.

The community is a growing, eclectic group of all kinds of people united by their passion for FreeCAD. All work on FreeCAD voluntarily, during their free time (although sometimes firms or individuals gather to pay a couple of programming hours to a developer to implement a specific function). Some are professional programmers, some are long-time FreeCAD users (some of them are true FreeCAD gurus, who know almost everything, and many of them end up knowing a lot about FreeCAD programming too), and many are new users of FreeCAD. There is nothing specific to do to be part of the community. Just use FreeCAD!

The main place where the community meets and discusses is the FreeCAD forum. All you need to do to participate in the discussions is to register an account on the forum (Your first post will need to be approved by a moderator before you can post more, to prevent spamming). The forum is a great place to ask questions when you are new to FreeCAD. Provided you made a good question (be sure to read the forum rules as they contain useful information to turn your question into a good question), you will usually get several replies within the same hour. If you think someone might have asked your question already, be sure to search, your answer might already be there.

The forum is also a great place to show what you achieved with FreeCAD, to help newcomers when you are more experienced, and to follow and give your opinions in more technical discussions about development. All the FreeCAD development is discussed on the forum, and anybody is free to read or participate.

There are also FreeCAD communities forming outside of the FreeCAD forum, for example on Facebook.

If you are becoming as enthusiastic about FreeCAD as we are, you might want to help the project. This can be done in many different ways, and there are tasks for everybody, programmers and non-programmers alike, for example:

- **Help by spreading the word**: Many people would get huge benefits from using a free, open-source 3D modeler like FreeCAD, but simply don't know it exists. Publishing the work you do with FreeCAD, talking about it on social networks, etc... helps these people discover FreeCAD.
- **Help newcomers**: The vast majority of discussions on the forum are questions asked by new users. You might have good answers for them.
- **Help by reporting bugs**: The stability of FreeCAD comes in large part from the fixing of bugs. Since it is not possible for the FreeCAD developers to test all possible use cases, users must report problems when they detect them. Be sure to read the guidelines if you think you found a bug, and then write a report on the bug tracker.
- **Help by writing documentation**: The FreeCAD documentation wiki is also written by community members. Some sections of it are still incomplete, or their information is incorrect or obsolete. You might be able to help fix that. To be able to

work on the wiki, you will need to familiarize yourself with wiki editing, and ask permission to edit the FreeCAD wiki on the forum.
- **Help to translate FreeCAD**: The translation of FreeCAD is done online by community members, on crowdin. If you don't see your language there, ask one of the administrators to have it added.
- **Help to translate the wiki documentation**: Every page of the wiki is translatable, and requires very little knowledge of the wiki syntax. Helping with translation is also a great way to learn FreeCAD.
- **Write scripts and macros**: FreeCAD has a growing list of Macros. If you wrote some interesting functionality, consider sharing it there.
- **Programming**: For this, you need to know how to program in Python or C++, and have a good knowledge of FreeCAD itself.

The source code of FreeCAD is located on the Github account of the FreeCAD project. Anybody can download, use and modify the code. You can publish your modifications (on Github or any other Git hosting service). If you made interesting modifications that you wish to see included in the FreeCAD source code, you must ask the community to have them included. This can be done using Github's pull requests mechanism, but the very best way is to discuss what you intend to do first on the forum, and then post an official request in the Pull requests section of the forum when your code is ready. This avoids you working on something that someone else is already working on too, and ensures that others agree with the way you are doing it so that there is no risk of having your work refused for some reason you didn't foresee.

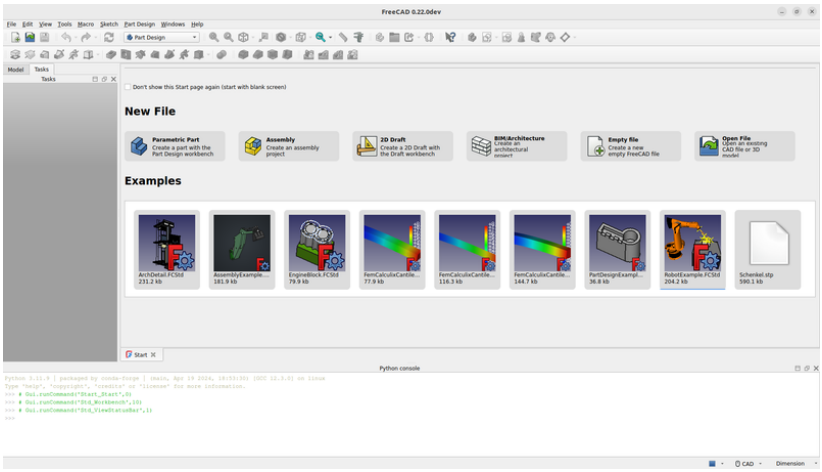
Hopefully, we managed to give you a good taste of FreeCAD in this manual, and you are already our newest community member. Welcome!

Read more

- The FreeCAD forum
- The source code of FreeCAD
- The Facebook FreeCAD community
- The FreeCAD documentation wiki
- Translating FreeCAD on crowdin
- The FreeCAD bug tracker

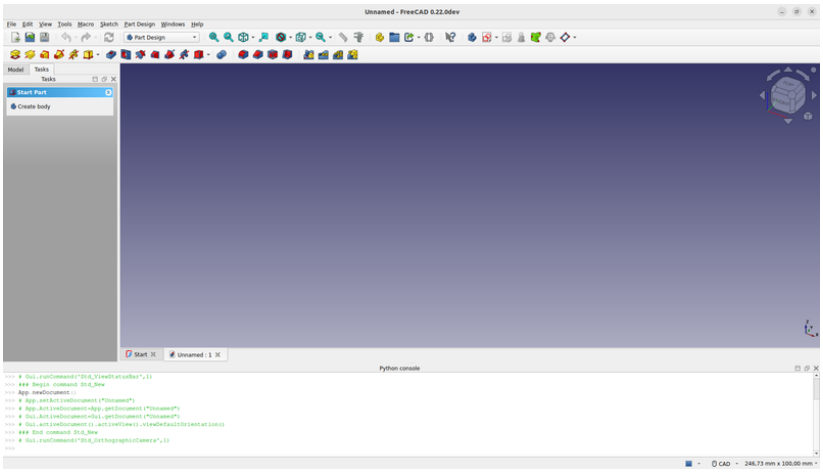
The FreeCAD Interface

FreeCAD is based on the Qt framework and is characterized by a simple and straightforward interface. More experienced CAD users will be able to identify similarities with other software, while new users will find it easy to navigate and discover the various options FreeCAD offers. Here is the default look of FreeCAD:



The Start Page serves as the welcoming screen, designed to facilitate quick and easy access to the main areas of FreeCAD that a user might wish to explore. Through it, users can effortlessly create new parts, open recent files, and initiate 2D drafting. Additionally, it features shortcuts to helpful resources like tutorials and user forums, which are invaluable for both beginners and experienced users seeking guidance or tips. Users can easily customize the appearance of the Start Page according to their preferences.

As you become more proficient with FreeCAD, you might adjust the settings under preferences. This can configure FreeCAD to open directly into one of the Workbenches with a new document ready to go when you launch it. Alternatively, you can simply close the Start Page tab and manually create a new document.



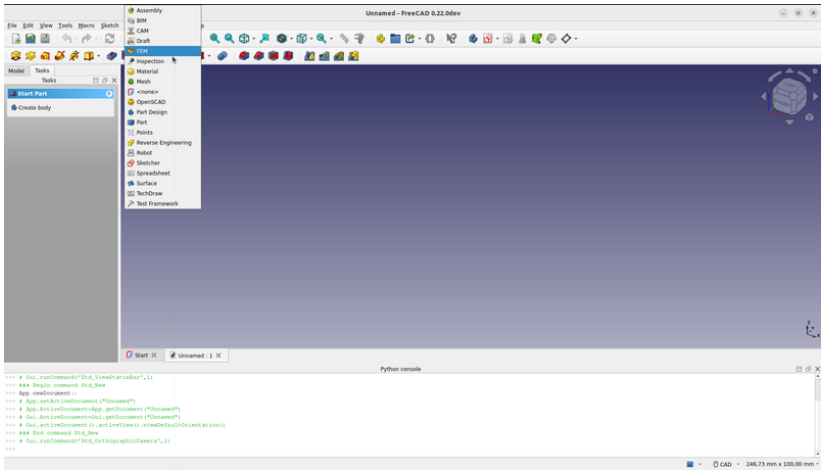
Workbenches[edit | edit source]

FreeCAD employs a system called "Workbenches," similar to the conceptual frameworks used in advanced design software like Revit or CATIA. The idea of a Workbench is analogous to specialized stations in a scientific lab, where different workstations are equipped for distinct types of experiments. In a lab, you might have one area dedicated to chemistry, another to biology, and a third to physics, each outfitted with the specific tools needed for those disciplines.

In the context of FreeCAD, each Workbench is tailored to a particular type of task, organizing all the necessary tools for that activity in one interface. When switching between Workbenches, the set of tools and controls visible in the user interface adjusts to reflect the needs of the selected task, though the actual project contents or "scene" you are working on does not change. This allows for seamless transitions in workflow, such as beginning a design with basic 2D shapes in the Draft Workbench and then elaborating on these designs with advanced modeling tools in the Part Workbench.

The terms "Workbench" and "Module" are sometimes used interchangeably, but they have distinct meanings within FreeCAD. A Module is any extension that adds functionality to FreeCAD, while a Workbench is a specific kind of Module equipped with its own user interface components such as toolbars and menus, designed to facilitate specific types of tasks. Thus, every Workbench is a Module, but not every Module qualifies as a Workbench.

The most important control of the FreeCAD interface is the Workbench selector, which you use to switch from one Workbench to the other:



Workbenches often confuse new users, since it's not always easy to know in which Workbench to look for a specific tool. But they are quick to learn, and after a short while, they will feel natural. New users quickly realize that Workbenches are a convenient way to organize the multitude of tools FreeCAD has to offer. In addition, Workbenches are also fully customizable.

Enabled workbenches[edit | edit source]

The following workbenches are enabled by default:

- The Assembly Workbench for building and solving mechanical assemblies. introduced in 1.0
- The BIM Workbench for working with architectural elements and creating BIM models. It combines the Arch Workbench and the formerly external BIM Workbench available in 0.21 and below.
- The CAM Workbench is used to produce G-Code instructions. This workbench was called "Path Workbench" in 0.21 and below.
- The Draft Workbench contains 2D tools and basic 2D and 3D CAD operations.
- The FEM Workbench provides Finite Element Analysis (FEA) workflow.
- The Material Workbench handles the FreeCAD material system. introduced in 1.0
- The Mesh Workbench for working with triangulated meshes.
- The Part Workbench for working with geometric primitives and boolean operations.
- The Part Design Workbench for building Part shapes from sketches.
- The Points Workbench for working with point clouds.
- The Reverse Engineering Workbench is intended to provide specific tools to convert shapes/solids/meshes into parametric FreeCAD-compatible features.
- The Sketcher Workbench for working with geometry-constrained sketches.
- The Spreadsheet Workbench for creating and manipulating spreadsheet data.
- The Surface Workbench provides tools to create and modify surfaces. It is similar to the Part Builder Face from edges option.
- The TechDraw Workbench for producing technical drawings from 3D models. It is the successor of the Drawing Workbench.

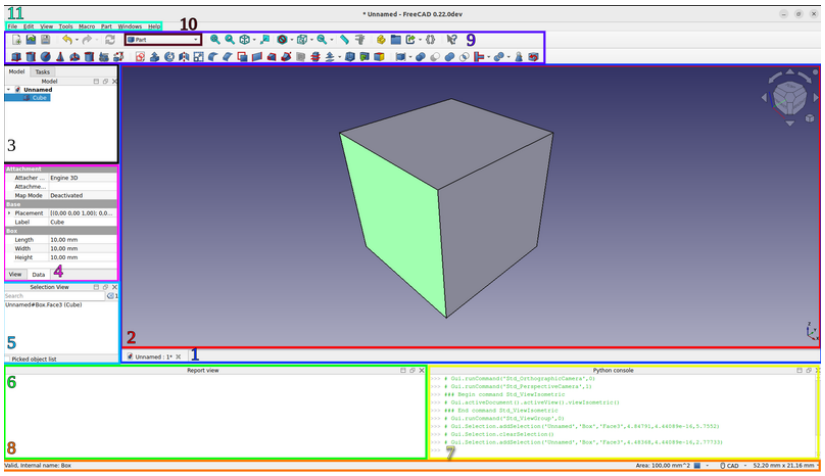
Disabled workbenches[edit | edit source]

introduced in 1.1 The following rarely used workbenches are disabled by default, they can be enabled in preferences:

- The Inspection Workbench is made to give you specific tools for the examination of shapes. Still in the early stages of development.
- The OpenSCAD Workbench for interoperability with OpenSCAD and repairing constructive solid geometry (CSG) model history.
- The Robot Workbench for studying robot movements. Currently unmaintained.
- The Test Framework Workbench is for debugging FreeCAD.

The interface[edit | edit source]

Let's have a better look at the different parts of the interface:



The main window of the application can be roughly divided into 11 sections:

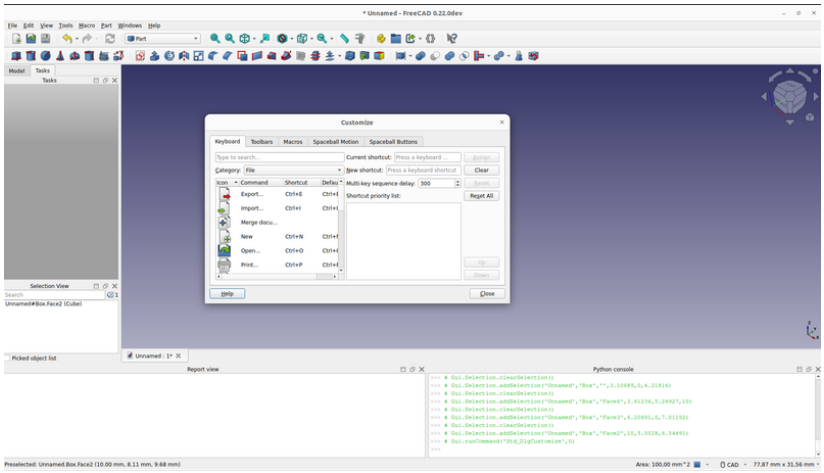
1. The Main view area, which can contain different tabbed windows.
2. The 3D view, normally embedded in the main view area. The 3D view is the central element of the interface, displaying and allowing manipulation of the objects you are working on. It is possible to have multiple views of the same document (or objects) or to have several documents open simultaneously. Additionally, each view can be detached from the main window separately.
3. The Model and the Tasks tab.
    1. The Model tab shows you the contents and structure of your document.
    2. The Tasks tab is where FreeCAD will prompt you for values specific to the workbench and tool you are currently using (for example dimensions of an object).
4. The Property editor which appears when the Model tab is active in the interface. It allows managing the publicly exposed properties of the objects in the document. It consists of the Data and View sections, showing the visualization properties and the parametric properties of the objects respectively.
5. The Selection view which indicates the objects or sub-elements of objects (vertices, edges, faces) that are selected.
6. The Report view where messages, warnings and errors are shown.
7. The Python console. The Python console, where all the commands executed are printed, and where you can enter Python code.
8. The Status bar where some messages and tooltips appear.
9. The toolbar area, where the toolbars are docked.
10. The workbench selector, where you select the active workbench.
11. The standard menu, which holds basic operations of the program.

Most of the above panels can be hidden or revealed using the **View → Panels menu**

Customizing the interface[edit | edit source]

The FreeCAD interface is designed for extensive customization. All toolbars and panels can be relocated, stacked, or even docked in various configurations according to user preference. Additionally, they can be closed and then reopened as required. Beyond these capabilities, users have numerous other options including the ability to create custom toolbars with tools selected from any of the available Workbenches, and to assign or modify keyboard shortcuts to streamline workflow. This flexibility ensures that users can tailor the environment to fit their specific needs and preferences.

These advanced customization options are available from the **Tools** → **Customize menu**:



Read more

- Getting started with FreeCAD
- Customizing the interface
- Workbenches
- More about Python

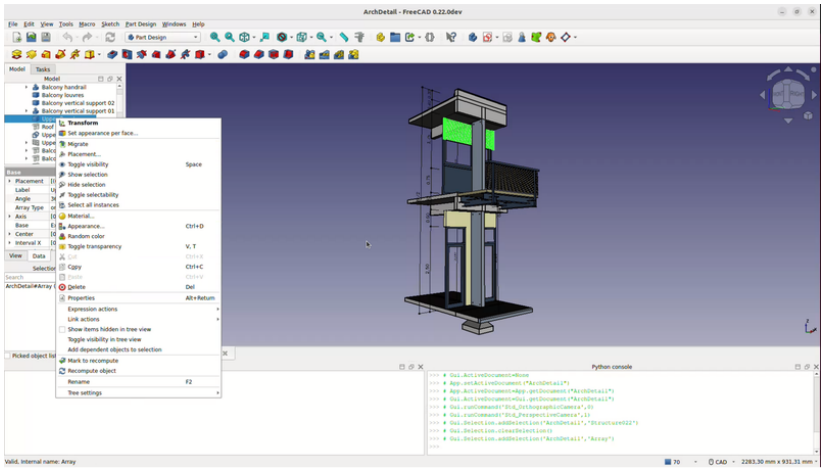
The FreeCAD document

A FreeCAD document forms the core of your design environment, encapsulating all the objects and elements that make up your scene. It can contain a variety of objects created across different workbenches, allowing for seamless integration and flexibility as you can switch between workbenches while working within the same document. A document is essentially the file that you save to your disk containing all your work. FreeCAD allows you to open multiple documents simultaneously, and can show multiple views of the same document.

Within the document, you can organize objects into groups, each with a unique name for easy identification and management. This management of groups, objects, and their names chiefly occurs within the tree view. Here, users can craft new groups, assign objects to these groups, and perform deletions. More detailed customizations such as renaming objects, changing their visual properties like color, or adjusting visibility, can be done by right-clicking an object or a group in the Tree View. Additional functionalities may also be available depending on the active workbench.

Objects in a FreeCAD document vary widely in type, as each workbench introduces its unique kind of objects. For example, the Mesh Workbench is known for adding mesh objects, while the Part Workbench provides Part objects. In any FreeCAD session, if at least one document is open, that document is considered active and is displayed in the current 3D view. This is the document you are actively modifying. Switching tabs to another document makes the newly selected tab the active document, and most operations will be applied to it.

FreeCAD documents are saved in the .FCStd file format, which utilizes a zip-based compound structure akin to formats used by software like LibreOffice. If technical issues arise, the document can often be unzipped, allowing direct access to its contents for troubleshooting or data recovery purposes. This capability provides an additional safety net, ensuring that your design work can be preserved and recovered even under unexpected circumstances.



Read more

- The FreeCAD document
- File Format FCStd

FreeCAD's Draft Workbench acts as a bridge between 2D drafting and 3D modeling, making it unique compared to traditional 2D CAD systems like AutoCAD. While FreeCAD operates in a fully 3D environment, the Draft Workbench is designed to provide users with familiar 2D drawing tools while offering the flexibility of transitioning smoothly between 2D sketches and 3D objects. For instance, you can set custom working planes to draw on specific surfaces or orientations, allowing for the construction of parametric models. Since FreeCAD is parametric, any changes made to dimensions update automatically across the entire project.

One of the strengths of the Draft Workbench is its comprehensive toolset, which includes both basic drawing and advanced modification tools. These tools can be used not only for 2D drafting but also for manipulating objects in 3D space. You can set working planes, define grids, and apply constraints to ensure geometric relationships remain intact across your design. Draft objects can be modified and repositioned using snapping modes and a variety of constraints, making precise drafting much easier.

Some of the tools found in the Draft Workbench:

1. Drawing Tools:

- Line, Wire (polyline): These tools allow for creating straight line segments or continuous polylines, which can be constrained and converted into 3D shapes.
- Circle, Ellipse, and Arc: Used to define basic circular and elliptical shapes, with options for further manipulation.

1. Manipulation Tools:

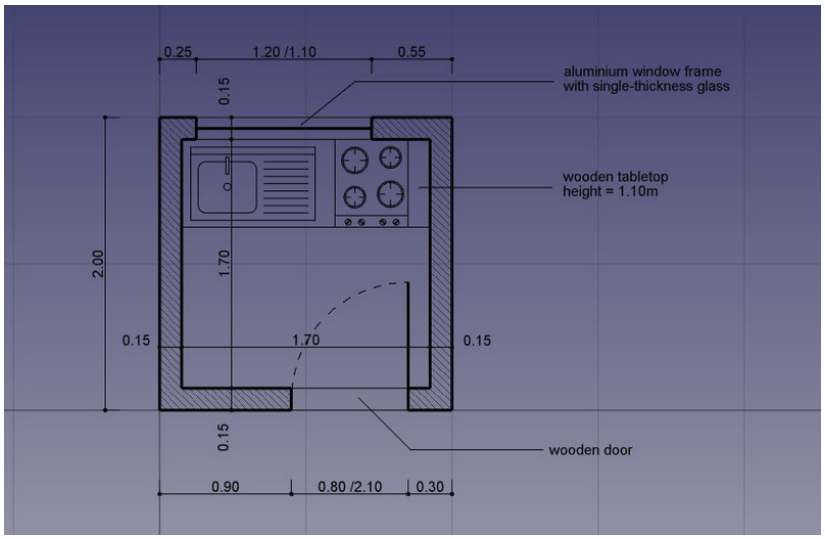
- Move, Rotate or Scale: These operations work in 3D as well as 2D, offering flexibility in positioning objects.
- Offset: Similar to traditional CAD systems, this allows you to create parallel lines or curves.
- Trimex and Stretch: Modify lines and shapes by trimming or extending them to intersect or meet other objects.
- Mirror and Array: These tools replicate and pattern objects, ideal for repetitive components.

The snapping system in FreeCAD’s Draft Workbench is designed for precision. Whether working in 2D or 3D, you can snap to critical points like endpoints, midpoints, and centers of circles, making it easy to position elements relative to one another. Modes such as perpendicular, tangent, and intersection snapping further enhance precision. Combined with the working plane and grid system, these tools ensure the accurate alignment of objects and components.

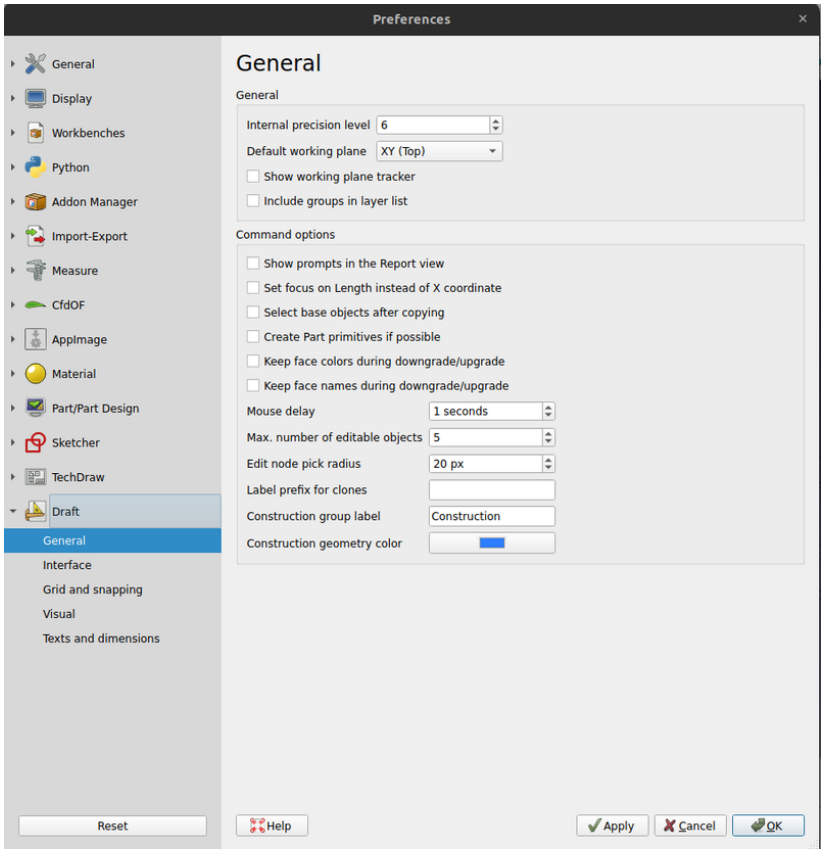
FreeCAD’s parametric nature enables constraints to be applied to drafted elements, ensuring geometric relationships stay intact. For example, you can make lines parallel or perpendicular and set fixed distances between elements. These constraints can be adjusted later, making design changes smooth and consistent across the project. The Draft Workbench also integrates seamlessly with other FreeCAD workbenches, such as Sketcher, which is designed for more constrained parametric 2D design, and TechDraw, which produces technical 2D drawings for documentation purposes.

Advanced features of the Draft Workbench include the ability to import and export files in formats like DXF and SVG, allowing you to work with or share designs with users of other CAD programs. Python scripting further enhances FreeCAD’s capabilities, allowing you to automate tasks or create custom workflows. You can write scripts that generate draft objects based on specific geometric rules, streamlining repetitive tasks.

To showcase the workflow and possibilities of the Draft Workbench, we will walk through a simple exercise, the result of which will be this little drawing, showing the floor plan of a small house that contains only a kitchen top (A pretty absurd floor plan, but we can do what we want here, can't we?):



- Switch to the **Draft Workbench**
- As in all technical drawing applications, it is wise to set up your environment correctly, it will save you a lot of time. If you wish to personalize your experience in the Draft Workbench, you can easily do so by adjusting various settings in the Draft Preferences Panel by going to **Edit** → **Preferences** → **Draft**. In this exercise, however, we will act as if these settings were left at their default values.



- The Draft Preferences Panel in FreeCAD allows you to customize various aspects of your 2D drafting environment. In the **General Settings**, you can define the default working plane, adjust decimal precision, and set the default line width, style, and color for objects. The **Grid and Snapping** section enables you to control grid

visibility, spacing, and snapping modes, allowing for precise alignment and positioning of elements. The **Visual Style** options let you customize the appearance of objects and the grid, including line and fill colors. In **Text and Dimensions**, you can configure the default text size, font, and color for annotations, ensuring clarity in technical drawings.

- Turning on all the snap buttons is convenient, but also makes drawing slower, as more calculation needs to be done when you move the mouse cursor. It is often better to keep only the ones you will actually use.

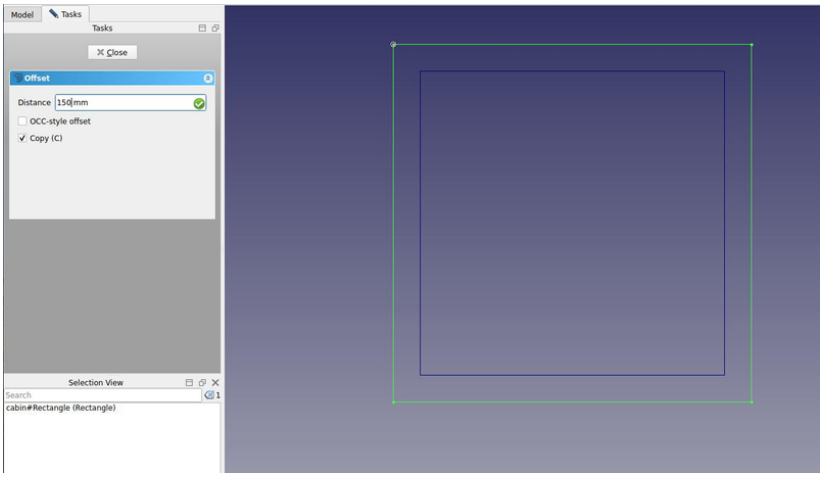
- Let's start by turning **construction mode** on, which will allow us to draw some guidelines on which we will draw our final geometry. You can do that by pressing on the Toggle construction mode command.
- If you prefer, you can set the working plane to XY. This will lock the working plane, ensuring it remains on the XY plane regardless of how you change the view. If you choose not to do this, the working plane will automatically adapt to the current view, meaning you'll need to ensure you're in the top view whenever you want to draw on the XY (ground) plane to avoid unintended shifts in orientation.

Let's begin laying out the basic shape of our floorplan.

- Press the **Draft Rectangle** button
- Draw a 2-meter by 2-meter rectangle starting from point (0,0,0), leave the Z coordinate at zero. You can efficiently complete this entire process using the keyboard, without needing the mouse. Simply type:
  - **re**, **Enter**, **Enter**, **Enter**, 2m, **Enter**, 2m, **Enter**, 0 and **Enter**.

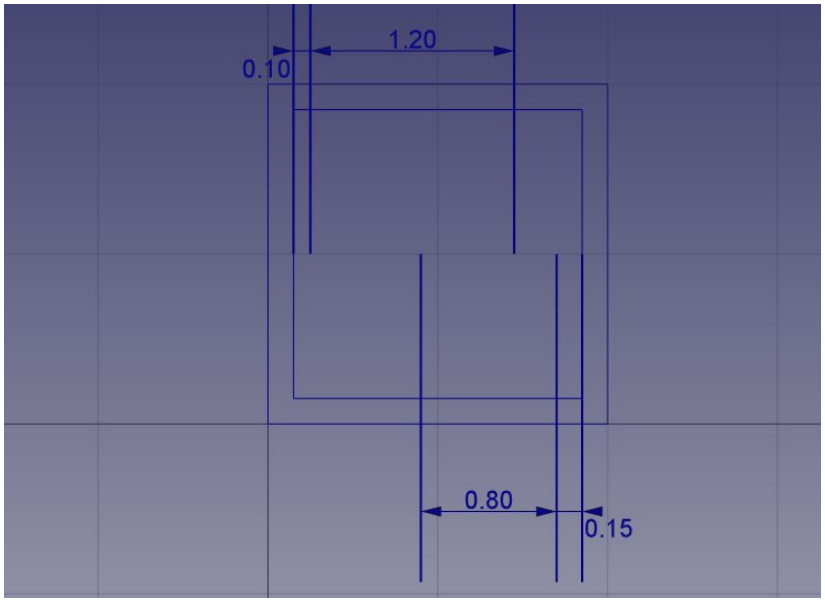
This keyboard-driven workflow speeds up drafting, especially when working with repetitive tasks or precision input, making it ideal for users looking to streamline their workflow. You can view the keystrokes for every object by hovering over the corresponding button.

- Duplicate that rectangle by 15cm inside, using the Offset tool, turning its Copy mode on, and giving it a distance of 15cm:

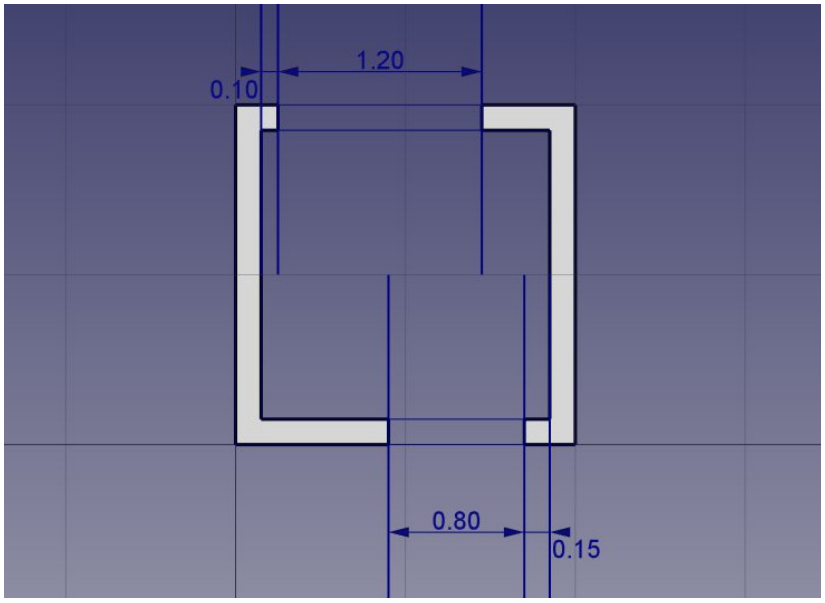


- We can then draw a couple of vertical lines to define where our doors and windows will be placed, using the Line tool (note that the "relative" mode box should be unchecked for this step). The crossing of these lines with our two rectangles will

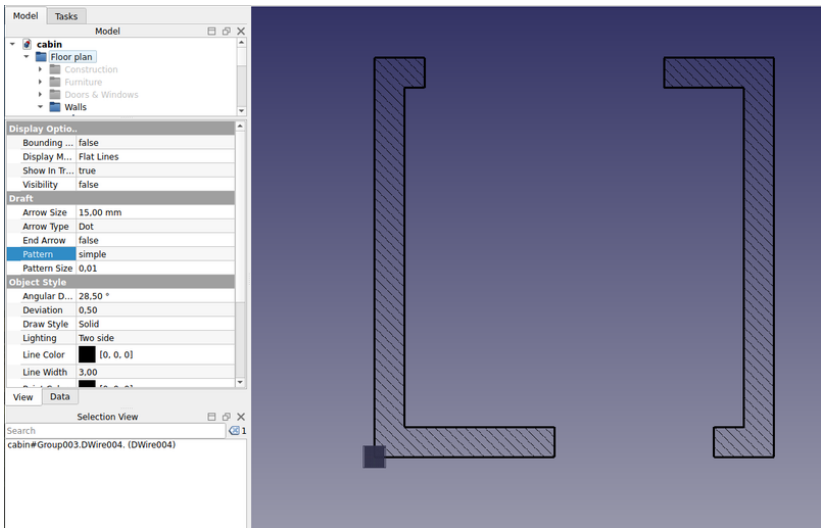
give us useful intersections to snap our walls to. Draw the first line by defining the following points:
    - **P1** (15cm, 1m, 0) and **P2** (15cm, 3m, 0).
- We will now duplicate this line 5 times. Press on the Move tool with Clone mode turned on. Make sure **Relative mode** is enabled. Copy mode ensures that each movement creates a new line instead of shifting the original, while Relative mode allows you to define movements based on relative distances, making positioning easier without needing to calculate exact coordinates. Start by selecting the original line, and initiate the move operation by choosing any starting point, such as (0,0,0). After each move, perform the next operation on the newly created line, so each copy builds on the previous one. Define the relative endpoints for each new line
    - line001: x: 10cm
    - line002: x: 120cm
    - line003: x: -55cm, y: -2m
    - line004: x: 80cm
    - line005: x: 15cm



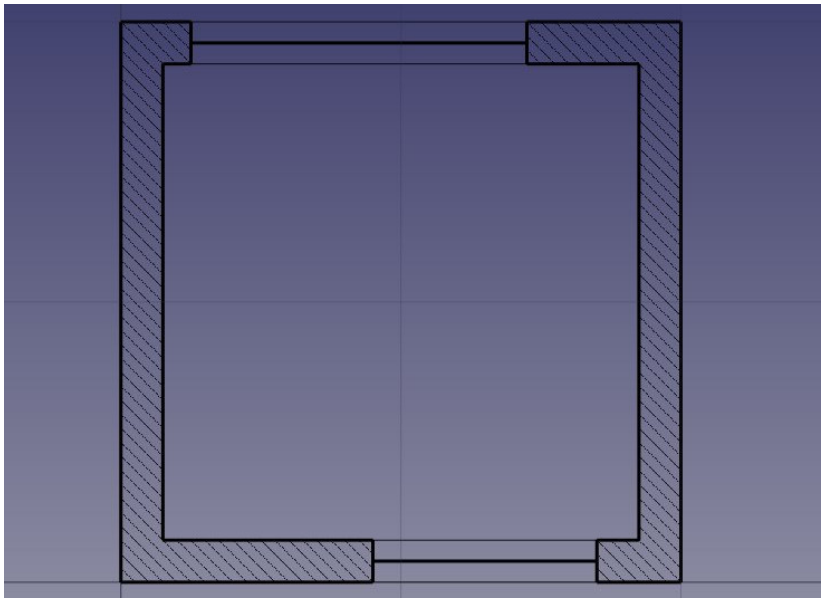
- That is all we need now, so we can switch construction mode off. Check that all the construction geometry has been placed into a "Construction" group, which makes it easy to hide it all at once or even delete it completely later on.
- Now let's draw our two wall pieces using the Wire tool. Make sure the intersection snap is turned on, as we will need to snap to the intersections of our lines and rectangles. Draw two wires as follows, by clicking all the points of their contours. To close them, either click on the first point again or press the **Close** button:



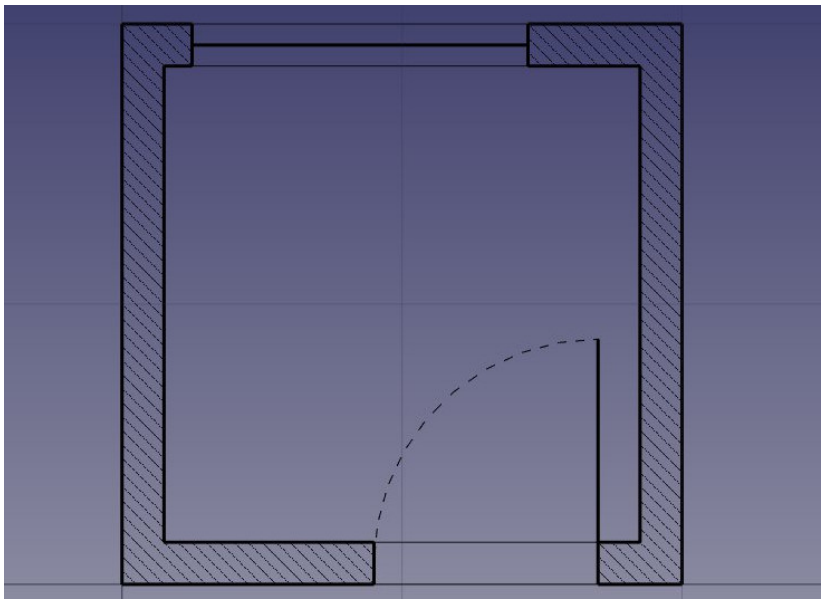
- We can give the walls a nice hatch pattern. Select both walls, make sure their **Make Face** property (located on the Data tab) is set to **TRUE**, then set their **Pattern** property (located on the View tab) to **Simple**, and their **Pattern size** to your liking, for example **0.01**.



- We can now hide the construction geometry by right-clicking the Construction group and choose **Hide Selection**.
- Let's now draw the windows and doors. Make sure the midpoint snap is turned on, and draw six lines as shown below:

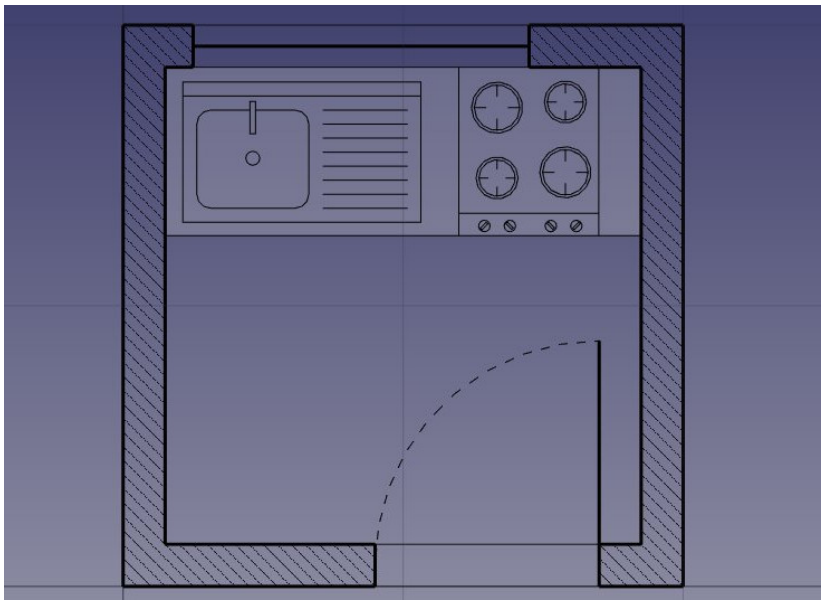


- We will now change the door line to create an opened door symbol. Start by rotating the line using the Rotate tool. Click the endpoint of the line as rotation center; then specify a base angle of **0**, and a rotation of **-90**.
- Then create the opening arc with the Arc tool. Pick the same point as the rotation center we used in the previous step as the center, click the other point of the line to give the radius, then the start and end points as follows:

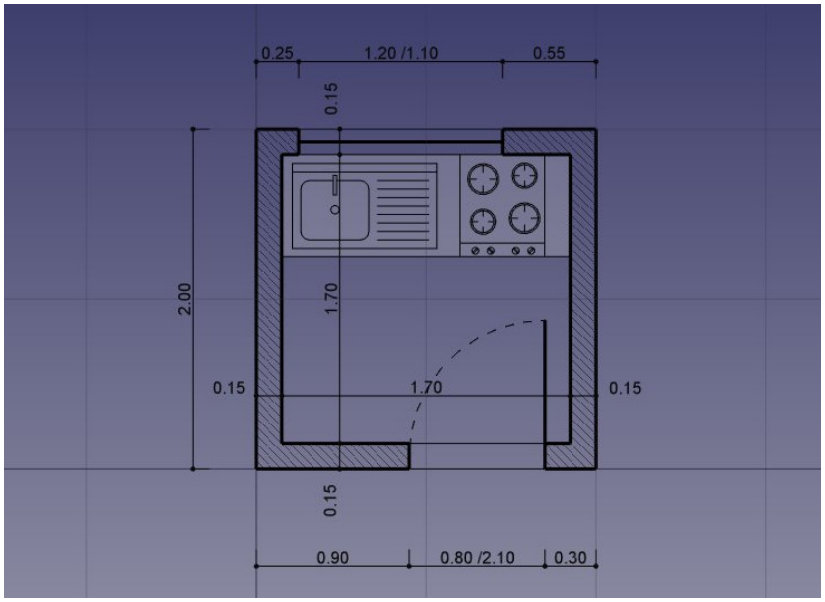


- We can now start placing some furniture. To begin with, let's place a counter by drawing a rectangle from the upper left inner corner, and giving it a width of 170cm and a height of -60cm. In the image below, the **Transparency** property of the rectangle is set to 80%, to give it a nice furniture look.
- Then let's add a sink and a cooktop. Drawing these kinds of symbols by hand can be very tedious, and they are usually easy to find on the internet, for example on http://www.cad-blocks.net. In the **Downloads** section below, for convenience, we separated a sink and a cooktop from this project and saved them as DXF files. You can download these two files by visiting the links below, and right-clicking the **Raw** button, then choosing **save as**.
- Inserting a DXF file into an opened FreeCAD document can be done either by choosing the **File** → **Import** menu option, or by dragging and dropping the DXF file from your file explorer into the FreeCAD window. The contents of the DXF files might not appear right on the center of your current view, depending on where they

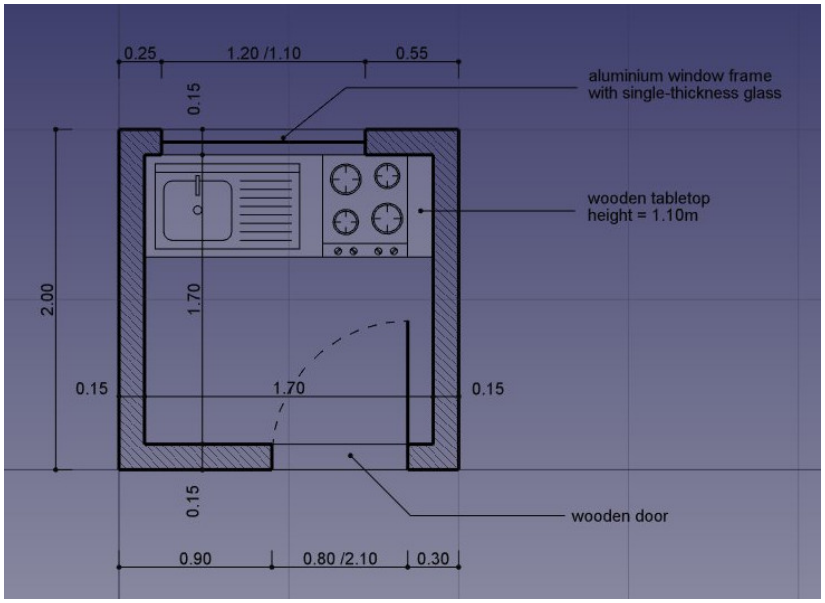
were in the DXF file. You can use the menu **View** → **Standard views** → **Fit all** to zoom out and find the imported objects. Insert the two DXF files, and move them to a suitable location on the tabletop:



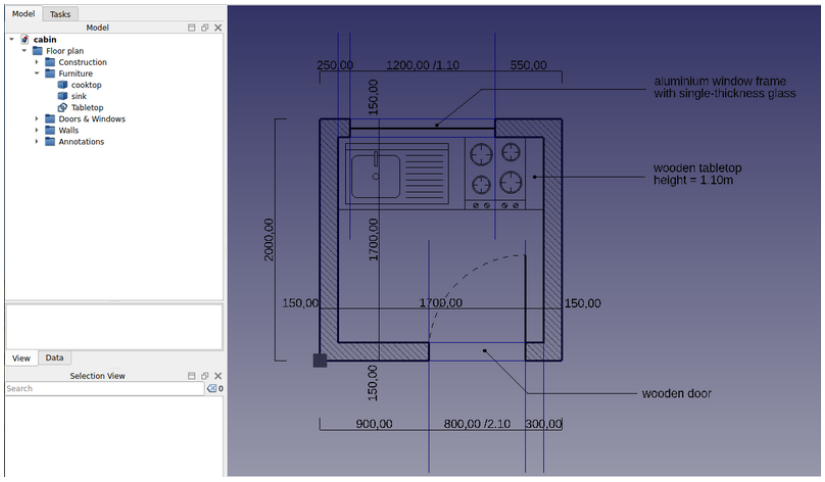
- We can now place a couple of dimensions using the Dimension tool. To create a dimension, you start by selecting three points: the first point sets the start of the measurement, the second point defines the endpoint, and the third point determines where the dimension line and text will be placed. By clicking these points carefully, you ensure that the dimension accurately represents the distance between the two selected points. If you want to force the dimension to be perfectly horizontal or vertical, even if the start and end points are not aligned, hold down the **Shift** key while clicking the second point. This locks the dimension into the desired orientation. You can further refine the dimension by adjusting properties such as text size, precision, and color in the properties panel, ensuring the dimensions fit your project's visual and technical standards.
- You can change the position of a dimension text by double-clicking the dimension in the tree view. A control point will allow you to move the text graphically. In our exercise, the "0.15" texts have been moved away for better clarity.
- You can change the contents of the dimension text by editing their **Override** property. In our example, the texts of the door and window dimensions have been edited to indicate their heights:



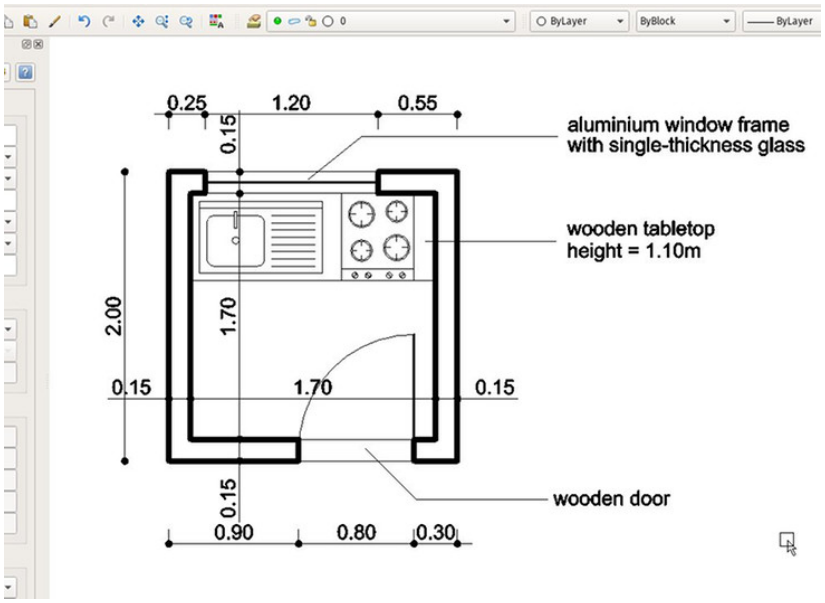
- Let's add some description texts using the Text tool. Click a point to position the text, then enter the lines of text, pressing Enter after each line. To finish, press Enter twice.
- The indication lines (also called "leaders") that link the texts to the item they are describing are simply done with the Wire tool. Draw wires, starting from the text position, to the place being described. Once that is done, you can add a bullet or arrow at the end of the wires by setting their **End Arrow** property to **TRUE**



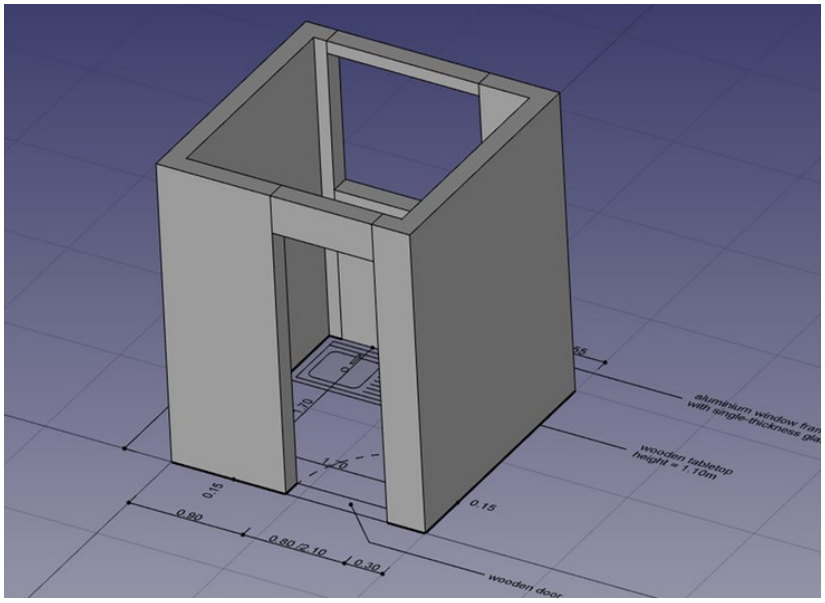
- Our drawing is now complete! Given the number of objects in the design, it’s a good idea to do some tidying up and restructuring before we consider it finished. Organizing everything into clear, logical groups will not only help keep the project well-structured but also make it significantly easier for others to navigate and understand the file. By grouping related elements—such as furniture, appliances, or architectural features—you can simplify the layout and improve the clarity of the design. This will also make future modifications or adjustments far more manageable, especially if the project needs to be shared or collaborated on. Additionally, a clean structure ensures that anyone reviewing the drawing can quickly locate specific elements without sifting through a cluttered workspace, ultimately contributing to a more professional and polished final product. Taking the extra time to organize now can save considerable time and effort later on.:



- We can now print our work by placing it on a Drawing sheet, which we will show later in this manual, or directly export our drawing to other CAD applications, by exporting it to a DXF file. Simply select our "Floor plan" group, select the menu **File** → **Export**, and select the **Autodesk DXF** format. The file can then be opened in any other 2D CAD application such as LibreCAD. You might notice some differences, depending on the configurations of each application.



- The most important aspect of the Draft Workbench, however, is that the 2D geometry you create can serve as the foundation for creating 3D objects. You can easily extrude these shapes into 3D using the Part Extrude tool found in the Part Workbench. Alternatively, if you prefer to stay within the Draft Workbench, you can use the Trimex tool, which combines trim, extend, and extrusion functionalities. The Trimex tool essentially performs a Part Extrusion under the hood but does so "the Draft way," allowing you to visually indicate and snap the extrusion length, giving you greater control and precision when working directly within your drafting environment. This flexibility makes transitioning from 2D to 3D seamless and intuitive, especially for those familiar with 2D workflows, while still offering advanced 3D modeling capabilities.
- By pressing the working plane button after selecting a face of an object, you are also able to place the working plane anywhere, and therefore draw Draft objects in different planes, for example on top of the walls. These can then be extruded to form other 3D solids. Experiment with setting the working plane on one of the top faces of the walls, then draw some rectangles up there.



- All kinds of openings can also be done as easily by drawing Draft objects on the faces of walls, then extruding them, and then using the boolean tools from the Part Workbench to subtract them from another solid, as we saw in the previous chapter.

Fundamentally, the Draft Workbench provides a more graphical and intuitive approach to creating basic operations, similar to those found in the Part Workbench. In the Part Workbench, positioning objects often involves manually adjusting parameters like the Placement values (for position, rotation, etc.), which gives you precise control but can sometimes feel less intuitive, especially for quick edits. In contrast, the Draft Workbench allows you to perform these same operations visually on-screen, making it easier to move, rotate, and manipulate objects directly in the workspace with snapping tools and relative positioning options.

This difference is where the workbenches complement each other. The Draft Workbench is ideal for fast, interactive design, allowing you to draw and position objects without constantly entering precise numerical values. On the other hand, the Part Workbench offers more detailed, parametric control over object properties, making it better suited for highly accurate adjustments, especially in engineering or technical design projects.

The beauty of FreeCAD is that you don't need to choose between one or the other. You can create custom toolbars by combining tools from both the Draft and Part Workbenches, giving you the flexibility to switch between graphical and parametric methods as needed. This allows you to enjoy the best of both worlds—quick, on-screen adjustments from the Draft Workbench and the precision of the Part Workbench—depending on the needs of your project. Additionally, using keyboard shortcuts and custom toolbars can speed up your workflow, making it easy to transition between different operations without interrupting your design process.

Downloads[edit | edit source]

- The file created during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/cabin.FCStd
- The sink DXF file: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/sink.dxf
- The cooktop DXF file: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/cooktop.dxf
- The final DXF file produced during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/cabin.dxf

Related[edit | edit source]

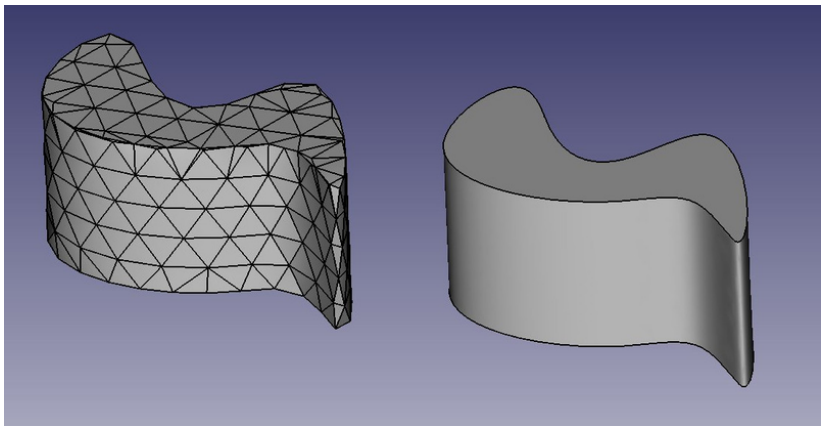
- The Draft Workbench
- Snapping
- The Draft working plane

Traditional modeling, the CSG way

CSG stands for Constructive Solid Geometry and describes the fundamental way to work with solid 3D geometry. It involves creating complex objects by adding or removing pieces from solids using Boolean operations such as union, subtraction, or intersection.

As discussed earlier in this manual, FreeCAD supports various types of geometry. However, the preferred and most practical type for designing real-world 3D objects in FreeCAD is solid BREP geometry, primarily handled by the Part Workbench. BREP defines 3D objects by specifying their spatial boundaries. The key aspects of BREP include: faces, the surface elements of the object; edges, the boundary lines where two faces meet; and vertices, the points where edges converge.

BREP offers several advantages. First, it defines surfaces using mathematical equations, enabling precise and accurate modeling. This precision is crucial for engineering applications where exact dimensions are required. Additionally, BREP provides smooth and detailed surfaces, unlike polygon meshes that approximate curved surfaces with facets. This is similar to the difference between vector images, which scale without losing quality, and bitmap images, which can appear pixelated when enlarged. BREP retains comprehensive topological information about the object, including relationships between faces, edges, and vertices, which is essential for complex operations like Boolean calculations and filleting.



On the left a mesh representation and on the right a BREP representation

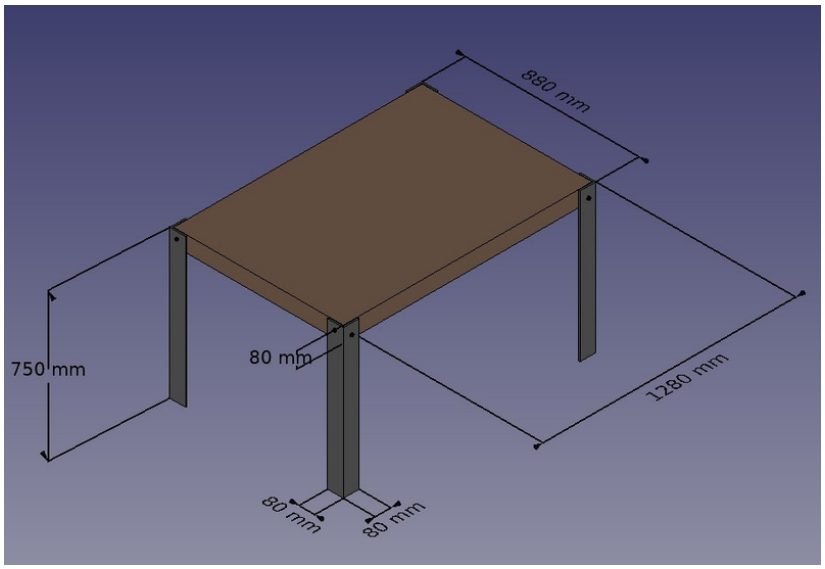
Polygon meshes consist of vertices, edges, and faces that form triangles or quadrilaterals. Meshes are simpler and faster for rendering but lack precision. When zoomed in or printed at large scales, meshes show faceted surfaces rather than smooth curves. In contrast, BREP utilizes curves and surfaces defined mathematically, offering superior accuracy and smoothness. BREP models are preferable for CAD applications where precision is required.

In FreeCAD, BREP-based geometry is managed by OpenCasCade, an open-source software library. The primary interface between FreeCAD and the OpenCasCade kernel is the Part Workbench, which serves as the foundation for most other workbenches, providing essential tools for creating and manipulating BREP objects. The Part Workbench includes tools for creating primitives, such as basic shapes like boxes, cylinders, and spheres. It also has tools for Boolean operations like fusing, intersecting, and subtracting shapes, as well as tools for moving, rotating, scaling, and cloning objects.

While other FreeCAD workbenches, such as the PartDesign and Surface Workbenches, offer more advanced tools for building and manipulating geometry, they rely on the underlying Part Workbench. Understanding how Part objects work internally and being

adept with the basic Part tools is beneficial. Often, these simpler tools can resolve issues that more complex tools may not handle effectively.

To illustrate the use of the Part Workbench, we will model this table using only CSG operations (except the screws, for which we will use one of the addons, and the dimensions, which will see in the next chapter):

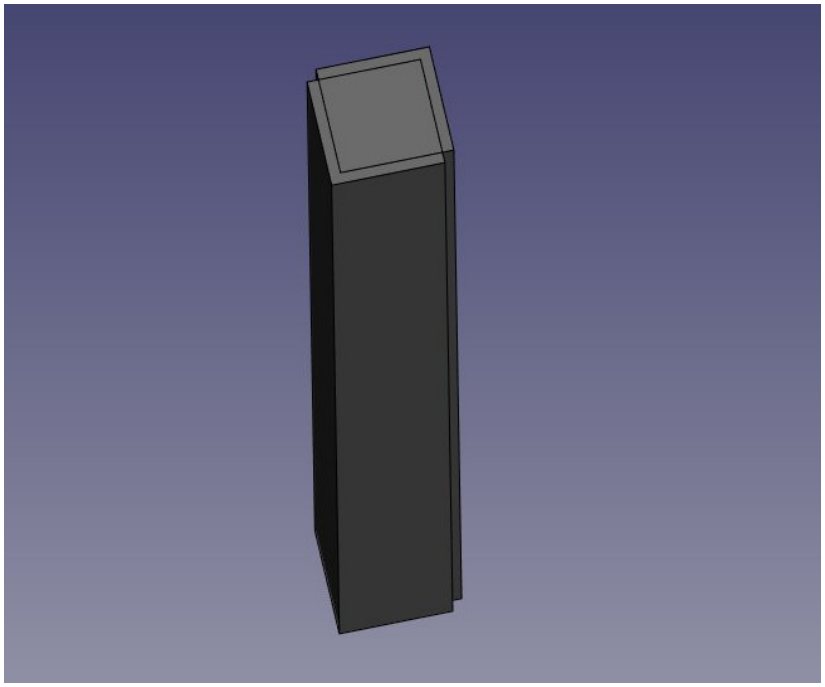


Let's create a new document (**Ctrl+N** or menu **File** → **New**) for our table design. The document is initially called "Unnamed" in the Model tab in the Combo View panel. If you save the document (**Ctrl+Shift+S** or menu **File** → **Save As**) as "table.FCStd", the document will be renamed "table", which more clearly identifies the project. We will use millimeters (mm) as our units of length. Feel free to change units by using the menu located on the lower right corner, according to your preference.

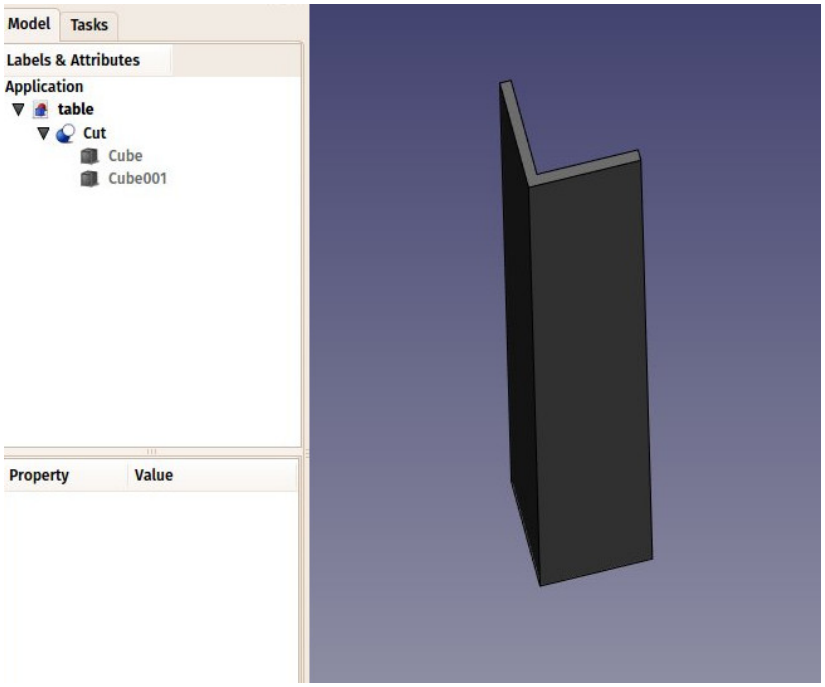
Now we can switch to the Part Workbench and create our first table leg.

- Press the **Cube** button
- Select the cube, then set the following properties (on the **Data** tab):
  - Length: 80 mm
  - Width: 80 mm
  - Height: 750 mm
- Duplicate the cube by pressing **Ctrl+C** then **Ctrl+V** (or menu **Edit** → **Copy** and **Paste**). You will not see any change in the 3D view, because the second object is overlaying the first. You can tell that the new cube has been pasted in because the Tree view now shows an object called "Cube001".
- Select Cube001 in the Tree view
- Change the position of Cube001 by editing its Placement property on the Data tab (click the arrow next to Position to expand):
  - Position x: 8 mm
  - Position y: 8 mm

You should now see two tall cubes, one shifted 8mm from the other in both X and Y directions:



- Now we can subtract one cube from the other to get our L-shaped table leg: Select the original cube, part of which will remain after the Cut operation. Then, with the Ctrl key pressed, select Cube001, which will be subtracted from the first. Note that the selection order determines the result of the Cut operation. Press the **Cut** button:

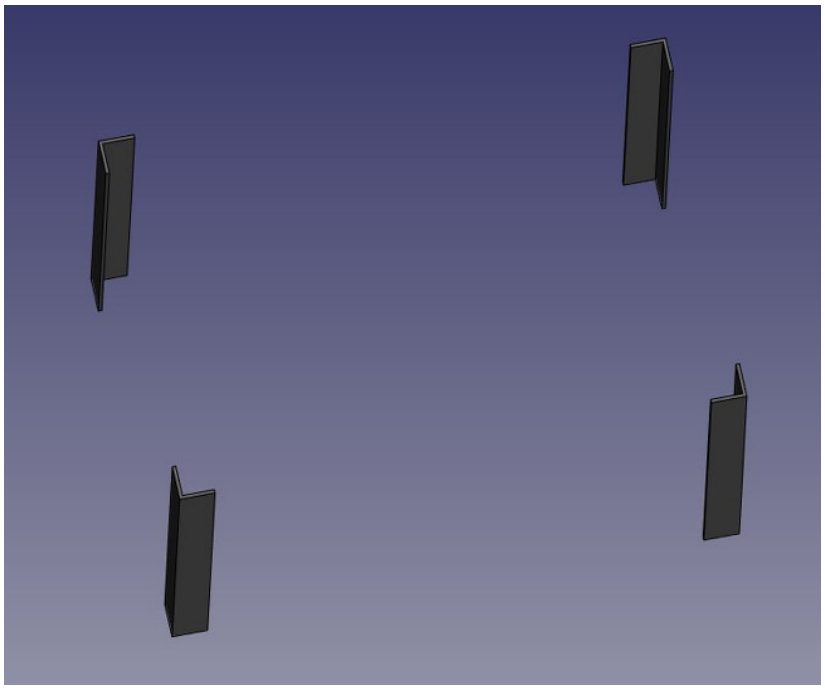


The newly created object, called "Cut", contains the two cubes we used as operands. In fact, the two cubes are still in the document, and have merely been hidden and grouped under the Cut object in the tree view. You can still select them by expanding the arrow next to the Cut object. If you wish, you can make them visible again by clicking the eye icons next to their object labels, in their right-click menus, or by changing their properties.

You can also perform cut and other Boolean operations with the Boolean tool.

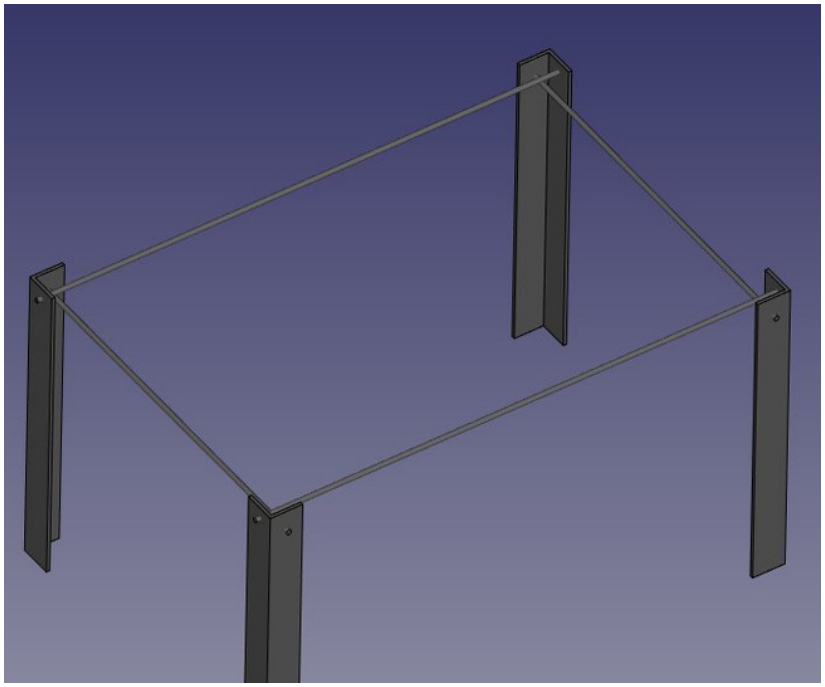
- Now let's create the three other table legs making six additional copies of our original cube. Since it is still copied on the clipboard, you can simply paste (Ctrl+V) 6 times. Change the position of each new cube as follows:
  - Cube002: x: 0, y: 800 mm
  - Cube003: x: 8 mm, y: 792 mm
  - Cube004: x: 1200 mm, y: 0
  - Cube005: x: 1192 mm, y: 8 mm
  - Cube006: x: 1200 mm, y: 800 m
  - Cube007: x: 1192 mm, y: 792 mm

- Now let's do three additional cut operations by selecting the "host" cube first and then the cube to be cut off. We now have four Cut objects:



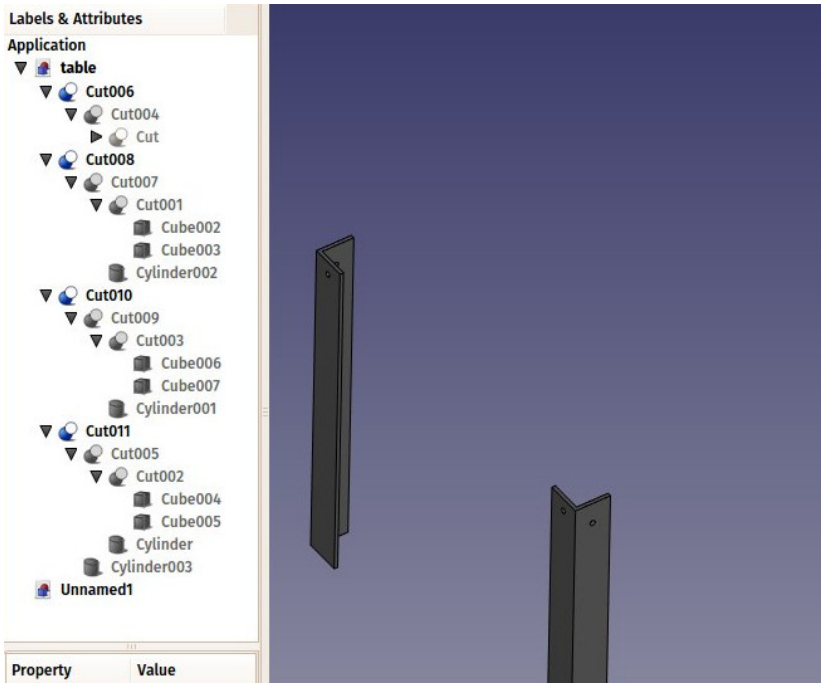
Instead of duplicating the base cube six times, we could have duplicated the complete leg three times by copying and pasting the Cut object created above from our first two cubes and rotating each cut leg into their proper orientation. In FreeCAD there are often multiple ways to achieve the same result. This is important to remember, because you may find it easier or more efficient to use different techniques in different contexts.

- We will now make holes for the screws, using the same Cut operation. Since we need 8 holes (two in each leg) we could make 8 objects to be subtracted. However, let's explore another way. We can make 4 cylinders, each intersecting a pair of legs. So, let's create them with the **Cylinder** tool. You can make one cylinder and duplicate it three times. Give each cylinder a radius of 6 mm. This time, we will need to rotate the cylinders using the **Placement** property under the Data tab *(**Note:** change the Axis property* before *setting the Angle, or the rotation will not be applied)*:
  - Cylinder: height: 1300 mm, angle: 90°, axis: x: 0, y: 1, z: 0, position: x: -10 mm, y: 40 mm, z: 720 mm
  - Cylinder001: height: 1300 mm, angle: 90°, axis: x: 0, y: 1, z: 0, position: x: -10 mm, y: 840 m, z: 720 mm
  - Cylinder002: height: 900 mm, angle: 90°, axis: x: -1, y: 0, z: 0, position: x: 40 mm, y: -10 mm, z: 700 m
  - Cylinder003: height: 900 mm, angle: 90°, axis: x: -1, y: 0, z: 0, position: x: 1240 mm, y: -10 mm, z: 700 mm



You will notice that the cylinders extend beyond the table legs. This is because, as in all solid-based 3D applications, boolean operations in FreeCAD sometimes fail when objects' faces are co-planar. We can avoid potential errors by putting the ends of the cylinders beyond the legs' surfaces.

- Now let's do the subtractions to create holes in the table legs. Select the first leg; then, with Ctrl pressed, select one of the cylinders that intersects it and press the **Cut** button. The hole will be created in the leg, and the cylinder will be hidden. You can find it in the tree view by expanding the leg's new cut object.
- Select the other leg that intersects by this hidden cylinder, and repeat the operation. This time, select the cylinder in the Tree view, since it is hidden in the 3D view. (Alternatively, you could make the cylinder visible again and select it in the 3D view.) Repeat this operation for the other legs until each of them has two holes:



As you can see, each leg is now described by a series of multiple operations nested in the Tree view. All the geometry we have created remains parametric, and you can change

any parameter of any of the older operations any time. In FreeCAD, we refer to this series as "modeling history", since it records the history of the operations we performed.

Another particularity of FreeCAD is that the concept of 3D object and the concept of 3D operation tend to blend into one. "Cut" refers to an operation, and also to the 3D object resulting from this operation. In FreeCAD this is sometimes called a "feature", rather than "object" or "operation".

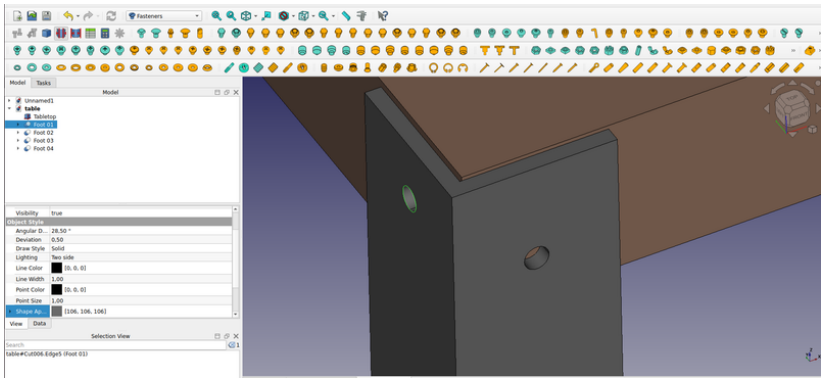
Now let's model the tabletop. It will be a simple rectangular prism, so let's start with another **Cube** and change its dimensions under the Data tab as follows:

- Box: length: 1260 mm, width: 860 mm, height: 80 mm, position: x: 10 mm, y: 10 mm, z: 670 mm.

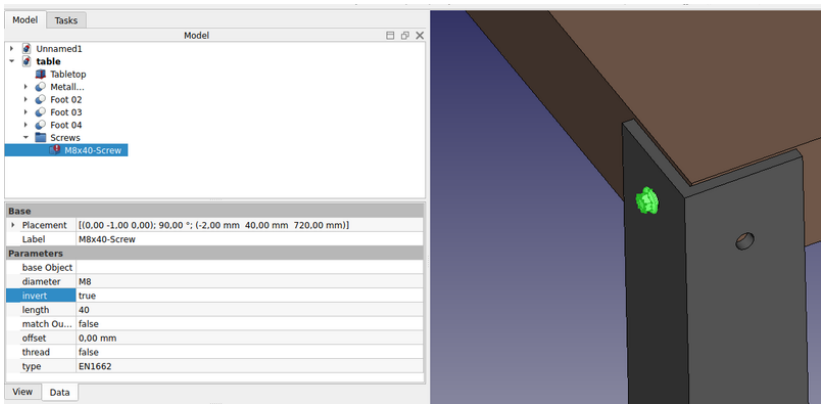
In the **View** tab, you can give it a brownish, wood-like color by changing its **Shape Appearance** property:

Now that our five pieces are complete, it is a good time to give them more descriptive names than "Cut015". By right-clicking each object in the tree view (or pressing **F2** with an object selected), you can rename them to something more meaningful. Properly naming your objects can be just as important as the way you model them.

- We will now insert some screws with an addon. Fasteners is an extremely useful addon developed by a member of the FreeCAD community. You can find it on the FreeCAD addons repository. Installing addon workbenches is easy! See the Addon manager for more information.
- Once you have installed the Fasteners Workbench and restarted FreeCAD, select Fasteners in the workbenches dropdown list. Let's add a screw to one of the holes we modeled above. First, select the circular edge of a hole in one of the table legs:



- Then, select one of the screws provided in the Fasteners Workbench. For this exercise, let's use the **EN 1665 Hexagon bolt with flanges, heavy series**. The screw will be placed in and aligned with our hole; and the diameter will automatically match the size of our hole. Sometimes the orientation of the screw will need to be flipped, using its **Invert** property:



- Repeat this for the other seven holes, and our table is complete!

As mentioned earlier, you can achieve the same result by following different steps. To demonstrate this, let's create the same table using a different methodology. Remember, there is no right or wrong way - just individual creativity.

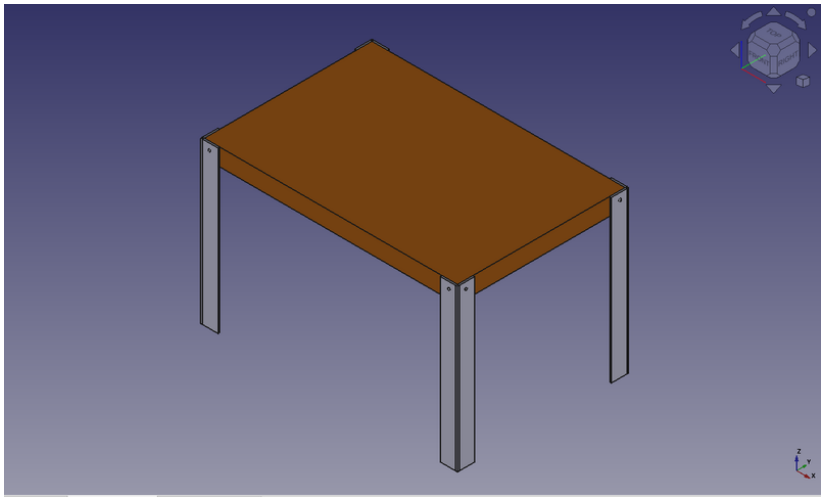
We will start in a similar way, by creating a cube with the following dimensions: length 80 mm, width 8 mm, and height 750 mm

- Create a cube by selecting the **Cube** button and set the following properties (in the **Data** tab):
  - Length: 80 mm
  - Width: 8 mm
  - Height: 750 mm
- Next, we will create a **Cylinder** with the following properties:
  - radius: 6 mm, height: 100 mm, angle: 90°, axis: x: 1, y: 0, z: 0, position: x: 40 mm, y: 40 mm, z: 720 mm
- Next, we will apply the Cut operation. Select the cube; then, hold the Ctrl key and select the cylinder. Keep in mind that the order is important to define which one stays. Then, press the **Cut** button.
- We will then copy and paste the cut object by pressing **Ctrl+C** then **Ctrl+V** (or menu **Edit** → **Copy** and **Paste**):
  - angle: 90°, axis: x: 0, y: 0, z: 1, position: x: 8 mm
- Select the two objects and apply the **Fuse** tool. Now the two objects are fused, and we have an L-shaped table leg.
- Copy and paste the fused leg, positioning it at
  - angle: 90°, axis: x: 0, y: 0, z: -1, position y: 800 mm.
- Select the two legs and create a **Compound**.
- Copy and paste the compound, positioning it at:
  - angle: 180°, axis: x:0, y:0, z:1, position x: 1200 mm, y: 800 mm. We have our legs.

Let's create the table top.

- Create a Cube, and edit its properties as follows:
  - Length: 1184 mm
  - Width: 784 mm
  - Height: 80 mm
  - position x: 8 mm, y: 8 mm, z: 670 mm.

Now, continue adding screws in the Fasteners Workbench as before.



The internal structure of Part objects

As we saw above, in FreeCAD it is possible to select not only whole objects but parts of them, e.g. the circular edge of our screw hole. This is a good time to have a quick look at how Part objects are constructed internally. Every workbench that produces Part geometry will be based on these:

- **Vertices**: These are points (usually endpoints) on which all the rest is built. For example, a line has two vertices.
- **Edges**: Edges can take the form of lines, arcs, ellipses or NURBS curves. They usually have two vertices, but some special cases have only one (e.g. a closed circle).
- **Wires**: A wire is a sequence of edges connected by their endpoints. A wire can contain edges of any type, and it can be closed or not.
- **Faces**: Faces can be planar or curved. They can be defined by one closed wire forming the border of the face, or by more than one if the face has holes.
- **Shells**: Shells are groups of faces connected by their edges. They can be open or closed.
- **Solids**: A closed shell can be turned into a solid. Solids carry the notion of inside and outside. Many workbenches rely on this to make sure the objects they produce can be built in the real world.
- **Compounds**: Compounds combine other shapes of one or more types into a single object.

In the 3D view, you can select individual **vertices**, **edges** or **faces**. Selecting one of these also selects the whole object it belongs to.

A note about shared design

You might look at the table above and think its design is not very good. The legs don't seem attached to the tabletop very securely! You might want to add reinforcing pieces, or you might have other improvements in mind. This is where sharing becomes interesting. You can download the file made during this exercise from the link below, and modify it to make it better. Then, if you share that improved file, others might be able to make it even better or use your well designed table in their projects. Your design might then give other ideas to other people, and maybe you will have contributed to a better world...

Downloads

- The file produced in this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/table.FCStd

Read more

- The Part Workbench
- The FreeCAD addons repository
- The Fasteners Workbench

Using spreadsheets

The Spreadsheet Workbench in FreeCAD allows users to create and manage spreadsheets, such as those made with Excel or Calc from LibreOffice, directly within their design projects. It enables inputting, organizing, and manipulating data in a table format, which can then be linked to various parameters and models in the project.

One of the key advantages is its use in parametric modeling. Spreadsheets can be linked to the dimensions and properties of 3D models, making them an essential tool for dynamic design changes. For example, adjusting a value in the spreadsheet will automatically update the corresponding dimension in the model.

In addition to managing values, the workbench is excellent for data management, storing critical information such as material properties, dimensions, and project-wide parameters. This becomes particularly useful in complex projects where multiple values need to be referenced or adjusted.

Spreadsheets also allow users to input formulas for calculations and data management. These formulas can reference other spreadsheet cells or parameters within the 3D model, making the entire design process adaptable and responsive to changes.

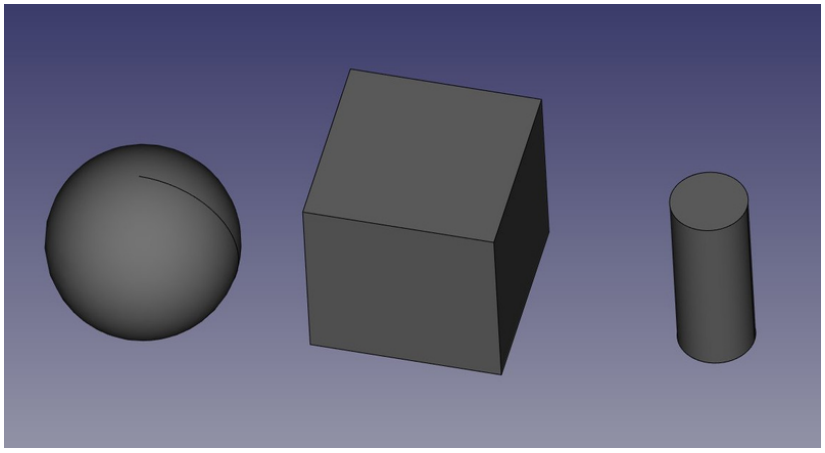
It can integrate seamlessly with other FreeCAD workbenches, allowing interaction between data and model components. This integration centralizes control over different aspects of the project, making it easier to manage. The interface is straightforward, resembling traditional spreadsheet software, which makes it familiar and easy to use for those already accustomed to programs like Excel or LibreOffice Calc.

In practice, the Spreadsheet Workbench is versatile for different use cases, including defining project-wide parameters, managing bills of materials (BOM), and performing custom calculations that influence design decisions. It simplifies complex projects by centralizing the control of parameters in one location.

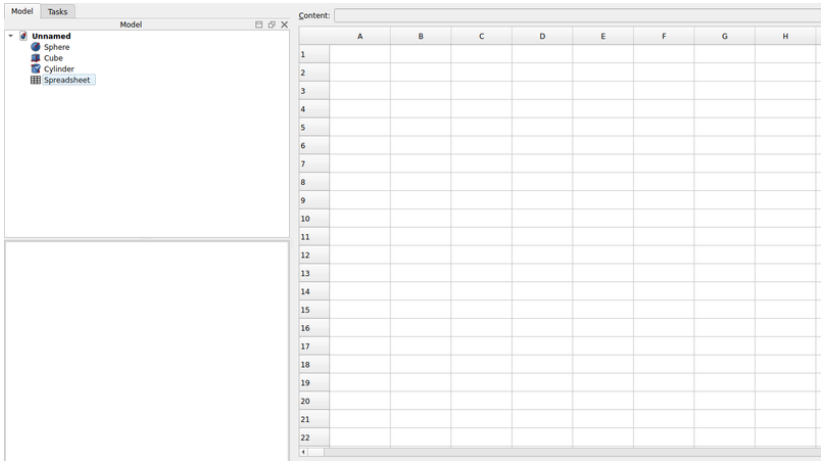
In the following example, we will create a couple of objects, retrieve some of their properties in a spreadsheet, then use the spreadsheet to directly drive the properties of other objects.

Reading properties[edit | edit source]

- Start by switching to the Part Workbench, and create a couple of objects: a box, a cylinder and a sphere.
- Edit their **Placement** property (or use the Draft Move tool) to place them a little apart, so we can better see the effects of what we'll do:



- Now, let's extract some information about these objects. Switch to the Spreadsheet Workbench
- Press the **New Spreadsheet** button
- Double-click the new Spreadsheet object in the tree view. The spreadsheet editor opens:

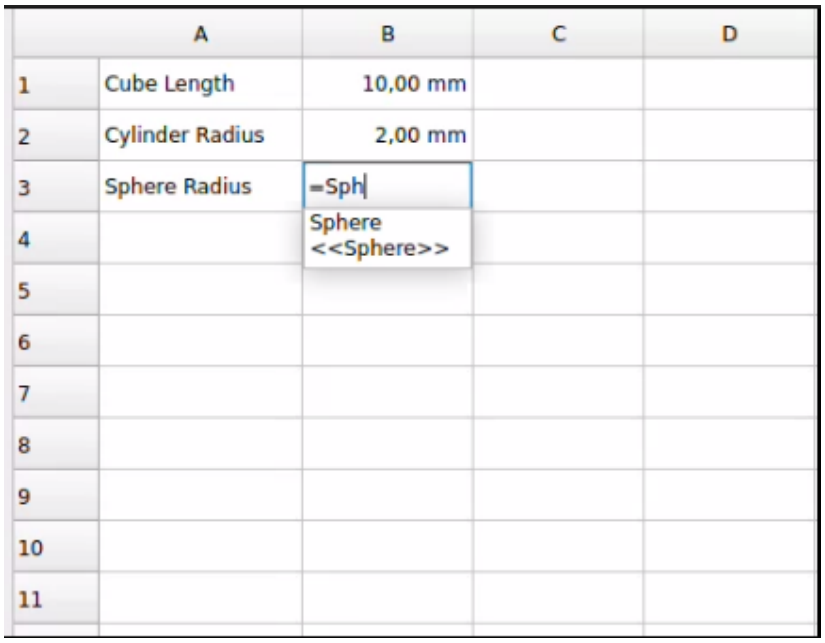


Although FreeCAD's spreadsheet editor is not as feature-rich as dedicated applications like Excel or LibreOffice Calc, it provides essential tools for most design tasks. Users can adjust cell properties such as size, color, and alignment, and merge or split cells for better organization. Basic formulas or references to other cells are supported, allowing for simple data manipulation. What sets it apart is its deep integration with FreeCAD's modeling environment, where changes in the spreadsheet can automatically update model dimensions in real time. While it may lack advanced features like pivot tables or charts, its focus on parametric-driven design makes it a powerful tool for managing design data directly within FreeCAD.

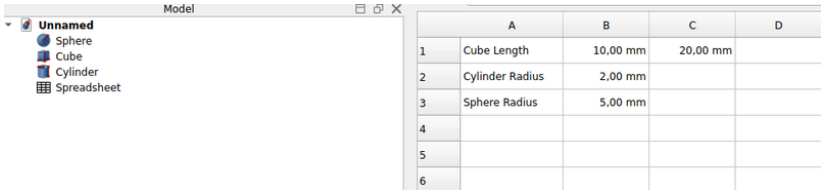
In FreeCAD, beyond the standard spreadsheet features, there is a particularly useful function: the ability to reference not just other cells, but also objects within the document and extract values from their properties. For instance, you can retrieve properties from 3D objects that are visible in the **Data** tab of the **Properties Editor** when an object is selected. This allows for seamless integration between the spreadsheet and the 3D model, making it easy to link and automate changes based on the parameters of objects within the design, offering a more dynamic and interconnected workflow.

- Let's start by entering a couple of texts in the cells A1, A2 and A3, so we remember what is what later on, for example **Cube Length**, **Cylinder Radius** and **Sphere Radius**. To enter text, just write in the "Contents" field above the spreadsheet, or double-click a cell.

- Now let's retrieve the actual length of our cube. In cell B1, type **=Cube.Length**. You will notice that the spreadsheet has an auto-completion mechanism, which is actually the same as the expression editor we used in the previous chapter.
- Do the same for cell B2 (**=Cylinder.Radius**) and B3 (**=Sphere.Radius**).



- Although these results are expressed with their units, the values can be manipulated as any number, try for example entering in cell C1: **=B1*2**.
- We can now change one of these values in the properties editor, and the change will be immediately reflected in the spreadsheet. For example, let's change the length of our cube to **20mm**:



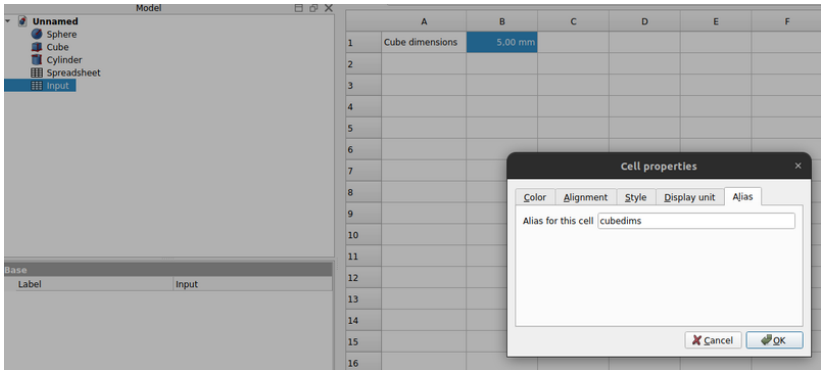
The Spreadsheet Workbench page will describe in more detail all the possible operations and functions available in spreadsheets.

Writing properties[edit | edit source]

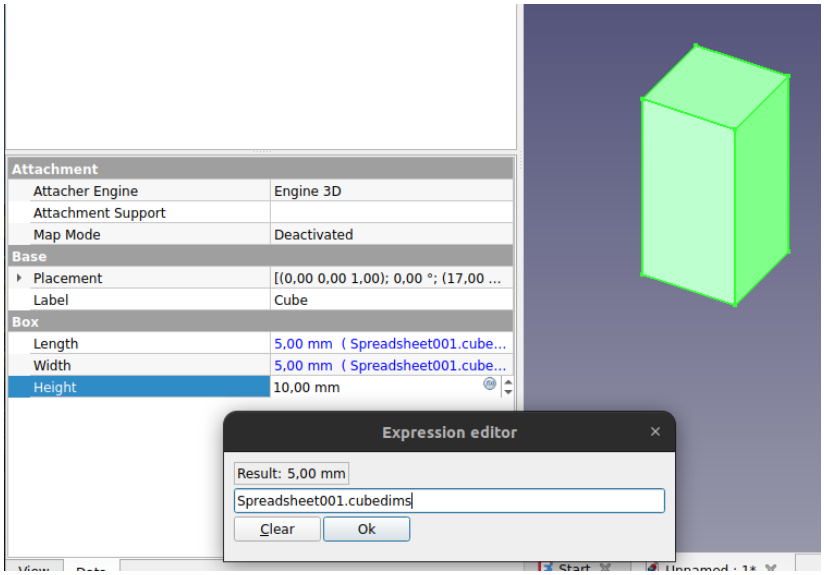
Another powerful feature of the Spreadsheet Workbench in FreeCAD is the ability to not only read values from the properties of 3D objects but also assign values to them. This allows for controlling the dimensions and attributes of objects directly from the spreadsheet. However, one of FreeCAD's fundamental rules is that circular dependencies are forbidden—meaning a spreadsheet cannot both read from and write to the same object. Doing so would create a situation where the object depends on the spreadsheet while the spreadsheet also depends on the object, leading to an invalid configuration. To avoid this, a second spreadsheet is typically created to handle writing values, ensuring a clear separation between the reading and writing processes.

- We can now close the spreadsheet tab (under the 3D view). This is not mandatory, there is no problem in keeping several spreadsheet windows open.
- Press the **New Spreadsheet** button again
- Change the name of the new spreadsheet to something more meaningful, such as **Input** (do this by right-clicking the new spreadsheet object, and choosing **Rename**).

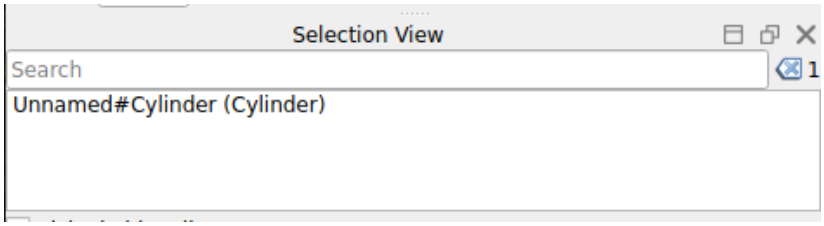
- Double-click the Input spreadsheet to open the spreadsheet editor.
- In cell A1, let's put a descriptive text, for example: "Cube dimensions"
- In cell B1, write **=5mm** (using the = sign makes sure the value is interpreted as a unit value, not a text).
- Now to be able to use this value outside the spreadsheet, we need to give a name, or alias, to the B1 cell. Right-click the cell, click **Properties** and select the **Alias** tab. Give it a name, such as **cubedims**:



- Press **OK**, then close the spreadsheet tab
- Select the cube object
- In the properties editor, click the little  **expression** icon at the right side of the **Length** field. This will open the expressions editor, where you can write **Spreadsheet001.cubedims**. Repeat this for **Height** and **Width**:



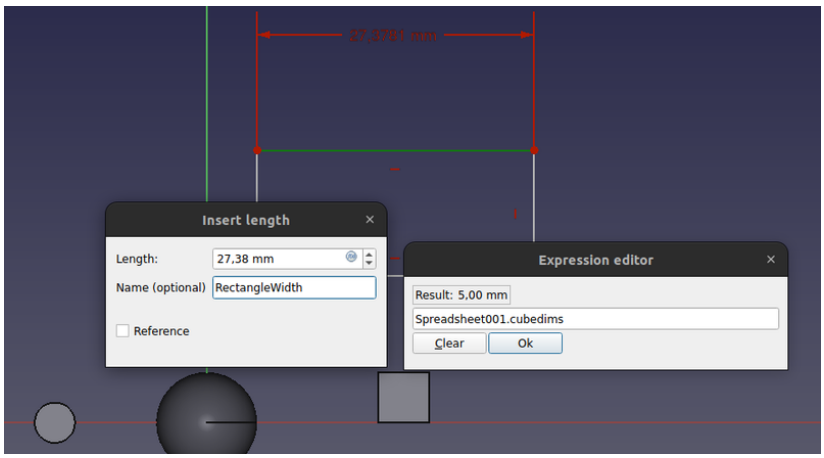
The reason we use "Spreadsheet001" instead of "Input" in the expression is that every object in a FreeCAD document has a unique internal name and a more user-friendly label. While the label is what appears in the tree view, the internal name is used to uniquely identify objects within the document. FreeCAD allows you to assign the same label to multiple objects if you adjust your preferences, but the internal name remains unique. For any operation that requires identifying an object without ambiguity, FreeCAD uses the internal name rather than the label, as the label could refer to more than one object. To find the internal name of an object, it's useful to keep the Selection Panel (accessible via View → Panels) open. This panel will always display the internal name of the selected object, ensuring you use the correct reference in your expressions.



By using cell aliases in FreeCAD's Spreadsheet Workbench, it's possible to store "master values" within the document, making it easy to manage and modify key parameters. For instance, a spreadsheet can hold the dimensions of a model, allowing these values to be referenced throughout the design. This approach simplifies the process of updating the model; if new dimensions are required, you can simply open the spreadsheet, adjust the values, and the model will automatically update to reflect these changes. This method streamlines versioning and improves efficiency, especially in parametric modeling, where dimensions frequently change based on project requirements.

Finally, note that the constraints inside a sketch can also receive the value of a spreadsheet cell:

You can also give aliases to dimensional constraints (horizontal, vertical or distance) in a sketch (you can then use that value from outside the sketch as well):



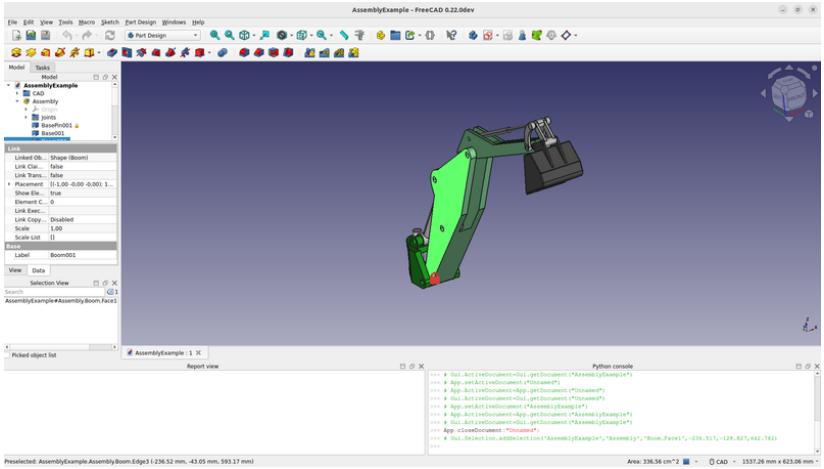
Download

- The file produced in this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/spreadsheet.FCStd

Read more

- The Spreadsheet Workbench
- The Expressions engine

What is FreeCAD



As mentioned in the previous section, FreeCAD is a parametric 3D computer-aided design (CAD) modeling application. Parametric design is a method of designing objects based on defining parameters and relationships between them. In this approach, changes to parameters automatically propagate throughout the associated elements of the design, allowing for rapid iterations and adaptations.

Commonly used in fields such as architecture, engineering, and product design, parametric design helps create complex shapes and forms efficiently, with the ability to adjust and fine-tune various design aspects through the modification of input values. This approach is especially powerful in software like FreeCAD, where users can set these parameters dynamically to automate and optimize the design process.

FreeCAD is a multiplatform application that runs on Windows, macOS, and Linux, and is completely open-source. This means it is free to use, modify, and distribute. FreeCAD is developed by an enthusiastic community rather than a single company, which allows for constant improvements and updates.

The software includes many open-source components and can be integrated into other applications. It is known for being highly customizable and supports a variety of file formats. This flexibility and the robust support from its community make FreeCAD a powerful tool in the open-source world. Developed collectively by a global community of programmers, enthusiasts, and users, FreeCAD exemplifies the collaborative spirit of open-source projects. It offers significant advantages such as customization and scriptability, bringing sophisticated features typically found in professional software within the reach of any user interested in 3D modeling and design.

The official website of FreeCAD is at https://freecad.org

Read more:

- About FreeCAD
- List of features
- Screenshots and user cases