FREECAD

A MANUAL

BY YORIK VAN HAVRE
AND THE FREECAD COMMUNITY

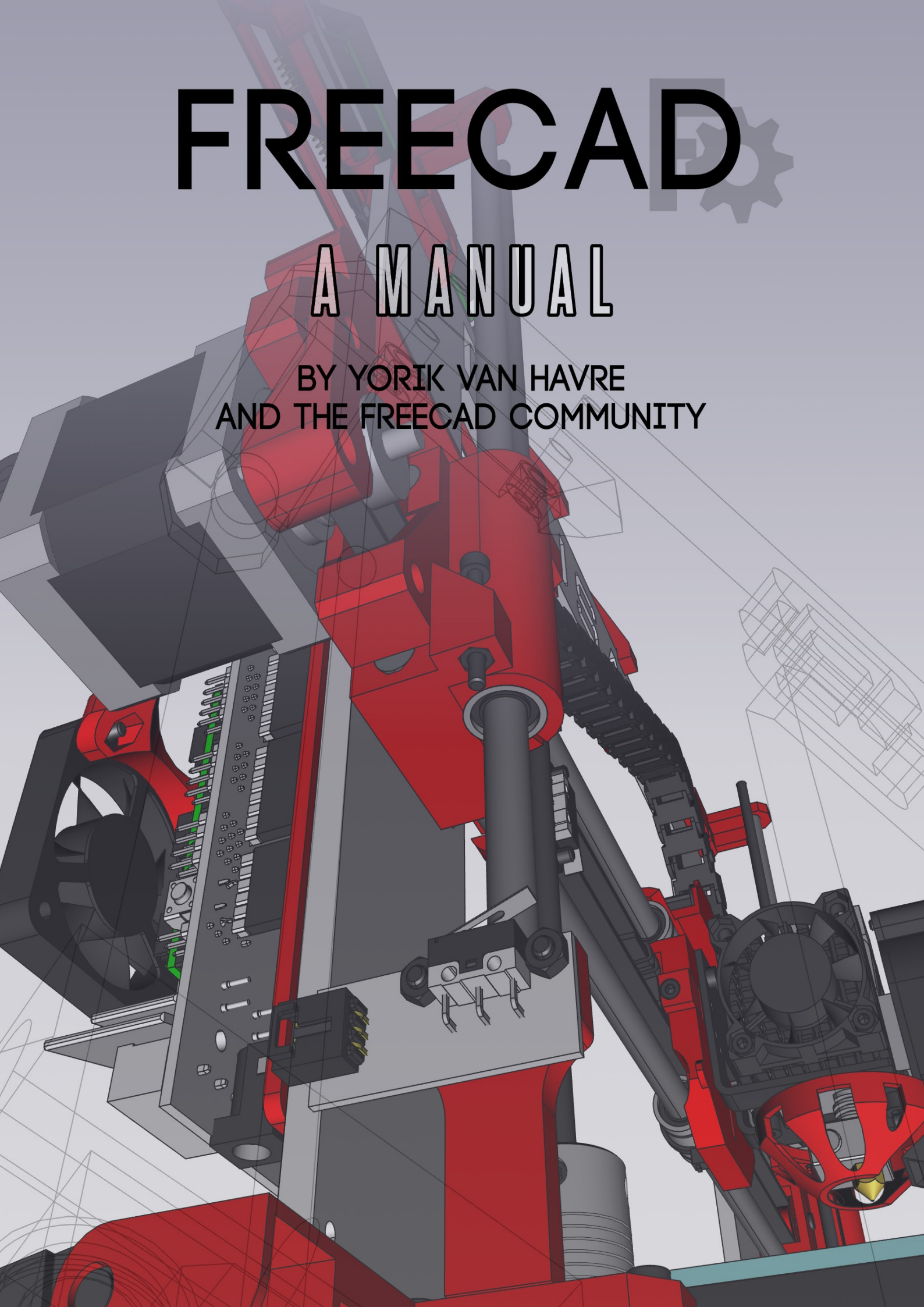


Table of Contents

A FreeCAD manual

Introduction

FreeCAD is a free, open-source parametric 3D modeling application. It is made primarily to model real-world objects, ranging from the small electronic components up to buildings and civil engineering projects, with a strong focus on 3D-printable objects. FreeCAD is free to download, use, distribute and modify, and its source code is open and published under the very permissive LGPL license. The data you produce with FreeCAD is fully yours, and can be recovered without FreeCAD.

FreeCAD is also fundamentally a social project, as it is developed and maintained by a community of developers and users united by their passion for FreeCAD.

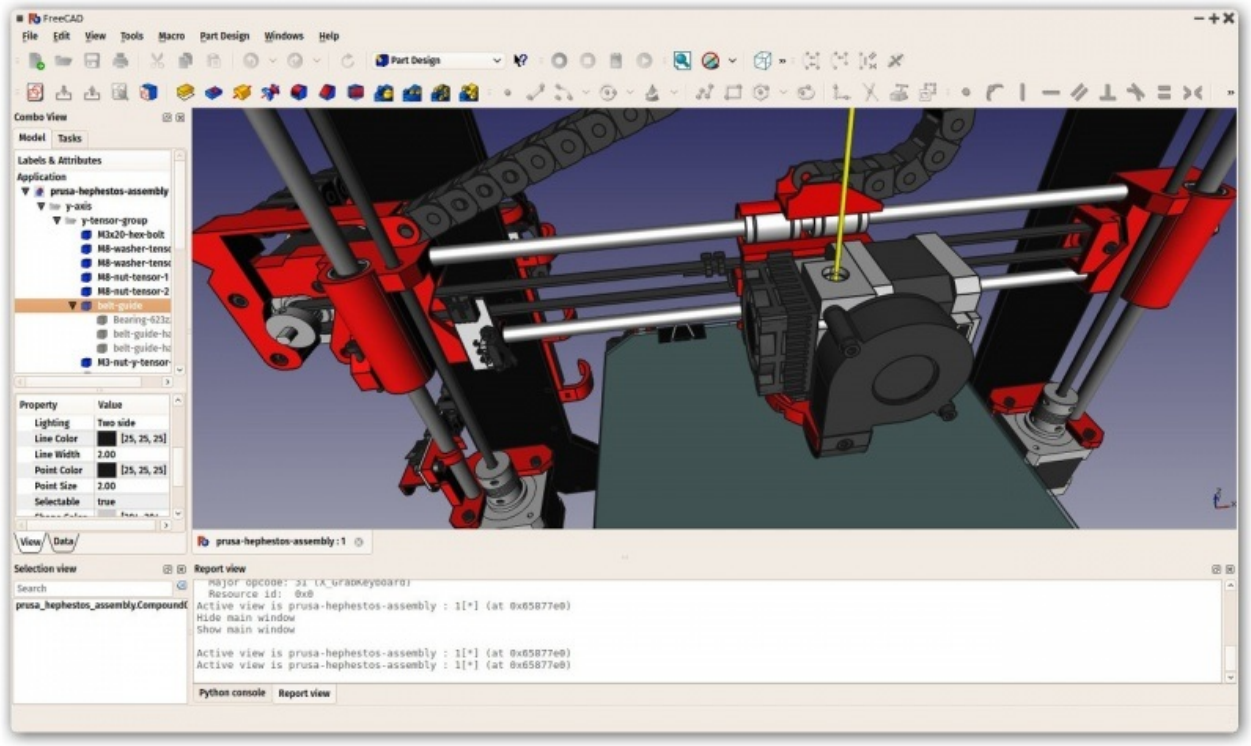
This manual is an experiment at taking the opposite way from the official FreeCAD documentation wiki. The wiki is written collaboratively by dozens of community members and, like most wikis, it contains huge amounts of information, but is very hard to access and navigate by newcomers. This turns it a precious resource for reference, but not a very practical tool to learn FreeCAD. This manual will walk you through the same information available on the wiki. However, we hope that the more step-by-step pace, based on examples, and the more unified tone given by a smaller number of authors, will make it more suitable for a first contact with FreeCAD, and that it will become a perfect companion for the wiki.

This manual has been written for the current stable version of FreeCAD which is version **0.16**.

This book has been written mostly by Yorik, but using a lot of information built by FreeCAD users, mostly from the FreeCAD wiki. The real author of this book is actually the whole FreeCAD community!

Discovering FreeCAD

What is FreeCAD?



FreeCAD is an open-source parametric 3D modeling application, made primarily to design real-life objects. Parametric modeling describes a certain type of modeling, where the shape of the 3D objects you design are controlled by parameters. For example, the shape of a brick might be controlled by three parameters: height, width and length. In FreeCAD, as in other parametric modelers, these parameters are part of the object, and stay modifiable at any time, after the object has been created. Some objects can have other objects as parameters, for example you could have an object that takes our brick as input, and creates a column from it. You could think of a parametric object as a small program that creates geometry from parameters.

FreeCAD is not designed for a particular kind of work, or to make a certain kind of objects. Instead, it allows a wide range of uses, and permits users to produce models of all sizes and purposes, from small electronic components to 3D-printable pieces and all the way up to buildings. Each of these tasks have different dedicated sets of tools and workflows available.

FreeCAD is also multiplatform (it runs exactly the same way on Windows, Mac OS and Linux platforms), and open-source. Being open-source, FreeCAD benefits from the contributions and efforts of a large community of programmers, enthusiasts and users worldwide. FreeCAD is essentially an application built by the people who use it, instead of being made by a company trying to sell you a product. And of course, it also means that FreeCAD is free, not only to use, but also to distribute, copy, modify, or even sell.

FreeCAD also benefits from the huge, accumulated experience of the open-source world. In its bowels, several other components of the open-source world are used, as FreeCAD itself can be used as a component in other applications. It also possesses all kinds of features that have become a standard in the open-source world, such as supporting a wide range of file formats, being hugely scriptable, customizable and modifiable. All made possible through a dynamic and enthusiast community of users.

The official website of FreeCAD is at http://www.freecadweb.org

Read more:

- About FreeCAD: http://www.freecadweb.org/wiki/index.php?title=About_FreeCAD
- List of features: http://www.freecadweb.org/wiki/index.php?title=Feature_list
- Screenshots and user cases: http://forum.freecadweb.org/viewforum.php?f=24

Installing

FreeCAD uses the LGPL license, which makes you free to download, install, redistribute and use FreeCAD the way you want, regardless of the type of work you'll do with it (commercial or non-commercial). You are not bound to any clause or restriction, and the files you produce with it are fully yours. The only thing that the license prohibits, really, is to claim that you programmed FreeCAD yourself!

FreeCAD runs without any difference on Windows, Mac OS and Linux. However, the ways to install it differ slightly depending on your platform. On Windows and Mac, the FreeCAD community provides precompiled packages (installers) ready to download, while on Linux, the source code is made available to Linux distributions maintainers, who are then responsible for packaging FreeCAD for their specific distribution. As a result, on Linux, you can usually install FreeCAD right from the software manager application.

The official FreeCAD download page for Windows and Mac OS is https://github.com/FreeCAD/FreeCAD/releases

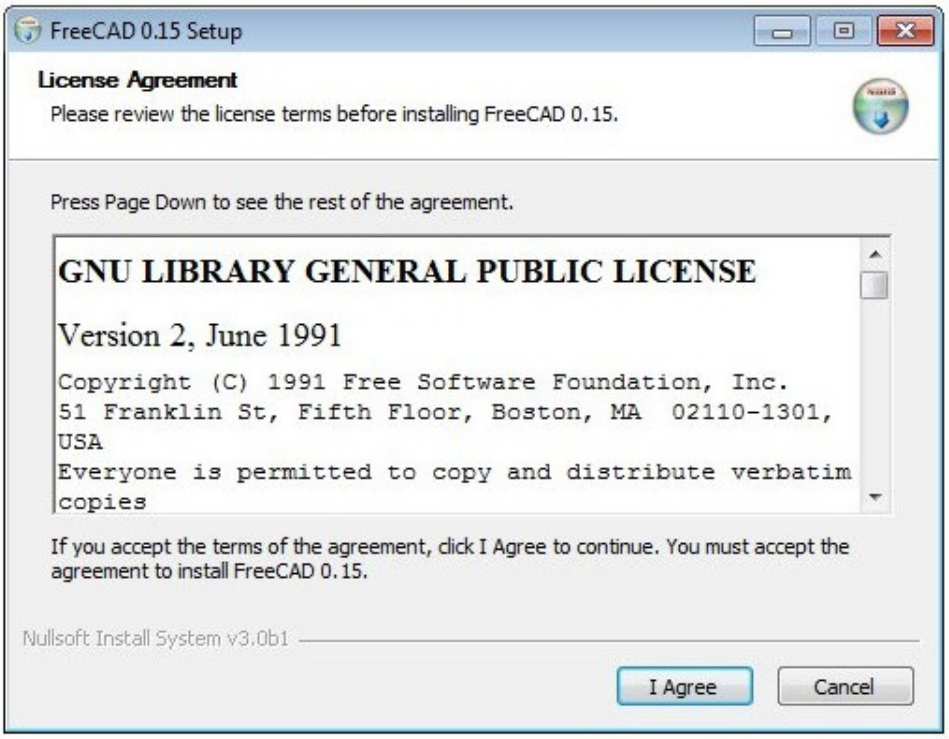
FreeCAD versions

The official releases of FreeCAD, that you can find on the page above and in your distribution's software manager, are stable versions. However, the development of FreeCAD is fast! New features and bug fixes are added almost every single day. Since it can sometimes take a long time between stable releases, you might be interested in trying a more bleeding-edge version of FreeCAD. These development versions, or pre-releases, are uploaded from time to time to the download page mentioned above, or, if you are using Ubuntu, the FreeCAD community also maintains PPA (Personal Package Archives) or 'daily builds' which are regularly updated with the most recent changes.

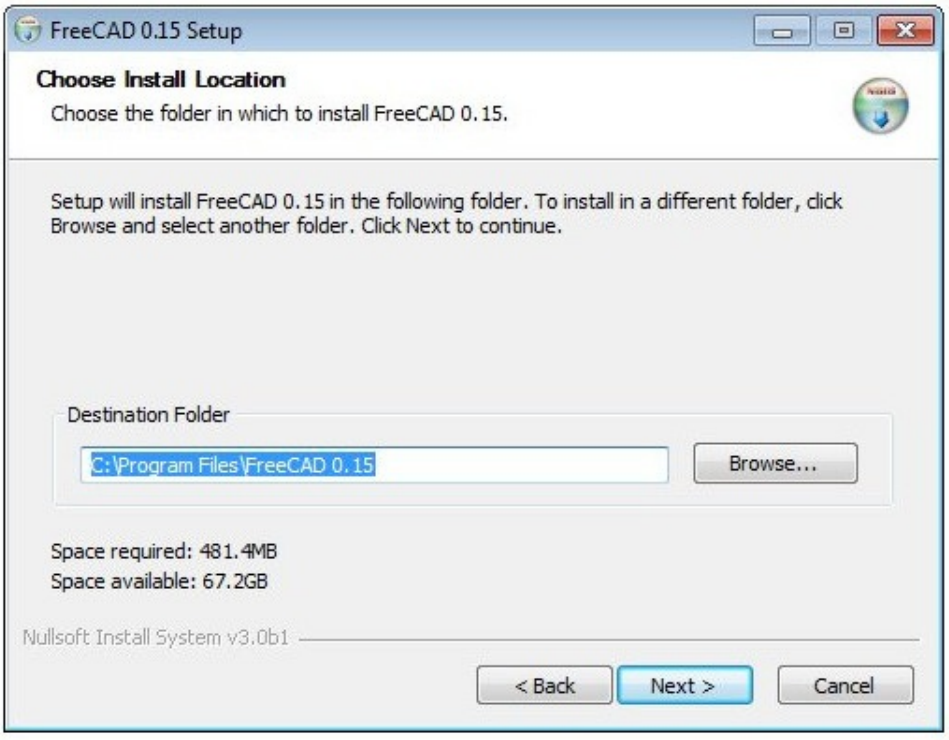
If you are installing FreeCAD in a virtual machine, please be aware that the performance might be low, or in some cases unusable due to the limits of OpenGL support on most virtual machines.

Installing on Windows

1. Download an installer (.exe) package corresponding to your version of Windows (32bit or 64bit) from the download page. The FreeCAD installers should work on any windows version starting from Windows 7.
2. Double-click the downloaded installer.
3. 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):

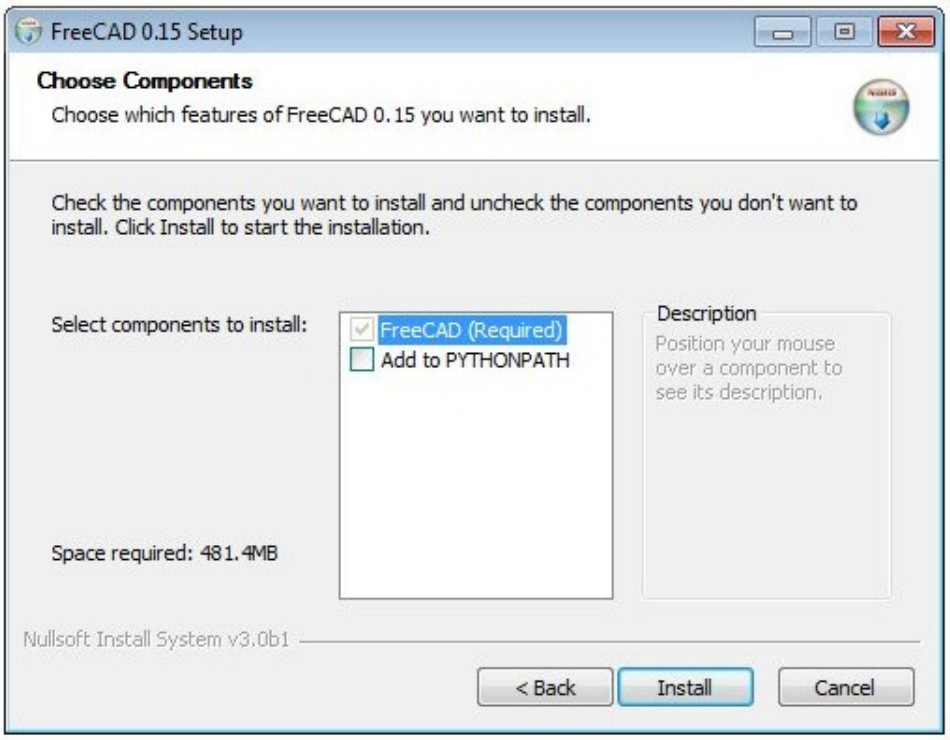


4. You can leave the default path here, or change if you wish:

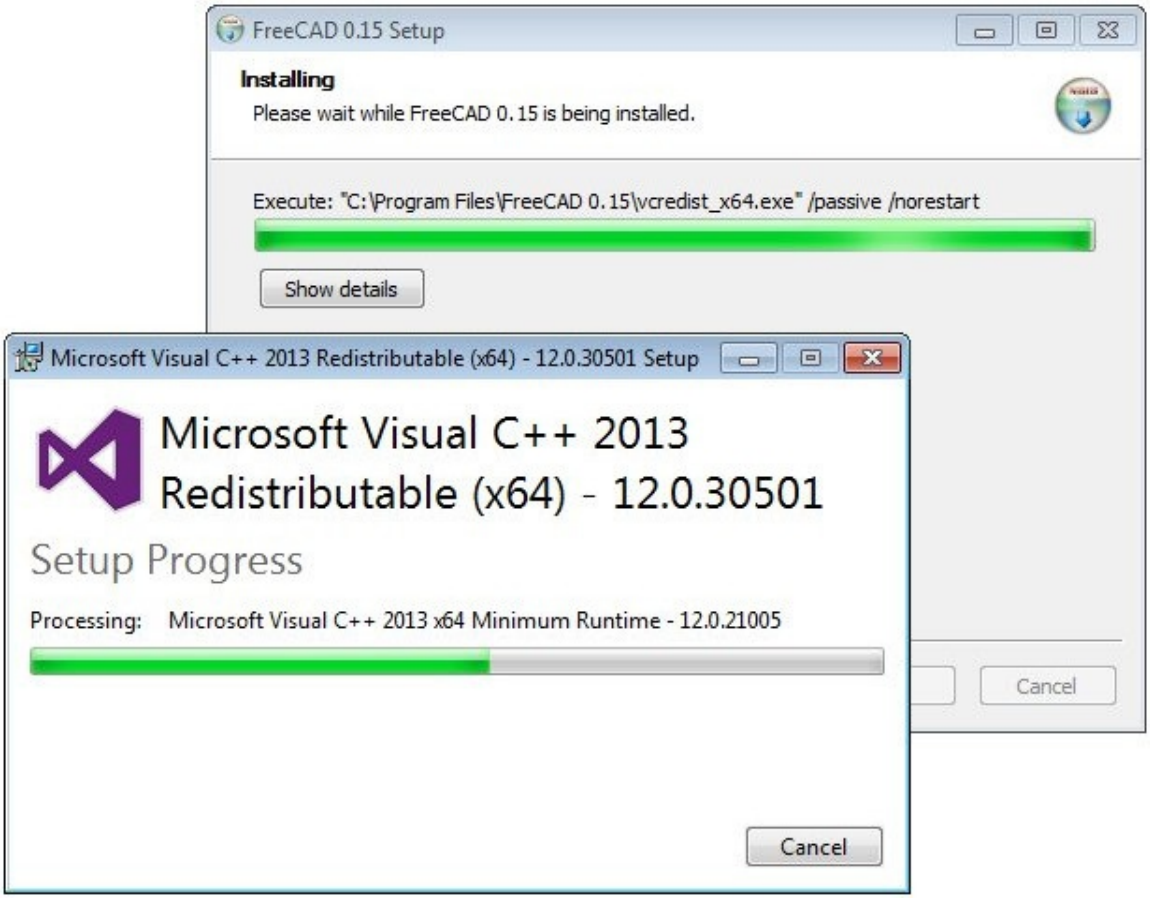


5. No need to set the PYTHONPATH variable, unless you plan to do some advanced

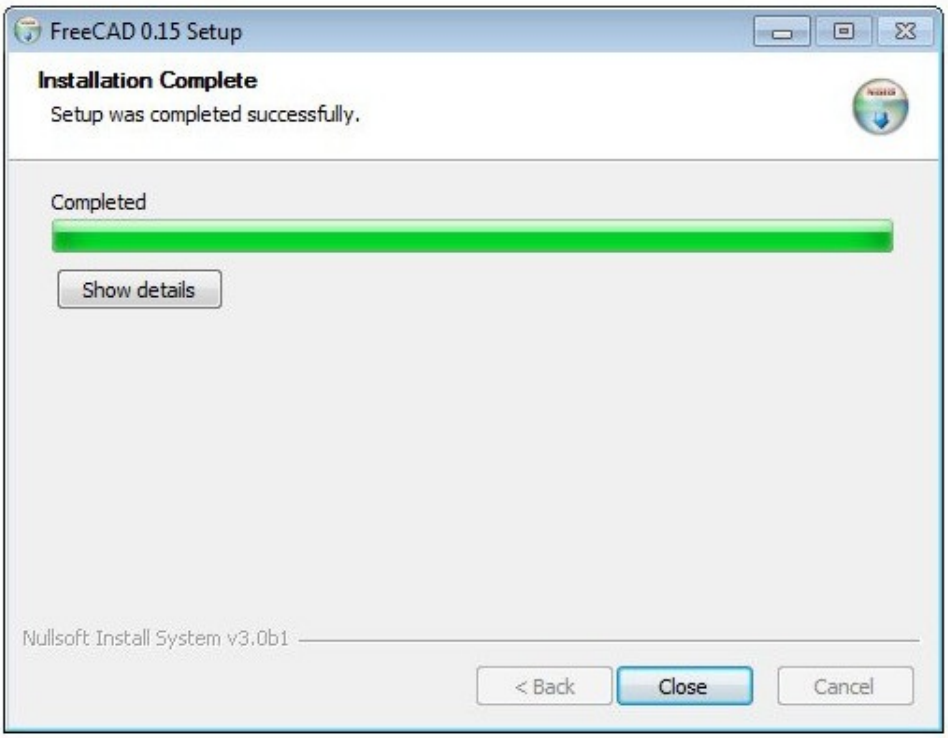
python programming, in which case you probably already know what this is for:



6. During the installation, a couple of additional components, which are bundled inside the installer, will be installed too:



7. That's it, FreeCAD is installed. You will find it in your start menu.



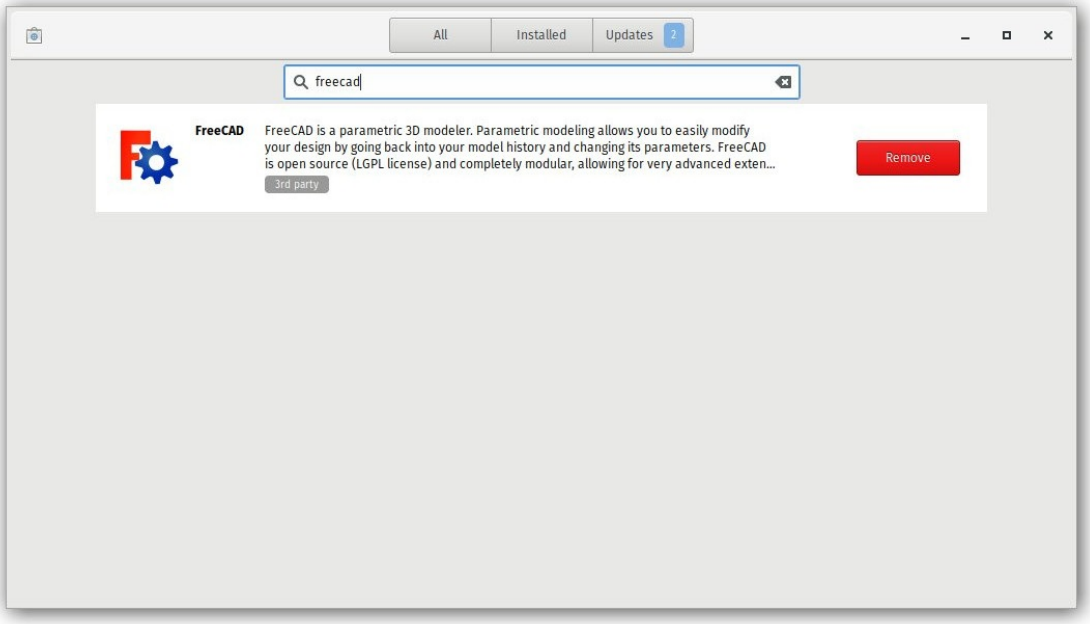
Installing a development version

Packaging FreeCAD and creating an installer takes some time and dedication, so usually, development (also called pre-release) versions are provided as .zip (or .7z) archives. These don't need to be installed, just unpack them and lauch FreeCAD by double-clicking the FreeCAD.exe file that you will find inside. This also allows you to keep both the stable and "unstable" versions together on the same computer.

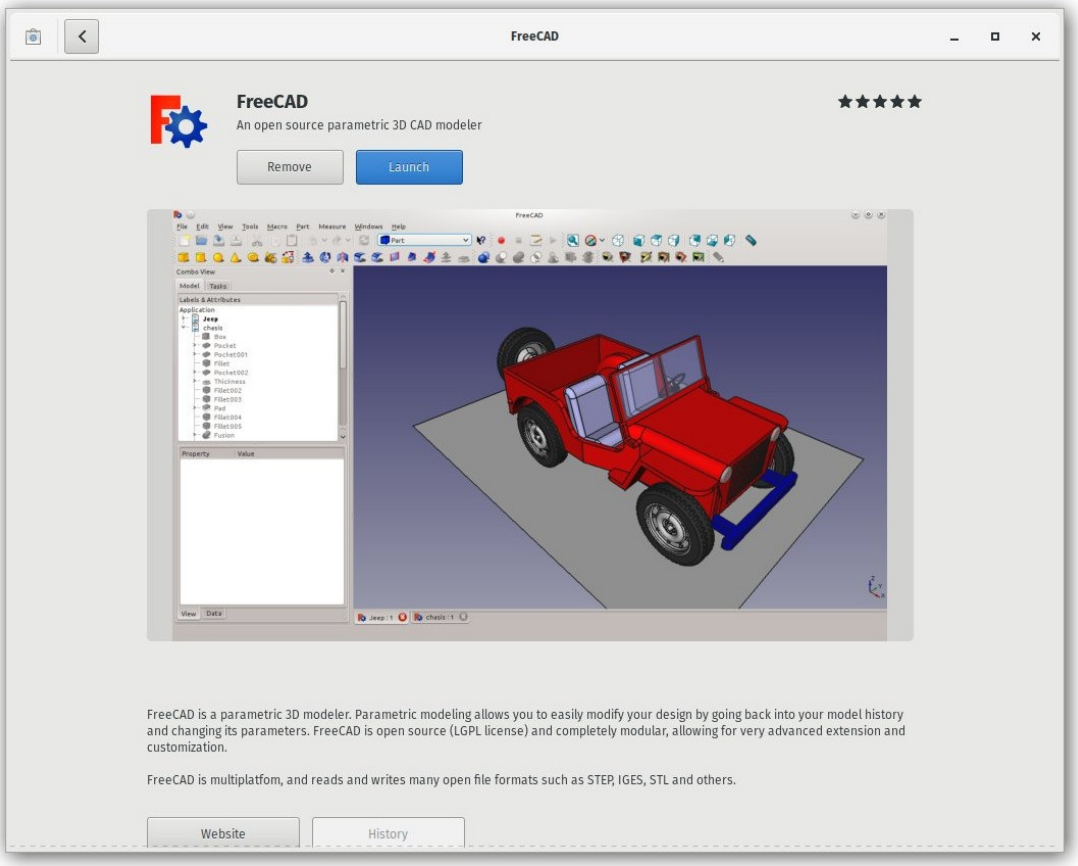
Installing on Linux

On most modern Linux distributions (Ubuntu, Fedora, OpenSUSE, Debian, Mint, Elementary, etc), FreeCAD can be installed with the click of a button, directly from the software management application provided by your distribution (the aspect of it can differ from the images below, each distribution uses its own tool).

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!



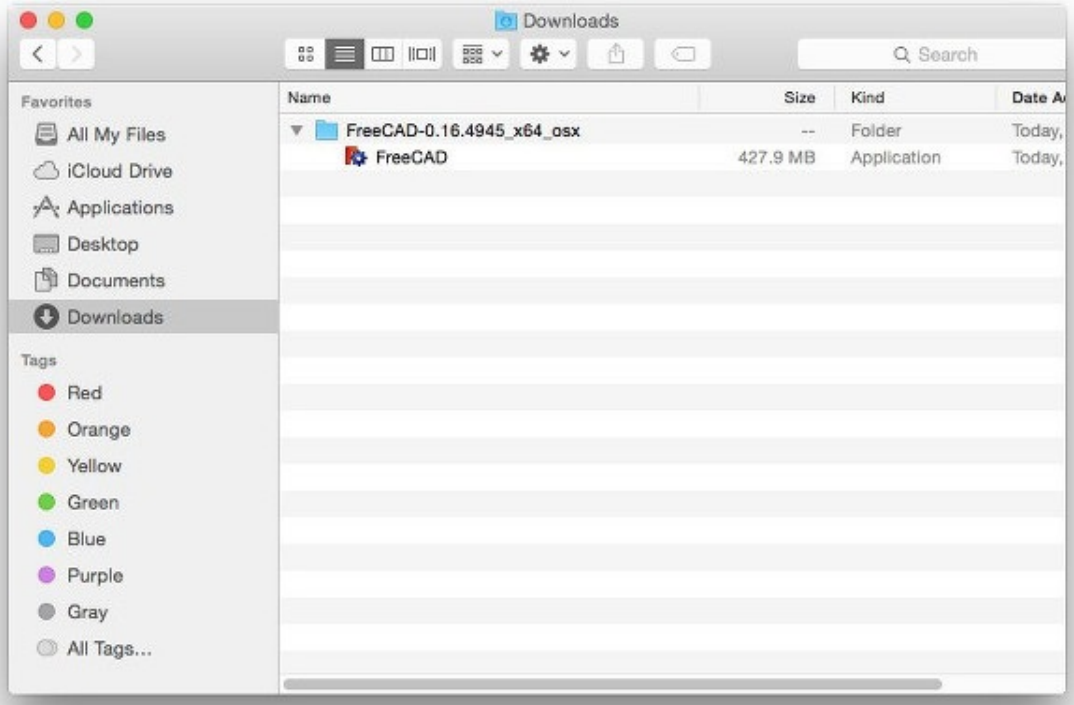
Alternative ways

One of the big joys of using Linux is the multitude of possibilities to tailor your software, so don't restrain yourself. On Ubuntu and derivatives, FreeCAD can also be installed from a PPA maintained by the FreeCAD community (it contains both stable and development versions) and since this is open-source software, you can also easily compile FreeCAD yourself.

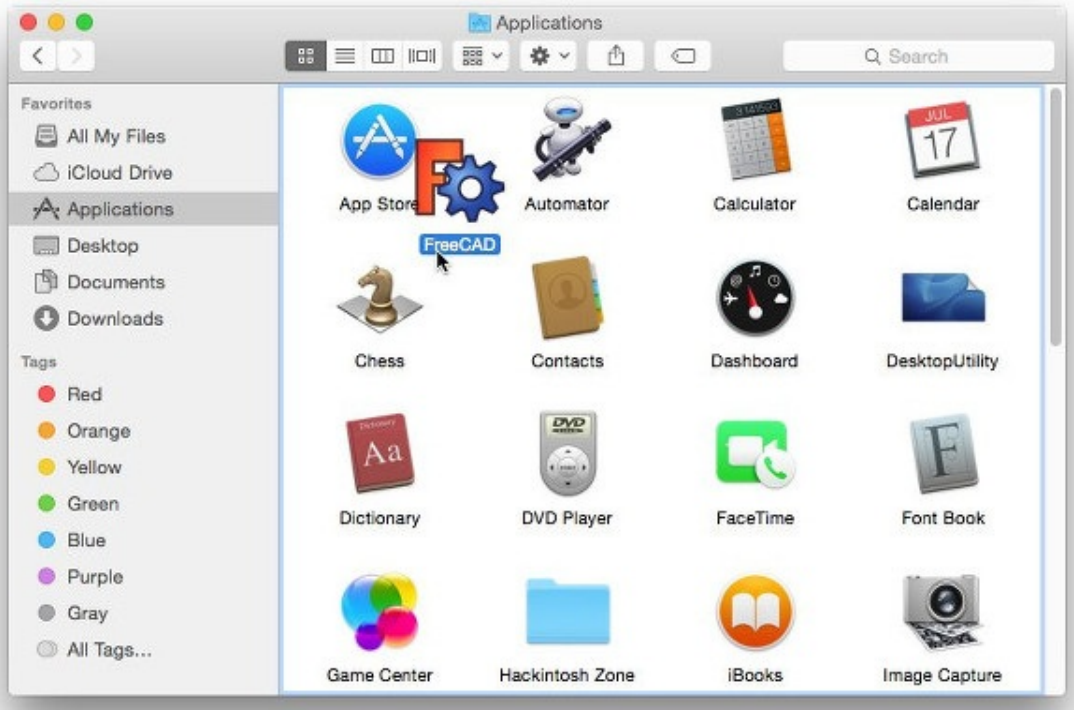
Installing on Mac OS

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

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

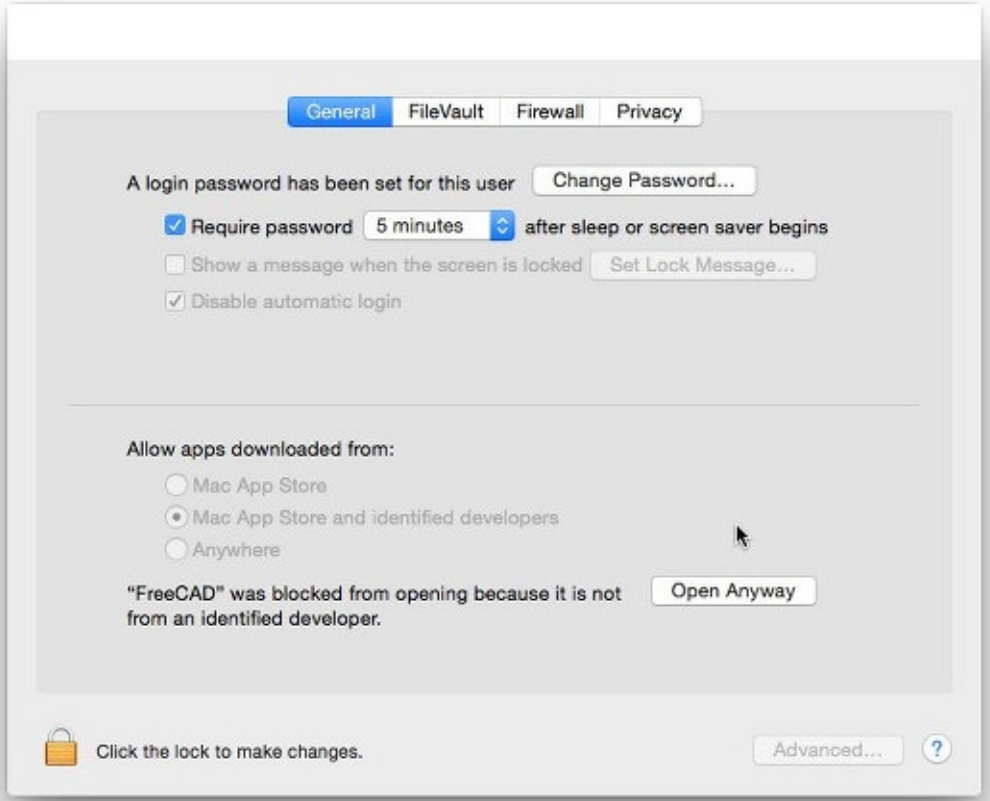


3. Drag the FreeCAD application from inside the zip to the Applications folder:



4. That's it, FreeCAD is installed!

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



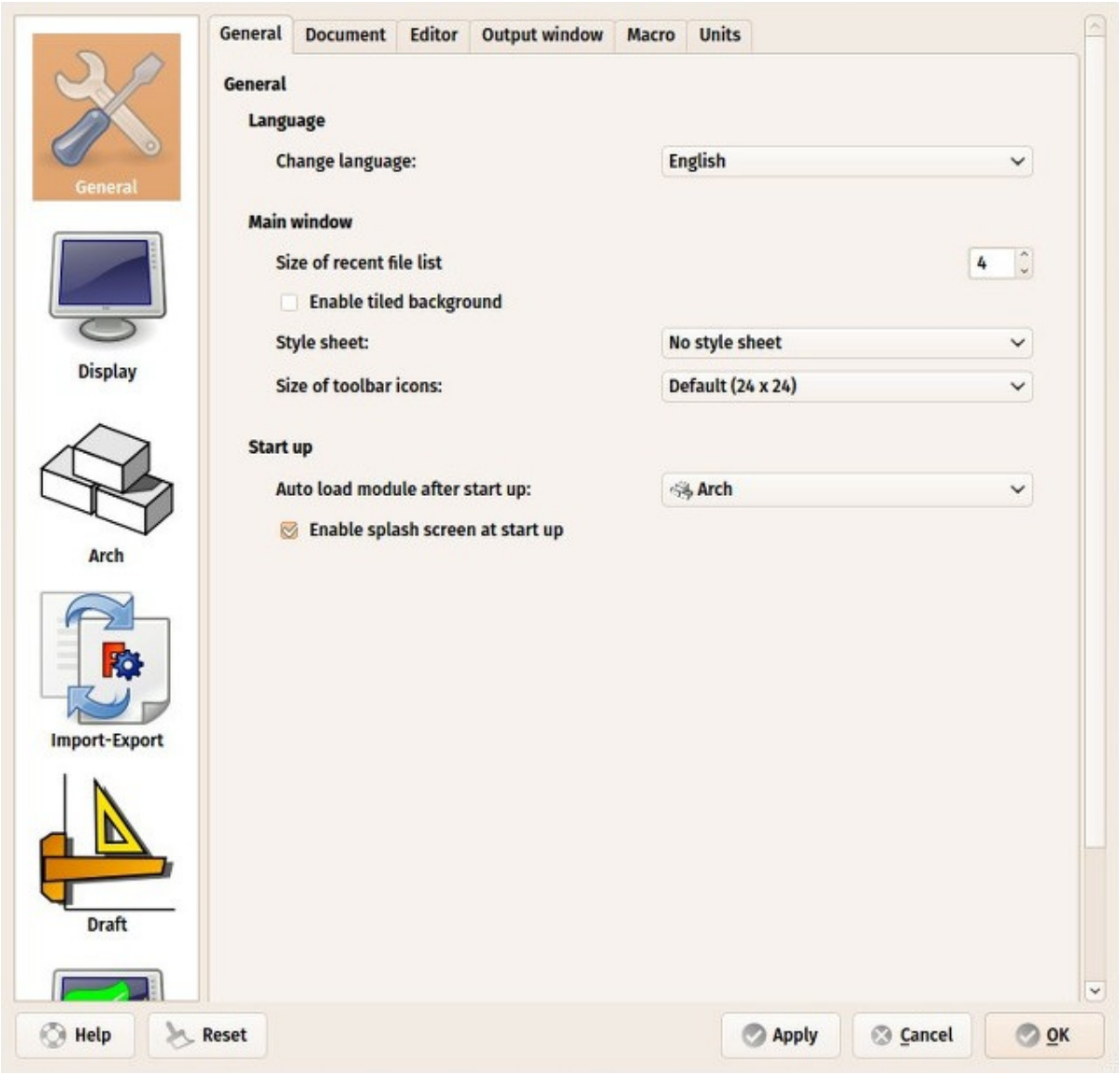
Uninstalling

Hopefully you won't want to do that, but it is good to know anyway. On Windows and Linux, uninstalling FreeCAD is very straightforward. Use the standard "remove software" option found in the control panel on Windows, or remove it with the same software manager you used to install FreeCAD on Linux. On Mac, all you need to do is remove it from the Applications folder.

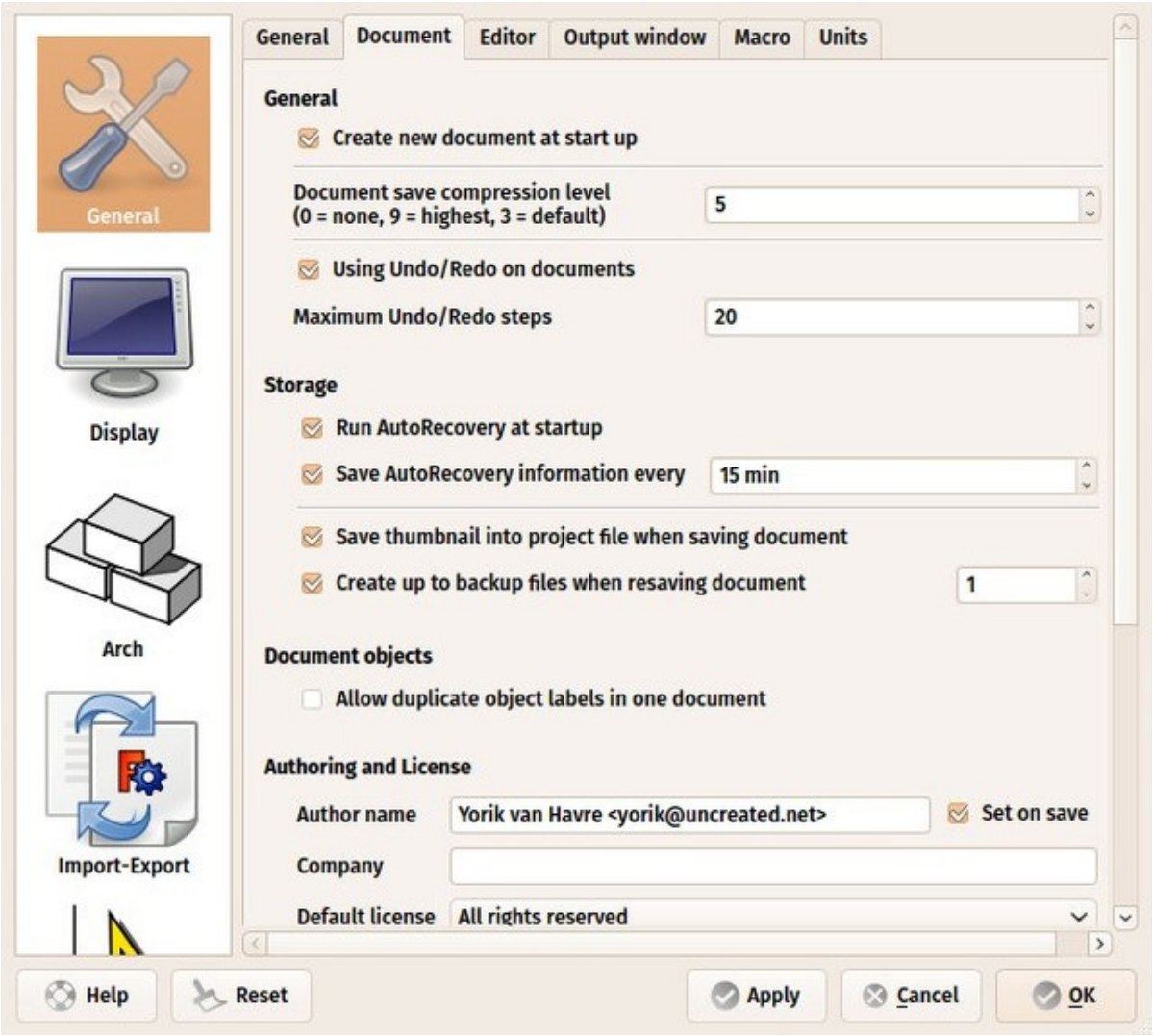
Setting basic preferences

Once FreeCAD is installed, you might want to open it and set a couple of preferences. Preferences settings in FreeCAD are located under menu **Edit -> Preferences**. You can browse through the different pages to see if there is anything else you would want to change, but here are a couple of basic ones:

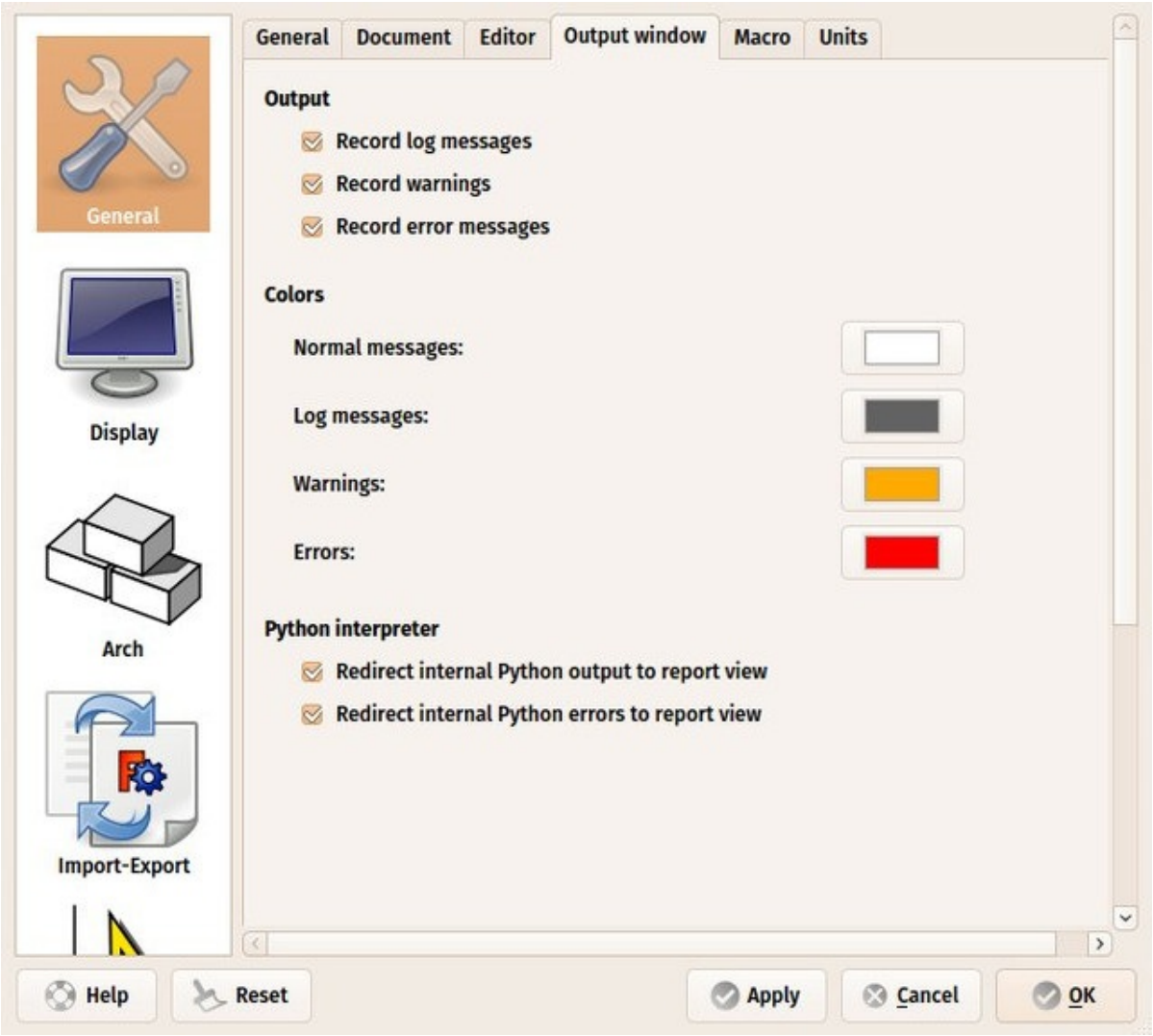
1. **Language**: FreeCAD will automatically pick the language of your operating system, but you might want to change that. FreeCAD is almost fully translated to 5 or 6 languages, plus many others that are at the moment only partially translated. You can easily help to translate FreeCAD.



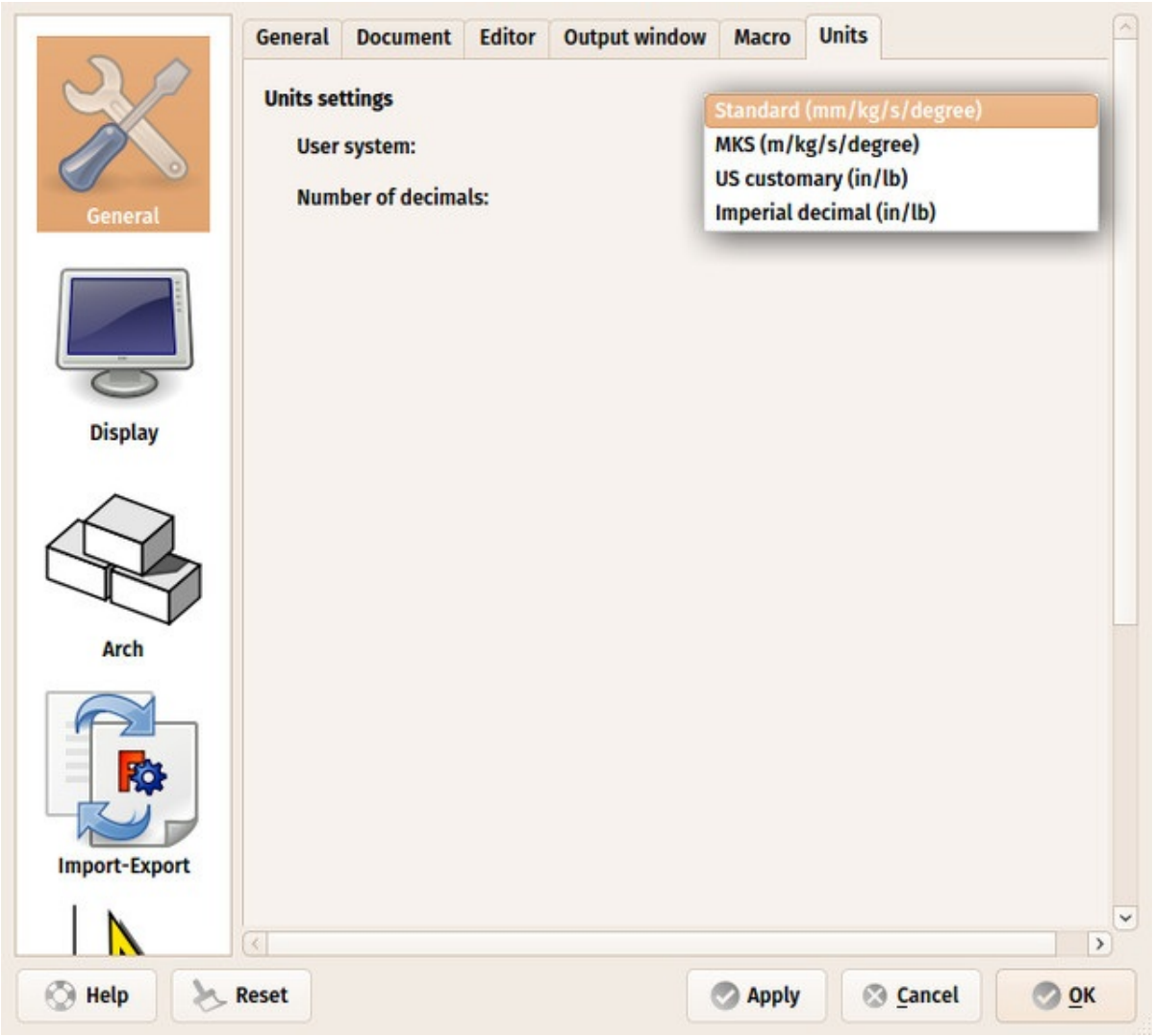
2. **Auto-load module**: Normally, FreeCAD will start showing you the start center page. You can skip this and begin a FreeCAD session directly in the workbench of your choice. Workbenches will be explained in detail in the next chapter.
3. **Create document at startup**: Combined with the option above, this starts FreeCAD ready for work.



4. **Storage options**: As any complex application, FreeCAD might crash from time to time. Here you can configure a few options that will help you to recover your work in case of a crash.
5. **Authoring and license**: You can set the default settings that will be used for your new files. Consider making your files shareable right from the start, by using a friendlier, copyleft license like Creative Commons.
6. **Redirect python messages to output view**: These two options are always good to mark, as they will permit you to see what's wrong in the Report View when there's a problem with running a particular python script.



7. **Units**: Here you can set the default units system you wish to use.



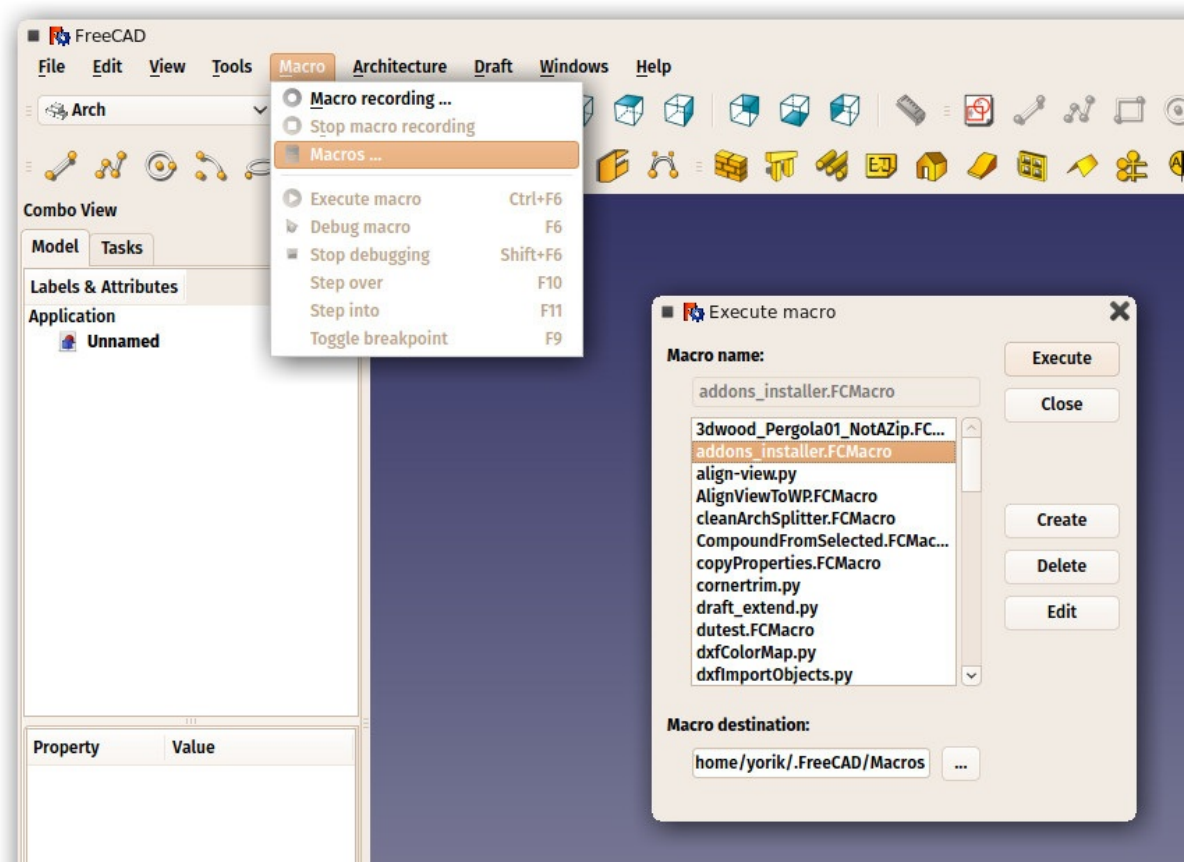
Installing additional content

As the FreeCAD project and its community grows quickly, and also because it is easy to extend, external contributions and side-projects made by community members and other enthusiasts begin to appear everywhere on the internet. There is an effort going on to gather all these interesting additions in one place, on the FreeCAD github page. There, among other things, you will find:

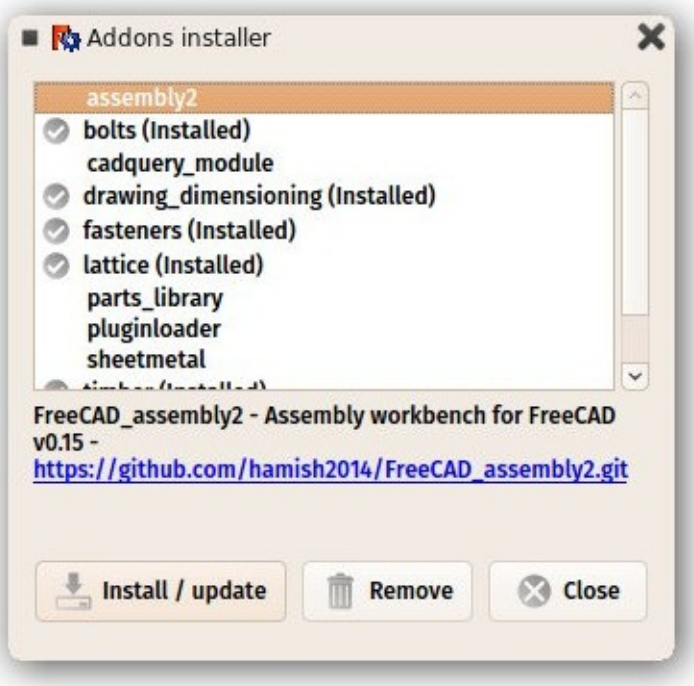
1. A Parts library, which contains all kinds of useful models, or pieces of models, created by FreeCAD users that can be freely used in your projects. The library can be used and accessed right from inside your FreeCAD installation.
2. A collection of addons, most of them additional workbenches, that extend the functionality of FreeCAD for certain tasks. Instructions for installing are given on each separate addon page.
3. A collection of macros, which are also available on the FreeCAD wiki along with documentation about how to use them. The wiki contains many more macros.

If you are using Ubuntu or any of its derivatives, the FreeCAD-extras PPA contains most of these addons. On other platforms, any of the addons, including the Parts library, can easily be installed using an addon-installer macro provided in the addons repository. The following procedure shows how to install the addon-installer (other macros can be installed the same way)

1. Download the addons-installer.FCMacro file from https://github.com/FreeCAD/FreeCAD-addons by clicking it, then right-clicking the "RAW" button, and choosing "Save as".
2. Place the macro in your FreeCAD Macros destination path. You FreeCAD Macros destination path is indicated at the bottom of the **Execute macro** dialog in FreeCAD:



3. Close and reopen the **Execute macro** dialog, and start the **addons_installer.FCMacro**. The installer will launch, from where you can install, update and uninstall any of the addons:

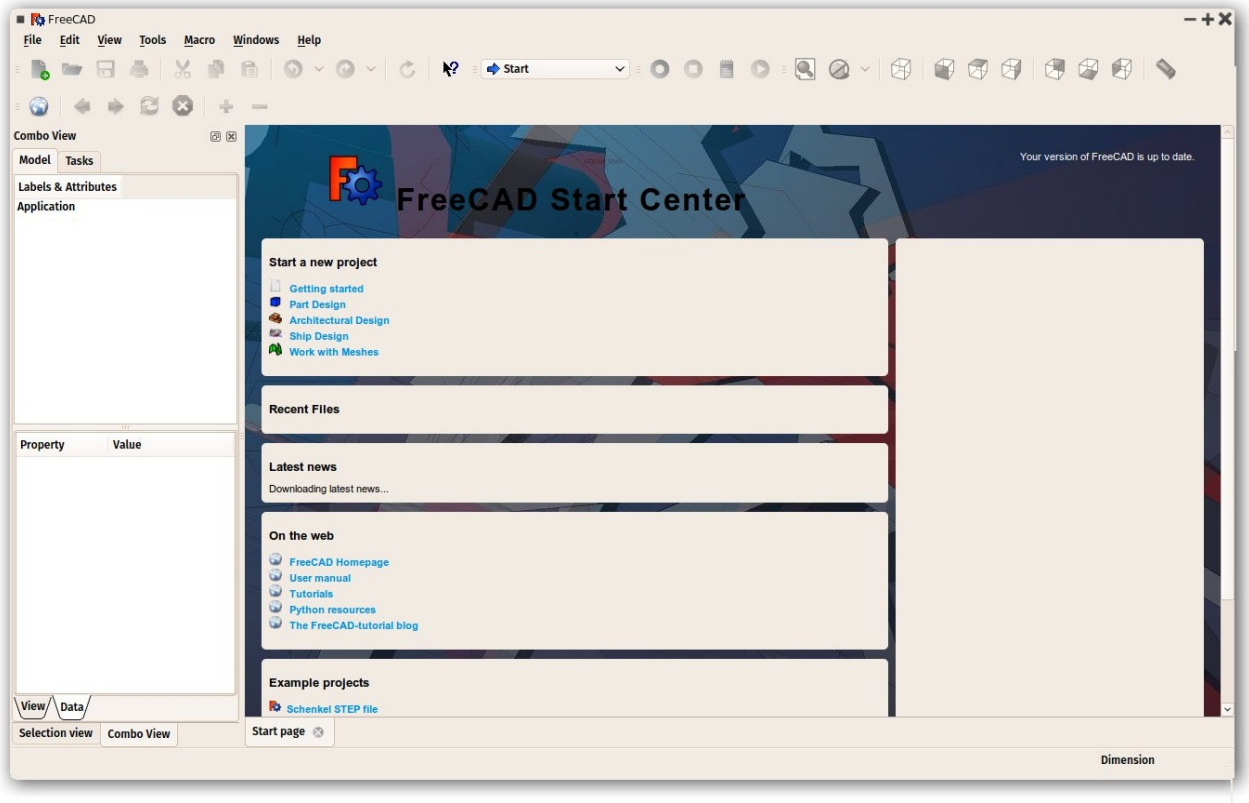


Read more

- More download options: http://www.freecadweb.org/wiki/index.php?title=Download
- Detailed installation instructions: http://www.freecadweb.org/wiki/index.php?title=Installing
- FreeCAD PPA for Ubuntu: https://launchpad.net/~freecad-maintainers
- FreeCAD addons PPA for Ubuntu: https://launchpad.net/freecad-extras
- Compile FreeCAD yourself: http://www.freecadweb.org/wiki/index.php?title=Compiling
- FreeCAD translations: https://crowdin.com/project/freecad
- FreeCAD github page: https://github.com/FreeCAD

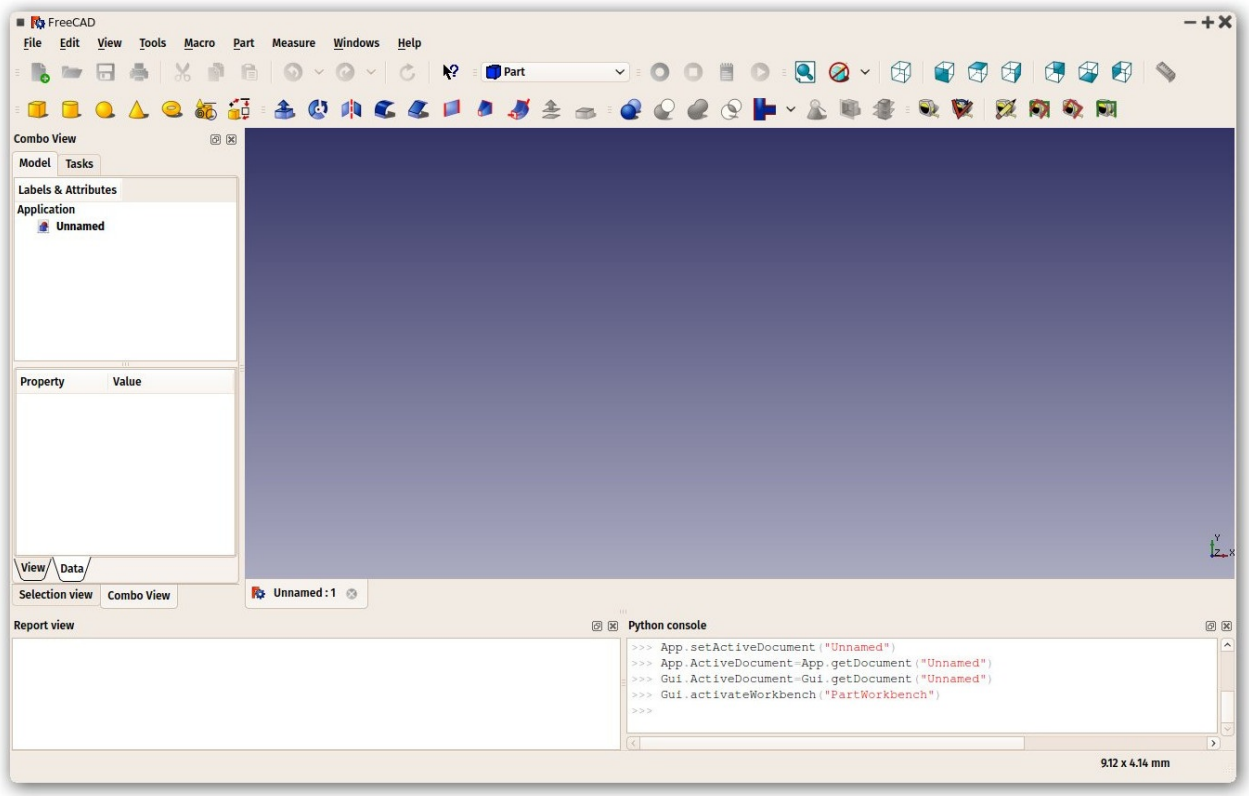
The FreeCAD interface

FreeCAD uses the Qt framework to draw and manage its interface. This framework is used in a wide range of applications, so the FreeCAD interface is very classical and presents no particular difficulty to understand. Most buttons are standard and will be found where you expect them (File -> Open, Edit -> Paste, etc). Here is the look of FreeCAD when you open it for the first time, just after installing, showing you the start center:



The start center is a convenient "welcome screen", that shows useful information for newcomers, like the latest files you have been working on, what's new in the FreeCAD world, or quick info about the most common Workbenches. It will also notify you if a new stable version of FreeCAD is available.

But after a while, or after you did some changes in the preferences, you will much more likely find yourself directly in one of the other Workbenches, with a new document open. Or simply, close the start page tab and create a new document:

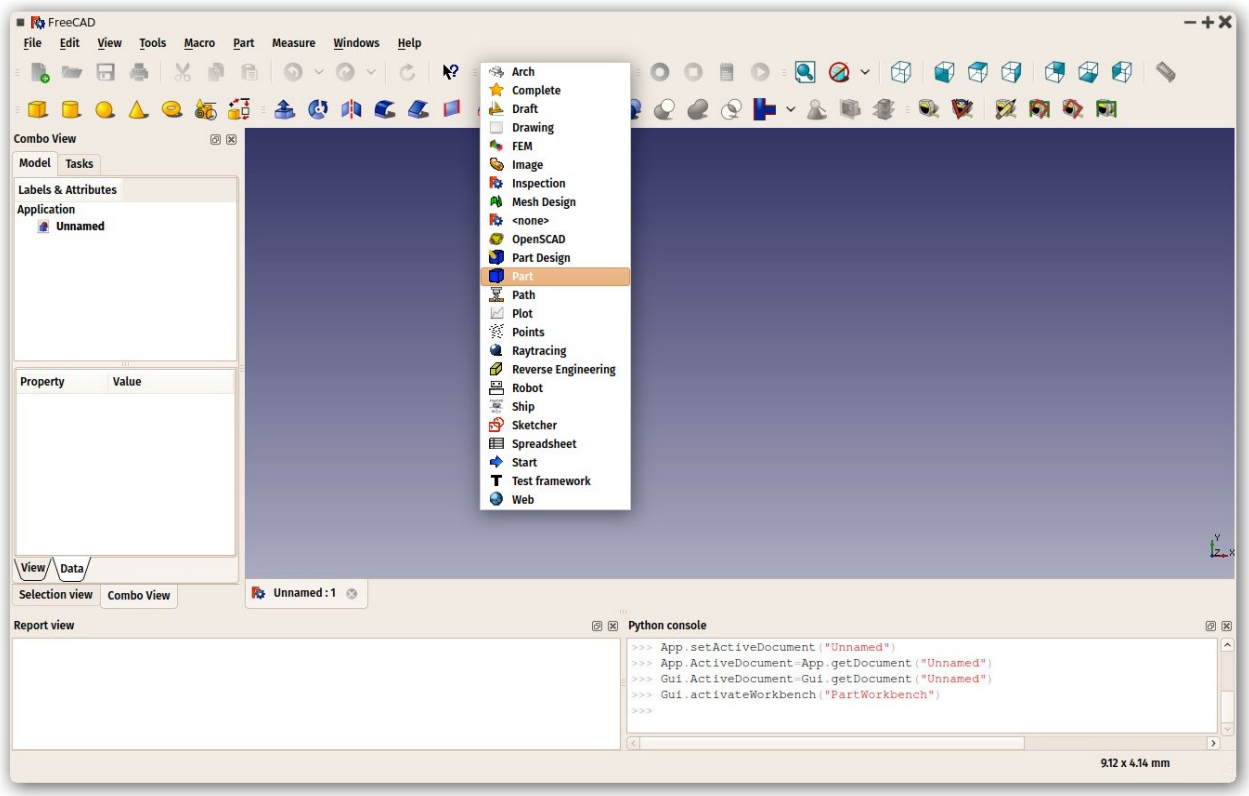


Workbenches

Note that some of the icons have changed between the two screencaptures above. This is where the most important concept used in the FreeCAD interface comes into play: Workbenches.

Workbenches are group of tools (toolbar buttons, menus, and other interface controls) that are grouped together by specialty. Think of a workshop where you have different people working together: A person who works with metal, another with wood. Each of them has, in their workshop, a separate table with specific tools for his/her job. However, they can all work on the same objects. The same happens in FreeCAD.

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

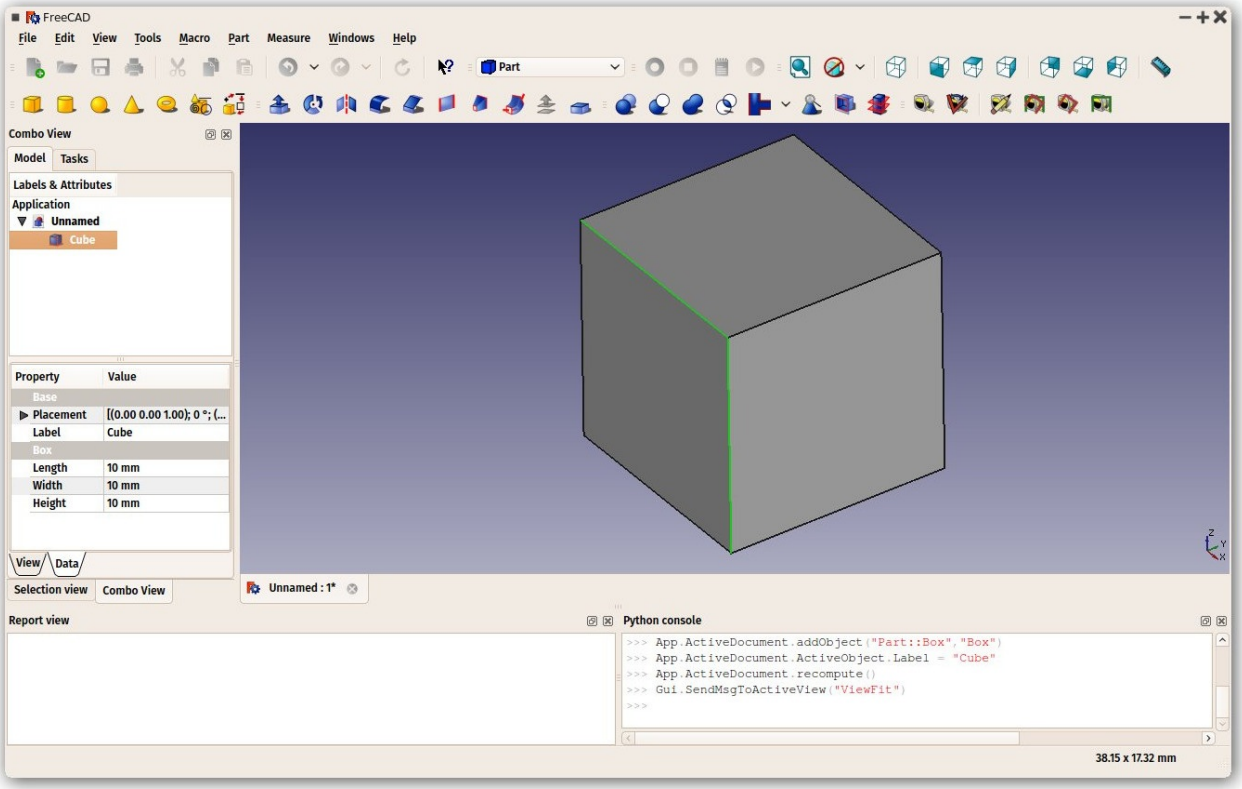


The Workbenches often confuse new users, since it's not always easy to know in which Workbench to look for a specific tool. But they are quick to learn, and after a short while, will feel nature--realizing it a convenient way to organize the multitude of tools FreeCAD has to offer, and they are also fully customizable (see below).

Later in this manual, you will also find a table showing the contents of all Workbenches.

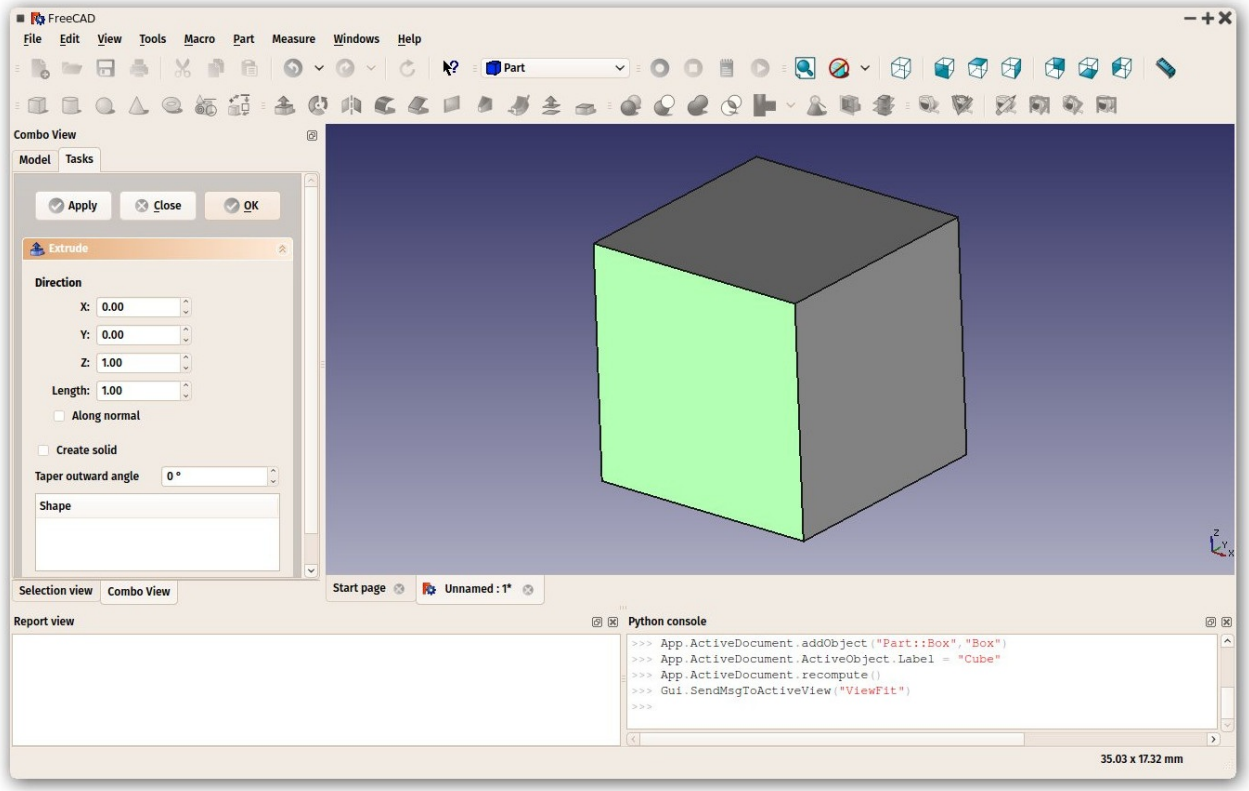
The interface

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



- **The 3D view** is the main component of the interface. It can be undocked out of the main window, you can have several views of the same document (or same objects), or several documents opened at the same time. You can select objects or parts of objects by clicking them, and you can pan, zoom and rotate the view with the mouse buttons. This will be explained further in the next chapter.
- **The combo view** has two tabs:
  - The Model tab shows you the contents and structure of your document above and the properties (or parameters) of the selected object(s) below. These properties are separated in two categories:
    - Data (properties which concern the geometry itself)
    - View (properties that affect how the geometry looks like on screen).
  - The Tasks tab is where FreeCAD will prompt you for values specific to the tool you're currently using at the time—for example, entering a 'length' value when the Line tool is being used. It will close automatically after pressing the OK (or Cancel) button. Also, by double-clicking the related object in the combo view, most tools will allow you to reopen that task panel in order to modify the settings.
- **The report view** is normally hidden, but it is a good idea to leave open as it will list any information, warnings or errors to help you decipher (or debug) what you may have done wrong.
- **The Python console** is also hidden by default. This is where you can interact with the contents of the document using the Python language. Since every action you do on the FreeCAD interface actually executes a piece of Python code, having this open allows you to watch the code unfold in real time—allowing you a wonderful and easy way to

learn a little Python on the way, almost without noticing it.

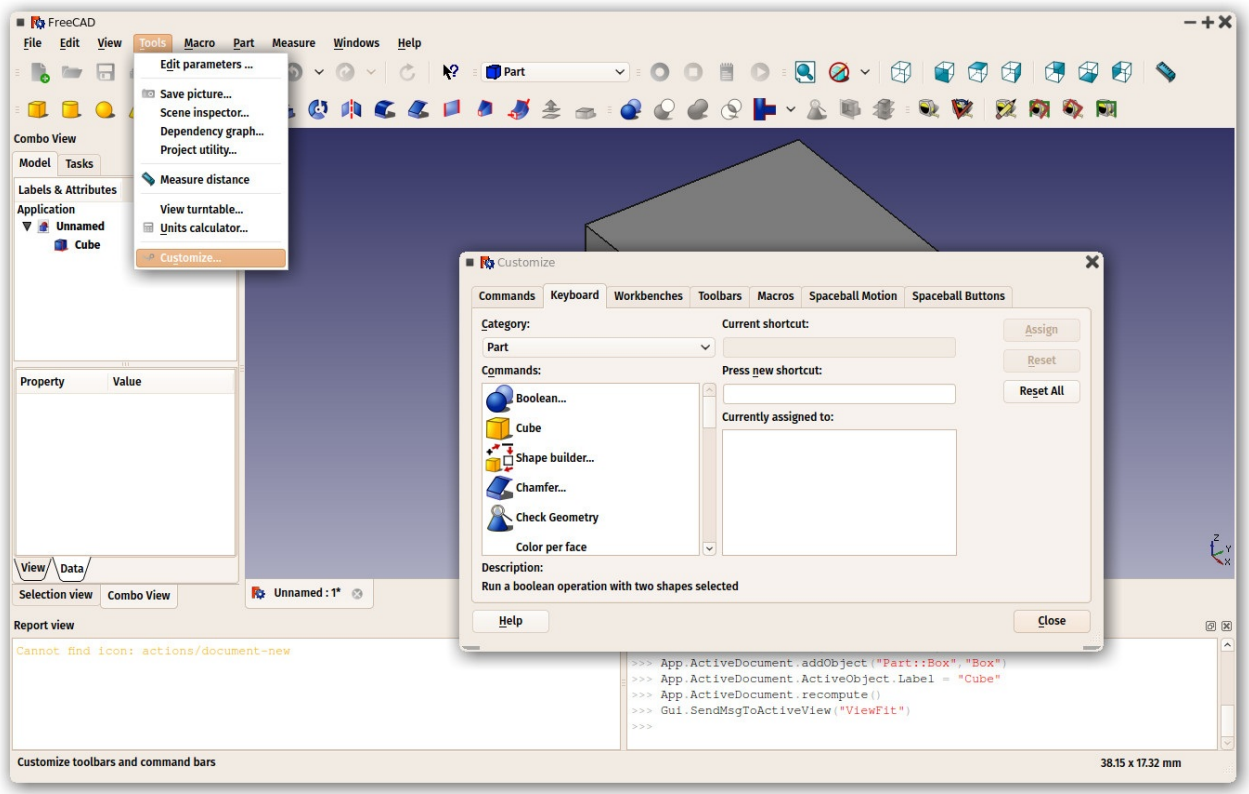


Any of the panels above can be turned on/off from menu View -> Panels.

Customizing the interface

The interface of FreeCAD is deeply customizable. All panels and toolbars can be moved to different places or stacked one with another. They can also be closed and reopened when needed from the View menu or by right-clicking on an empty area of the interface. There are, however, many more options available, such as creating custom toolbars with tools from any of the Workbenches, or assigning and changing keyboard shortcuts.

These advanced customization options are availabe from the Tools -> Customize menu:



Read more

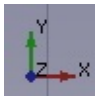
- Getting started with FreeCAD: http://www.freecadweb.org/wiki/index.php?title=Getting_started
- Customizing the interface: http://www.freecadweb.org/wiki/index.php?title=Interface_Customization
- Workbenches: http://www.freecadweb.org/wiki/index.php?title=Workbenches
- More about Python: https://www.python.org

Navigating in the 3D view

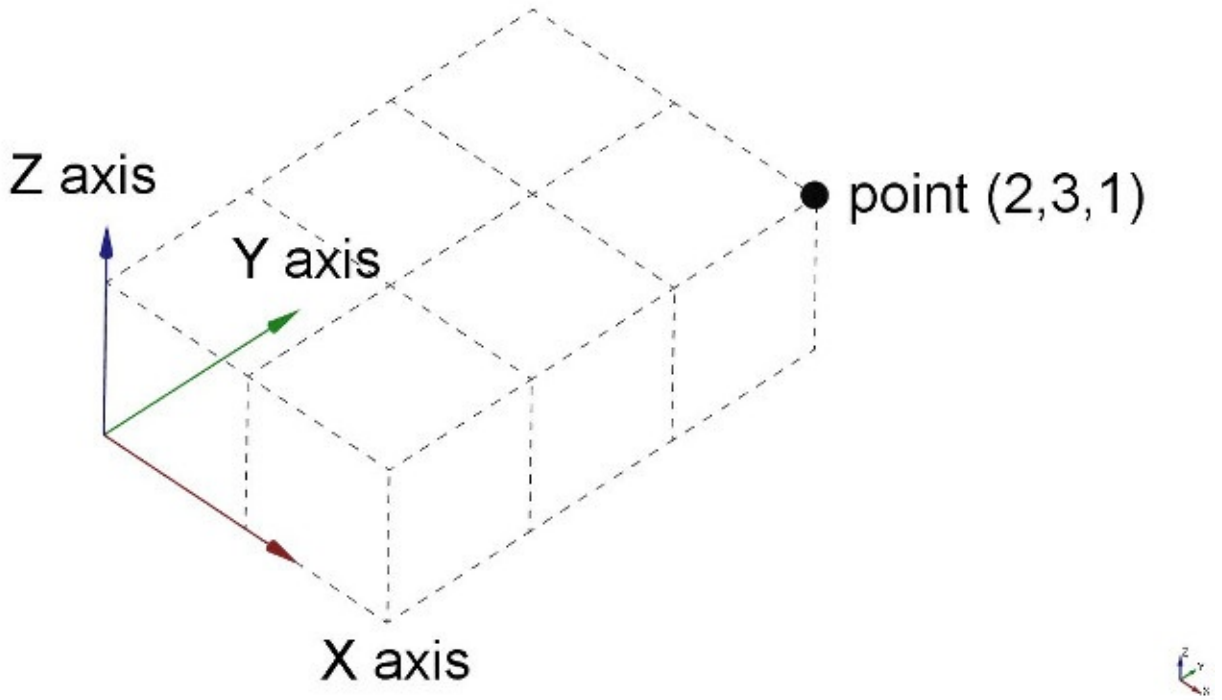
A word about the 3D space

If this is your first contact with a 3D application, you will need to grab some concepts first. If not, you can safely skip this section.

The FreeCAD 3D space is an euclidian space. It has an origin point and three axes: X, Y and Z. If you look at your scene from above, conventionally, the X axis points to the right, the Y axis to the back, and the Z axis upwards. In the lower right corner of the FreeCAD view, you can always see from where you are viewing the scene:



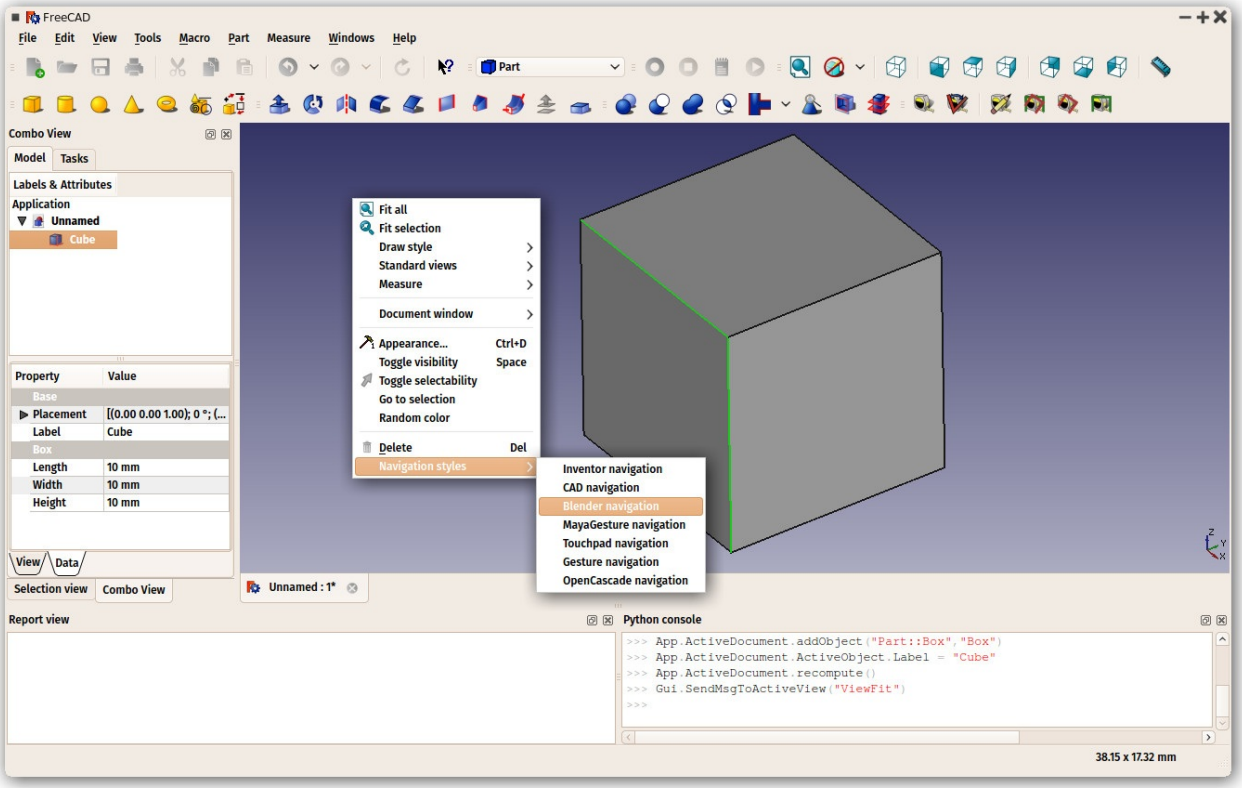
Every point of every object that exists in that space can be located by its (x,y,z) coordinates. For example, a point with coordinates (2,3,1) will lie at 2 units on the X axis, 3 units on the Y axis, and 1 unit on the Z axis:



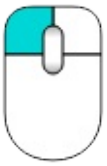
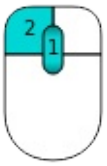
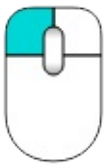
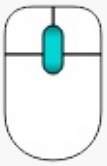
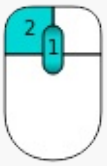
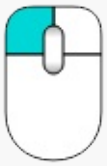
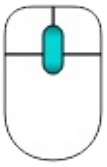
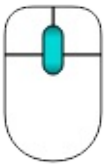
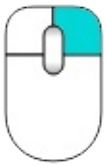
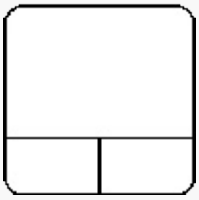
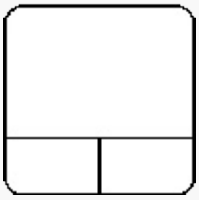
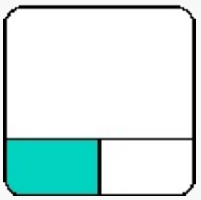
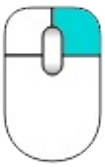
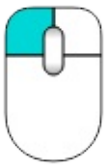
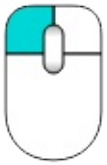
You can look at that scene from any angle, like if you were holding a camera. That camera can be moved left, right, up and down (pan), rotated around what it is looking at (rotate) and brought closer or further from the scene (zoom).

The FreeCAD 3D view

Navigating in the FreeCAD 3D view can be done with a mouse, a Space Navigator device, the keyboard, a touchpad, or a combination of those. FreeCAD proposes several navigation modes, which determine how the three basic view manipulation operations (pan, rotate and zoom) are done, as well as how to select objects on the screen are performed. Navigation modes are accessed from the Preferences screen, or directly by right-clicking anywhere on the 3D view:



Each of these modes attributes different mouse buttons, or mouse + keyboard combinations, or mouse gestures, to these four operations. The following table shows the principal available modes:

Mode	Pan	Rotate	Zoom	Select
Inventor				CTRL + 
CAD (default)				
Blender	SHIFT + 			
Touchpad	SHIFT + 	ALT + 	PGUP / PGDOWN	
Gesture	 + DRAG	 + DRAG		

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

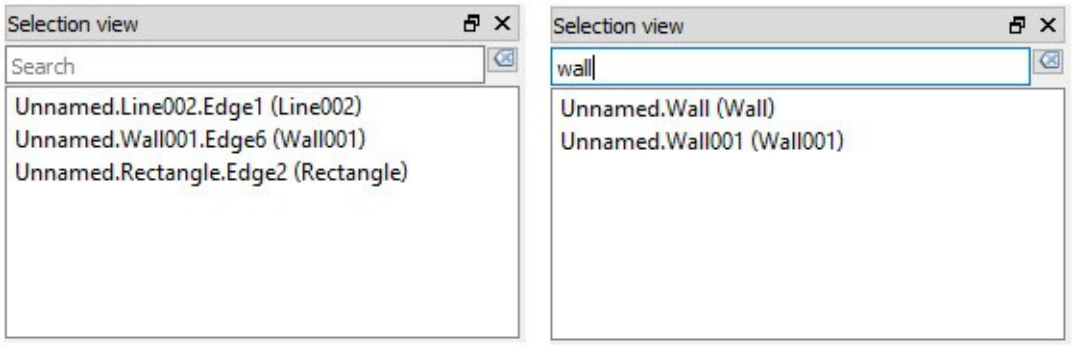
- **CTRL +** and **CTRL -** to zoom in and zoom out
- The **arrow keys** to shift the view left/right and up/down
- **SHIFT + left arrow** and **SHIFT + right arrow** to rotate the view by 90 degrees
- the numeric keys, **1 to 6**, for the six standard views, top, front, right, bottom, back and left
- **O** will set the camara in orthographic mode,
- while **P** sets it in perspective mode.
- **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.

Selecting objects

Objects in the 3D view can be selected by clicking them with the corresponding mouse button, depending on the navigation mode. A single click will select the object, and one of its subcomponents (edge, face, vertex). Double-clicking will select the object, and all its subcomponents. You can select more than one subcomponent, or even different subcomponents from 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:



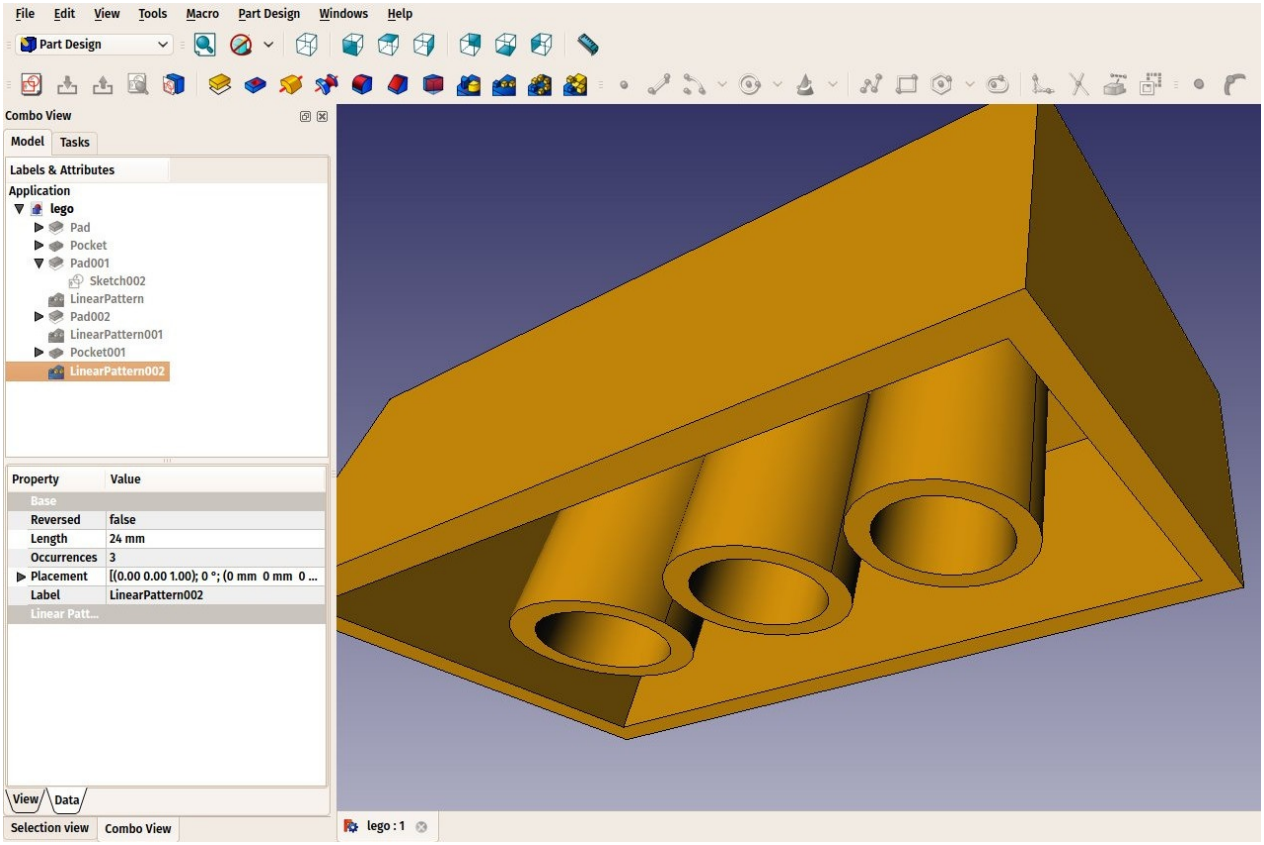
You can also use the Selection View to select objects by searching for a particular object.

Read more

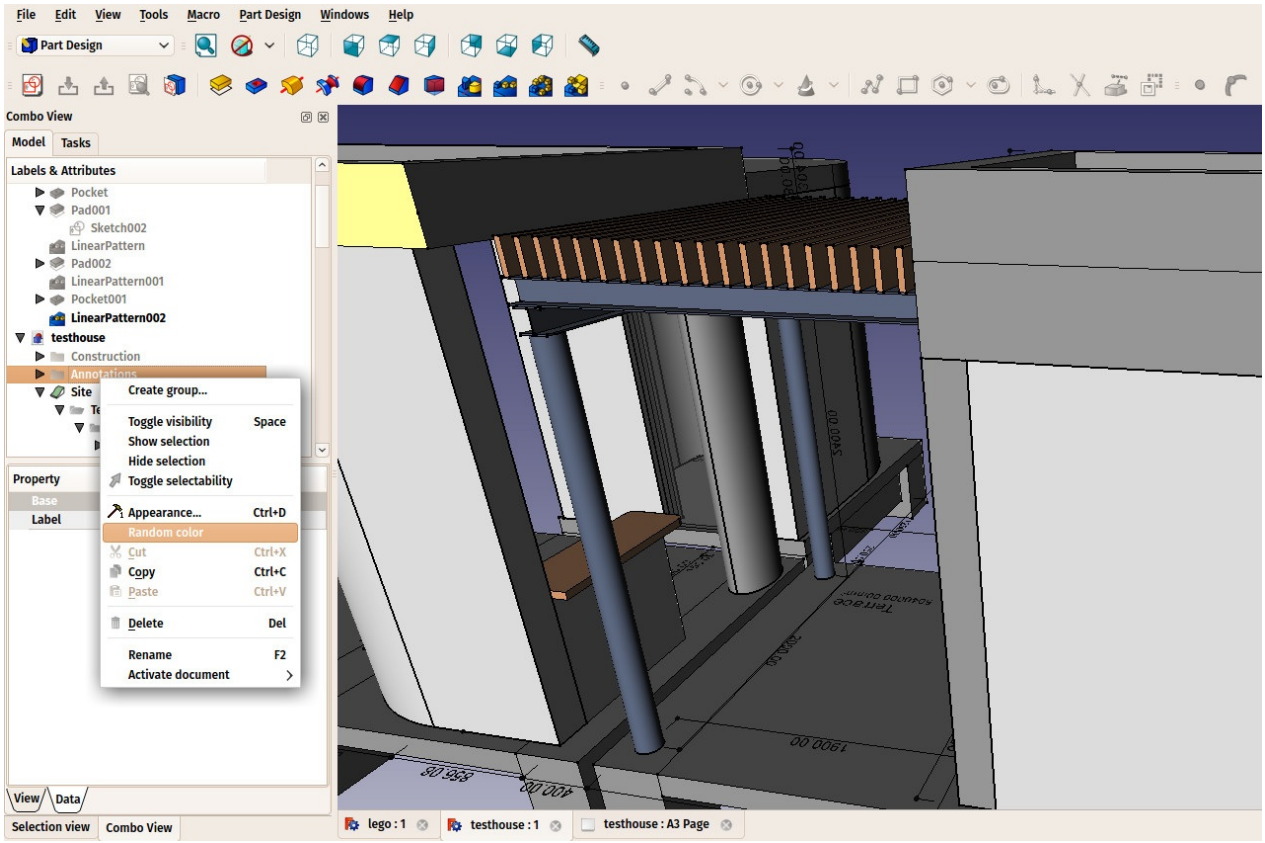
- The FreeCAD navigation modes: http://www.freecadweb.org/wiki/index.php?title=Mouse_Model

The FreeCAD document

A FreeCAD document contains all the objects of your scene. It can contain groups and objects made with any workbench. You can therefore switch between workbenches, and still work on the same document and/or objects within that document. The document is what gets saved to disk when you save your work. You can also open several documents at the same time in FreeCAD, and open several views of the same document.



Inside the document, the objects can be moved into groups, and have a unique name. Managing groups, objects and object names is done mainly from the Tree view. There, you can create groups, move objects to groups, delete objects or groups, by right-clicking in the tree view or on an object, you can rename objects, change their color, hide or show them, or possibly other operations, depending on the current workbench.



The objects inside a FreeCAD document can be of different types. Each workbench can add its own types of objects, for example the Mesh Workbench adds mesh objects, the Part Workbench adds Part objects, etc.

If there is at least one document open in FreeCAD, there is always one and only one active document. That's the document that appears in the current 3D view, the document you are currently working on. If you switch tabs to another document, that one becomes the active document. Most operations always work on the active document.

FreeCAD documents are saved with the .FcStd extension, which is a zip-based compound format, similar to LibreOffice. If something goes very wrong, it is often possible to unzip it and fix the problem or rescue the data.

Read more

- The FreeCAD document: http://www.freecadweb.org/wiki/index.php?title=Document_structure
- The FcStd file format: http://www.freecadweb.org/wiki/index.php?title=File_Format_FCStd

Parametric objects

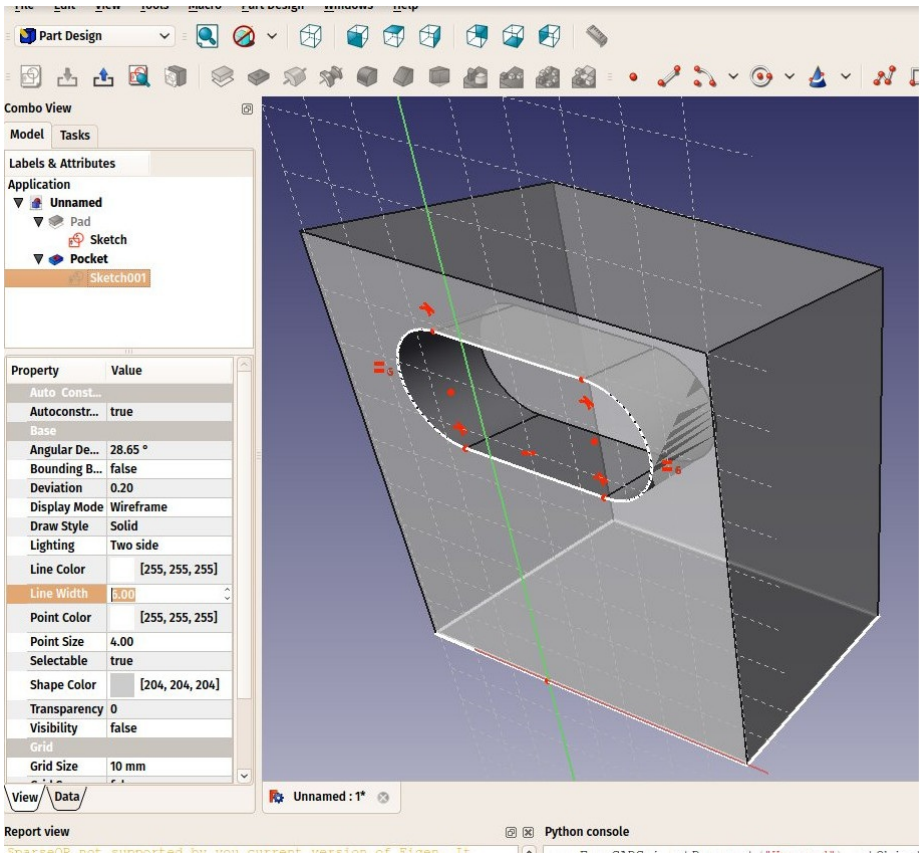
FreeCAD is designed for parametric modeling. This means that the geometry that you create, instead of being freely sculptable, is produced by rules and parameters. For example, a cylinder might be produced from a radius and a height. With these two parameters, the program has enough information to build the cylinder.

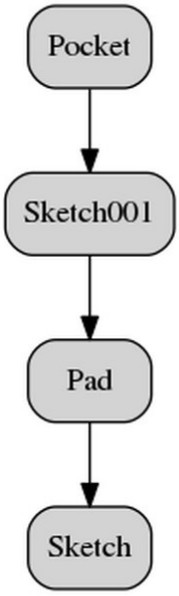
Parametric objects, in FreeCAD, are in reality small pieces of a program that run whenever one of the parameters has changed. Objects can have a lot of different kinds of parameters: numbers (integers like 1, 2, 3 or floating-point values like 3.1416), real-world sizes (1mm, 2.4m, 4.5ft), (x,y,z) coordinates, text strings ("hello!") or even another object.

This last type allows to quickly build complex chains of operations, each new object being based on a previous one, and adding new features to it.

In the example below, a solid, cubic object (Pad) is based on a rectangular 2D shape (Sketch) and has an extrusion distance. With these two properties, it produces a solid shape by extruding the base shape by the given distance. You can then use this object as a base for further operations, such as drawing a new 2D shape on one of its faces (Sketch001) and then making a subtraction (Pocket), until arriving at your final object.

All the intermediary operations (2D shapes, pad, pocket, etc) are still there, and you can still change any of their parameters anytime. The whole chain will be rebuilt (recomputed) whenever needed.





Two important things are necessary to know:

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 to have all the marked objects recomputed.
2. The dependency tree must always flow in the same direction. Loops are forbidden. You can have object A which depends on object B which depend 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.

Not all objects are parametric in FreeCAD. Often, the geometry that you import from other files won't contain any parameter, 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 pyth 

Read more

- The properties editor: http://www.freecadweb.org/wiki/index.php?title=Property_editor
- How to program parametric objects: http://www.freecadweb.org/wiki/index.php?title=Scripted_objects
- Positioning objects in FreeCAD: http://www.freecadweb.org/wiki/index.php?title=Placement
- Enabling the dependency graph: http://www.freecadweb.org/wiki/index.php?title=Std_DependencyGraph

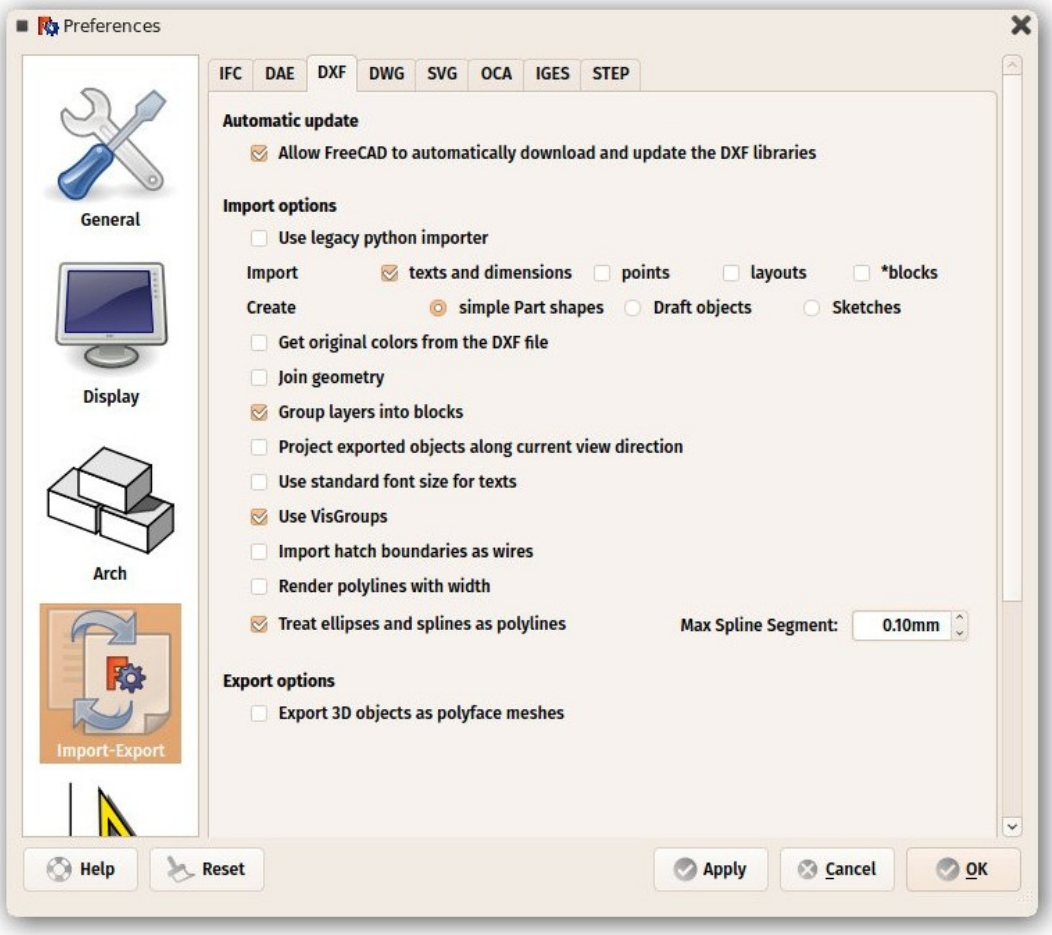
Import and export to other filetypes

FreeCAD can import and export to many filetypes. 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 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 is 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: http://www.freecadweb.org/wiki/index.php?title=Import_Export
- Working with DXF files in FreeCAD: http://www.freecadweb.org/wiki/index.php?title=Draft_DXF
- Enabling DXF and DWG support: http://www.freecadweb.org/wiki/index.php?title=Dxf_Importer_Install
- Working with SVG files in FreeCAD: http://www.freecadweb.org/wiki/index.php?title=Draft_SVG

- Importing and exporting to IFC: http://www.freecadweb.org/wiki/index.php?title=Arch_IFC
- OpenCasCade: http://www.opencascade.com
- Teigha File Converter: https://www.opendesign.com/guestfiles
- The IFC format: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/index.htm
- IfcOpenShell: http://ifcopenshell.org/

Working with FreeCAD

All workbenches at a glance

One of the biggest difficulty for new users of FreeCAD, is to know in which workbench to find a specific tool. The table below will give you an overview of the most important workbenches and their tools. Refer to each workbench page in the FreeCAD documentation for a more complete list.

Four workbenches are also designed to work in pairs, and one of them is fully included into the other: Arch contains all the Draft tools, and PartDesign all the Sketcher tools. However, for clarity, they are separated below.

Part

The Part Workbench provides basic tools for working with solid parts: primitives, such as cube and sphere, and simple geometric operations and boolean operations. Being the main anchor point with OpenCasCade, the Part workbench provides the foundation of FreeCAD's geometry system, and almost all other workbenches produce Part-based geometry.

Tool	Description	Tool	Description
 Box	Draws a box	 Cone	Draws a cone
 Cylinder	Draws a cylinder	 Sphere	Draws a sphere
 Torus	Draws a torus (ring)	 Create Primitives	Creates various other parametric geometric primitives
 Shape Builder	Create more complex shapes from primitives	 Fuse	Fuses (unions) two objects
 Common	Extracts the common (intersection) part of two objects	 Cut	Cuts (subtracts) one object from another
 Join Connect	Connects interiors of walled objects	 Join Embed	Embeds a walled object into another walled object
 Join Cutout	Creates a cutout in a wall of an object for another walled object	 Extrude	Extrudes planar faces of an object
 Fillet	Fillets (rounds) edges of an object	 Revolve	Creates a solid by revolving another object (not solid) around an axis
 Section	Creates a section by intersecting an object with a section plane	 Section Cross	Creates multiple cross sections along an object
 Chamfer	Chamfers edges of an object	 Mirror	Mirrors the selected object on a given mirror plane
 Ruled Surface	Create a ruled surface between selected curves	 Sweep	Sweeps one or more profiles along a path
 Loft	Lofts from one profile to another	 Offset	Creates a scaled copy of the original object
 Thickness	Assign a thickness to the faces of a shape		

Draft

The Draft Workbench provides tools to do basic 2D CAD drafting tasks: lines, circles, etc... and a series of generic handy tools such as move, rotate or scale. It also provides several drawing aids, such as grid and snapping. It is principally meant to draw the guidelines for Arch objects, but also serves as FreeCAD's "swiss knife".

Tool	Description	Tool	Description
 Line	Draws a line segment between 2 points	 Wire	Draws a line made of multiple line segments (polyline)
 Circle	Draws a circle from center and radius	 Arc	Draws an arc segment from center, radius, start angle and end angle
 Ellipse	Draws an ellipse from two corner points	 Polygon	Draws a regular polygon from a center and a radius
 Rectangle	Draws a rectangle from 2 opposite points	 Text	Draws a multi-line text annotation
 Dimension	Draws a dimension annotation	 BSpline	Draws a B-Spline from a series of points
 Point	Inserts a single point	 Shape String	The ShapeString tool inserts a compound shape representing a text string at a given point in the current document
 Facebinder	Creates a new object from selected faces on existing objects	 Bezier Curve	Draws a Bezier curve from a series of points
 Move	Moves or copies objects from one location to another	 Rotate	Rotates objects by a certain angle around a point
 Offset	Offsets an object to a certain distance	 Trimex	Trims, extends or extrudes an object
 Upgrade	Turns or joins objects into a higher-level object	 Downgrade	Turns or separtes objects into lower-level objects
 Scale	Scales objects in relation to a point	 Shape2D View	Creates a 2D object which is a flattened view of another object

Draft2Sketch	Converts a Draft object to a Sketch and vice-versa	Array	Creates a polar or rectangular array from an object
PathArray	Creates an array from an object by placing copies along a path	Clone	Creates linked copies of objects
Mirror	Mirrors objects across a line		

Sketcher

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

Tool	Description	Tool	Description
Point	Draws a point	Line by 2 points	Draws a line segment from 2 points
Arc	Draws an arc segment from center, radius, start angle and end angle	Arc by 3 points	Draws an arc segment from two endpoints and another point on the circumference
Circle	Draws a circle from center and radius	Circle by 3 points	Draws a circle from three points on the circumference
Ellipse by center	Draws an ellipse by center point, major radius point and minor radius point	Ellipse by 3 points	Draws an ellipse by major diameter (2 points) and minor radius point
Arc of ellipse	Draws an arc of ellipse by center point, major radius point, starting point and ending point	Polyline	Draws a line made of multiple line segments. Several drawing modes available
Rectangle	Draws a rectangle from 2 opposite points	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	 Slot	Draws an oval by selecting the center of one semicircle and an endpoint of the other semicircle
 Fillet	Makes a fillet between two lines joined at one point	 Trim	Trims a line, circle or arc with respect to a clicked point
 External Geometry	Creates an edge linked to external geometry	 Construction Mode	Toggles an element to/from construction mode. A construction object will not be used in a 3D geometry operation and is only visible while editing the Sketch that contains it
 Coincident constraint	Affixes a point onto (coincident with) one or more other points.	 Point On Object constraint	Affixes a point onto another object such as a line, arc, or axis.
 Vertical constraint	Constrains the selected lines or polyline elements to a true vertical orientation. More than one object can be selected before applying this constraint.	Horizontal constraint	Constrains the selected lines or polyline elements to a true horizontal orientation. More than one object can be selected before applying this constraint.
 Parallel constraint	Constrains two or more lines parallel to one another.	 Perpendicular constraint	Constrains two lines perpendicular to one another, or constrains a line perpendicular to an arc endpoint.
 Tangent constraint	Creates a tangent constraint between two selected entities, or a co-linear constraint between two line segments.	 Equal Length constraint	Constrains two selected entities equal to one another. If used on circles or arcs their radii will be set equal.
 Symmetric constraint	Constrains two points symmetrically about a line, or constrains the first two selected points	 Lock constraint	Constrains the selected item by setting vertical and horizontal distances relative to the origin,

constraint	symmetrically about a third selected point.		thereby locking the location of that item
 Horizontal Distance constraint	Fixes the horizontal distance between two points or line endpoints. If only one item is selected, the distance is set to the origin.	 Vertical Distance constraint	Fixes the vertical distance between 2 points or line endpoints. If only one item is selected, the distance is set to the origin.
 Length constraint	Defines the distance of a selected line by constraining its length, or defines the distance between two points by constraining the distance between them.	 Radius constraint	Defines the radius of a selected arc or circle by constraining the radius.
 Internal Angle constraint	Defines the internal angle between two selected lines.	 Snell's Law constraint	Constrains two lines to obey a refraction law to simulate the light going through an interface
 Internal Alignment constraint	Aligns selected elements to selected shape (e.g. a line to become major axis of an ellipse)	 Map sketch to face	Maps a sketch to the previously selected face of a solid
 Merge	Merge two or more sketches	 Mirror	Mirrors selected elements of a sketch

Part Design

The Part Design Workbench contains advanced tools to build solid parts. It also contains all the tools from the sketcher. Since it can only produces solid shapes (the rule number one of Part Design), it is the main workbench to use when designing pieces (parts) to be manufactured or 3D-printed, as you will always obtain a printable object.

Tool	Description	Tool	Description
 Pad	Extrudes a solid object from a selected sketch	 Pocket	Creates a pocket from a selected sketch. The sketch must be mapped to an existing solid object's face
 Revolution	Creates a solid by revolving a sketch around an axis	 Groove	Creates a groove by revolving a sketch around an axis
 Fillet	Fillets (rounds) edges of an object	 Chamfer	Chamfers edges of an object
 Draft	Applies angular draft to faces of an object	 Mirrored	Mirrors features on a plane or face
 Linear Pattern	Creates a linear pattern of features	 Polar Pattern	Creates a polar pattern of features
 Scaled	Scales features to a different size	 MultiTransform	Allows creating a pattern with any combination of the other transformations
 Shaft wizard	Generates a shaft from a table of values and allows to analyze forces and moments	 Involute Gear wizard	Allows you to create several types of gears

Arch

The Arch Workbench contains tools to work with BIM projects (civil engineering and architecture). It also contains all the tools from the Draft workbench. The main use of the Arch Workbench is to create BIM objects or give BIM attributes to objects built with other workbenches, in order to export them to IFC.

Tool	Description	Tool	Description
 Wall	Creates a wall from scratch or using a selected object as a base	 Structure	Creates a structural element from scratch or using a selected object as a base
 Reinforcement Bar	Creates a reinforcement bar in a selected structural element	 Floor	Creates a floor including selected objects
 Building	Creates a building including selected objects	 Site	Creates a site including selected objects
 Window	Creates a window using a selected object as a base	 Section Plane	Adds a section plane object to the document
 Axes	Adds an axes system to the document	 Roof	Creates a sloped roof from a selected face
 Space	Creates a space object in the document	 Stairs	Creates a stairs object in the document
 Panel	Creates a panel object from a selected 2D object	 Frame	Creates a frame object from a selected layout
 Equipment	Creates an equipment or furniture object	 Set Material	Attributes a material to selected objects
 Schedule	Creates different types of schedules	 Cut Plane	Cut an object according to a plan.
 Add Component	Adds objects to a component	 Remove Component	Subtracts or removes objects from a component
 Survey Mode	Enters or leaves surveying mode		

Drawing

The Drawing Workbench handles the creation and manipulation of 2D drawing sheets, used for displaying views of your 3D work in 2D. These sheets can then be exported to 2D applications in SVG or DXF formats, to a PDF file or printed.

Tool	Description	Tool	Description
New sheet	Creates a new drawing sheet	Insert view	Inserts a view of the selected object in the active drawing sheet
Annotation	Adds an annotation to the current drawing sheet	Clip	Adds a clip group to the current drawing sheet
Browser preview	Opens a preview of the current sheet in the browser	Ortho Views	Automatically creates orthographic views of an object on the current drawing sheet
Symbol	Adds the contents of a SVG file as a symbol on the current drawing sheet	Draft View	Inserts a special Draft view of the selected object in the current drawing sheet
Export	Saves the current sheet as a SVG file		

Other built-in workbenches

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

- 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 Raytracing Workbench offers tools to interface with external renderers such as povray or luxrender. Right from inside FreeCAD, this workbench allows you to produce high-quality renderings from your models.
- 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, allowing to use them as master data structures.
- The FEM Workbench deals with Finite Elements Analysis, and permits the performing of pre- and post-processing FEM calculations and to display the results graphically.

External workbenches

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 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 Assembly2 Workbench offers a series of tools to mount and work with assemblies.

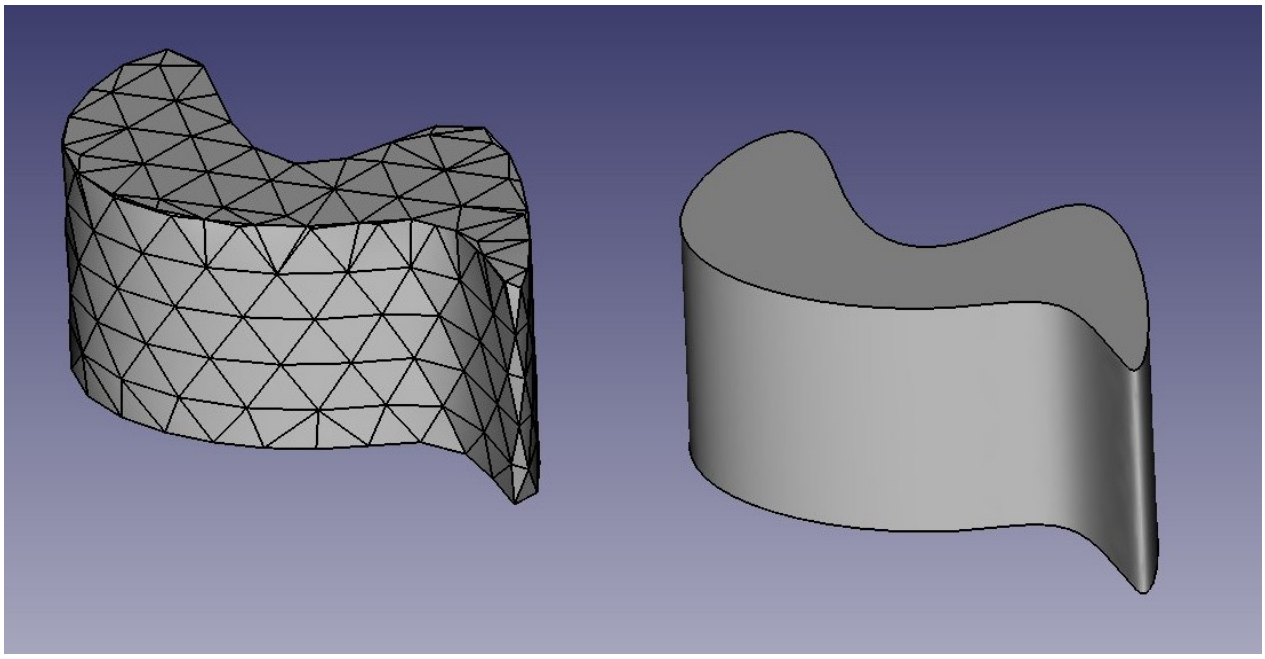
Read more

- The complete list of workbenches: http://www.freecadweb.org/wiki/index.php?title=Workbenches
- The Part Workbench: http://www.freecadweb.org/wiki/index.php?title=Part_Module
- The Draft Workbench: http://www.freecadweb.org/wiki/index.php?title=Draft_Module
- The Sketcher and Part Design Workbench: http://www.freecadweb.org/wiki/index.php?title=PartDesign_Workbench
- The Arch Workbench: http://www.freecadweb.org/wiki/index.php?title=Arch_Module
- The Drawing Workbench: http://www.freecadweb.org/wiki/index.php?title=Drawing_Module
- The FEM Workbench: http://www.freecadweb.org/wiki/index.php?title=Fem_Workbench
- The FreeCAD-addons repository: https://github.com/FreeCAD/FreeCAD-addons

Traditional modeling - the CSG way

CGS stands for Constructive Solid Geometry and describes the most basic way to work with solid 3D geometry, which is creating complex objects by adding and removing pieces to/from solids by using Boolean operations such as union, subtraction or intersection.

As we saw earlier in this manual, FreeCAD can handle many types of geometry, but the preferred and most useful type for the kind of 3D objects that we want to design with FreeCAD, that is, real-world objects, is, without a doubt, solid, BREP geometry, that is mainly handled by the Part Workbench. Unlike polygon meshes, which are made only of points and triangles, BREP objects have their faces defined by mathematical curves, which permits abolute precision, no matter the scale.

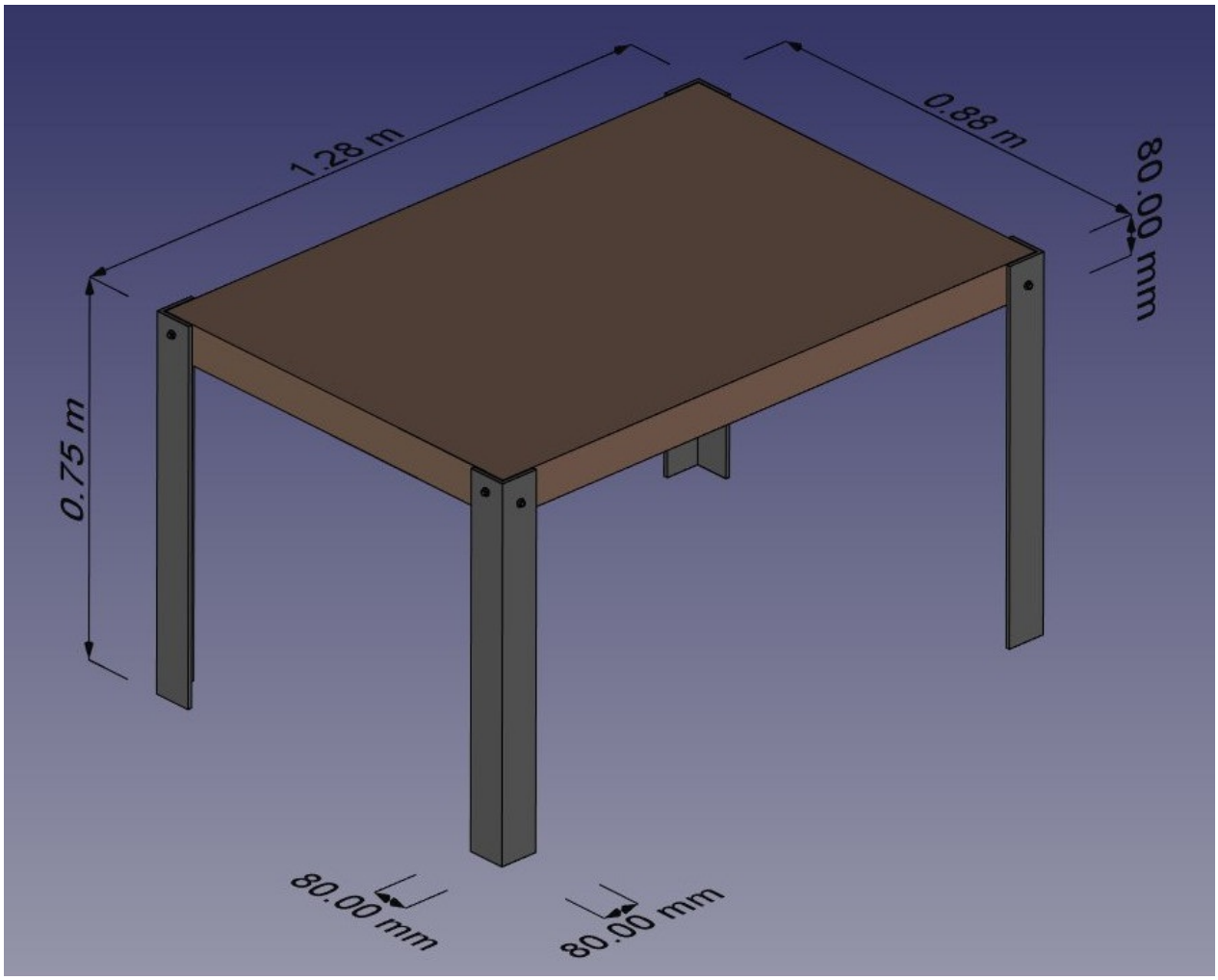


The difference between the two can be compared to the difference between bitmap and vectorial images. As with bitmap images, polygon meshes have their curved surfaces fractionned in a series of points. If you look at it from very close, or print it very large, you will see not a curved but a faceted surface. In both vectorial images and BREP data, the position of any point on a curve is not stored in the geometry but calculated on the fly, with exact precision.

In FreeCAD, all BREP-based geometry is handled by another piece of open-source software, OpenCasCade. The main interface between FreeCAD and the OpenCasCade kernel is the Part Workbench. Most other workbenches build their functionality on top of the Part Workbench.

Although other workbenches often offer more advanced tools to build and manipulate geometry, since they all actually manipulate Part objects, it is very useful to know how these objects work internally, and be able to use the Part tools since, being more simple, they can very often help you to work around problems that the more intelligent tools fail to solve properly.

To illustrate the working 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):

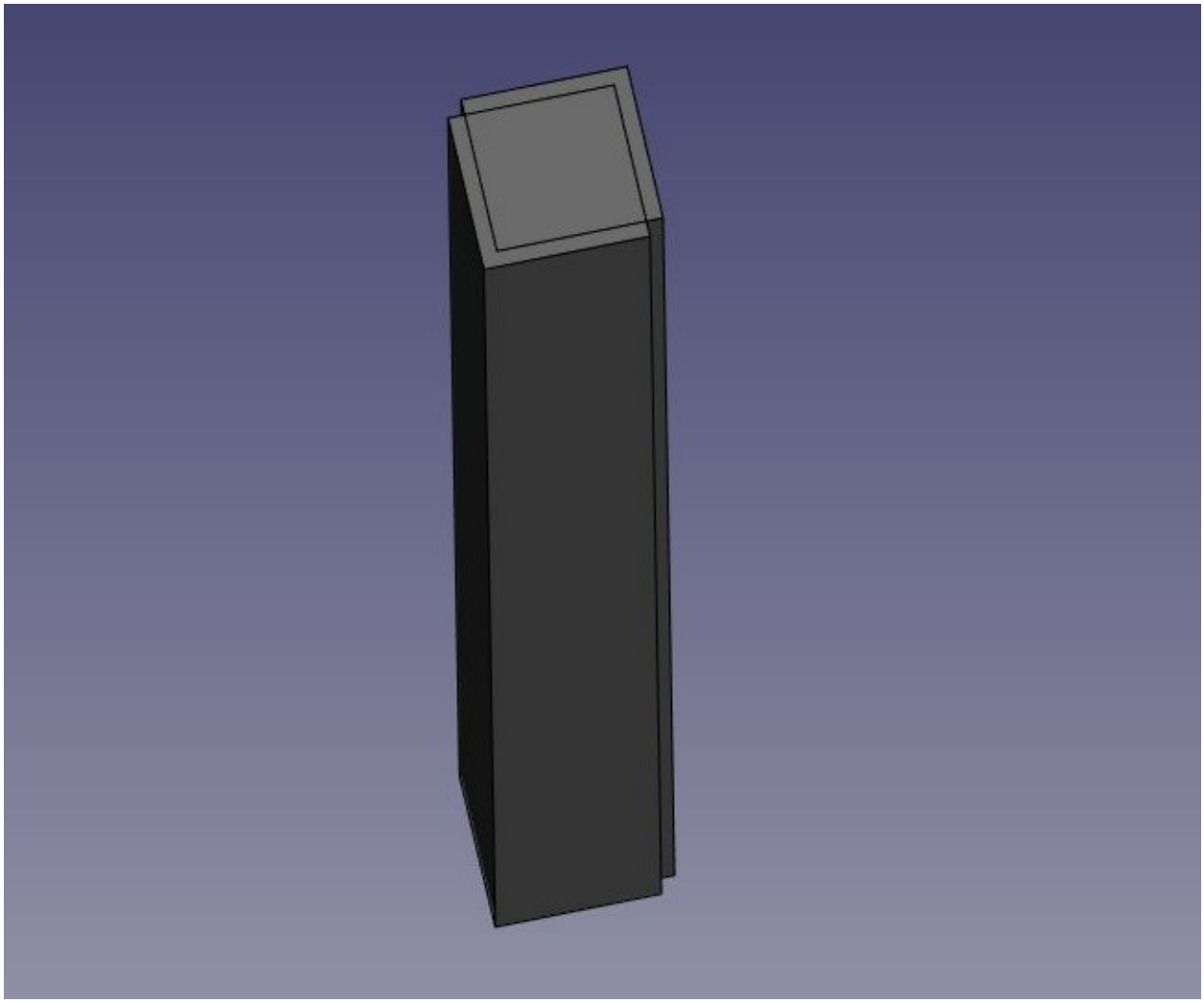


Let's create a new document (**Ctrl+N** or menu File -> New Document), switch to the Part Workbench, and begin with the first foot:

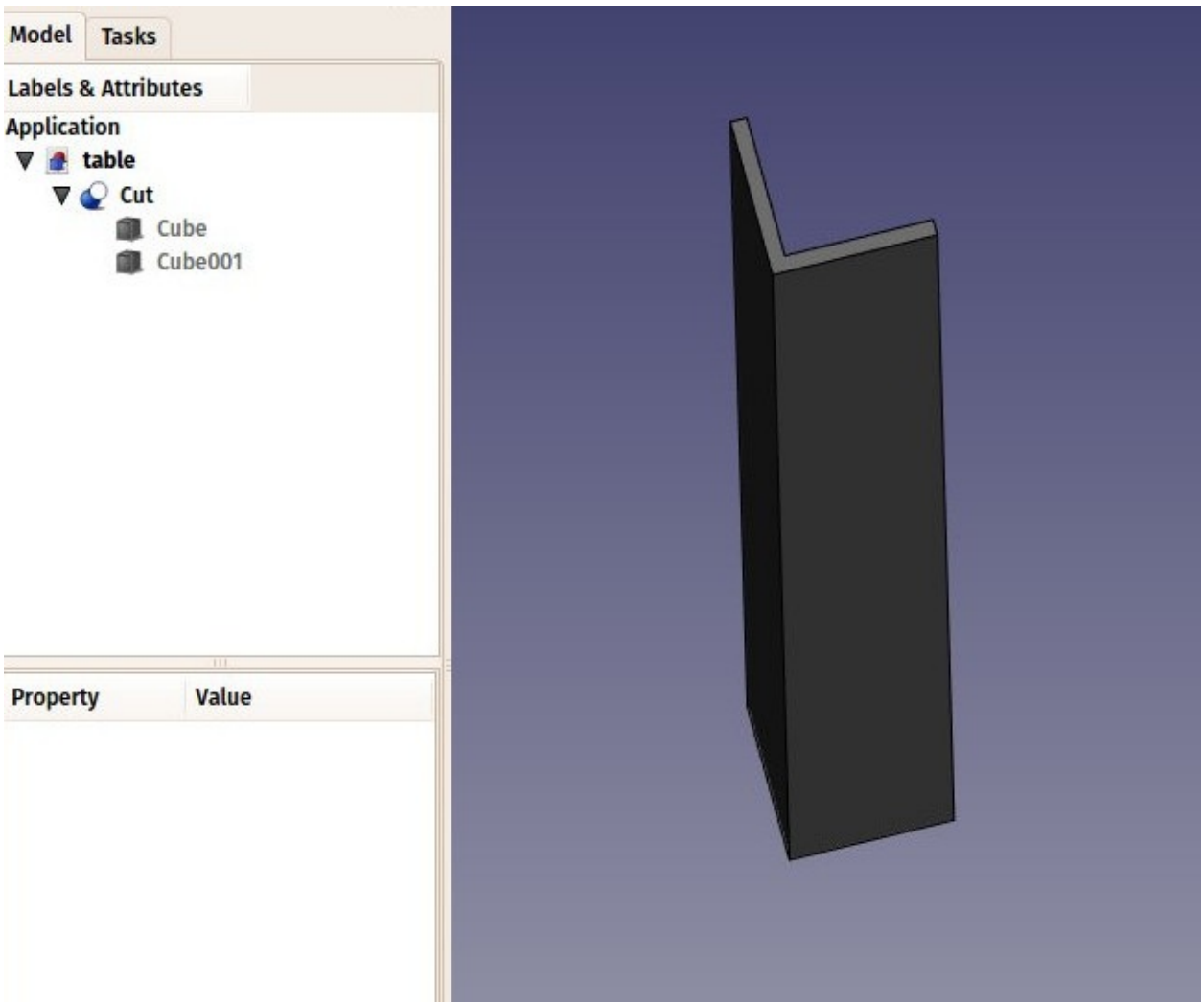
- Press the **Box** button
- Select the box, then set the following properties (in the **Data** tab):
  - Length: 80mm (or 8cm, or 0.8m, FreeCAD works in any unit)
  - Width: 80mm
  - Height: 75cm
- Duplicate the box by pressing **Ctrl+C** then **Ctrl+V** (or menu Edit -> Copy and Paste)
- Select the new object that has been created

- Change its position by editing its Placement property:
  - Position x: 8mm
  - Position y: 8mm

You should obtain two high boxes, one 8mm apart from the other:

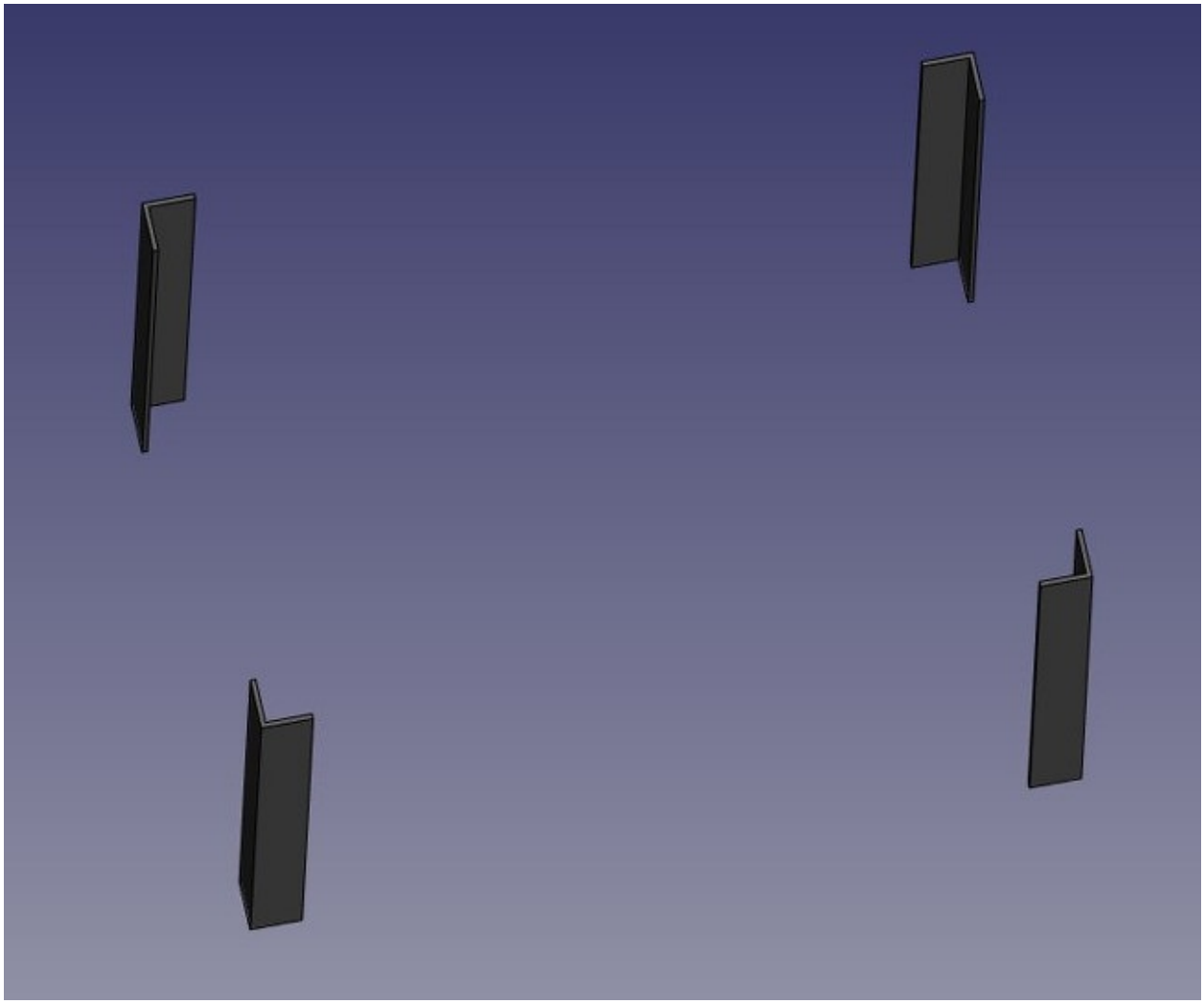


- Now we can subtract one from the other: Select the **first** one, that is, the one that will **stay**, then, with the CTRL key pressed, select the **other** one, that will be **subtracted** (the order is important) and press the **Cut** button:



Observe that the newly created object, called "Cut", still contains the two cubes we used as operands. In fact, the two cubes are still there in the document, they 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, and, if you wish, turn them visible again by right-clickig them or change any of their properties.

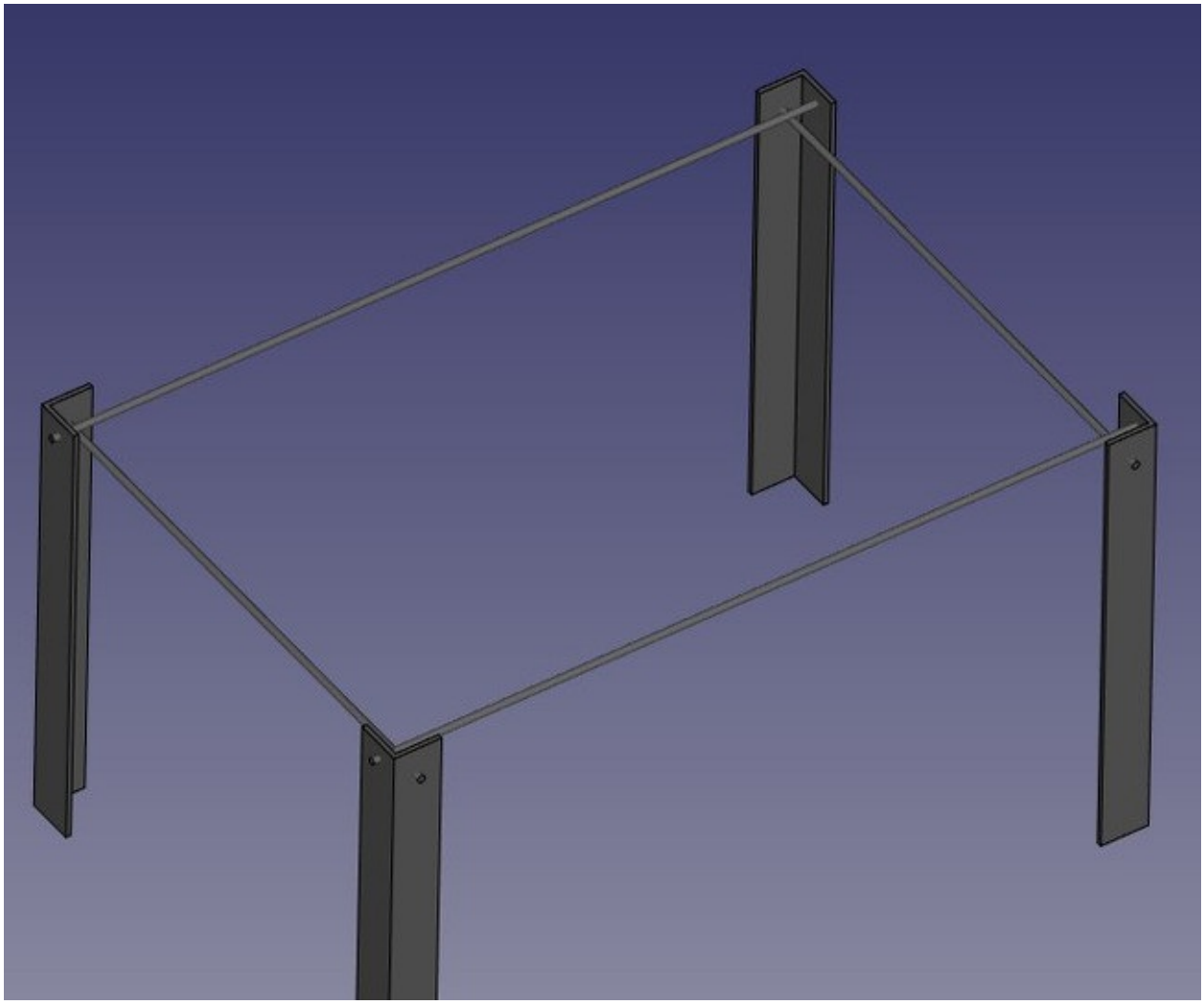
- Now let's create the three other feet by duplicating our base cube 6 other times. Since it is still copied, you can simply paste (Ctrl+V) 6 times. Change their position as follows:
    - cube002: x: 0, y: 80cm
    - cube003: x: 8mm, y: 79.2cm
    - cube004: x: 120cm, y: 0
    - cube005: x: 119.2cm, y: 8mm
    - cube006: x: 120cm, y: 80cm
    - cube007: x: 119.2cm, y: 79.2cm
- Now let's do the three other cuts, selecting first the "host" cube then the cube to be cut off. We now have four Cut objects:



You might have been thinking that, instead of duplicating the base cube six times, we could have duplicated the complete foot three times. This is totally true, as always in FreeCAD, there are many ways to achieve a same result. This is a precious thing to remember, because, as we will advance into more complex objects, some operations might not give the correct result and we often need to try other ways.

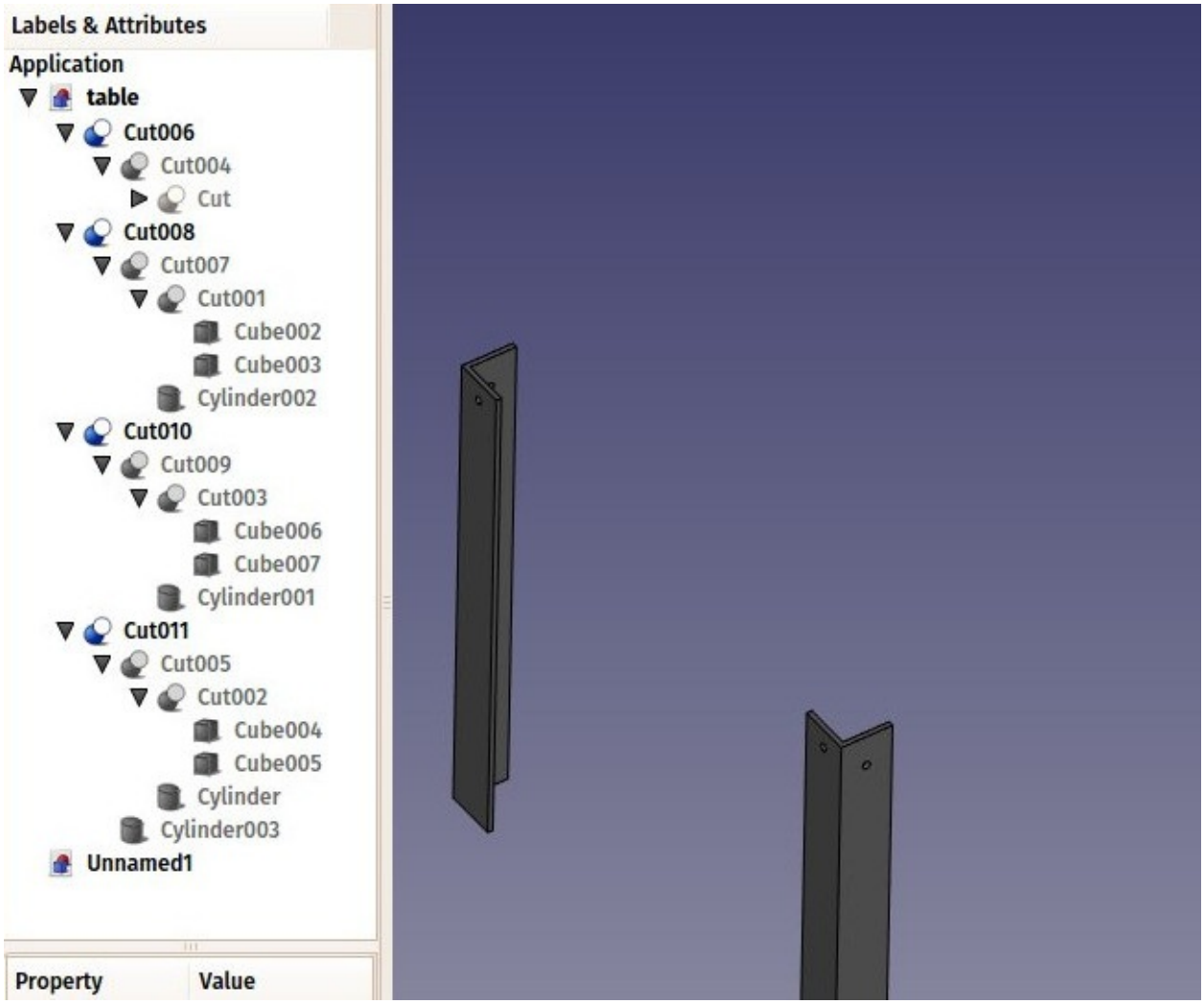
- We will now make holes for the screws, using the same Cut method. Since we need 8 holes, two in each foot, we could make 8 objects to be subtracted. Instead, let's explore other ways and make 4 tubes, that will be reused by two of the feet. So let's create four tubes by using the **Cylinder** tool. You can again, make only one and duplicate it afterwards. Give all cylinders a radius of 6mm. This time, we will need to rotate them, which is also done via the **Placement** property:
  - cylinder: height: 130cm, angle: 90°, axis: x:0,y:1, position: x:-10mm, y:40mm, z:72cm
  - cylinder001: height: 130cm, angle: 90°, axis: x:0,y:1, position: x:-10mm, y:84cm, z:72cm
  - cylinder002: height: 90cm, angle: 90°, axis: x:-1,y:0, position: x:40mm, y:-10mm, z:70cm

- cylinder003: height: 90cm, angle: 90°, axis: x:-1,y:0, position: x:124cm, y:-10mm, z:70cm



You will notice that the cylinders are a bit longer than needed. This is because, as in all solid-based 3D applications, boolean operations in FreeCAD are sometimes oversensitive to face-on-face situations and might fail. By doing this, we put ourselves on the safe side.

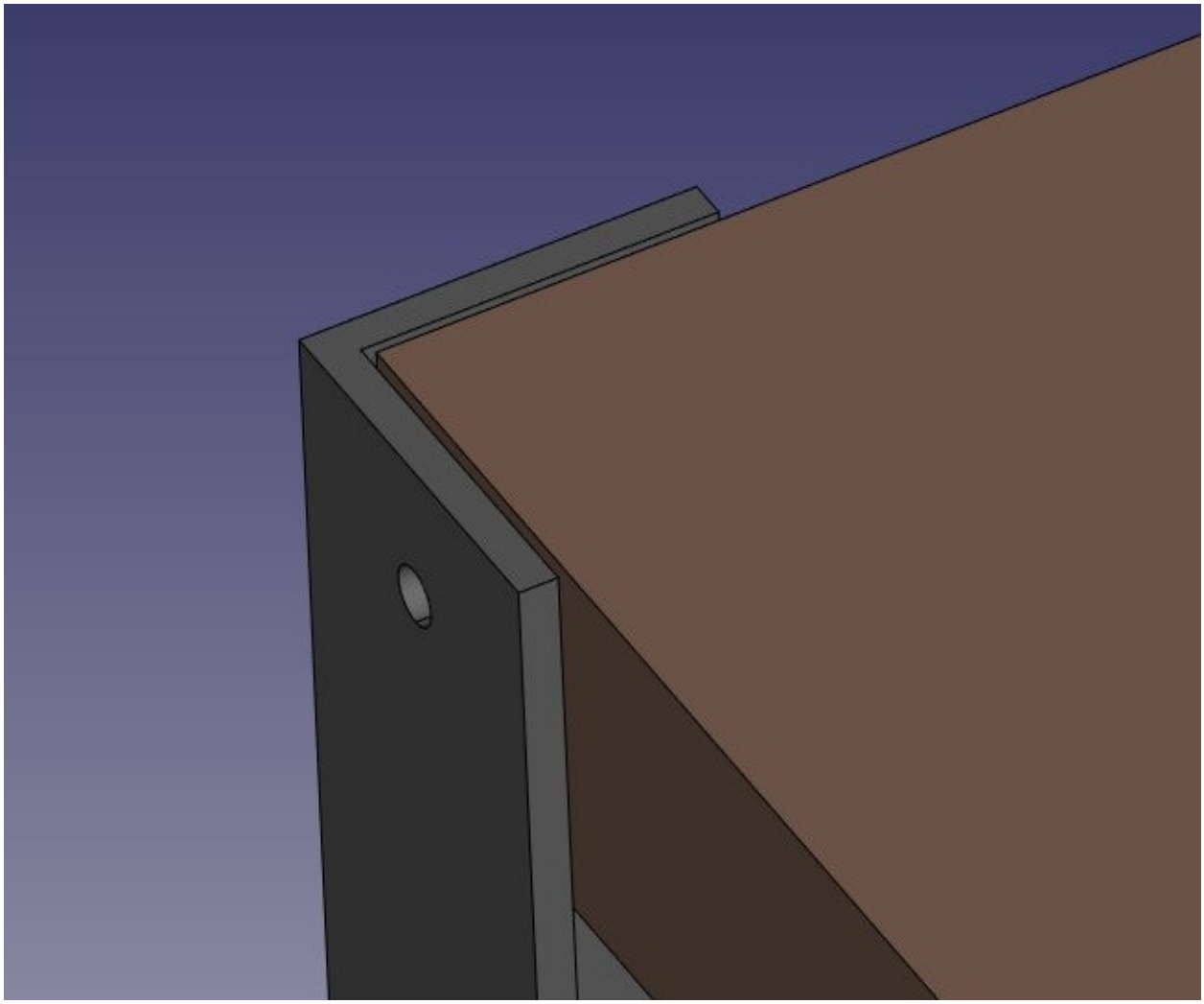
- Now let's do the subtractions. Select the first foot, then, with CTRL pressed, select one of the tubes that crosses it, press the **Cut** button. The hole will be done, and the tube hidden. Find it in the tree view by expanding the pierced foot.
- Select another foot pierced by this hidden tube, then repeat the operation, this time Ctrl+ selecting the tube in the tree view, as it is hidden in the 3D view (you can also make it visible again and select it in the 3D view). Repeat this for the other feet until each of them has its two holes:



As you can see, each foot has become a quite long series of operations. All this stays parametric, and you can go change any parameter of any of the older operations anytime. In FreeCAD, we often refer to this pile as "modeling history", since it in fact carries all the history of the operations you did.

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

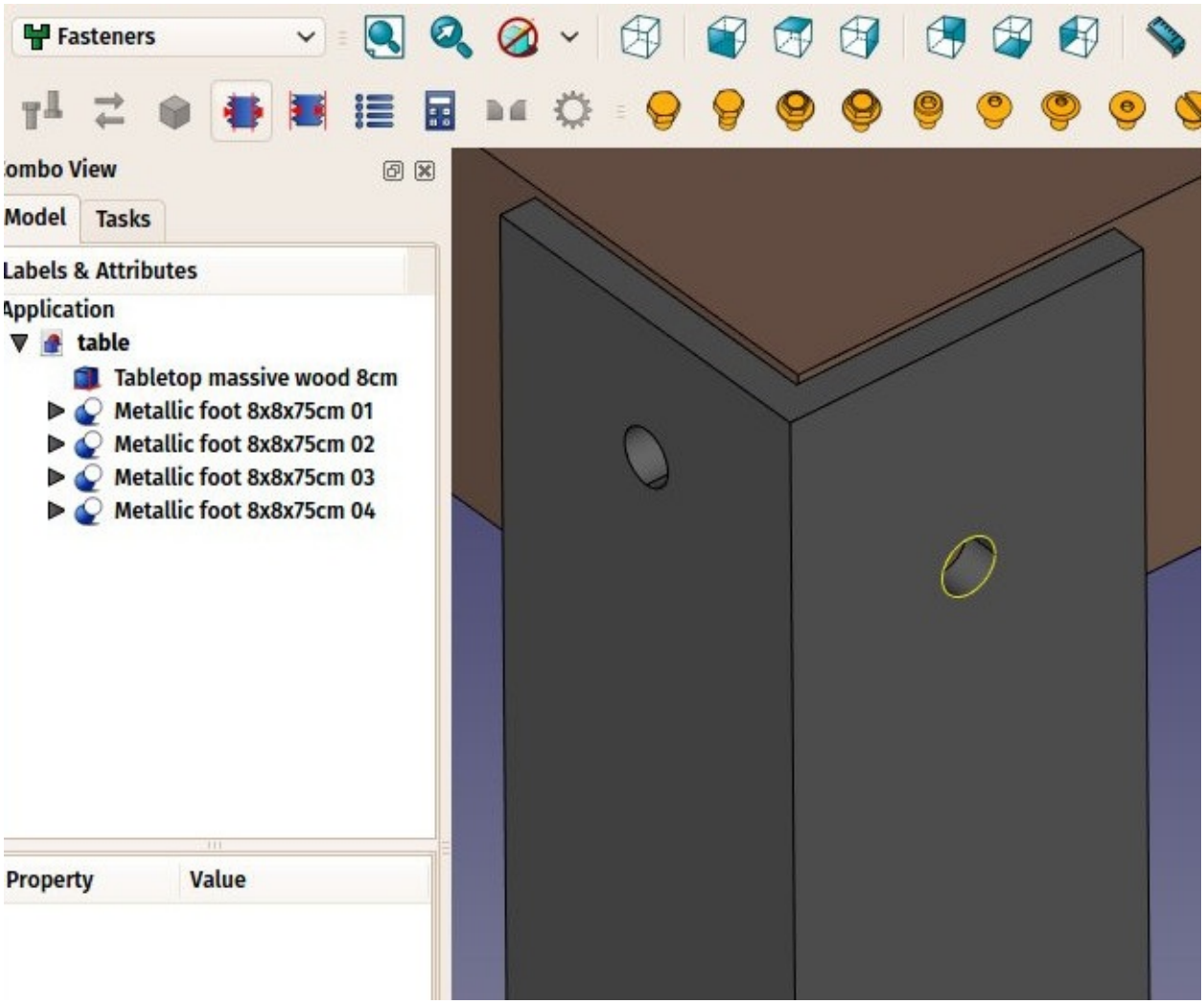
- Now let's do the tabletop, it will be a simple block of wood, let's do it with another **Box** with length: 126cm, width: 86cm, height: 8cm, position: x: 10mm, y: 10mm, z, 67cm. In the **View** tab, you can give it a nice brownish, wood-like color by changing its **Shape Color** property:



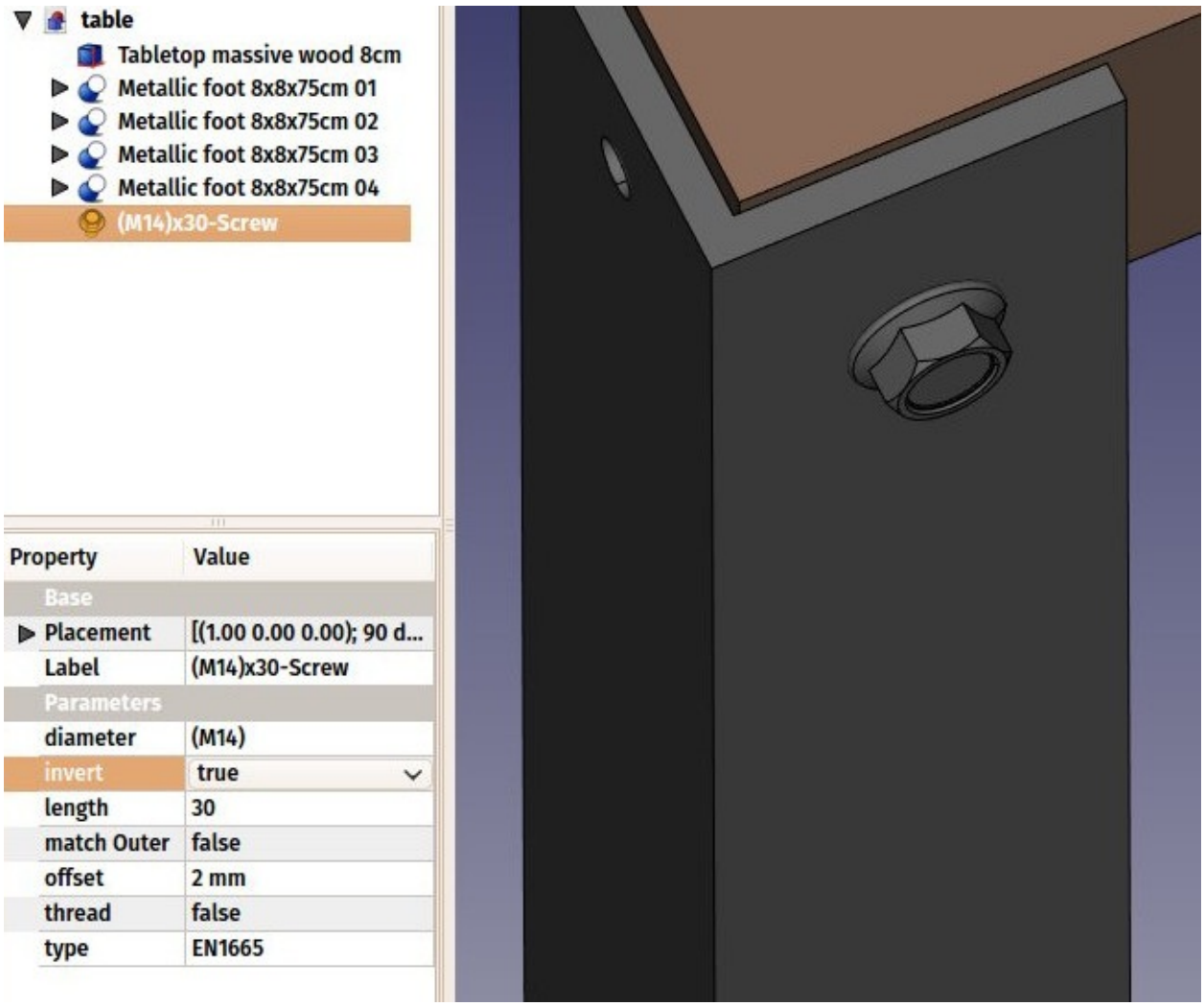
Notice that, although the legs are 8mm thick, we placed it 10mm away, leaving 2mm between them. This is not necessary, of course, it won't happen with the real table, but it is a common thing to do in that kind of "assembled" models, it helps people who look at the model to understand that these are independent parts, that will need to be attached together manually later.

Now that our five pieces are complete, it is a good time to give them more proper names than "Cut015". By right-clicking the objects in the tree view (or pressing **F2**), you can rename them to something more meaningful to yourself or to another person who would open your file later. It is often said that simlpy giving proper names to your objects is much more important than the way you model them.

- We will now place some screws. There is nowadays an extremely useful addon developed by a member of the FreeCAD community, that you can find on the FreeCAD addons repository, called Fasteners, that makes the insertion of screws very easy. Installing additional workbenches is easy and described on the addons pages.
- Once you have installed the Fasteners Workbench and restarted FreeCAD, it will appear in the workbenches list, and we can switch to it. Adding a screw to one of our holes is done by first selecting the circular edge of our hole:



- Then we can press one of the screw buttons of the Fasteners Workbench, for example the *EN 1665 Hexagon bolt with flanges, heavy series*. The screw will be placed and aligned with our hole, and the diameter will automatically be selected to match the size of our hole. Sometimes the screw will be placed inverted, which we can correct by flipping its **invert** proprty. We can also set its offset to 2mm, to follow the same rule we used between the tabletop and the feet:



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

The internal structure of Part objects

As we saw above, it is possible in FreeCAD to select not only whole objects, but parts for them, such as the circular border 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**: the edges are linear geometry like lines, arcs, ellipses or NURBS curves. They usually have two vertices, but some special cases have only one (a closed circle for example).
- **Wires**: A wire is a sequence of edges connected by their endpoints. It can contain edges of any type, and it can be closed or not.
- **Faces**: Faces can be planar or curved, and can be formed by one closed wire, which forms the border of the face, or more than one, in case the face has holes.
- **Shells**: Shells are simply a group of faces connected by their edges. It can be open or

closed.

- **Solids**: When a shell is tighly closed, that is, it has no "leak", it becomes a solid. Solids carry the notion of inside and outside. Many workbench rely on this to make sure the objects they produce can be built in the real world.
- **Compounds**: Compounds are simply aggegates of other shapes, no matter their type, into a single shape.

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

A note about shared design

You might look at the table above, and think its design is not good. The tightening of the feet with the tabletop is probably too weak. You might want to add reforcing pieces, or simply you have other ideas to make it better. 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 helped a tiny bit to make 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: http://www.freecadweb.org/wiki/index.php?title=Part_Module
- The FreeCAD addons repository: https://github.com/FreeCAD/FreeCAD-addons
- The Fasteners Workbench: https://github.com/shaise/FreeCAD_FastenersWB

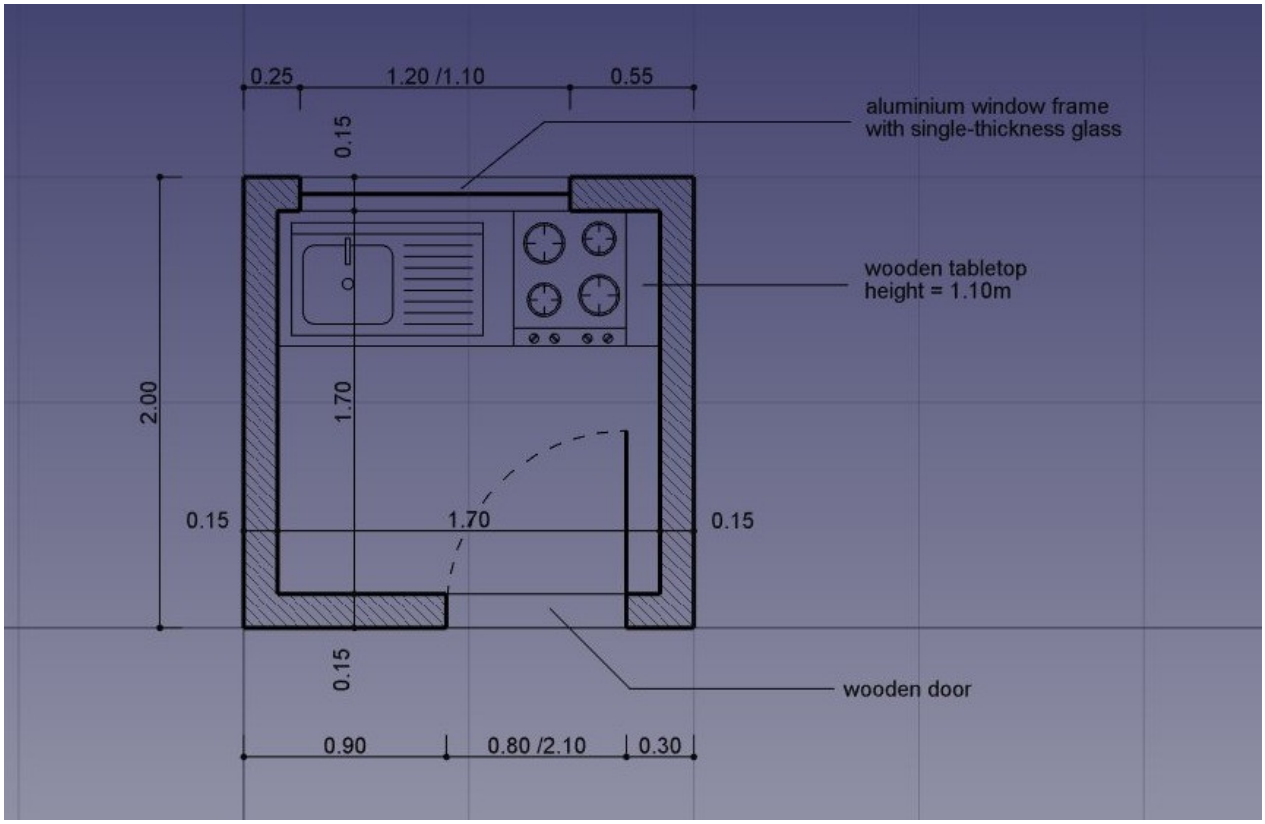
Traditional 2D drafting

You might be interested by FreeCAD because you already have some technical drawing experience, for example with software like AutoCAD. Or you already know something about design, or you prefer to draw things before building them. In either cases, FreeCAD features a more traditional workbench, with tools found in most 2D CAD applications: The Draft Workbench.

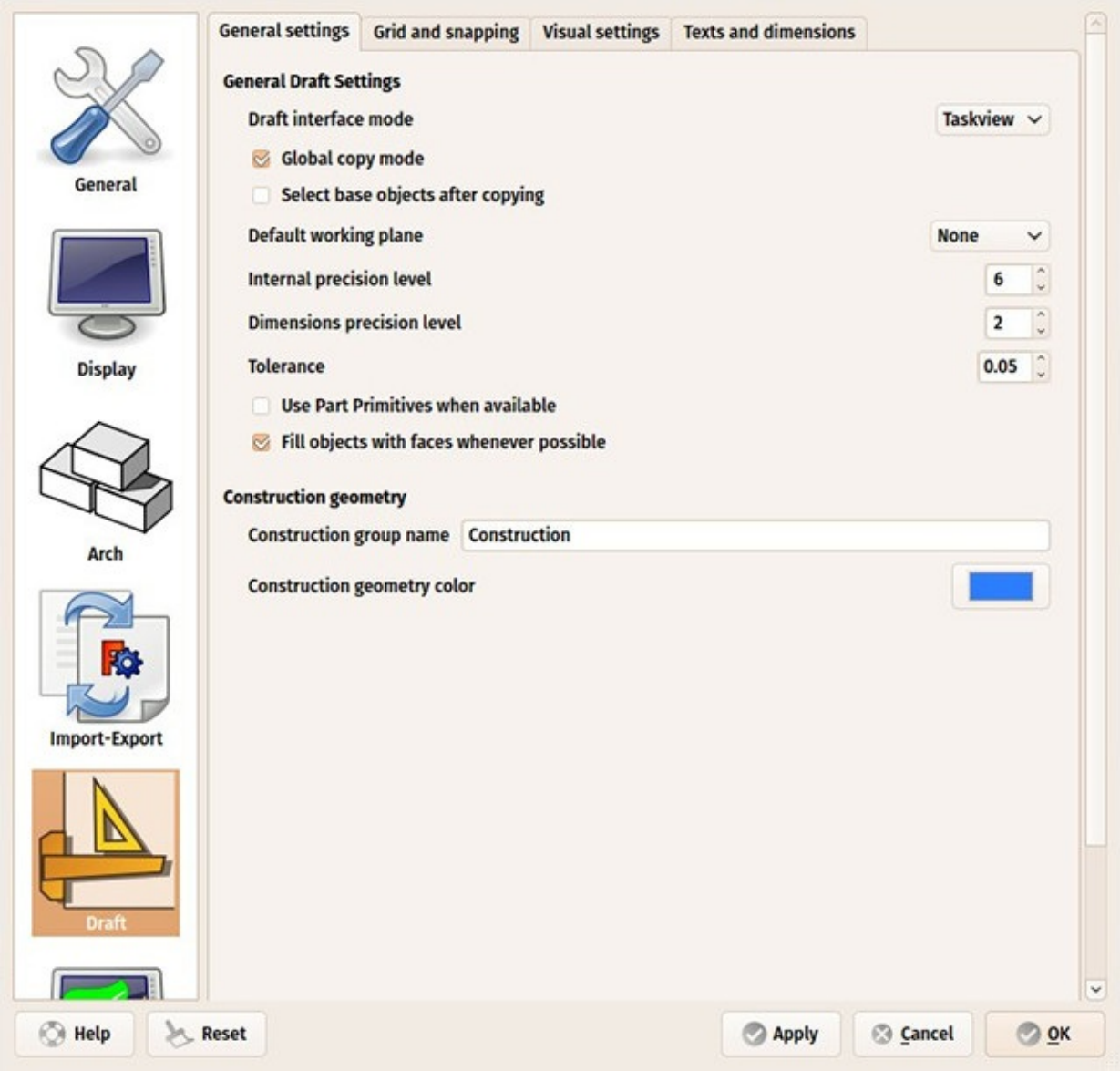
The Draft Workbench, although it adopts ways of working inherited from the traditional 2D CAD world, is not limited at all to the 2D realm. All its tools work in the whole 3D space and many of the Draft tools, for example Move or Rotate, are commonly used all over FreeCAD because they are often more intuitive than changing placement parameters manually.

Among the tools offered by the Draft Workbench, you will find traditional drawing tools like Line, Circle, or Wire (polyline), modification tools like Move, Rotate or Offset, a working plane/grid system that allows you to define precisely in which plane you are working, and a complete snapping system that makes it very easy to draw and position elements precisely in relation to each other.

To showcase the working 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. Configure the grid and working plane, text and dimensions settings to your likings in menu **Edit -> Preferences -> Draft**. In this exercise, however, we will act as if these preference settings were left to their default values.

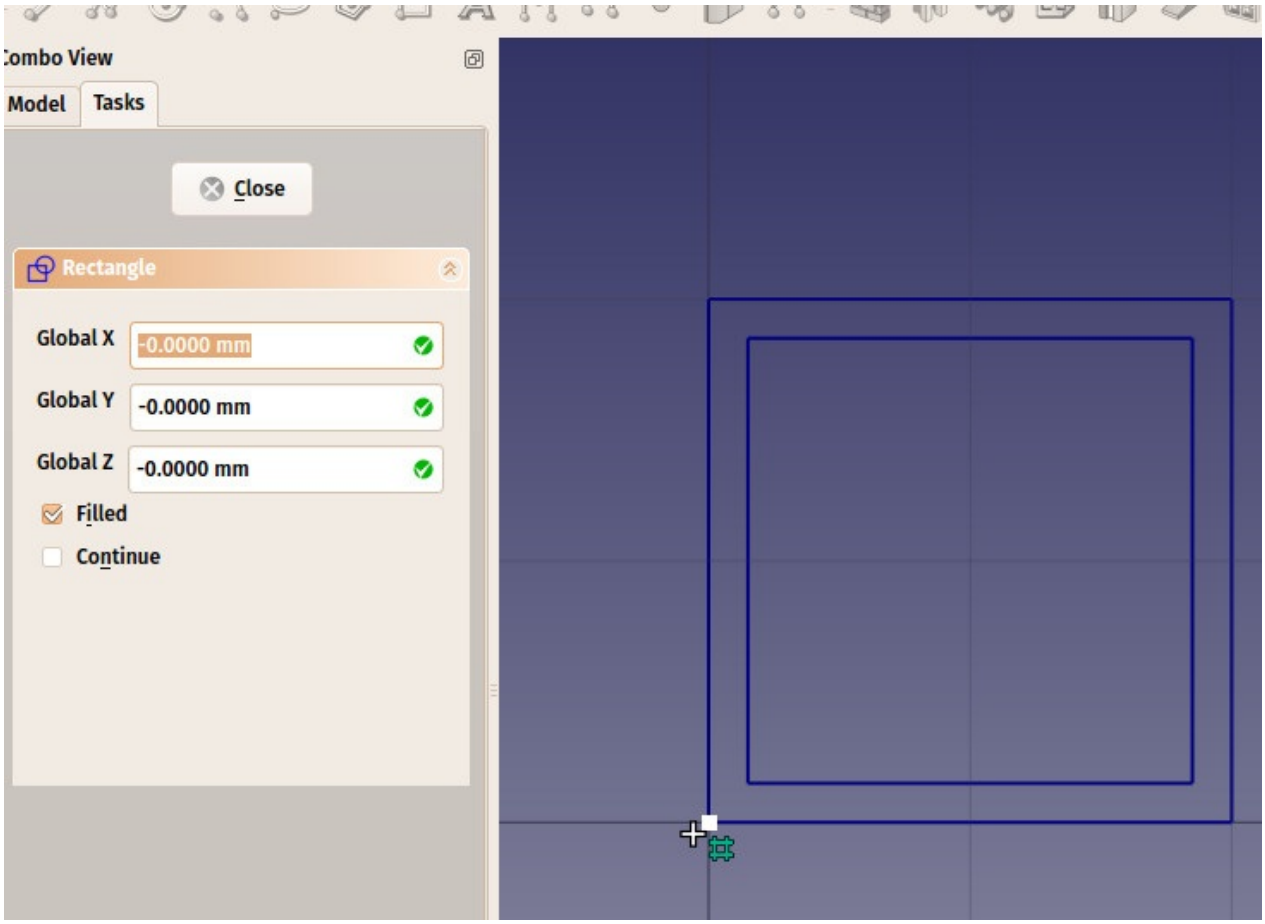


- The Draft Workbench also has two special toolbars: One with **visual settings**, where you can change the current working plane, turn construction mode on/off, set the line color, face color, line weight and text size to be used for new objects, and another one with **snap locations**. There, you can turn the grid on/of and set/unset individual Snap locations:

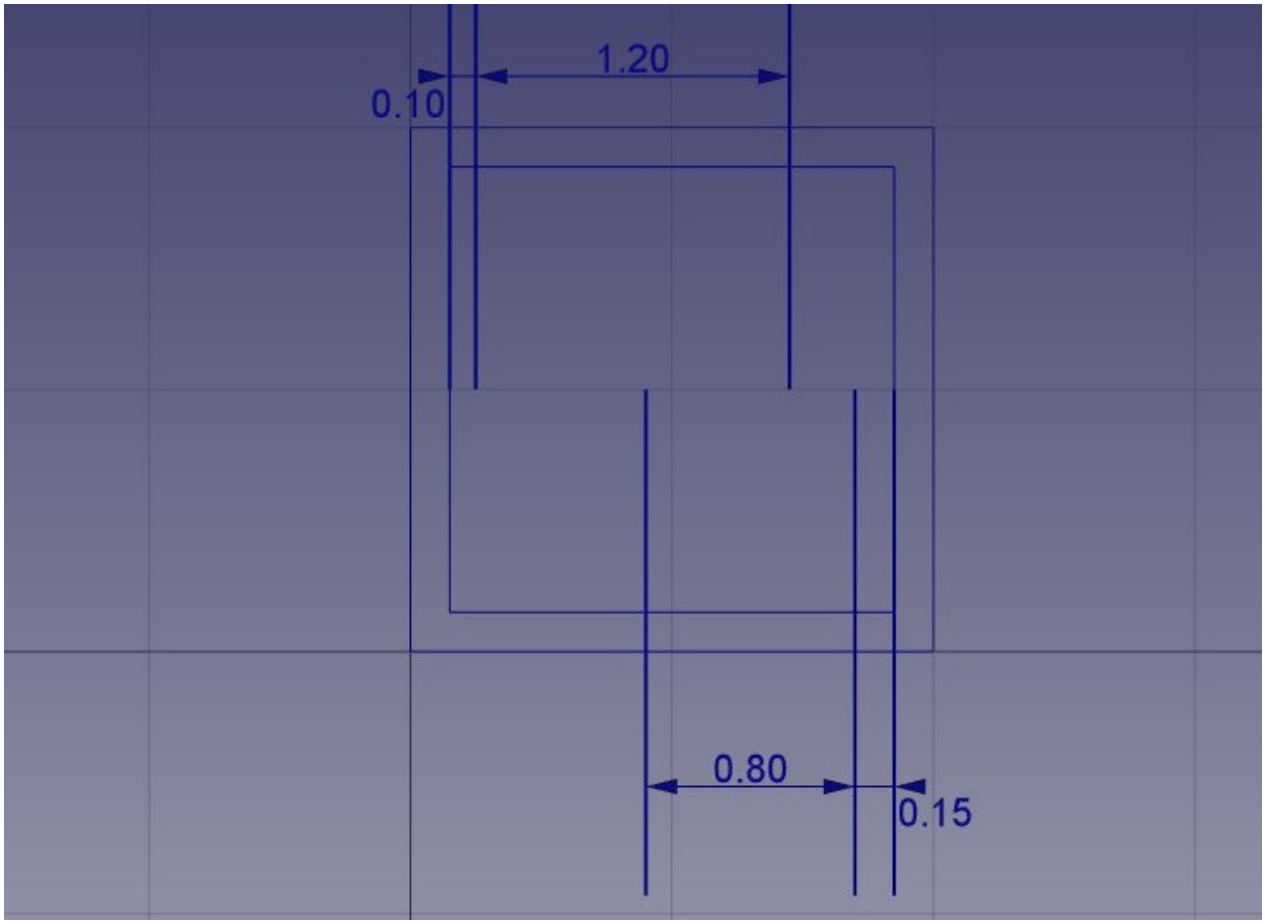
- Let's start by turning **construction mode** on, which will allow us to draw some guidelines on which we will draw our final geometry.
- If you wish, set the **working plane** to *XY*. If you do this, the working plane won't change, no matter the current view. If not, the working plane will adapt automatically to the current view, and you should take care of staying in top view whenever you want to draw on the XY (ground) plane.
- Then, select the Rectangle tool and draw a rectangle, starting at point (0,0,0), of 2 meters by 2 meters (leave the Z at zero). Note that most of the Draft commands can be

fully performed from the keyboard, without touching the mouse, using their two-letter shortcut. Our first 2x2m rectangle can be done like this: `re 0` **Enter** `0` **Enter** `0` **Enter** `2m` **Enter** `2m` **Enter** `0` **Enter**.

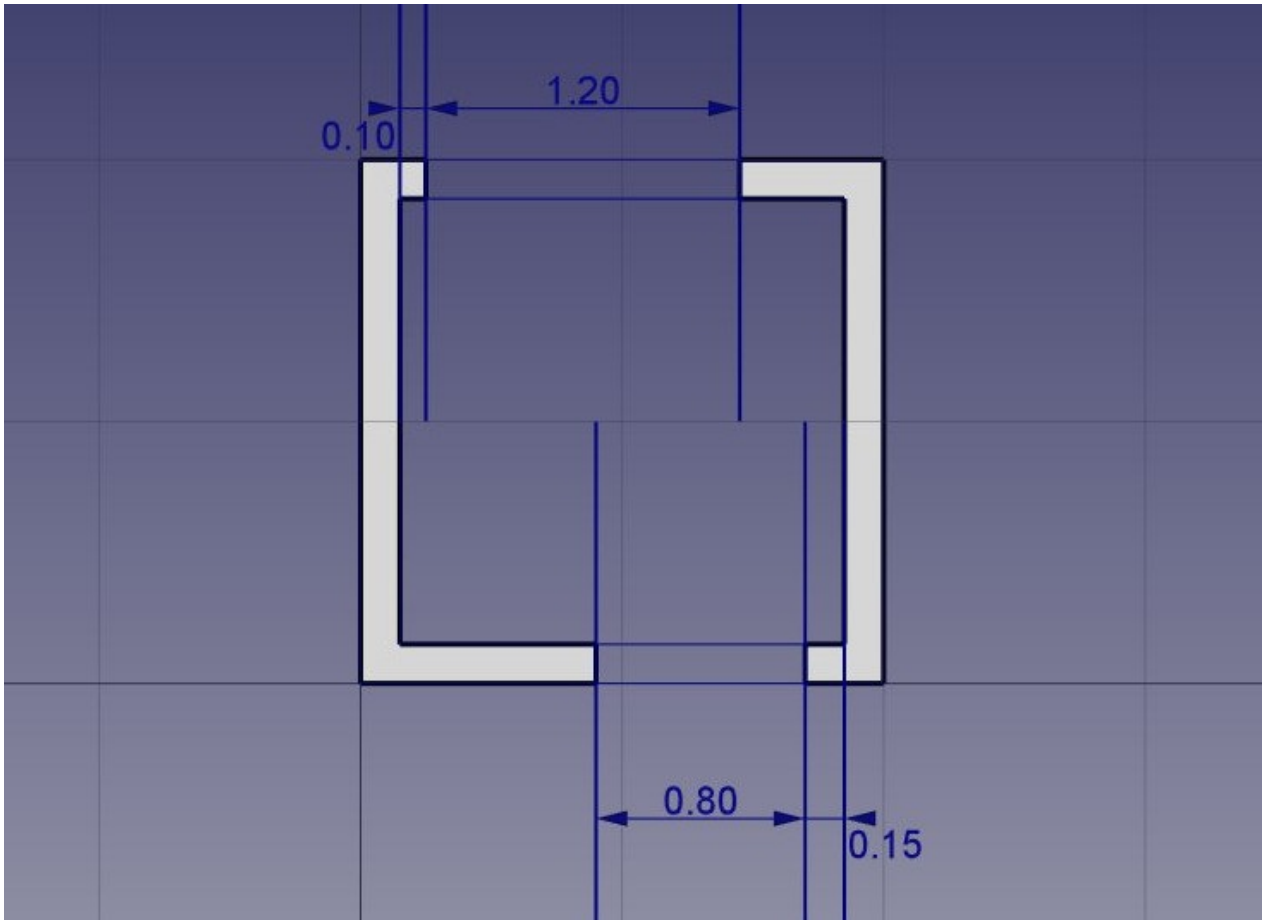
- Duplicate that rectangle by 15cm inside, using the Offset tool, turning its Copy mode on, and giving it a distance of 15cm:



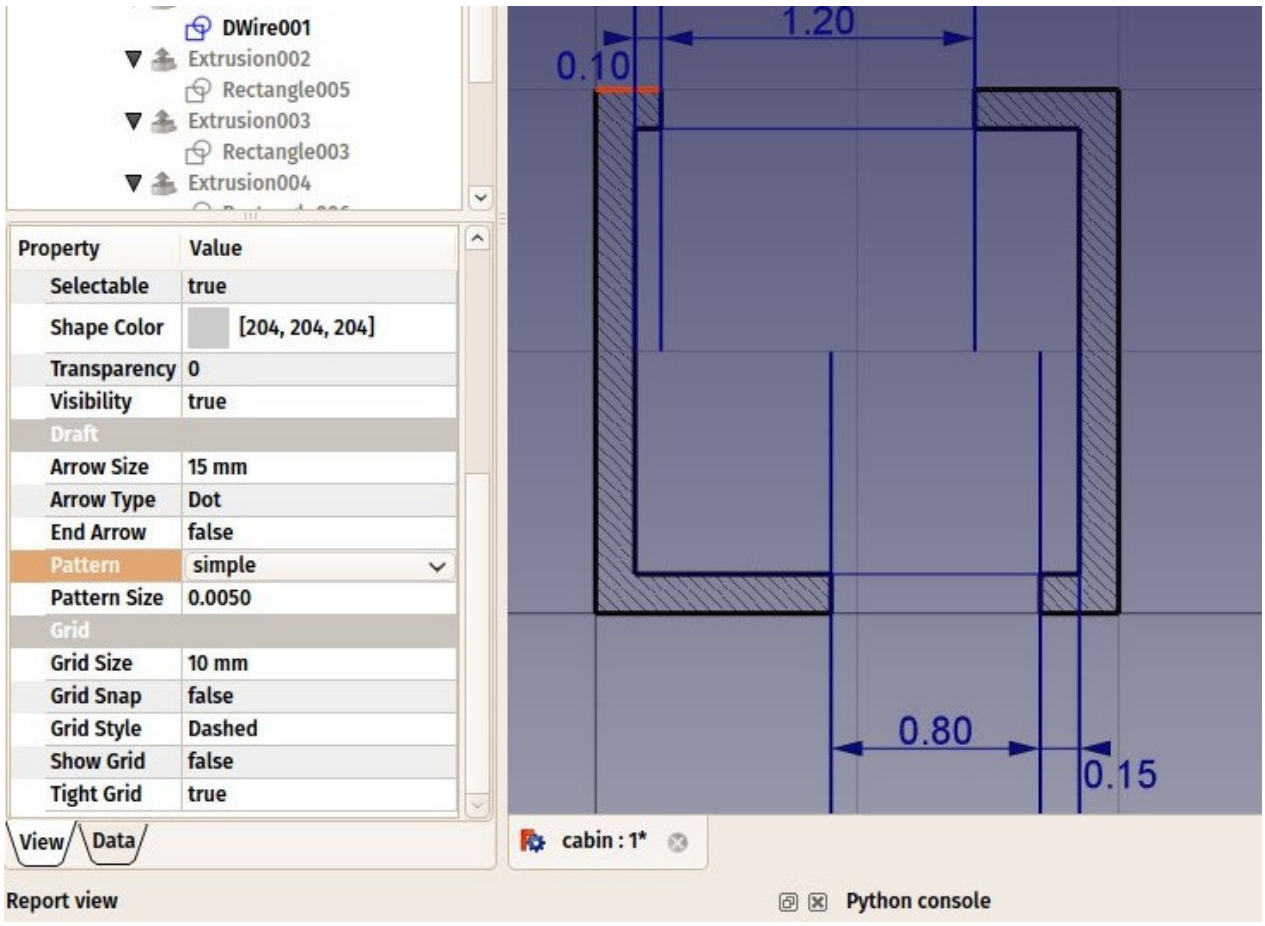
- We can then draw a couple of vertical lines to define where our doors and windows will be placed, using the Line tool. The crossing of these lines with our two rectangles will give us useful intersections to snap our walls to. Draw the first line from point (15cm, 1m, 0) to point (15cm, 3m, 0).
- Duplicate that line 5 times, using the Move tool with Copy mode turned on. Turn also the Relative mode on, which will allow us to define movements in relative distances, which is eaier than calculate the exact position of each line. Give each new copy any start point, you can leave it at (0,0,0) for example, and the following relative endpoints:
  - line001: x: 10cm
  - line002: x: 120cm
  - line003: x: -55cm, y: -2m
  - line004: x: 80cm
  - line005: x: 15cm



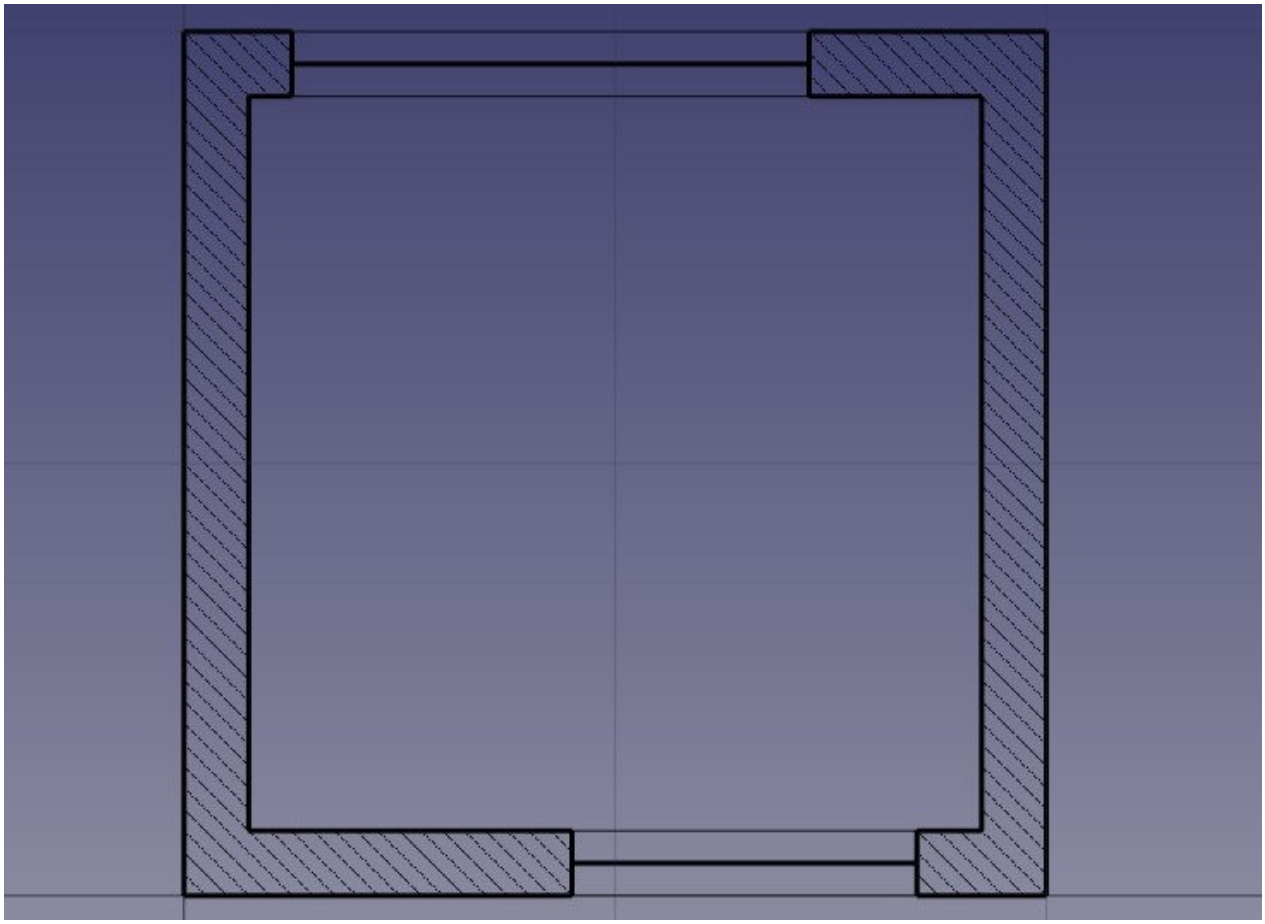
- That is all we need now, so we can switch construction mode off. Check that all the construction geometry has been placed into a "Construction" group, which makes it easy to hide it all at once or even delete it completely later on.
- Now let's draw our two wall pieces using the Wire tool. Make sure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



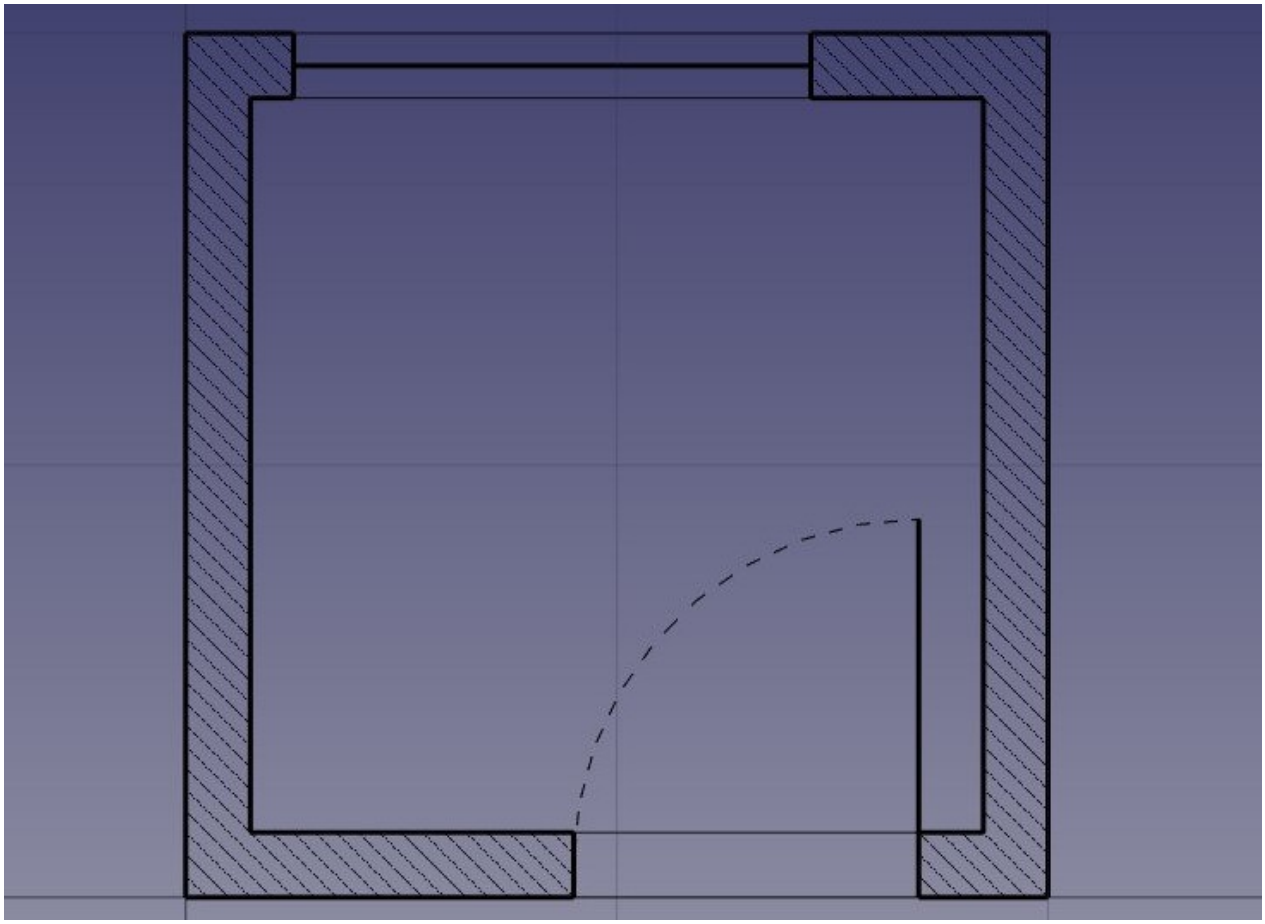
- We can change their default grey color to a nice hatch pattern, by selecting both walls, then setting their **Pattern** property to *Simple*, and their **Pattern size** to your liking, for example *0.005*.



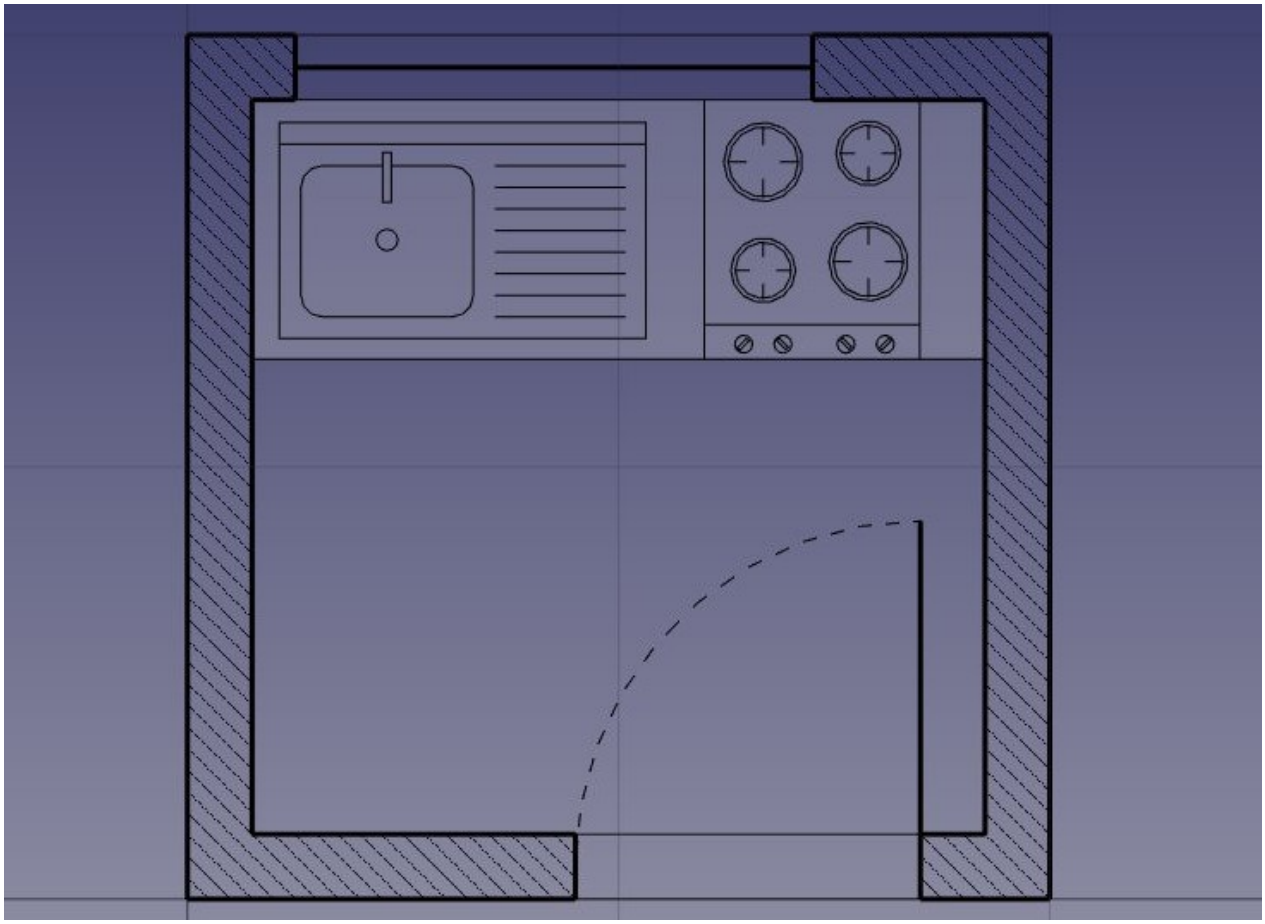
- We can now hide the construction geometry by right-clicking the Construction group and choose **Hide Selection**.
- Let's now draw the windows and doors. Make sure the midpoint snap is turned on, and draw six lines as follow



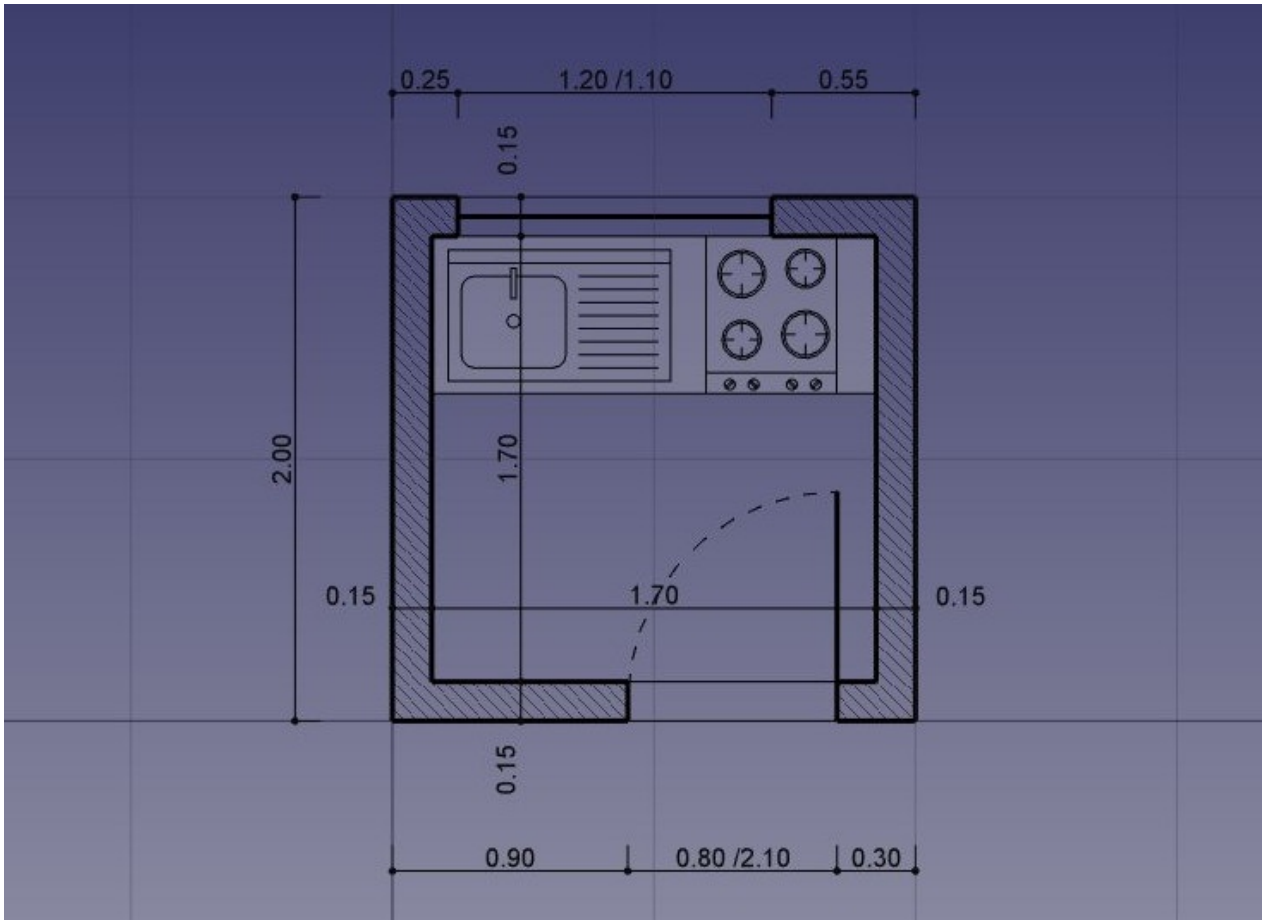
- We will now change the door line to create an opened door symbol. Start by rotating the line using the Rotate tool. Click the enpoint of the line as rotation center, give it a start angle of **0**, and an end angle of **-90**.
- Then create the opening arc with the Arc tool. Pick the same point as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



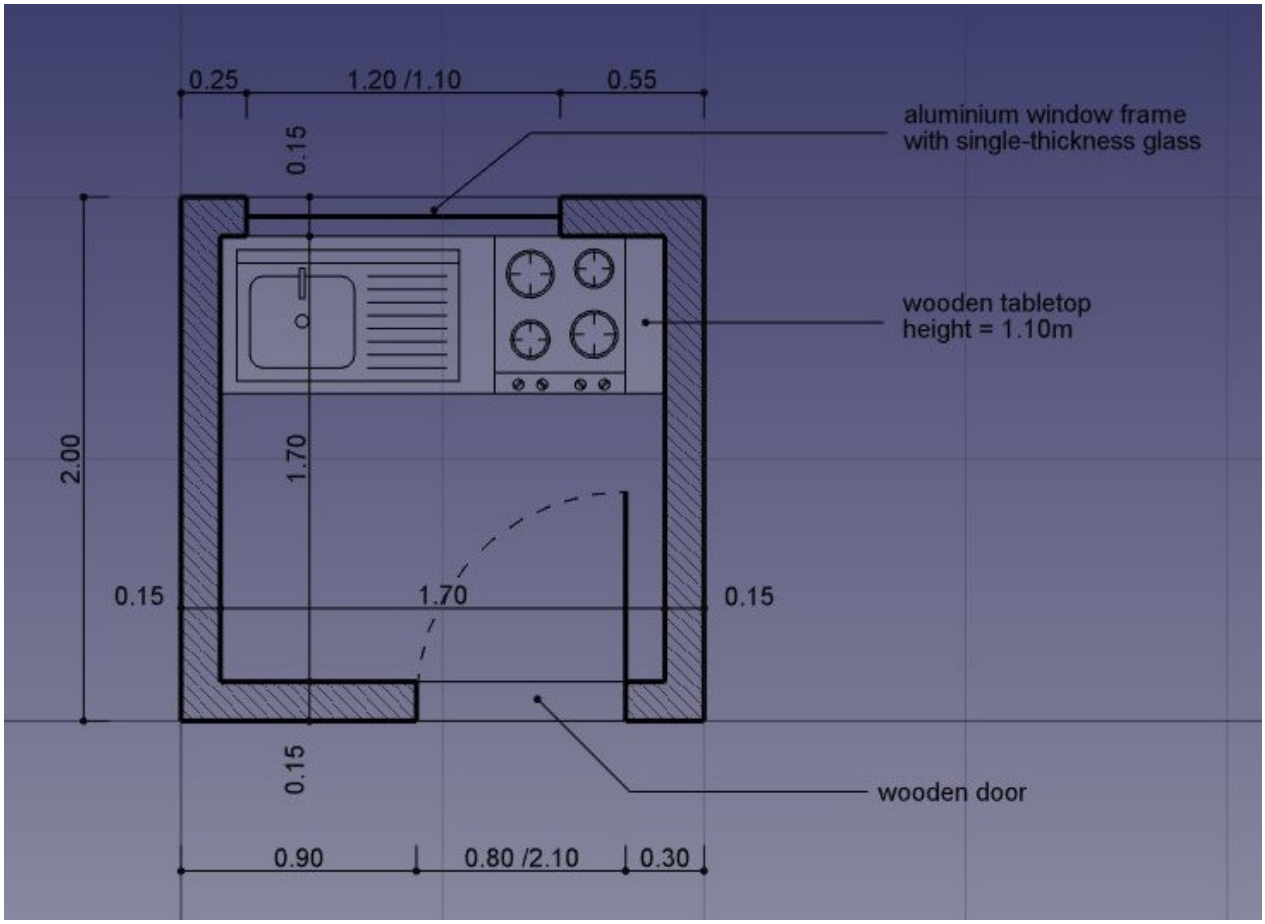
- We can now start placing some furniture. To begin with, let's place a counter by drawing a rectangle from the upper left inner corner, and giving it a width of 170cm and a height of -60cm. In the image below, the **Transparency** property of the rectangle is set to 80%, to give it a nice furniture 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 site, 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 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:



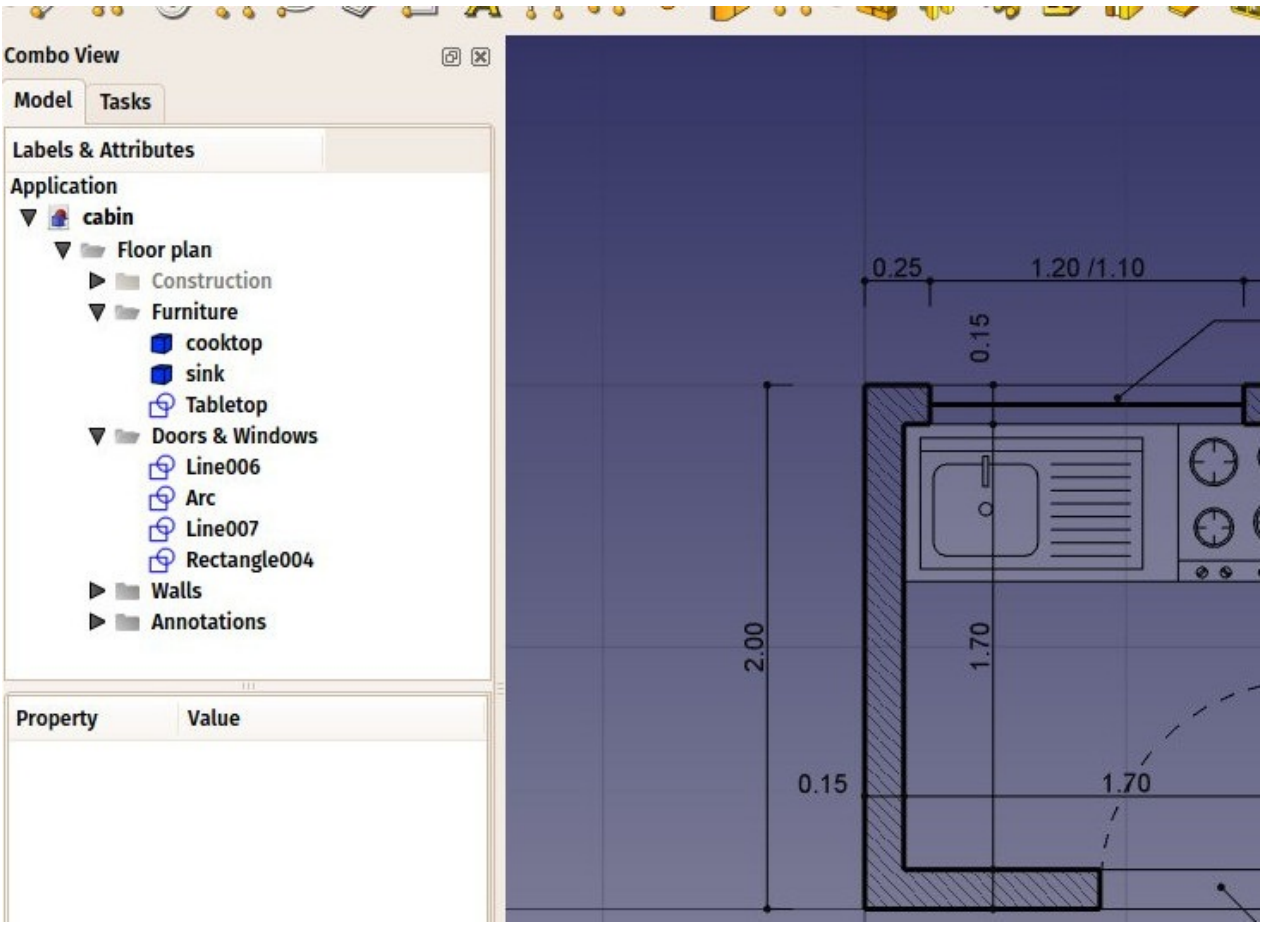
- We can now place a couple of dimensions using the Dimension tool. Dimensions are drawn by clicking 3 points: the start point, an end point, and a third point to place the dimension line. To make horizontal or vertical dimensions, even if the two first points are not aligned, press **Shift** while clicking the second point.
- 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 windows dimensions have been edited to indicate their heights:



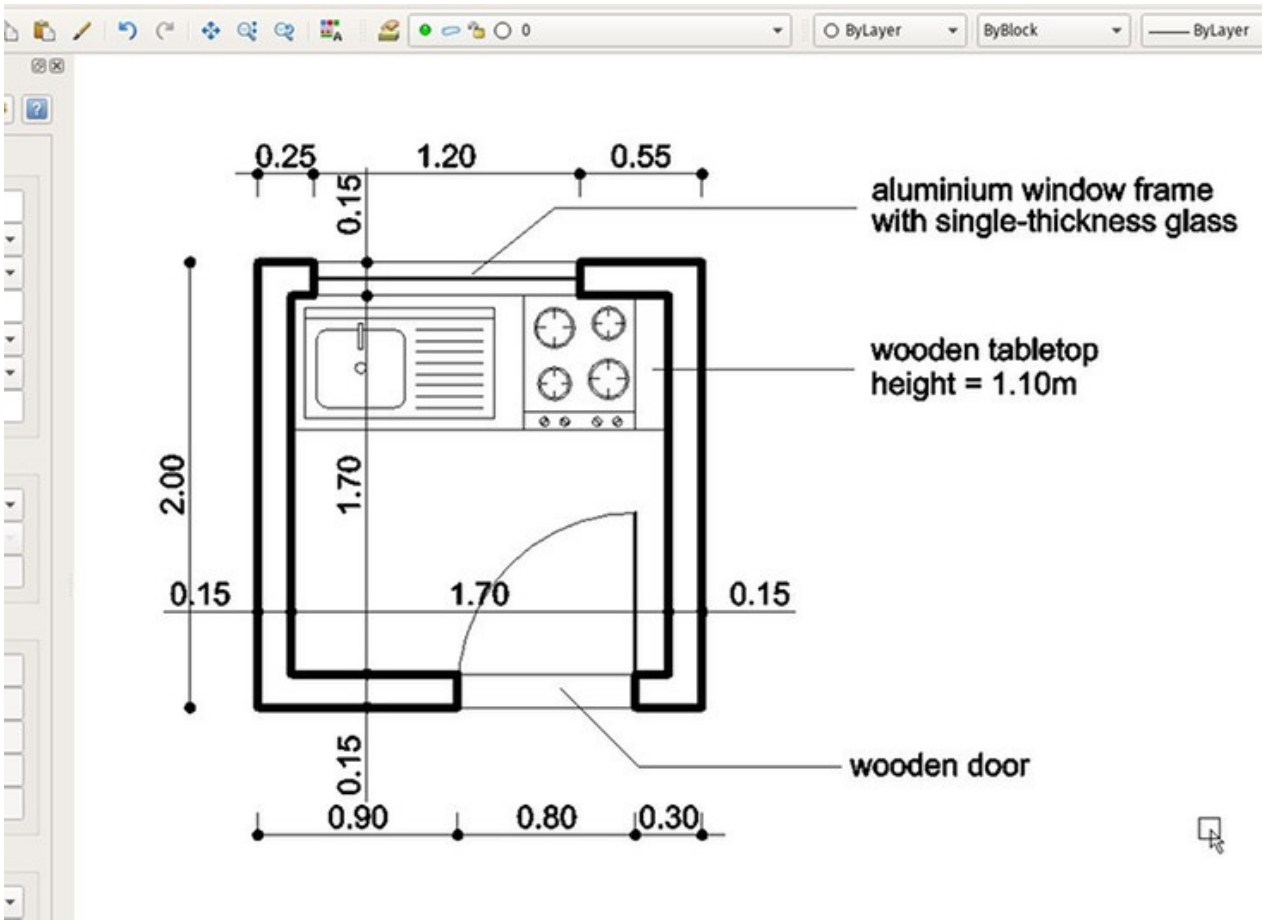
- Let's add some description texts using the Text tool. Click a point to position the text, then enter the lines of text, pressing Enter after each line. To finish, press Enter twice.
- The indication lines (also called "leaders") that link the texts to the item they are describing are simply done with 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**



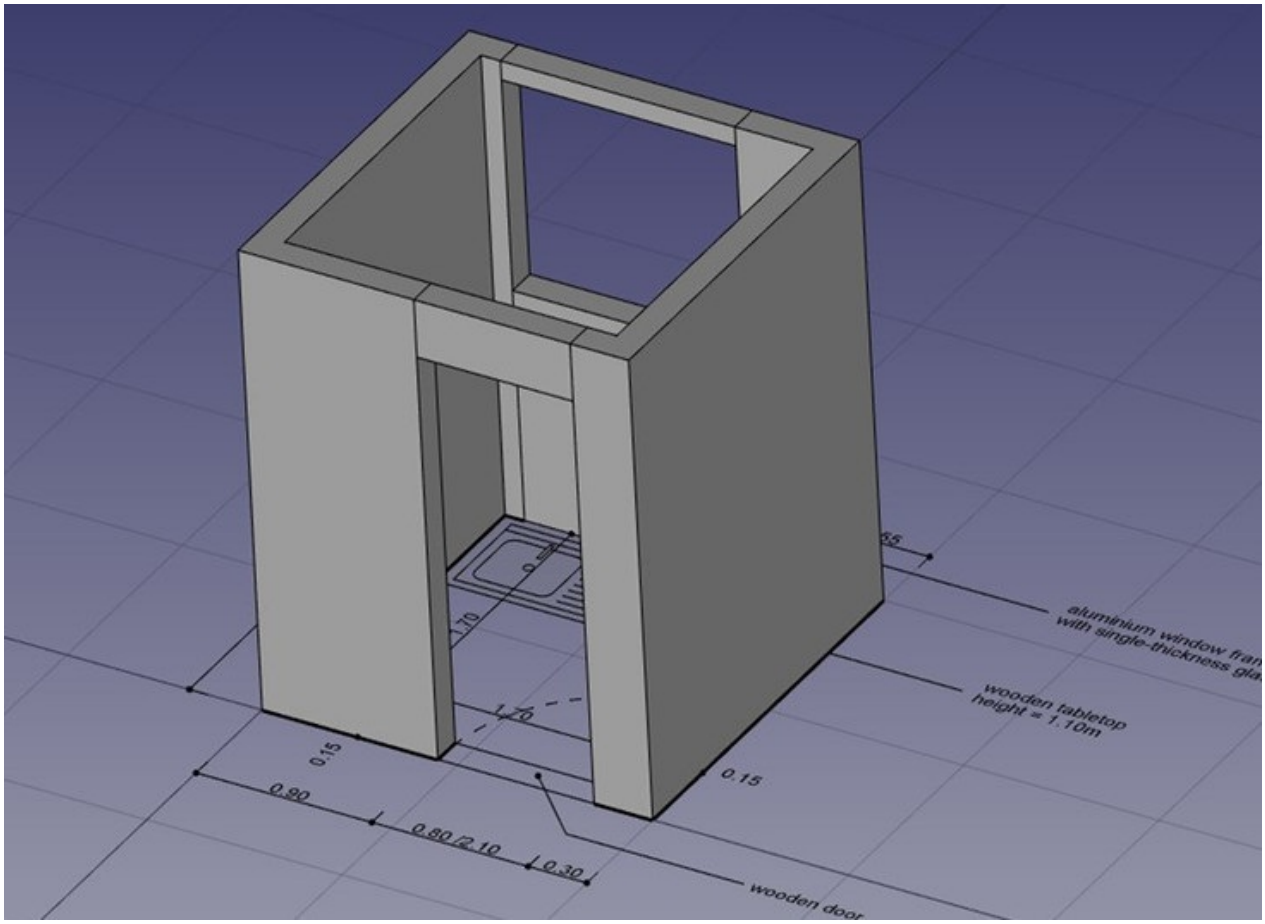
- Our drawing is now complete! Since there begins to be quite a number of objects there, it would be wise do some cleaning and place everything in a nice structure of groups, to make the file easier to understand to another person:



- We can now print our work by placing it on a Drawing sheet, which we will show further 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 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 thing about the Draft Workbench, however, is that the geometry you create with it can be used as a base or easily extruded into 3D objects, simply by using the Extrude tool from the Part Workbench, or, to stay in Draft, the Trimex (Trim/Extend/Extrude) tool, which under the hood performs a Part Extrusion, but does it "the Draft way", that is, allows you to indicate and snap the extrusion length graphically. Experiment extruding our walls as shown below.
- 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 other3D solids. Experiment 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, then using the boolean tools from the Part Workbench to subtract them from another solid, as we saw in the previous chapter.

Fundamentally, what the Draft Workbench does is provide graphical ways to create basic Part operations. While in Part you will usually position objects by setting their placement property by hand, in Draft you can do it on-screen. There are times when one is better, other times when the other is preferable. Don't forget, you can create custom toolars in one of these workbenches, add the tools from the other, and get the best of both worlds.

Downloads

- 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

Read more

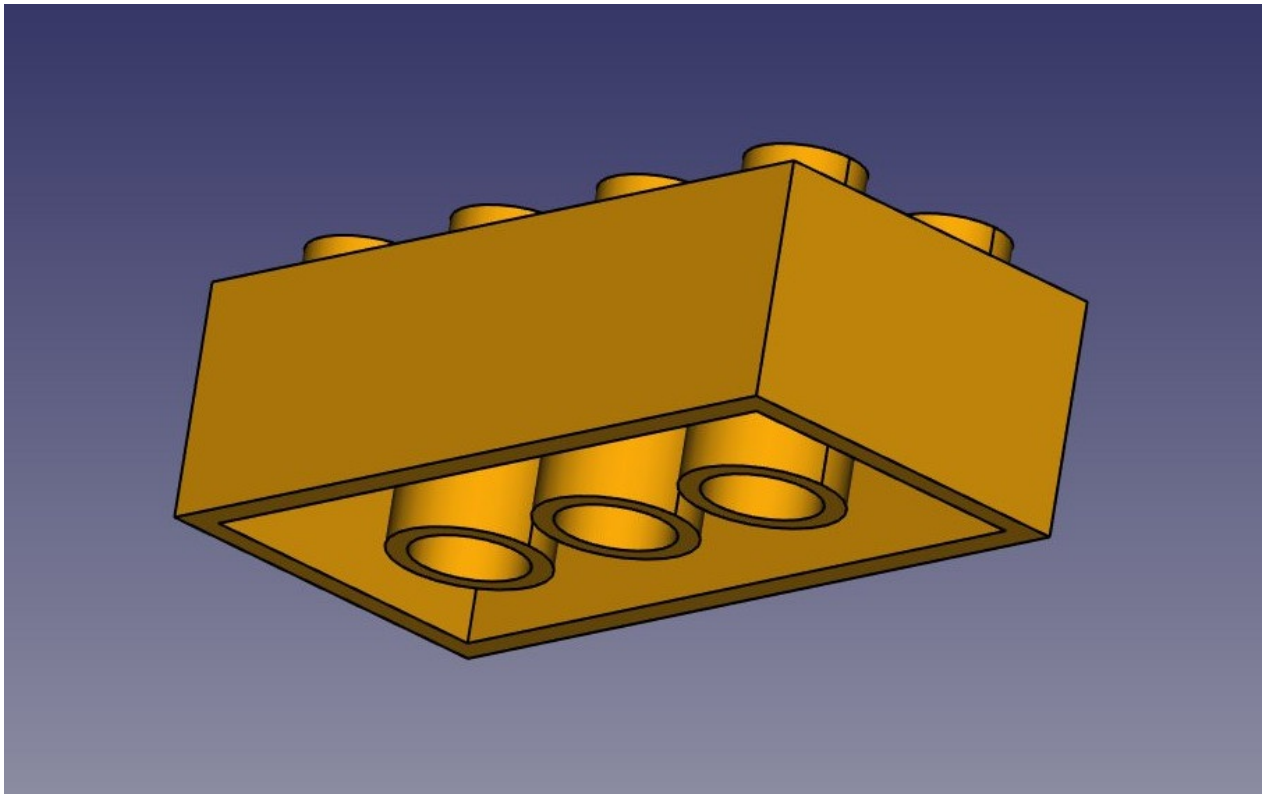
- The Draft Workbench: http://www.freecadweb.org/wiki/index.php?title=Draft_Module
- Snapping: http://www.freecadweb.org/wiki/index.php?title=Draft_Snap
- The Draft working plane: http://www.freecadweb.org/wiki/index.php?title=Draft_SelectPlane

Modeling for product design

Product design is originally a comercial term, but in the 3D world, it often means modeling something with the idea to have it 3D-printed or, more generally, manufactured by a machine, being a 3D printer or a CNC machine.

When you print objects in 3D, it is of ultimate importance that your objects are **solid**. As they will become real, solid objects, this is obvious. Nothing prevent them from being hollow inside, of course. But you always need to have a clear notion of which point is inside the material, and which point is outside, because the 3D printer or the CNC machine needs to know exactly what is filled with material and what is not. For this reason, in FreeCAD, the Part Design Workbench is the perfect tool to build such pieces, because it will always take care for you that your objects stay solid and buildable.

To illustrate how the PartDesign Workbench works, let's model this well-known piece of Lego



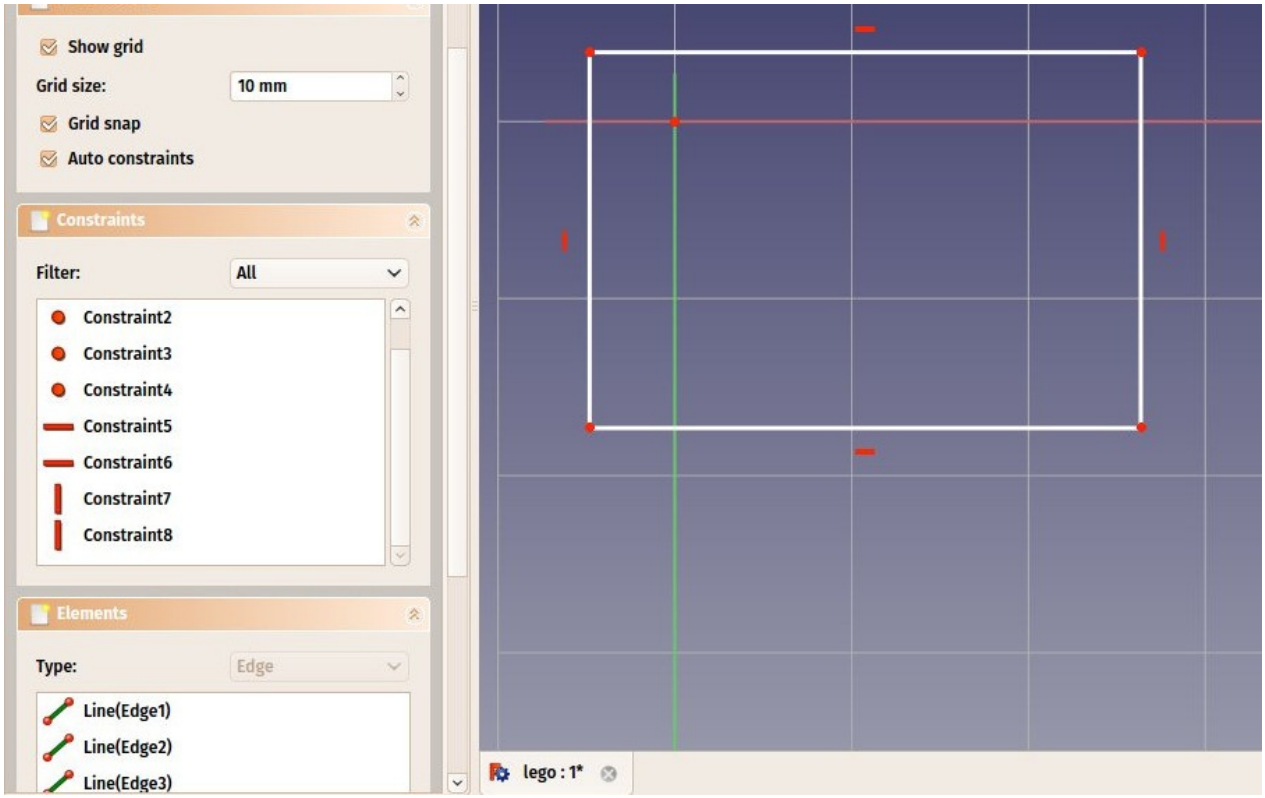
The cool thing with Lego pieces is that the dimensions are easy to obtain on the internet, at least for the standard pieces. These are pretty easy to model and print on a 3D printer, and with a bit of patience (3D printing often requires much adjustment and fine-tuning) you can make pieces that are totally compatible and click perfectly into original Lego blocks.In the example below, we will make a piece that is 1.5 times bigger than the original.

We will now use exclusively the Sketcher and Part Design 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.

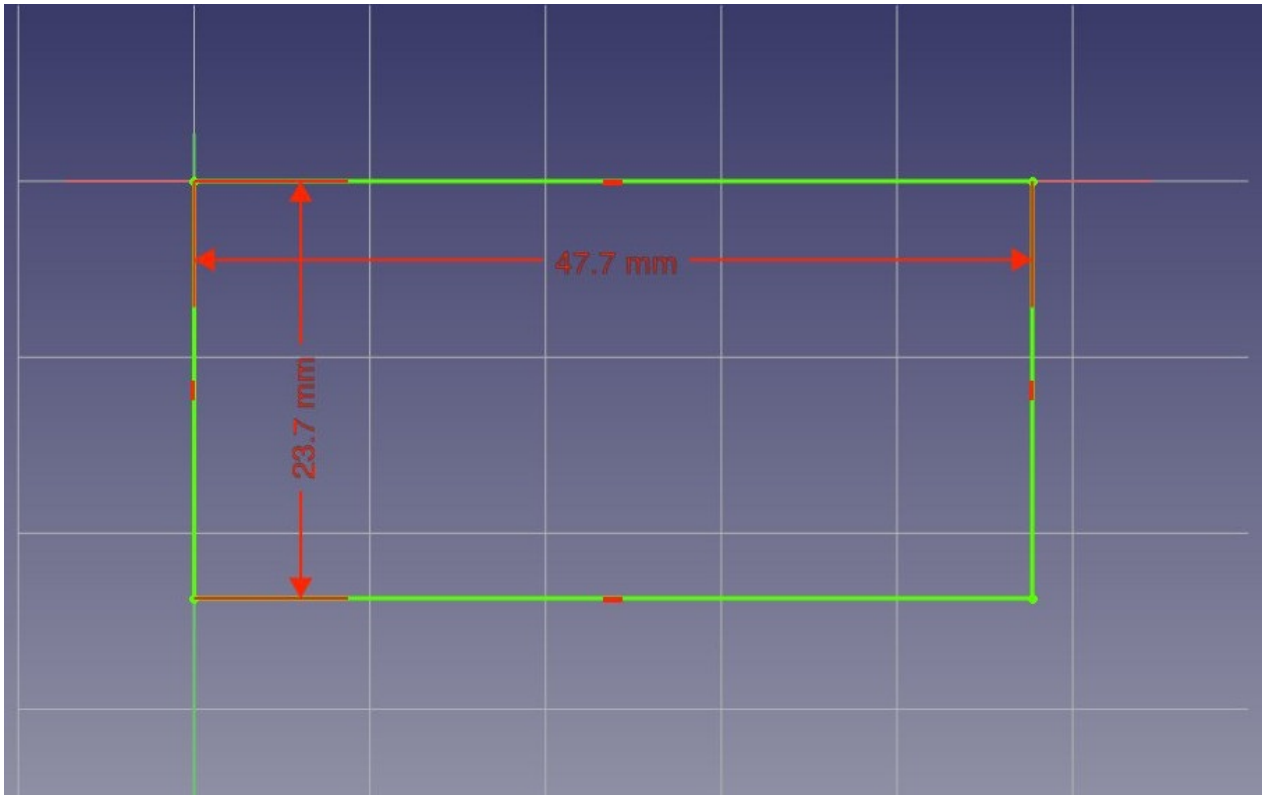
Part Design objects are fully based on **Sketches**. A Sketch is a 2D object, made of linear segments (lines, arcs of circle or ellipses) and constraints. These constraints can be applied either on linear segments or on their endpoints or center points, and will force the geometry to adopt certain rules. For example, you can place a vertical constraint on a line segment to force it to stay vertical, or a position (lock) constraint on an endpoint to prohibit it to move. When a sketch has an exact amount of constraints that prohibits any point of the sketch to be moved anymore, we talk about a fully constrained sketch. when there are redundant constraints, that could be removed without allowing the geometry to be moved, it is called over-constrained. This should be avoided, and FreeCAD will notify you if such case occurs.

Sketches have an edit mode, where their geometry and constraints can be changed. When you are done with editing, and leave edit mode, sketches behaves 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.

- Let's start by modeling a cubic shape that will be the base of our Lego brick. Later on we will carve the insides, and add the 8 dots on top of it. So let's start this by making a rectangular sketch that we will then extrude:
- Switch to the Part Design Workbench
- Click on the New Sketch button. A dialog will appear asking where you want to lie the sketch, 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.
- Now we can draw a rectangle, by selecting the Rectangle tool and clicking 2 corner points. You can place the two points anywhere, since their correct location will be set in the next step.
- 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 moving the rectangle around by dragging its lines with the mouse, all the geometry will keep obeying the constraints.

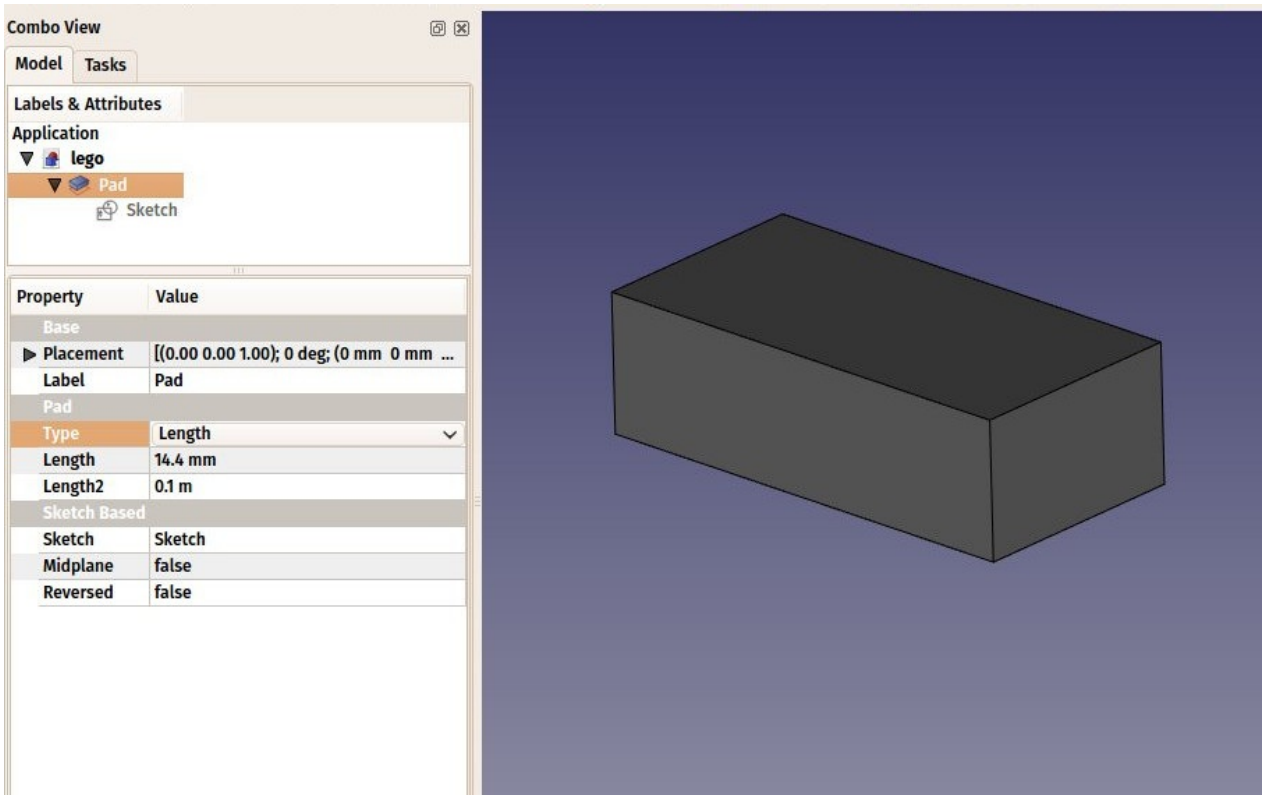


- Now, let's add three more constraints:
  - Select one of the vertical segments and add a Vectical Distance Constraint. Give it a size of 23.7mm.
  - Select one of the horizontal segments and add a Horizontal Distance Constraint. Make it 47.7mm.
  - Finally, select one of the corner points, then the origin point (which is the dot at the crossing of the red and green axes), then add a Point-on-Point Constraint. The rectangle will then jump to the origin point, and your sketch will turn green, meaning it is now fully constrained. You can try moving its lines or points, nothing will move anymore.



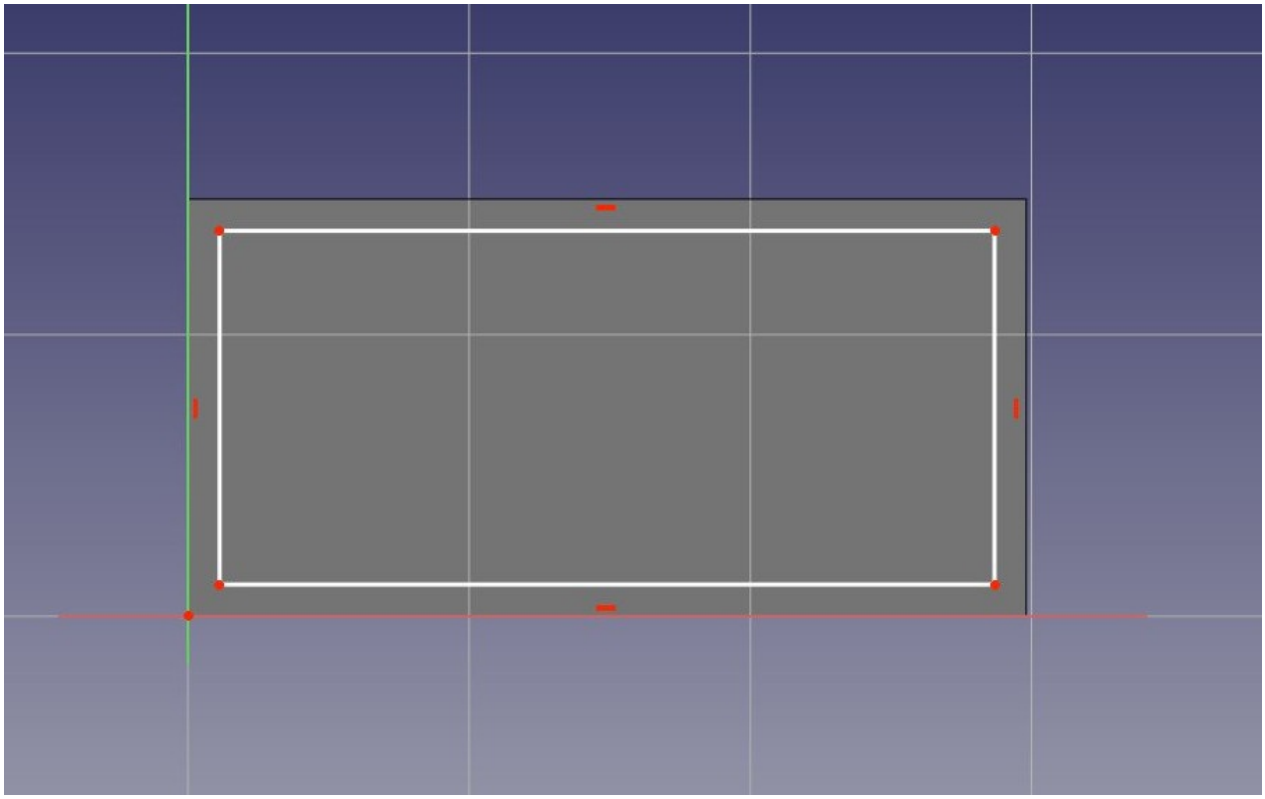
Note that the last point-on-point constraint was not absolutely necessary. You are never forced to work with fully constrained sketches. However, if we are going to print this block in 3D, it will be necessary to maintain our piece close to the origin point (which will be the center of the space where the printer head can move). By adding that constraint we are making sure that our piece will always stay "anchored" to that origin point.

- 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.
- Let's extrude it by using the Pad tool, and giving it a distance of 14.4mm. The other options can be left at their default values:

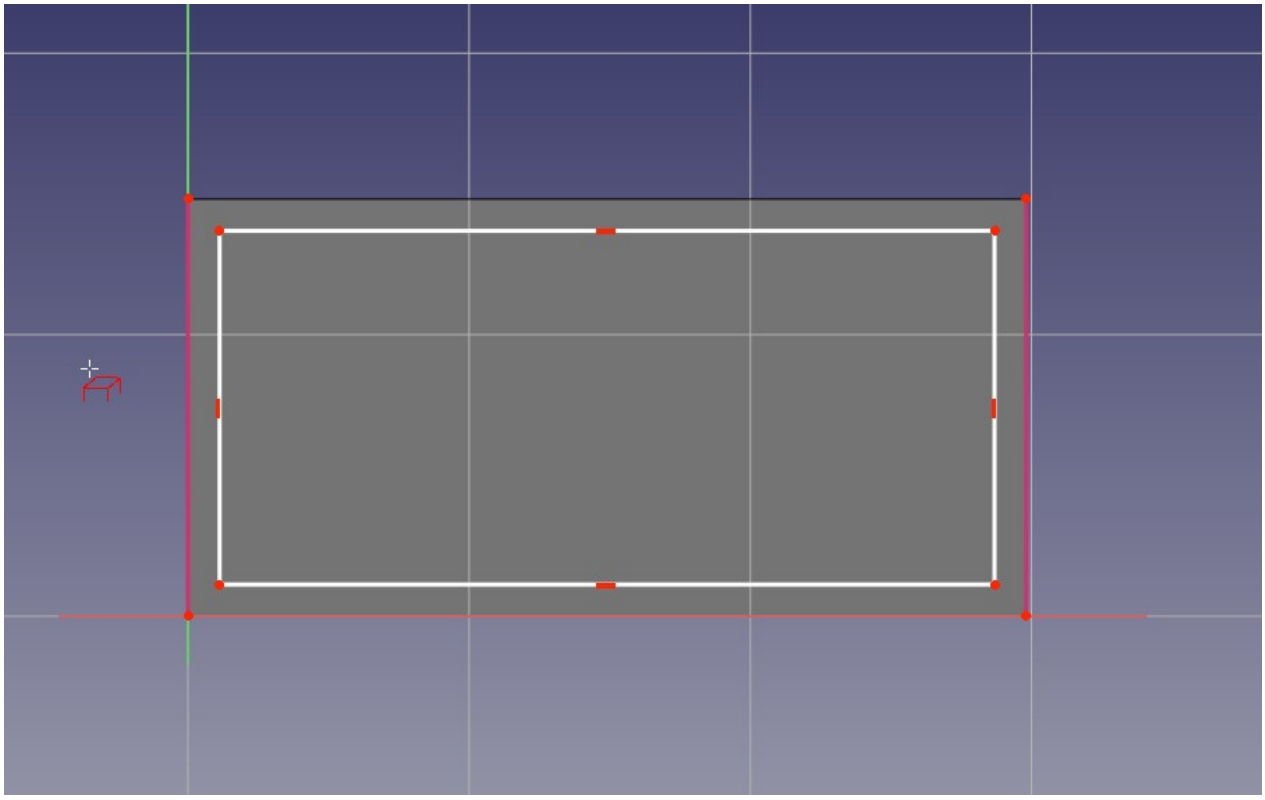


The **Pad** behaves very much like the Part Extrude tool that we used in the previous chapter. There are a couple of differences, though, the main one being that a pad cannot be moved. It is attached forever to its sketch. If you want to change the position of the pad, you must move the base sketch. In the current context, where we want to be sure nothing will move out of position, this is an additional security.

- 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.



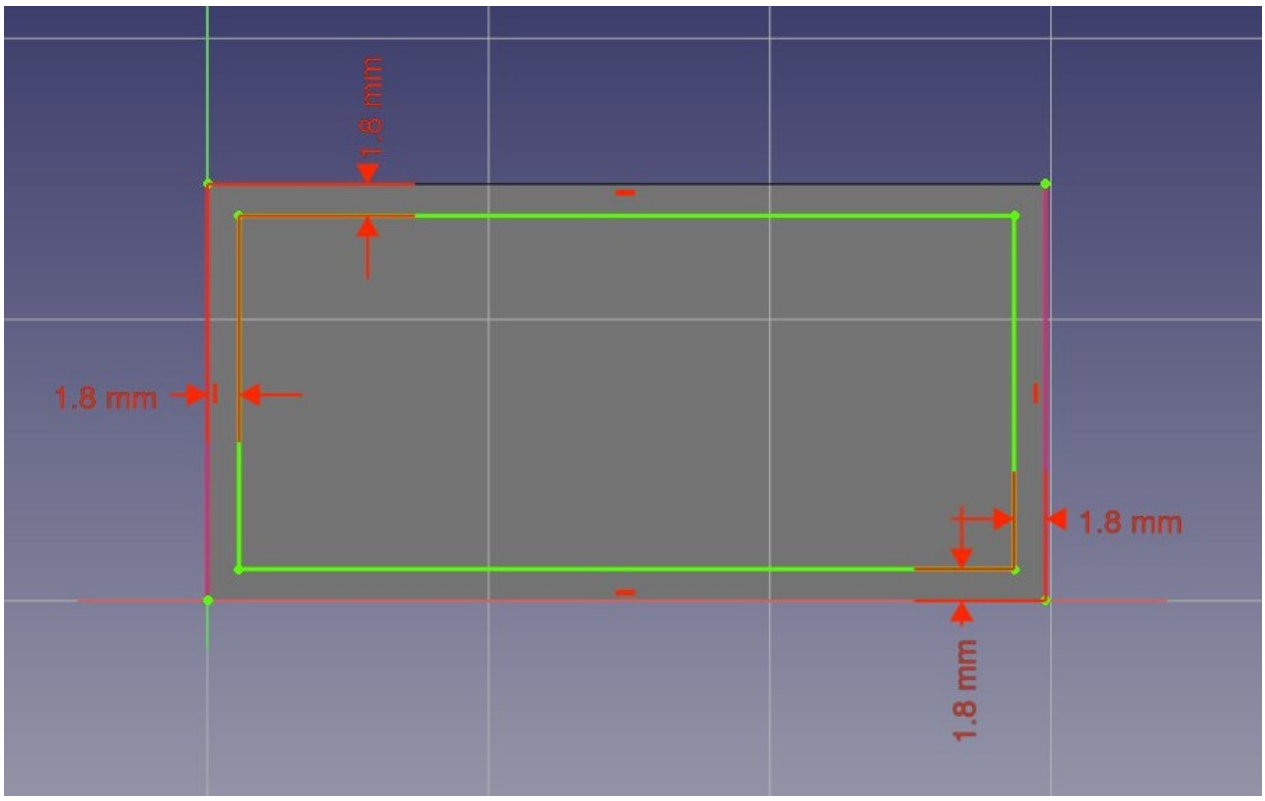
- We will now constrain the rectangle in relation to the bottom face. To do this, we need to "import" some edges of the face with the External geometry tool. Use this tool on the two vertical lines of the bottom face:



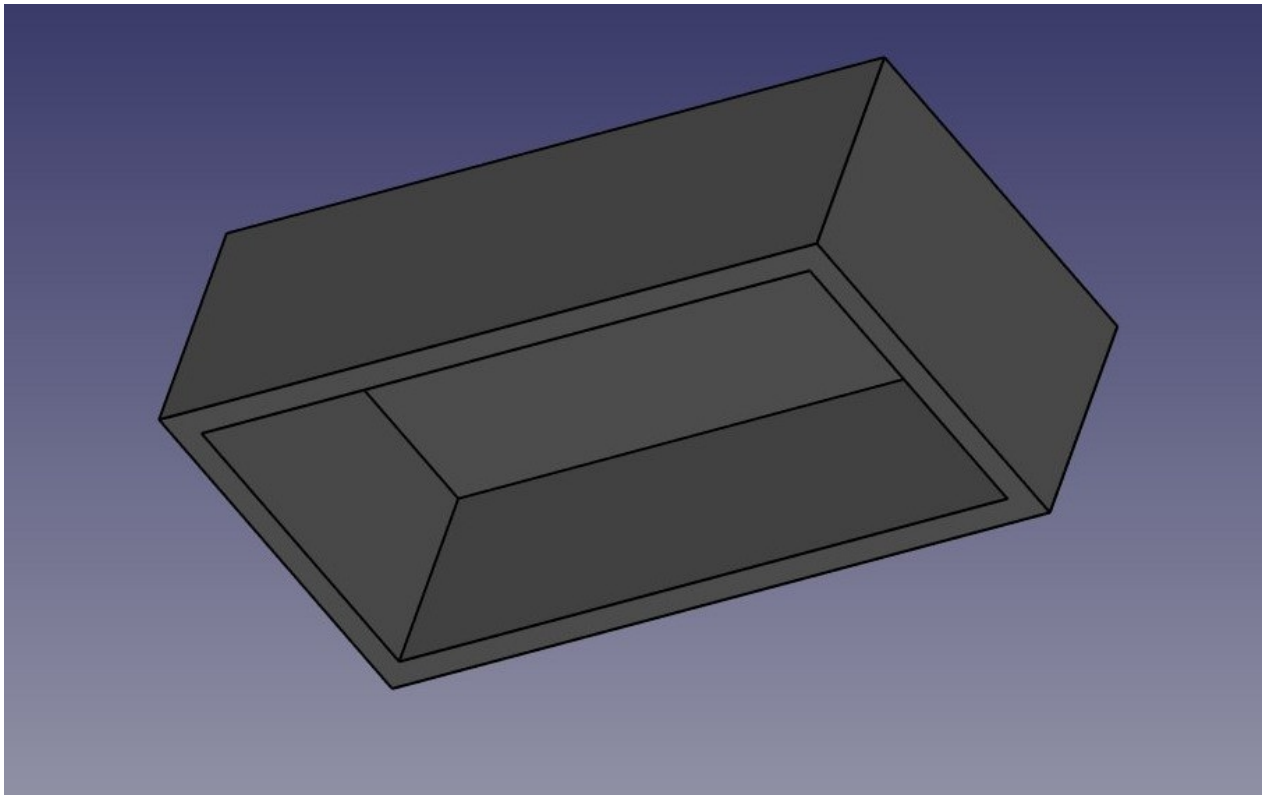
You will notice that only edges from the the base face can be added by this tool. When you create a sketch with a face selected, a relation is created between that face and the sketch, which is important for further operations. You can always remap a sketch to another face

later with the Map Sketch tool.

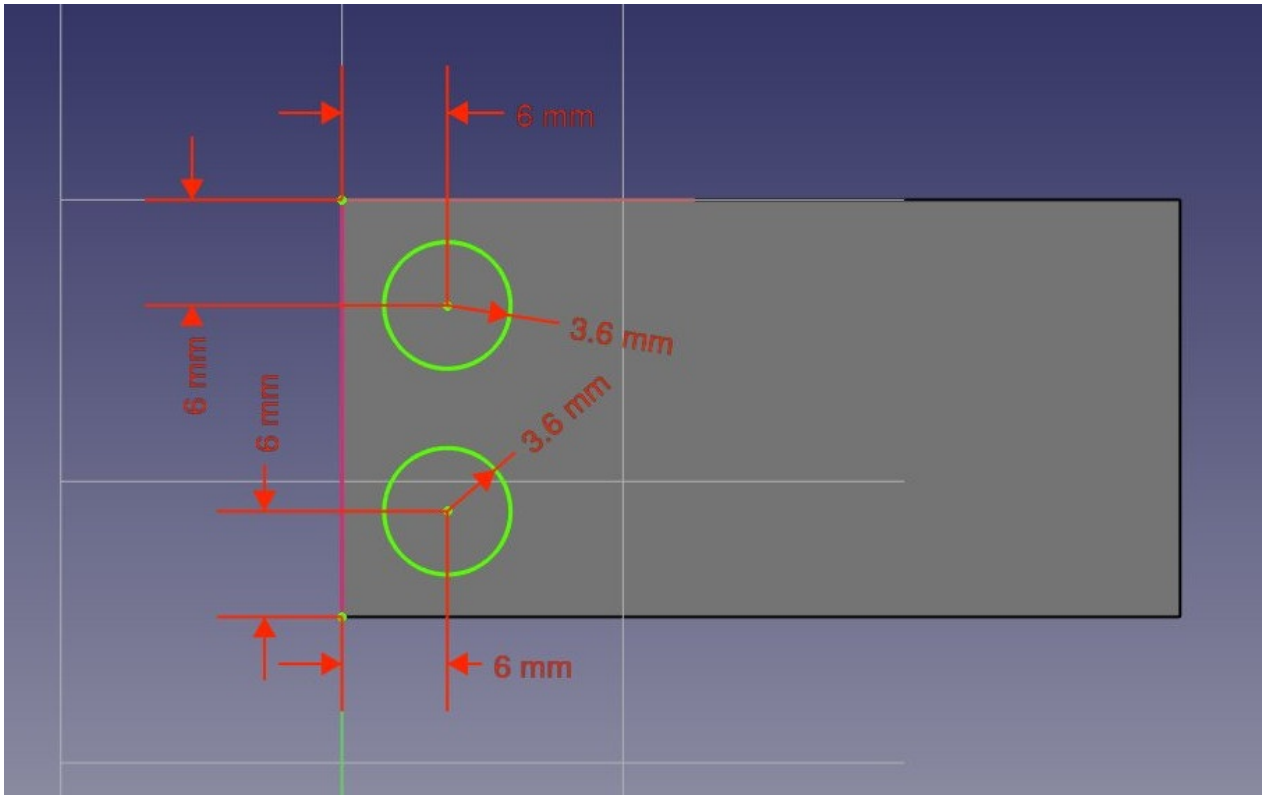
- The external geometry is not "real", it will be hidden when we leave edit mode. But we can use it to place constraints. Place the 4 following constraints:
  - Select the two upper left points of the rectangle and the left imported line and add a Horizontal Distance Constraint of 1.8mm
  - Select again the two upper left points of the rectangle and the left imported line and add a Vectical Distance Constraint of 1.8mm
  - Select the two lower right points of the rectangle and the right imported line and add a Horizontal Distance Constraint of 1.8mm
  - Select again the two lower right points of the rectangle and the right imported line and add a Vectical Distance Constraint of 1.8mm



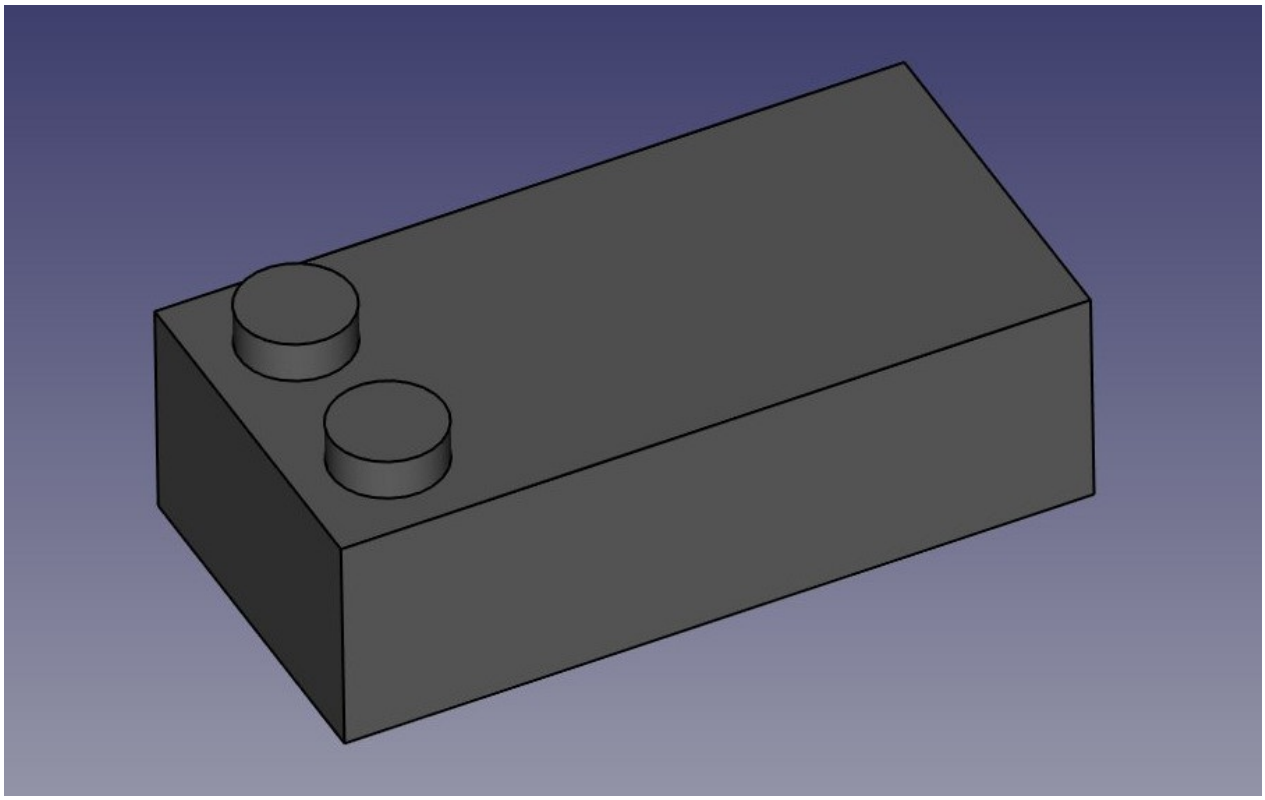
- Leave edit mode and we can now perform the pocket operation: With the sketch selected, press the Pocket buttton. Give it a length of 12.6mm, which will leave the upper face of our pad with a thickness of 1.8mm (remember, the total height of our pad was 14.4mm).



- We will now attack the 8 dots on the top face. To do this, since they are a repetition of a same feature, we will use the handy Linear Pattern tool of the Part Design Workbench, which allows to model once and repeat the shape.
- Start by selecting the top face of our block
- Create a New Sketch.
- Create two circles.
- Add a Radius Constraint of 3.6mm to each of them
- Import the left edge of the base face with the External geometry tool.
- Place two vertical constraints and two horizontal constraints of 6mm between the center point of each circle and the corner points of the imported edge, so each circle has its center at 6mm from the border of the face:



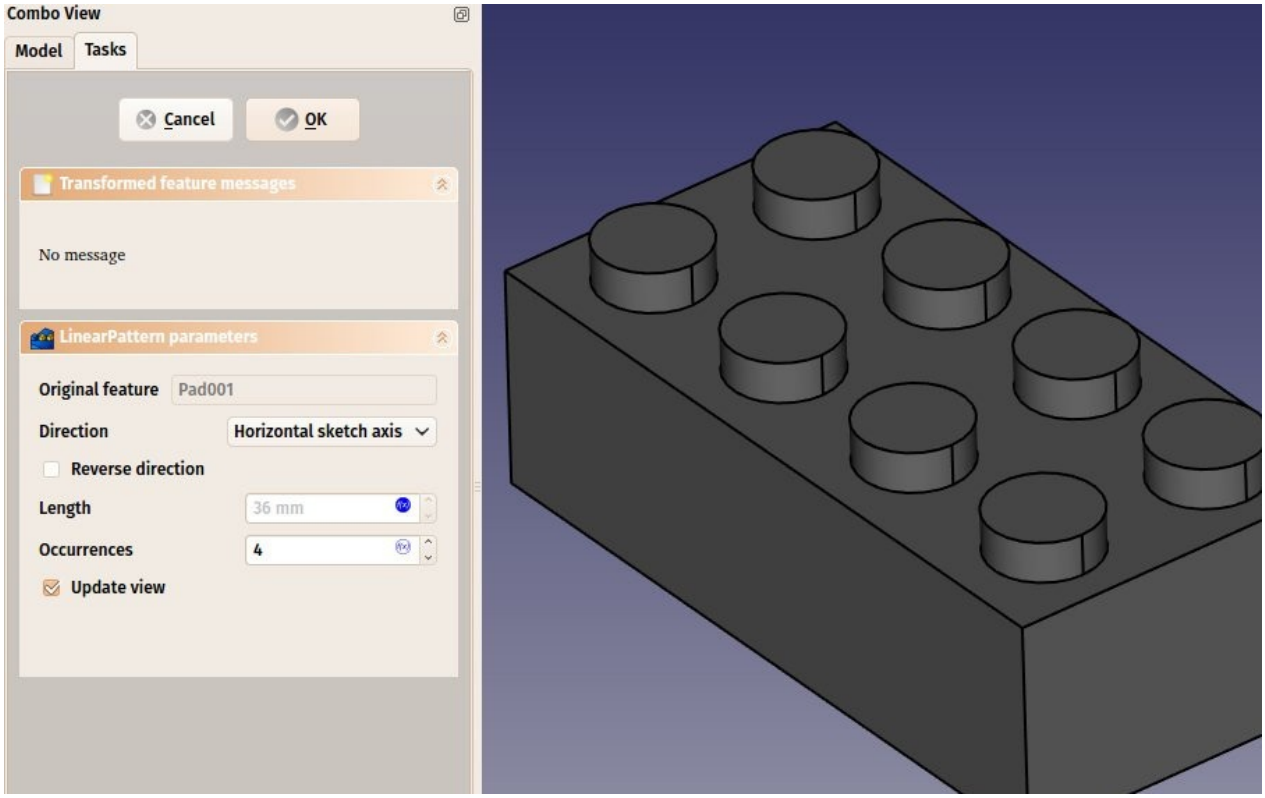
- Notice how, once again, when you lock the position and dimension of everything in your sketch, it becomes fully constrained. This always keeps you on the safe side. You could change the first sketch now, everything we did afterwards would keep tight.
- Leave edit mode, select this new sketch, and create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



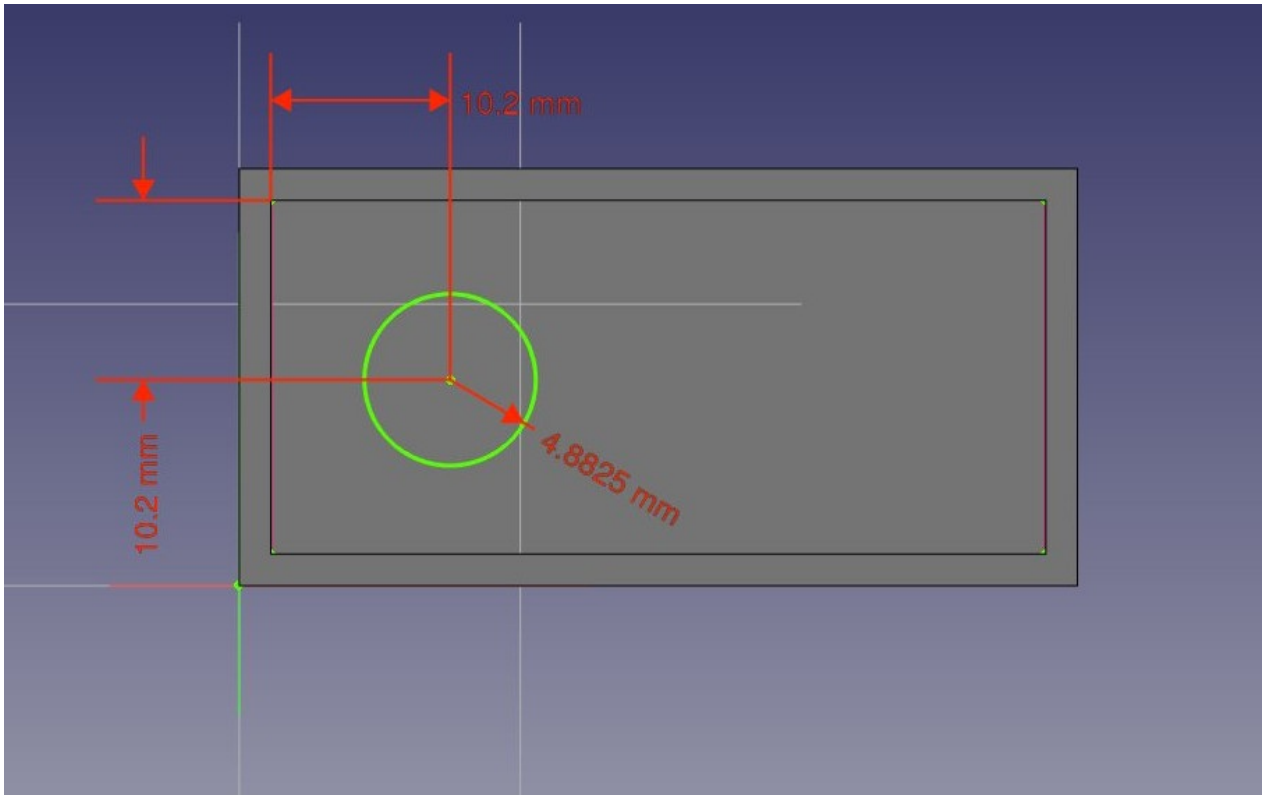
- Notice that, as earlier with the pocket, since we used the top face of our base block as a base for this latest sketch, any PartDesign operation we do with this sketch will correctly

be built on top of the base shape: The two dots are not independent objects, they have been extruded directly from our brick. This is the great advantage of working with the Part Design Workbench, as long as you take care of always building one step on top of the previous one, you are actually building one final solid object.

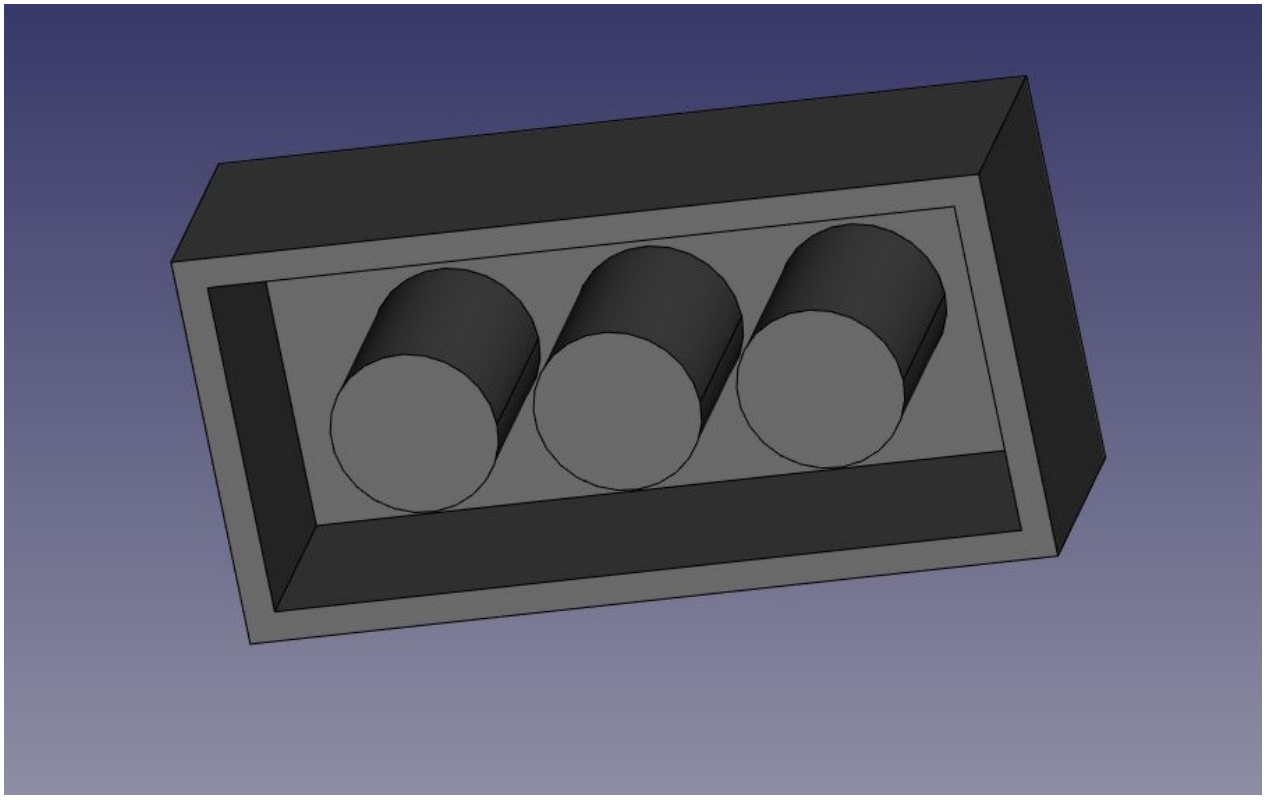
- We can now duplicate our two dots four times, so we get eight. Select the latest Pad we just created.
- Press the Linear Pattern button.
- Give it a length of 36mm (which is the total "span" we want our copies to fit in), in the "horizontal sketch axis" direction, and make it 4 occurences:



- Once again, see that this is not just a duplication of an object, it is a *feature* of our shape that has been duplicated, the final object is still only one solid object.
- Now let's work on the three "tubes" that fill the void we created on the bottom face. We have several possibilities: create a sketch with three circles, pad it then pocket it three times, or create a base sketch with one circle inside the other and pad it to form the complete tube already, or even other combinations. Like always in FreeCAD, there are many ways to do achieve a same result. Sometimes one way will not work the way we want, and we must try other ways. Here, we will take the safest approach, and do things one step at a time.
- Select the face that is at the bottom of the hollow space we carved earlier inside the block.
- Create a new sketch, add a circle with a radius of 4.8825mm, import the left border of the face, and constrain it vertically and horizontally at 10.2mm from the upper corner of the face:

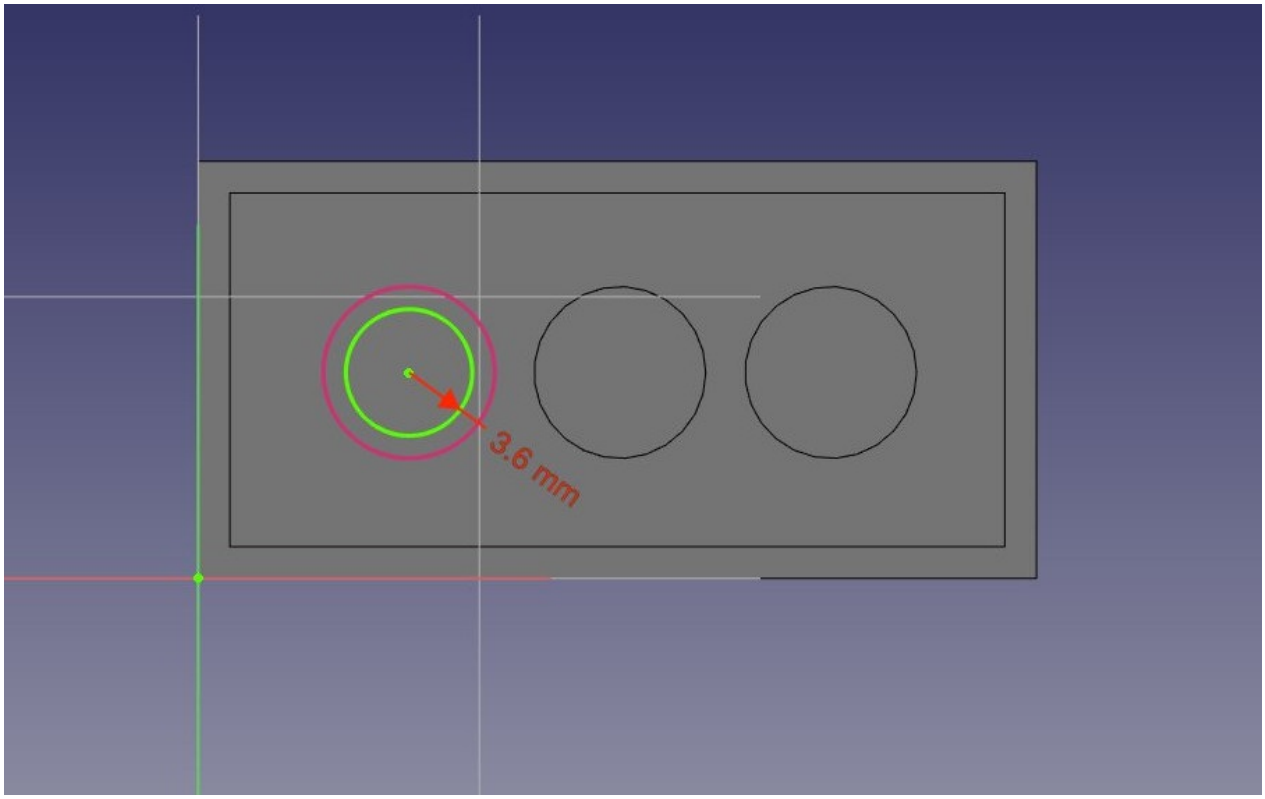


- Leave edit mode, and pad this sketch with a distance of 12.6mm
- Create a linear pattern from this last pad, give it a length of 24mm and 3 occurences. We now have three filled tubes filling the hollow space:

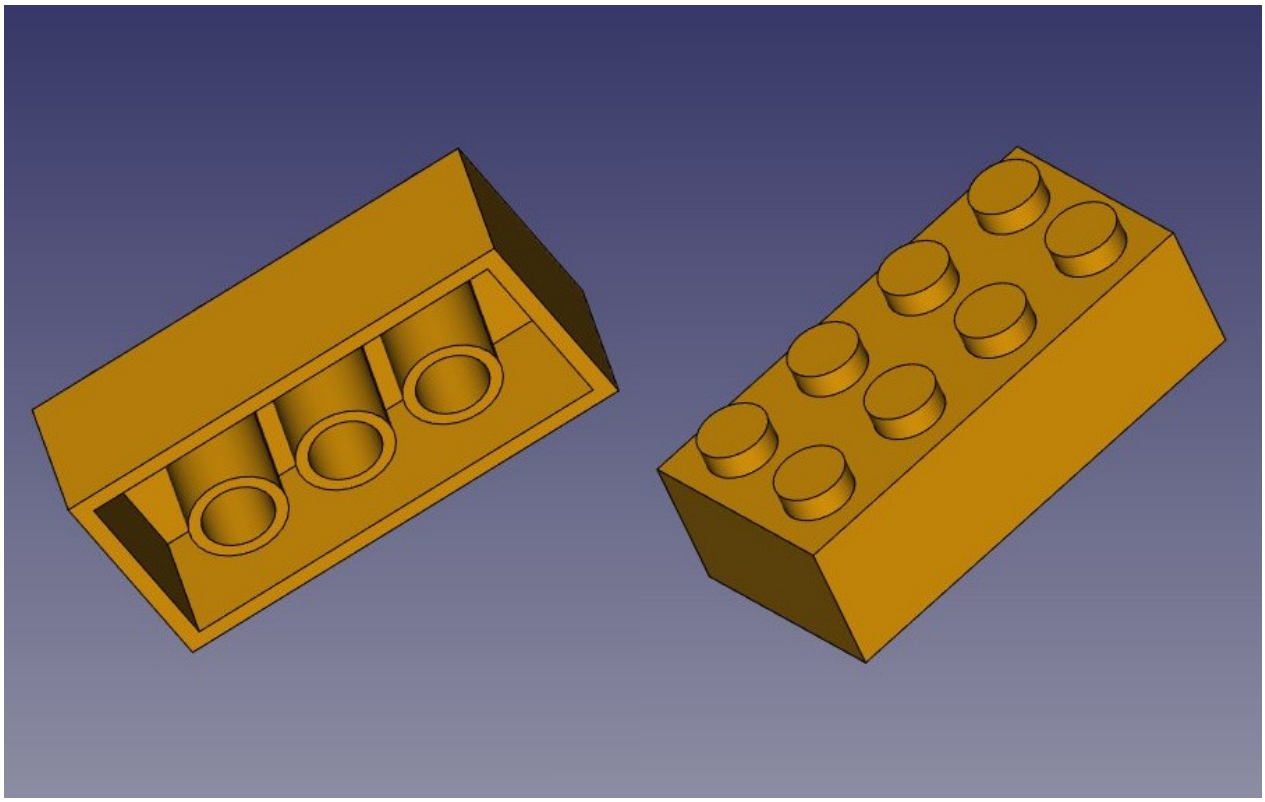


- Now let's make the final holes. Select the circular face of the first of our three "pins"
- Create a new sketch, import the circular boder of our face, create a circle with a radius constraint of 3.6mm, and add a Point-on-Point Constraint between the center of the

imported circle and our new circle. We now have a perfectly centered circle,and once again fully constrained:



- Leave edit mode, and create a pocket from this sketch, with a length of 12.6mm
- Create a linear pattern from this pocket, with a length of 24mm and 3 occurences. That's the last step, our piece of lego is now complete, we can give it a nice color of Victory!



You will notice that our modeling history (what appears in the tree view) has become quite long. This is of precious because every single step of what we did can be changed later on. Adapting this model for another kind of brick, for example one with 2x2 dots, instead of 2x4, would be a piece of cake, we would just need to change a couple of dimensions and the number of occurences in linear patterns. We could as easily create bigger pieces that don't exist in the original Lego game.

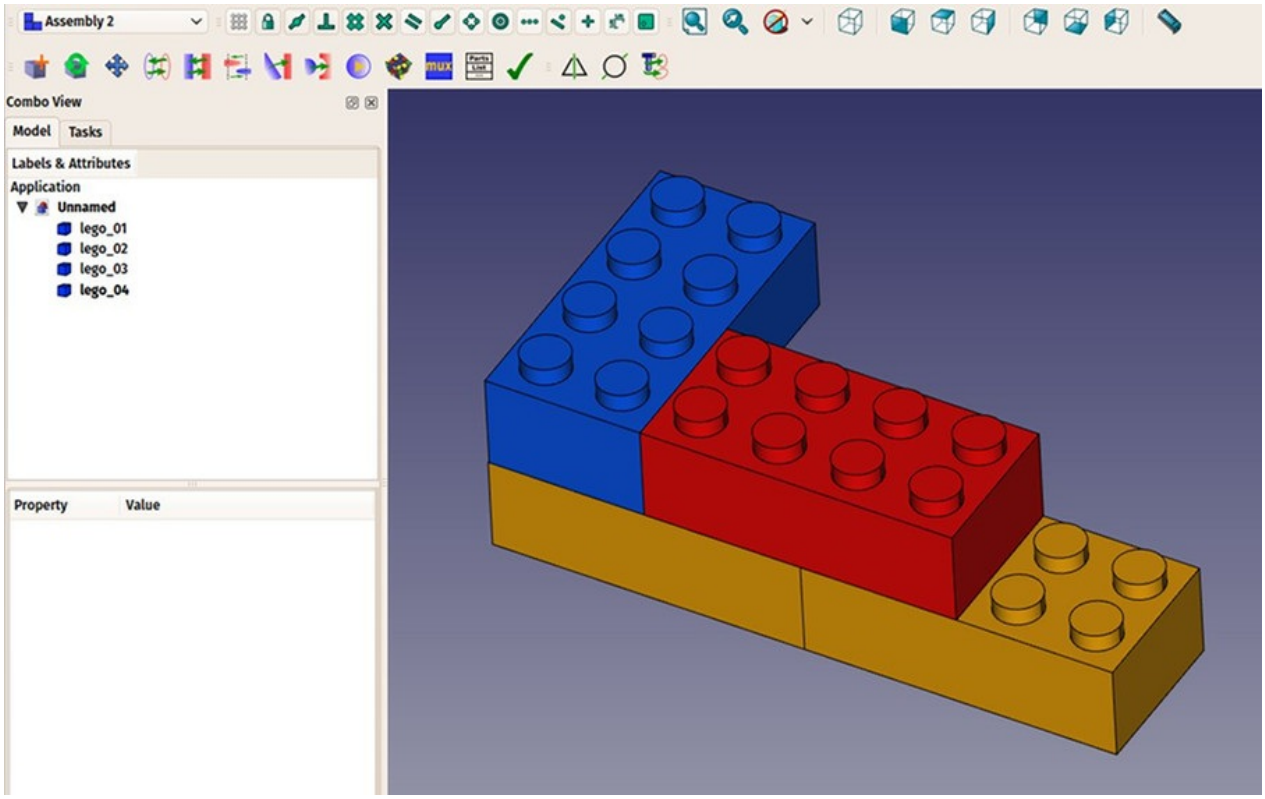
But we could also need to get rid of the history, for example if we are going to model a castle with this brick, and we don't want to have this whole history repeated 500 times in our file.

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 reimporting it.

Assembling

But the best of both worlds also exists, which is the Assembly2 Workbench, an addon that can be installed from the FreeCAD-addons repository. This Workbench is named "2" because there is also an official built-in Assembly Workbench in development, which is not ready yet. The Assembly2 Workbench, however, already works very well to construct assemblies, and also features a couple of object-to-object constraints which you can use to constrain the position of one object in relation to another. In the example below, however, it will be quicker and easier to position the pieces using Draft Move and Draft Rotate than using the Assembly2 constraints.

- Save the file we did until now
- Install the Assembly2 Workbench and restart FreeCAD
- Create a new empty document
- Switch to the Assembly2 workbench
- Press the **Import a part from another FreeCAD document** button
- Select the file we saved above
- The final piece will be imported in the current document. The Assembly2 workbench will determine automatically what is the final piece in our file that needs to be used, and the new object stays linked to the file. If we go back and modify the contents of the first file, we can press the **Update parts imported into the assembly** button to update the pieces here.
- By using the **Import a part from another FreeCAD document** button several times, and moving and rotating the pieces (with the Draft tools or by manipulating their Placement property), we can quickly create a small assembly:



Downloads

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

Read more

- The Sketcher: http://www.freecadweb.org/wiki/index.php?title=Sketcher_Module
- The Part Design Workbench: http://www.freecadweb.org/wiki/index.php?title=PartDesign_Workbench
- The Assembly2 Workbench: https://github.com/hamish2014/FreeCAD_assembly2

Preparing models for 3D printing

One of the main uses of FreeCAD is to produce real-world objects. These can be designed in FreeCAD, and then made real by different ways, such as communicated to other people who will then build them, or, more and more fequently, sent directly to a 3D printer or a CNC mill. This chapter will show you how to get your models ready to send to these machines.

If you have been cautious while modeling, most of the difficulty you might encounter when printing your model in 3D has already been avoided. This involves basically:

- Making sure that your 3D objects are **solid**. Real-world objects are solid, the 3D model must be solid too. We saw in earlier chapters that FreeCAD helps you a lot in that regard, and that the Part Design Workbench will notify you if you do an operation that prevents your model to stay solid. The Part Workbench also contains a Check Geometry tool that is handy to check further for possible defects.
- Making sure about the **dimensions** of your objects. One millimeter will be one millimeter in real-life. Every dimension matters.
- Controlling the **degradation**. No 3D printing or CNC milling system can take FreeCAD files directly. Most of them will only understand a machine language called G-Code. G-code has dozens of different dialects, each machine or vendor usually using its own. The conversiom of your models into G-Code can be easy and automatic, but you can also do it manually, with total control over the output. In any case, some loss of quality of your model will unavoidably occur during the process. When printing in 3D, you must always make sure this loss of quality stays below your minimal requirements.

Below, we will assume that the first two criterias are met, and that by now you are able to produce solid objects with correct dimensions. We will now see how to address the third point.

Exporting to slicers

This is the technique most commonly used for 3D printing. The 3D object is exported to another program (the slicer) which will generate the G-code from the object, by slicing it into thin layers (hence the name), which will reproduce the movements that the 3D printer will do. Since many of those printers are home-built, there are often small differences from one to the other. These programs usually offer advanced configuration possibilities that allow to tailor the output exactly for the particularities of your 3D printer.

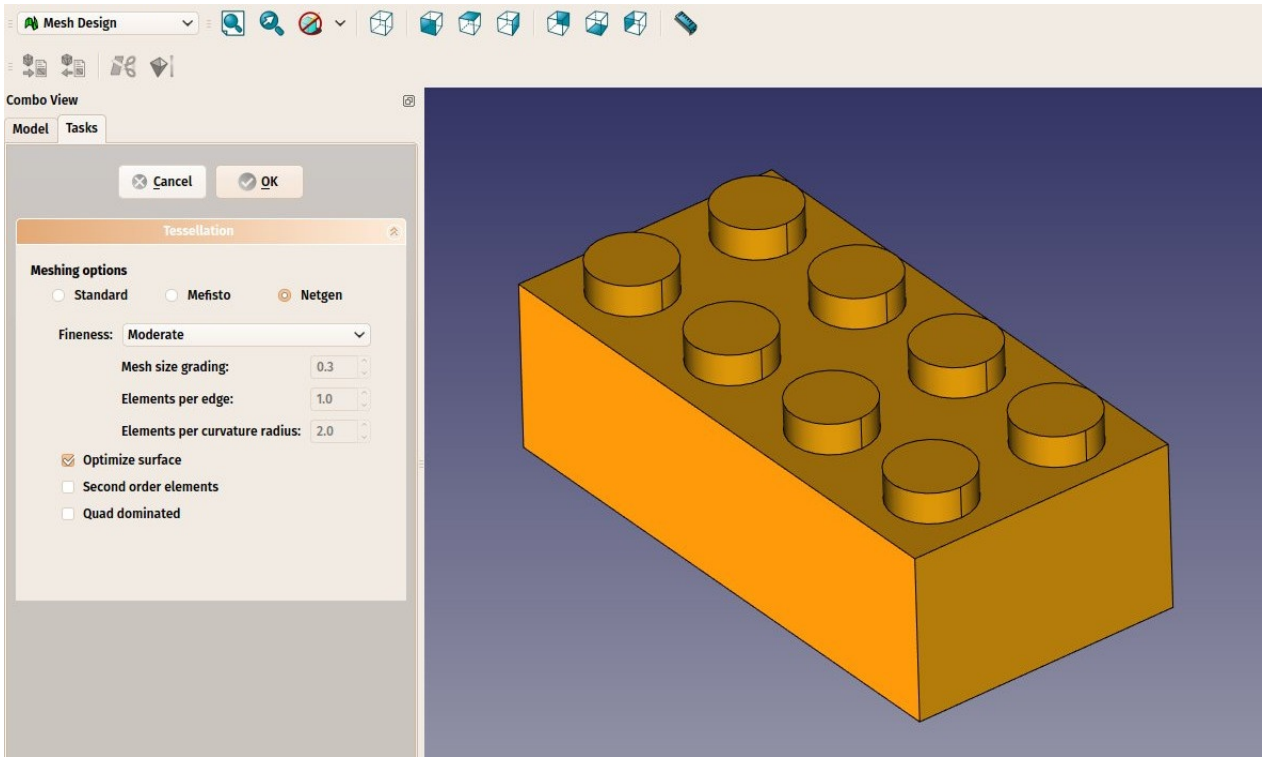
Actual 3D printing, however, is a too vast subject for this manual. But we will see how to export and use these slicers to check that the output is correct.

Converting objects to meshes

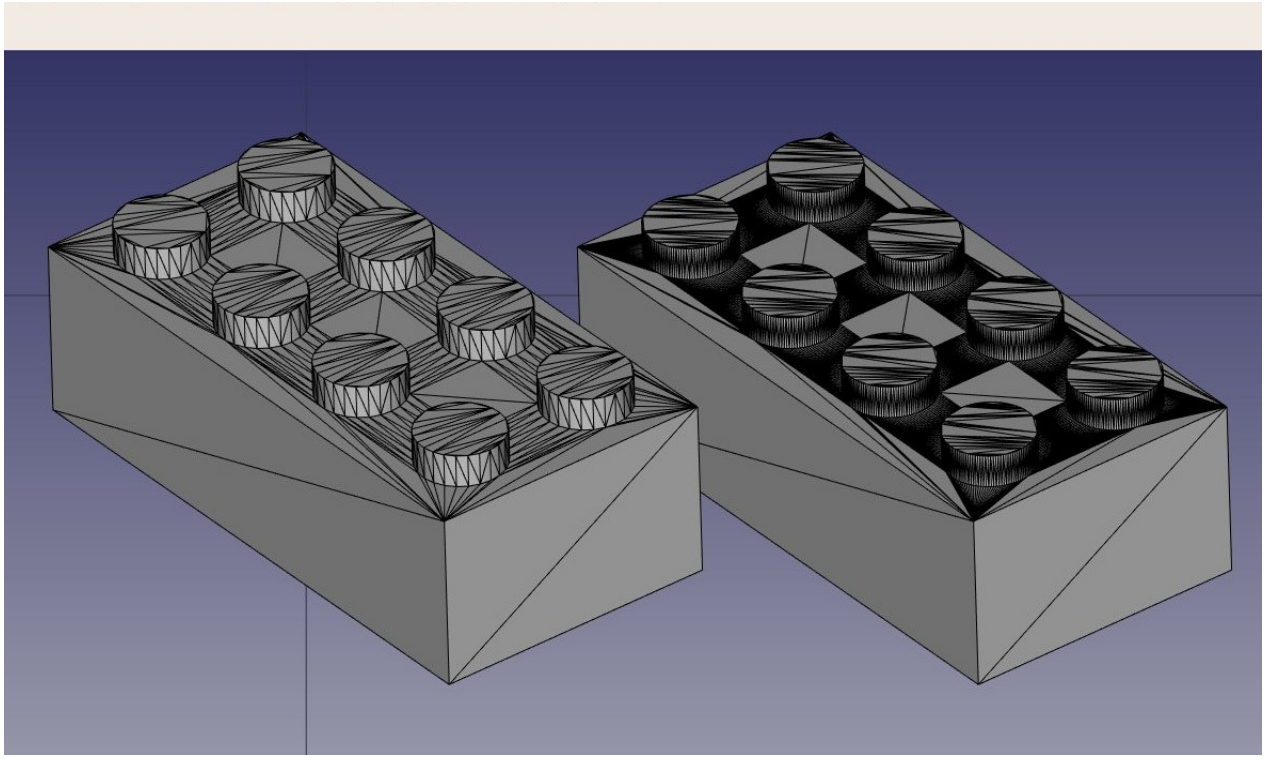
None of the slicers will, at this date, take directly solid geometry as we produce in FreeCAD. So we will need to convert any object we want to 3D print into a mesh first, that the slicer can open. Fortunately, as much as converting a mesh to a solid is a complicated operation, the contrary, converting a solid to a mesh, is very straightforward. All we need to be careful about, is that it is now that the degradation we mentioned above will occur. We need to check that the degradation stays inside acceptable limits.

All the mesh handling, in FreeCAD, is done by another specific workbench, the Mesh Workbench. This workbench contains, apart from the most important, the tools that convert between Part and Mesh objects, several utilities meant to analyze and repair meshes. Although working with meshes is not the focus of FreeCAD, when working with 3D modeling, you often need to deal with mesh objects, since their use is very widespread among other applications. This workbench allows you to handle them fully in FreeCAD.

- Let's convert one of the objects we modelled in the previous chapters, such as the lego piece (which can be downloaded from the end of the previous 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 less 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 apart, 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 occured.
- 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.

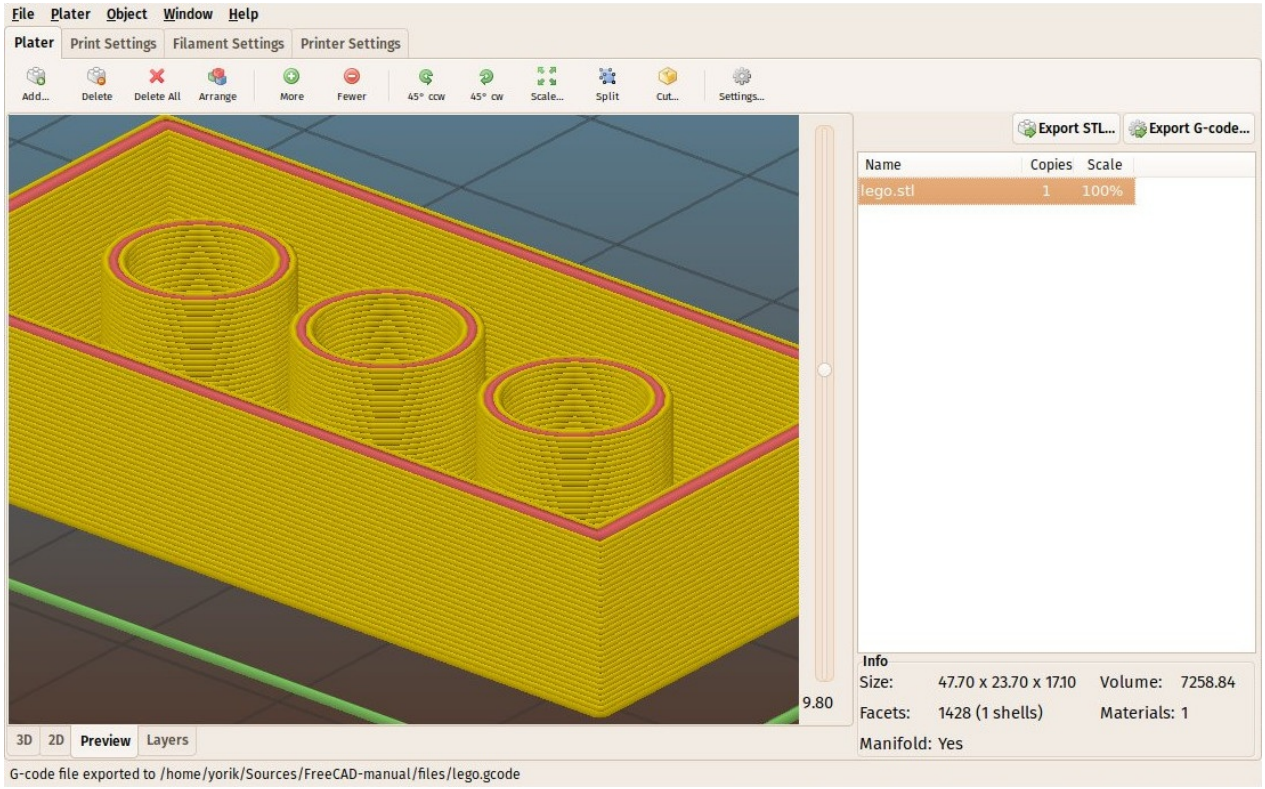
- We can now export our mesh to a mesh format, such as STL, which is currently the most widely used format in 3D printing, by using menu **File -> Export** and choosing the STL file format.

If you don't own a 3D printer, it is usually very easy to find commercial services that will print and send you the printed objects by mail. Among the famous ones are Shapeways and Sculpteo, but you will also usually find many others in your own city. In all major cities, you will also nowadays find Fab labs, which are workshops equipped with a range of 3D manufacturing machines, almost always including at least one 3D printer. Fab labs are usually community spaces, that will let you use their machines, for a fee or for free depending on the Fab lab, but also teach you how to use them, and promote other activities around 3D manufacturing.

Using Slic3r

Slic3r is an application that converts STL objects into G-code that can be sent directly to 3D printers. Like FreeCAD, it is free, open-source and runs on Windows, Mac OS and Linux. Correctly configurating things for 3D printing is a complicated process, where you must have a good knowledge of your 3D printer, so it is not very useful to generate G-code before actually going to print (your G-code file might not work well on another printer), but it is useful for us anyway, to check that our STL file will be printable without problems.

This is our exported STL file opened in Slic3r. By using the **preview** tab, and moving the right slider, we can visualize the path that the 3D printer head will follow to construct our object.



Using the Cura addon

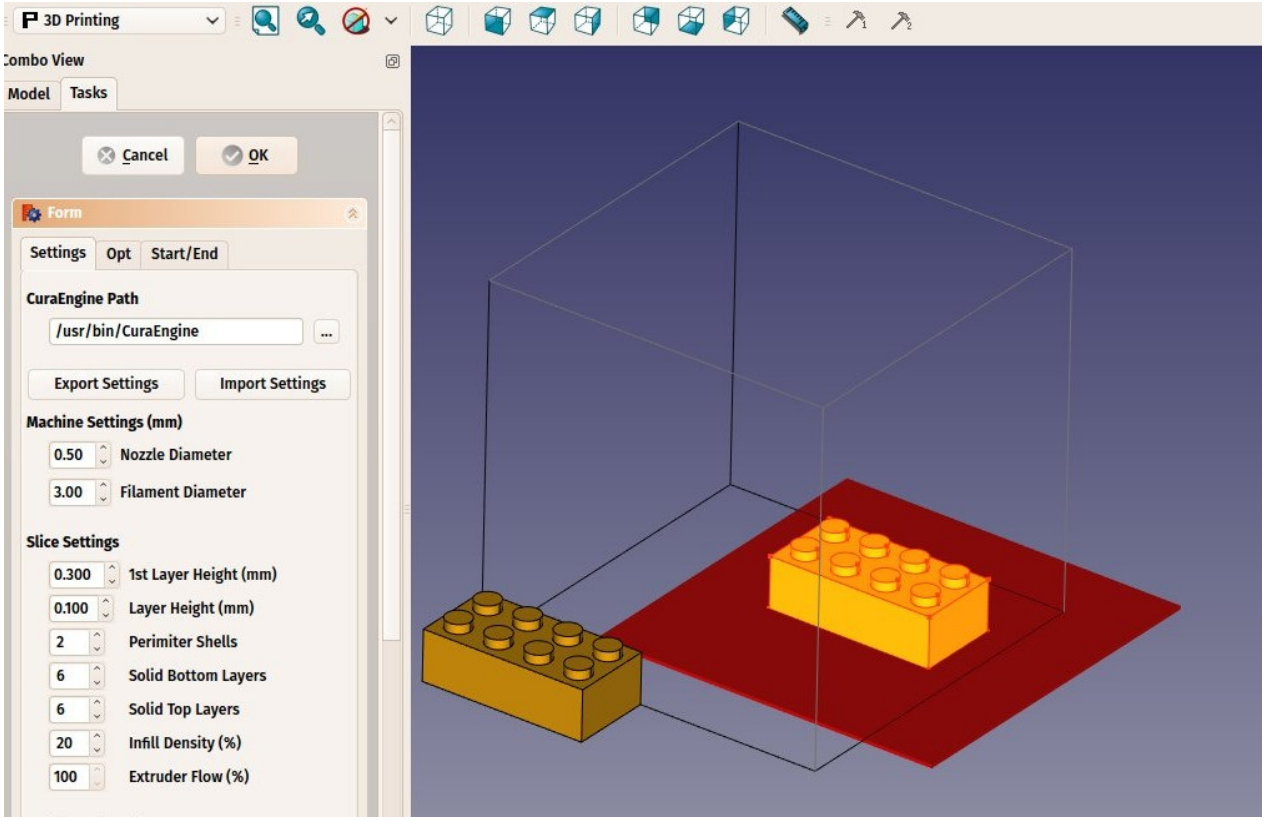
Cura is another free and open-source slicer application for Windows, Mac and Linux, maintained by the 3D printer maker Ultimaker. Some FreeCAD users have created a Cura Workbench that uses cura internally. The Cura Workbench is available from the FreeCAD addons repository. To use the Cura Workbench, you also need to install Cura itself, which is not included in the workbench.

Once you have installed both Cura and the Cura Workbench, you will be able to use it to produce the G-code file directly from Part objects, without the need to convert them to meshes, and without the need to open an external application. Producing another G-code file from our Lego brick, using the Cura Workbench this time, is done as follows:

- Load the file containing our Lego brick (it can be downloaded at the end of the previous chapter)
- Switch to the Cura Workbench
- Setup the printer space by choosing menu **3D printing -> Create a 3D printer definition**. Since we aren't going to print for real, we can leave the settings as they are. The geometry of the printing bed and available space will be shown in the 3D view.
- Move the Lego brick to a suitable location, such as the center of the printing bed.

Remember that PartDesign objects cannot be moved directly, so you need either to move its very first sketc (the first rectangle), or to move (and print) a copy, which can be made with the Part -> Create Simple Copy tool. The copy can be moved, for example with Draft -> Move.

- Select the object to be printed, and select menu **3D printing -> Slice with Cura Engine**.
- In the task panel that will open, make sure the path to the Cura executable is correctly set. Since we are not going to really print, we can leave all other options as they are. Press **Ok**. Two files will be generated in the same directory as your FreeCAD file, an STL file and a G-code file.



- The generated G-code can also be reimported into FreeCAD (using the slic3r preprocessor) for checking.

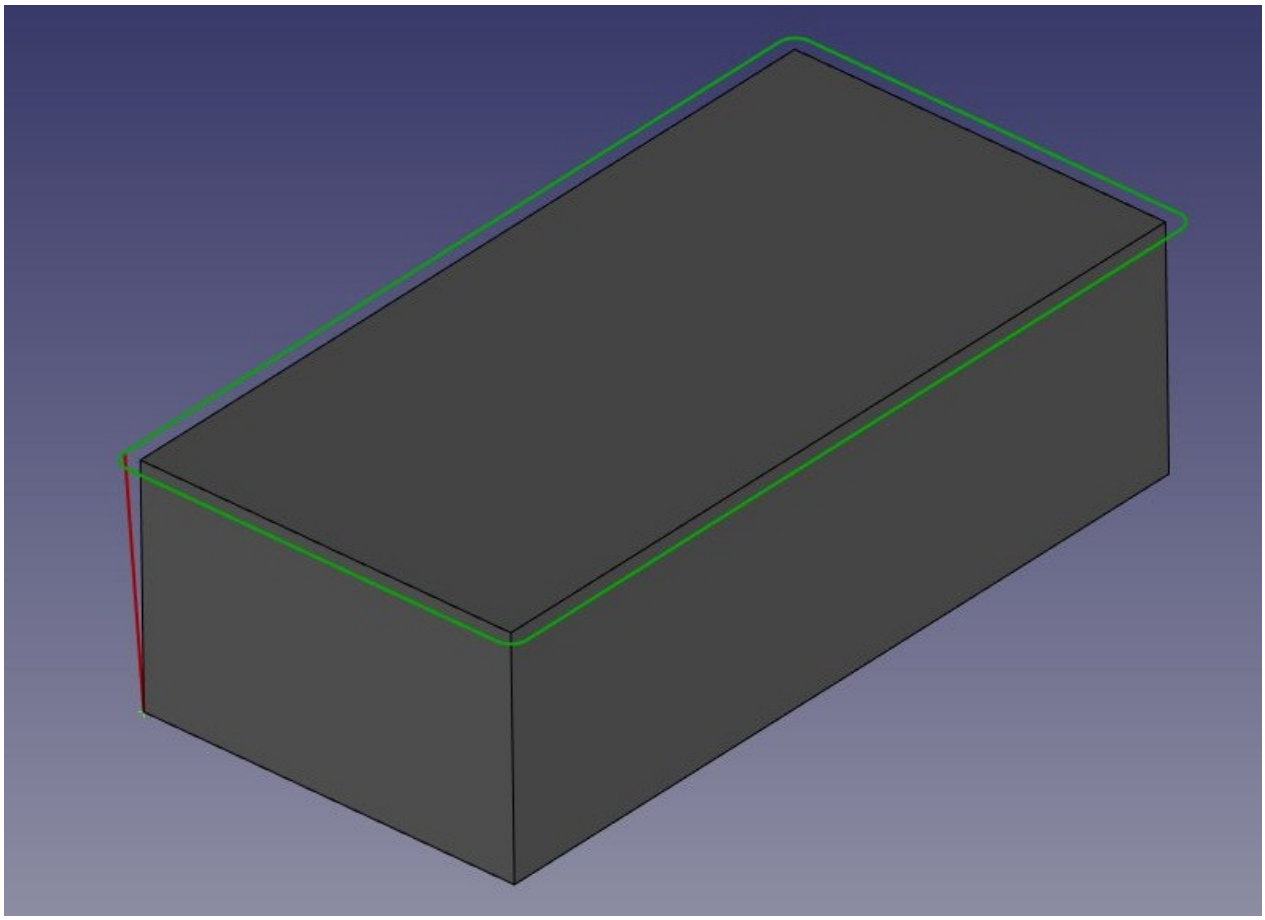
Generating G-code

FreeCAD also offers more advanced ways to generate G-code directly. This is often much more complicated than using automatic tools as we saw above, but has the advantage to let you fully control the output. This is usually not needed when using 3D printers, but becomes very important when dealing with CNC milling, as the machines are much more complex.

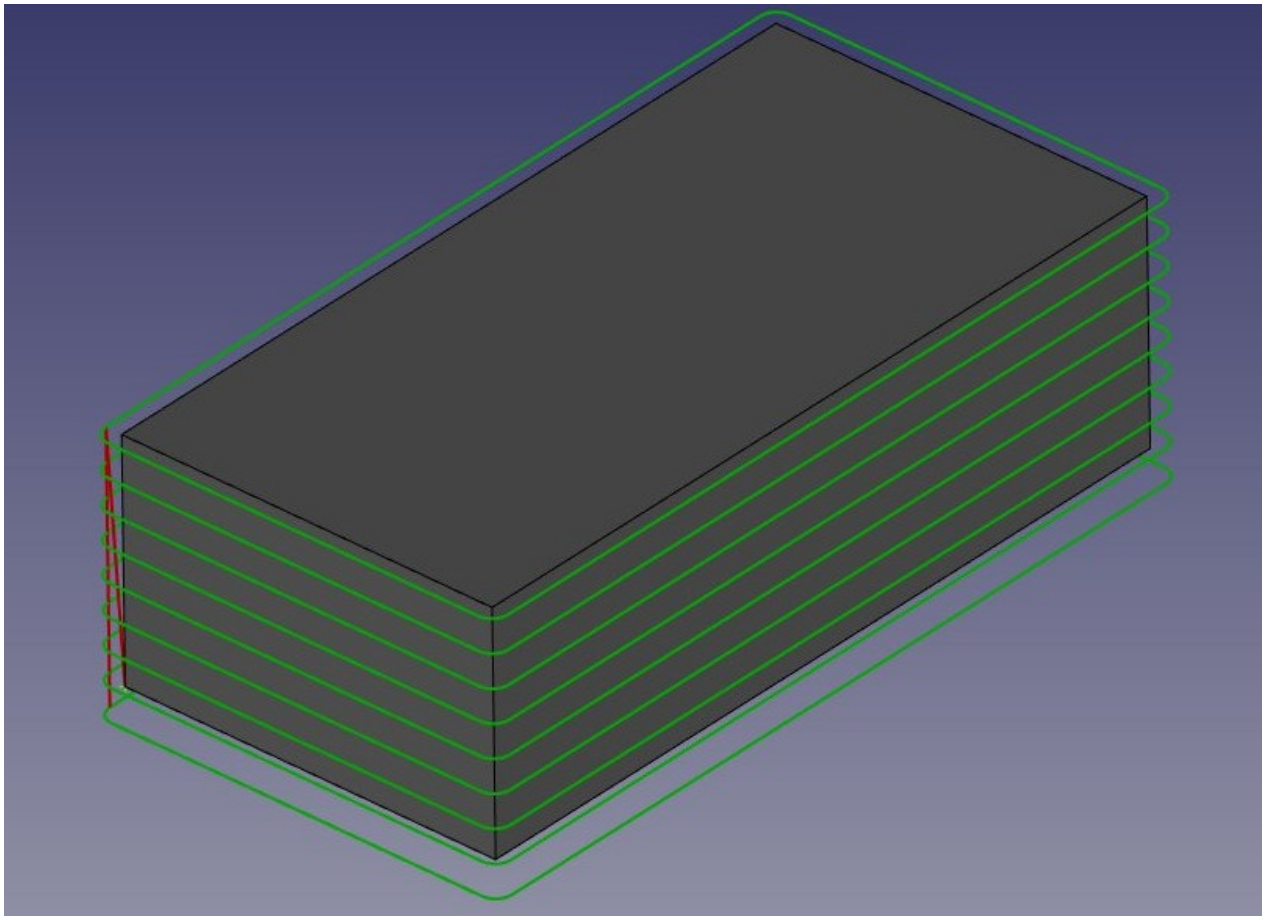
G-code path generation in FreeCAD is done with the Path Workbench. It features tools that generate full machine paths and others that generate only parts of a G-code project, that can be assembled to form a whole milling operation.

Generating CNC milling paths is another subject that is much too vast to fit in this manual, so we are going to show how to build a simple Path project, without caring much about most of the details of real CNC machining.

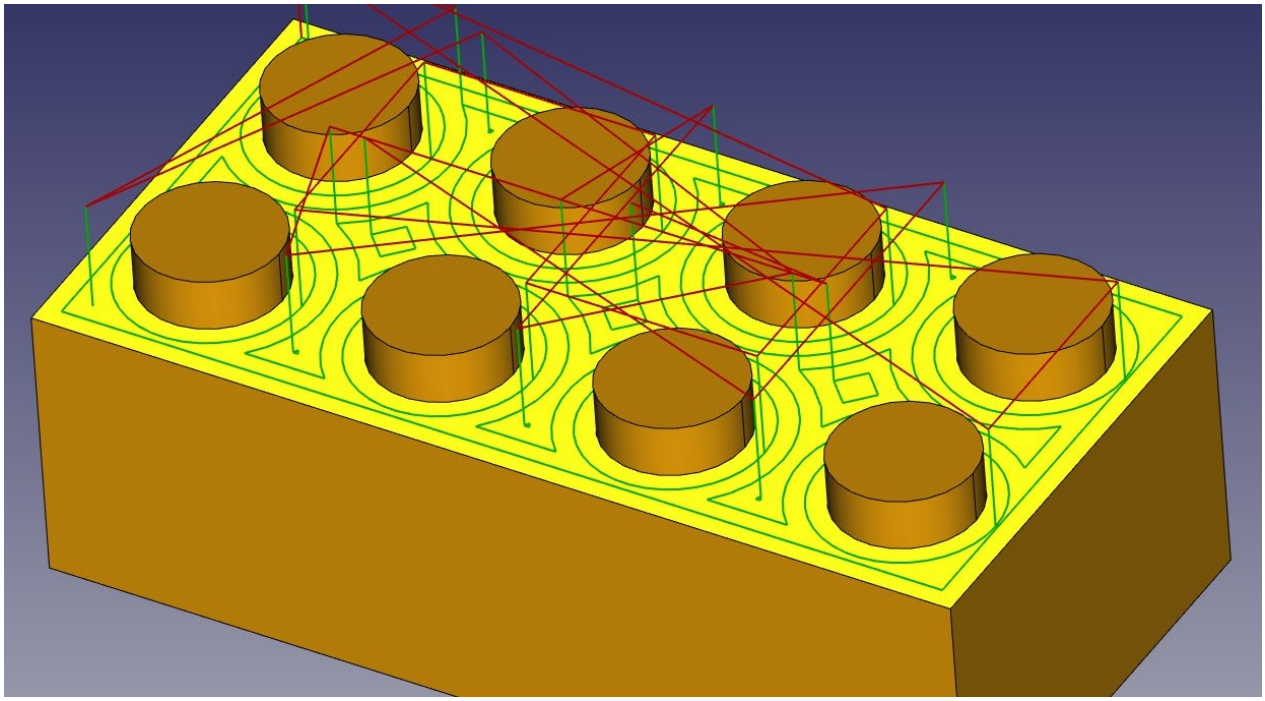
- Load the file containing our lego piece, and switch to the Path Workbench.
- Since the final piece doesn't contain anymore a rectangular top face, hide the final lego piece, and show the first cubic pad that we did, which has a rectangular top face.
- Select the top face and press the Face Profile button.
- Set its **Offset** property to 1mm.



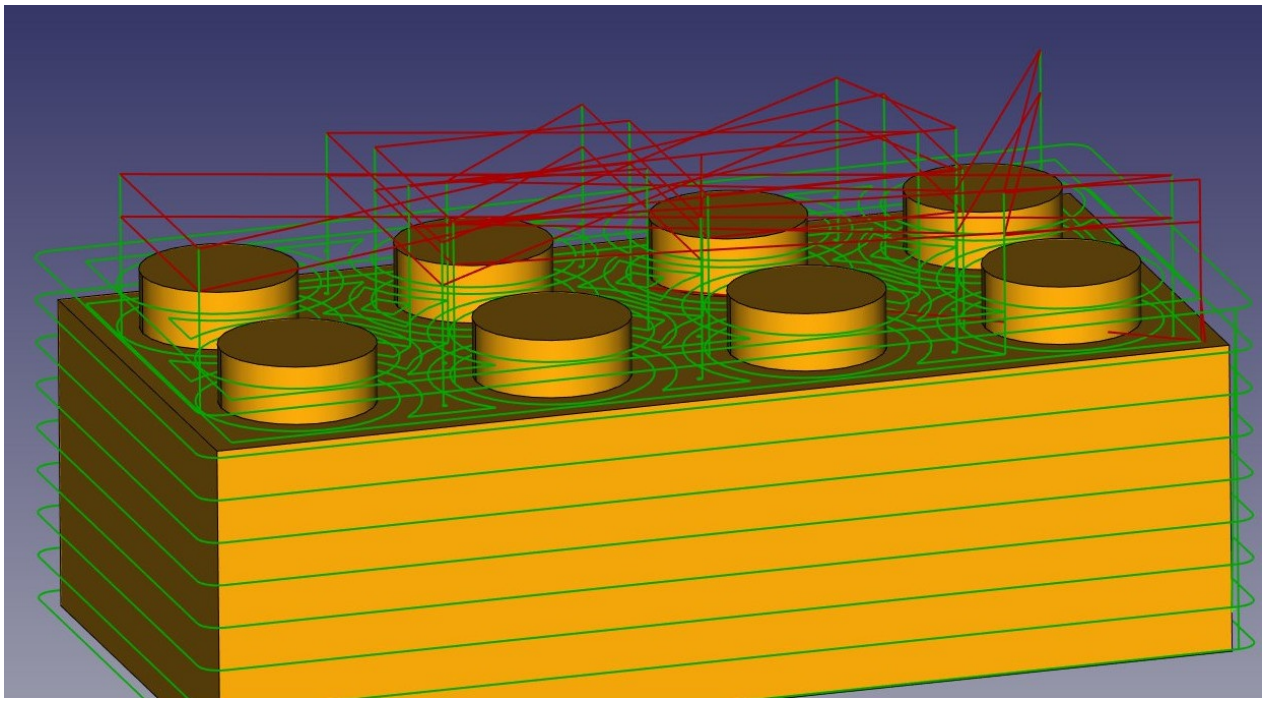
- Then, let's duplicate this first loop a couple of times, so the tool will carve out the whole block. Select the FaceProfile path, and press the Array button.
- Set the **Copies** property of the array to 8, and its **Offset** to -2mm in the Z direction, and move the placement of the array by 2mm in the Z direction, so the cutting will start a bit above the pad, and include the height of the dots too.



- Now we have defined a path that, when followed by the milling machine, will carve a rectangular volume out of a block of material. We now need to carve out the space between the dots, in order to reveal them. Hide the Pad, and show the final piece again, so we can select the face that lies between the dots.
- Select the top face, and press the Face Pocket button. Set the **Offset** property to 1mm, and the **retraction height** to 20mm. That is the height to where the cutter will travel when switching from one loop to another. Otherwise, the cutter might cut right through one of our dots:



- Once again, make an array. Select the FacePocket object, and press the Array button. Set the **Copies** number to 1 and the **offset** to -2mm in the Z direction. Move the placement of the array by 2mm in the Z direction. Our two operations are now done:



- Now all that is left to do is to join these two operations into one. This can be done with a Path Compound or a Path Project. Since we will need nothing more and will be ready to export already, we will use the project. Press the Project button.
- Set the **Use Placements** property of the project is to True, because we changed the placement of the arrays, and we want that to be taken into account in the project.
- In the tree view, drag and drop the two arrays into the project. You can reorder the arrays inside the project if needed, by double-clicking it.

- The project can now be exported to G-code, by selecting it, choosing menu **File -> Export**, selecting the G-code format, and in the pop-up dialog that will open, selecting a post-procesing script according to your machine.

There are many applications available to simulate the real cutting, one of them that is also multi-platform and open-source, like FreeCAD, is Camotics.

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: http://www.freecadweb.org/wiki/index.php?title=Mesh_Module
- The STL file format: https://en.wikipedia.org/wiki/STL_%28file_format%29
- Slic3r: http://slic3r.org/
- Cura: https://ultimaker.com/en/products/cura-software
- The Cura Workbench: https://github.com/cblt2l/FreeCAD-CuraEngine-Plugin
- The Path Workbench: http://www.freecadweb.org/wiki/index.php?title=Path_Workbench
- Camotics: http://camotics.org/

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 that. In technical fields (engineering, architecture, etc), this is usually done with drawings, that are handed over to the person responsible for assembling the final product, that will explain how to do it.

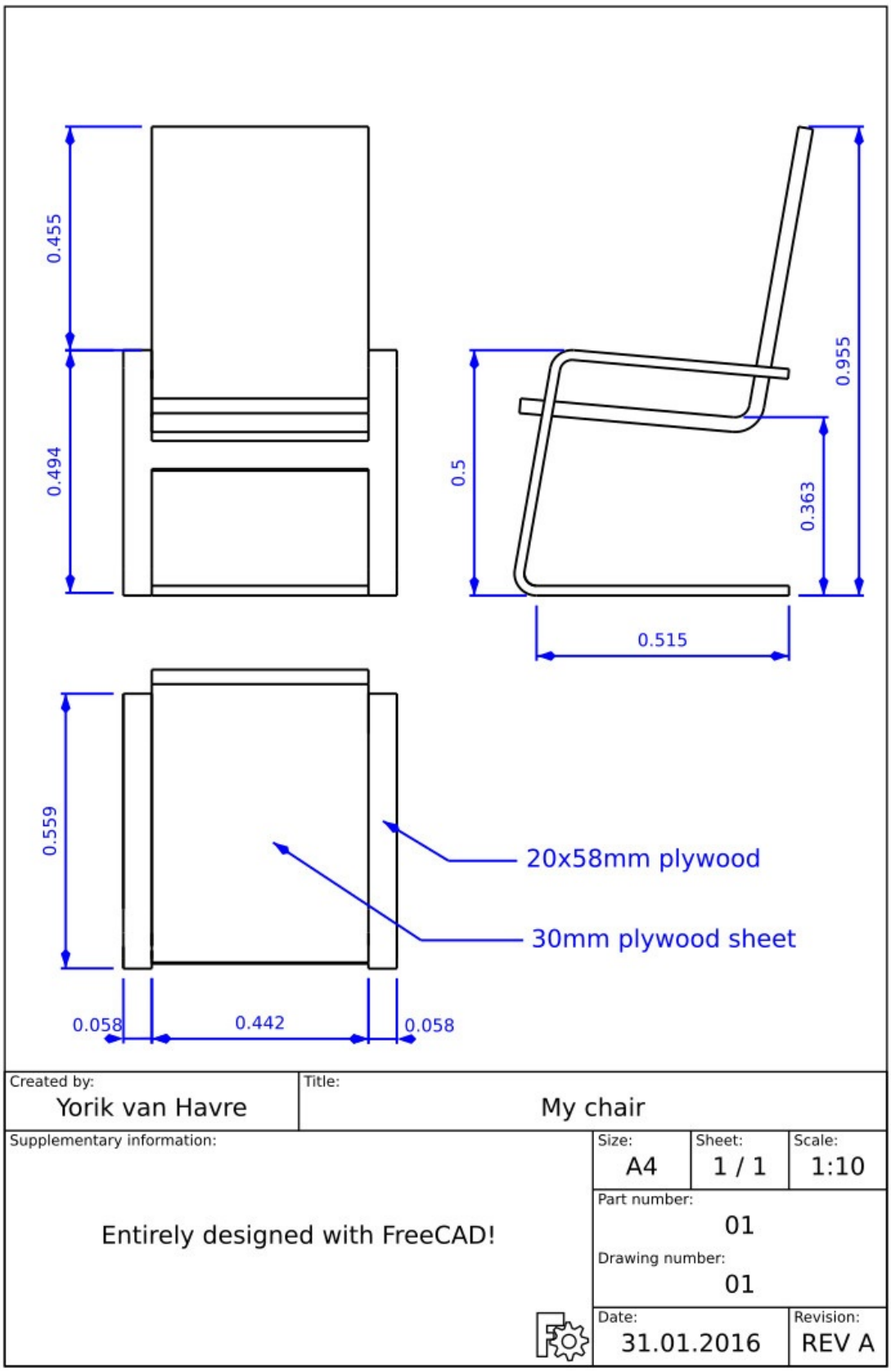
Typical examples are Ikea instructions, architectural drawings or blueprints. These drawings usually contain not only the drawing itself, but also many annotations, such as texts, dimensions, numbers, symbols that will help other people to understand what needs to be done and how.

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

The Drawing 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 title. On these sheets, you can then place views of the 3D objects you modeled previously, and configure how these views must appear on the sheet. Finally, thanks to an addon called Drawing Dimensioning Workbench, you can also place all kinds of annotations on the sheet, such as dimensions, texts, and other usual symbols commonly used in technical drawings.

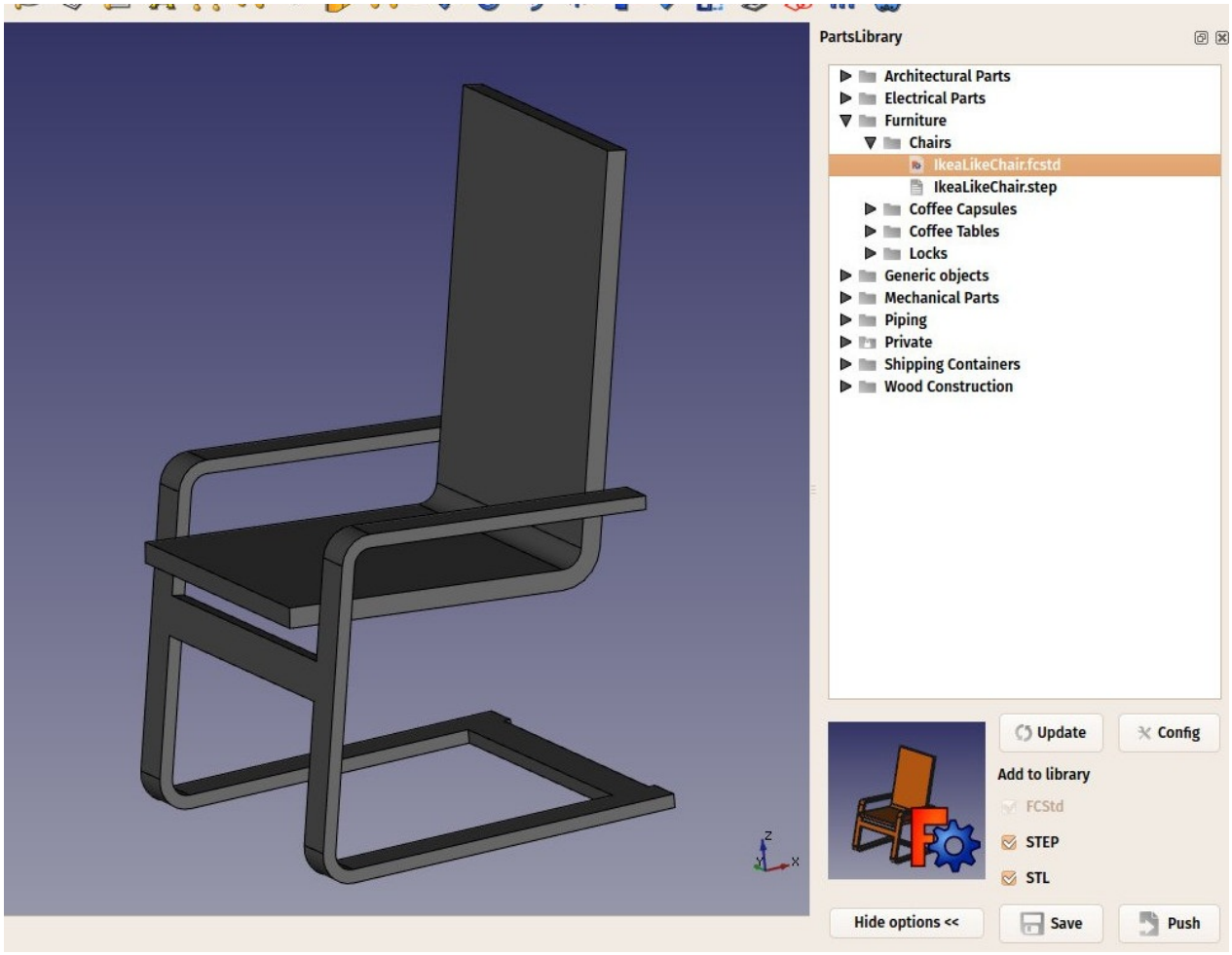
Drawing sheets, once complete, can be printed or exported as SVG, PDF or DXF files.

In the following exercise, we will see how to create a simple drawing of a chair model found in the FreeCAD library (Furniture -> Chairs -> IkeaChair). 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.

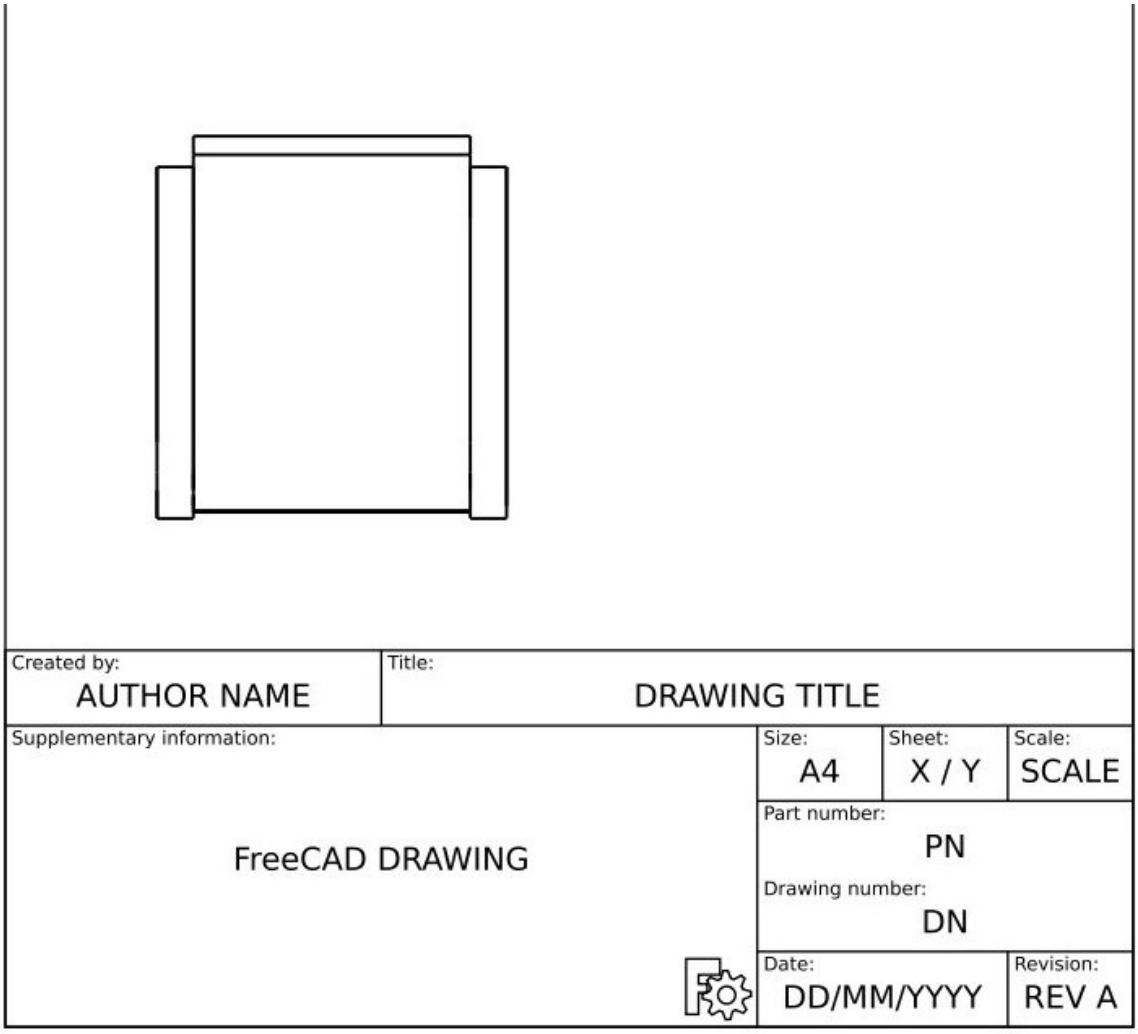


- Load the IkeaChair file from the library. You can choose between the .FCStd version, which will load the full modeling history, or the .step version, which will create only one object, without the history. Since we won't need to model any further now, it is best to

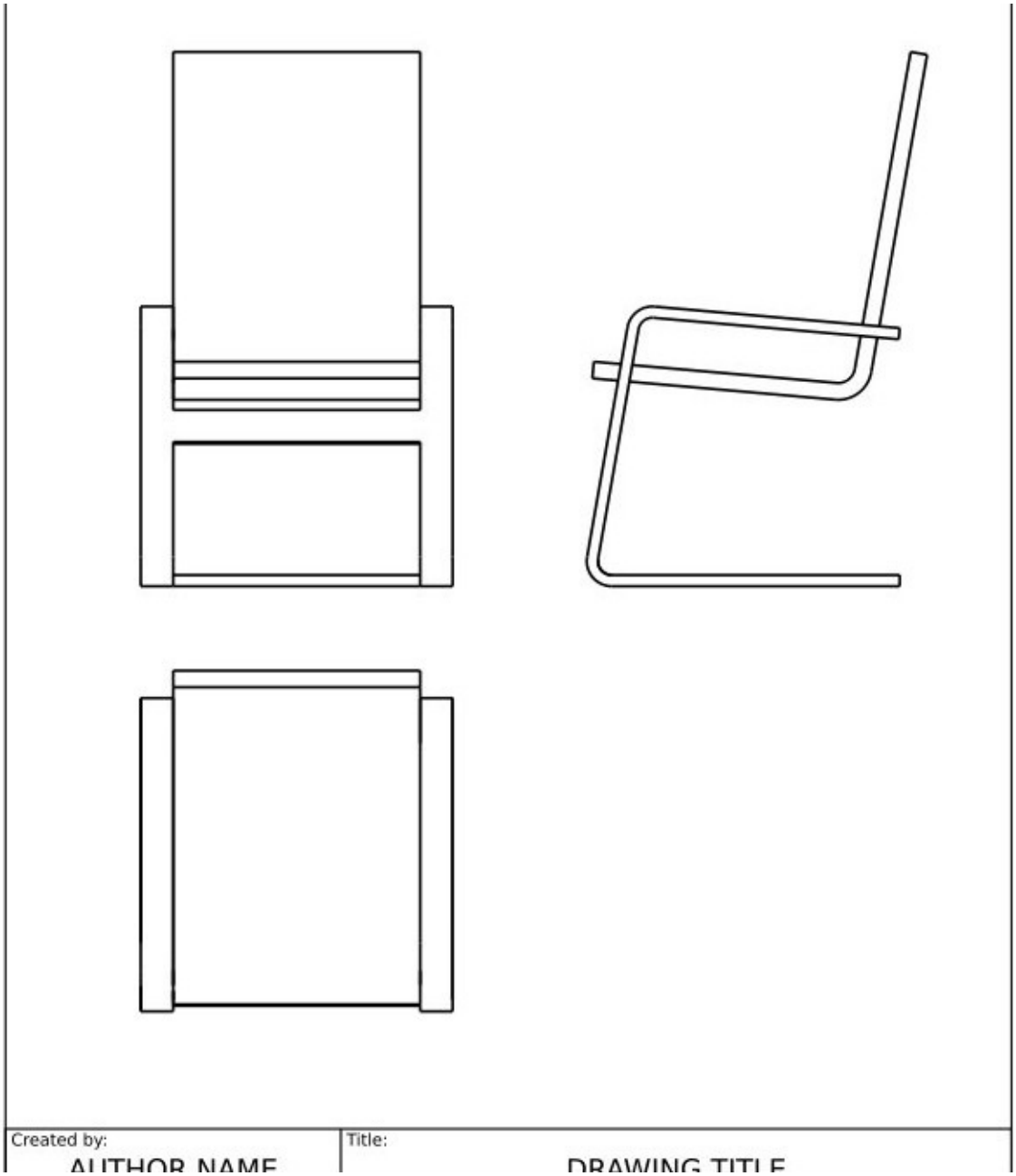
choose the .step version, as it will be easier to manipulate.



- Switch to the Drawing Workbench
- Press the little arrow next to the New Drawing Page button.
- Select the **A4 Portrait / ISO7200** 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.
- Press the Insert view button.
- A View object will be created on our page. Give the view the following properties:
  - X: 100
  - Y: 150
  - Scale: 0.1
  - Rotation: 270
- We now have a nice top view (which is the default projection) of our chair:



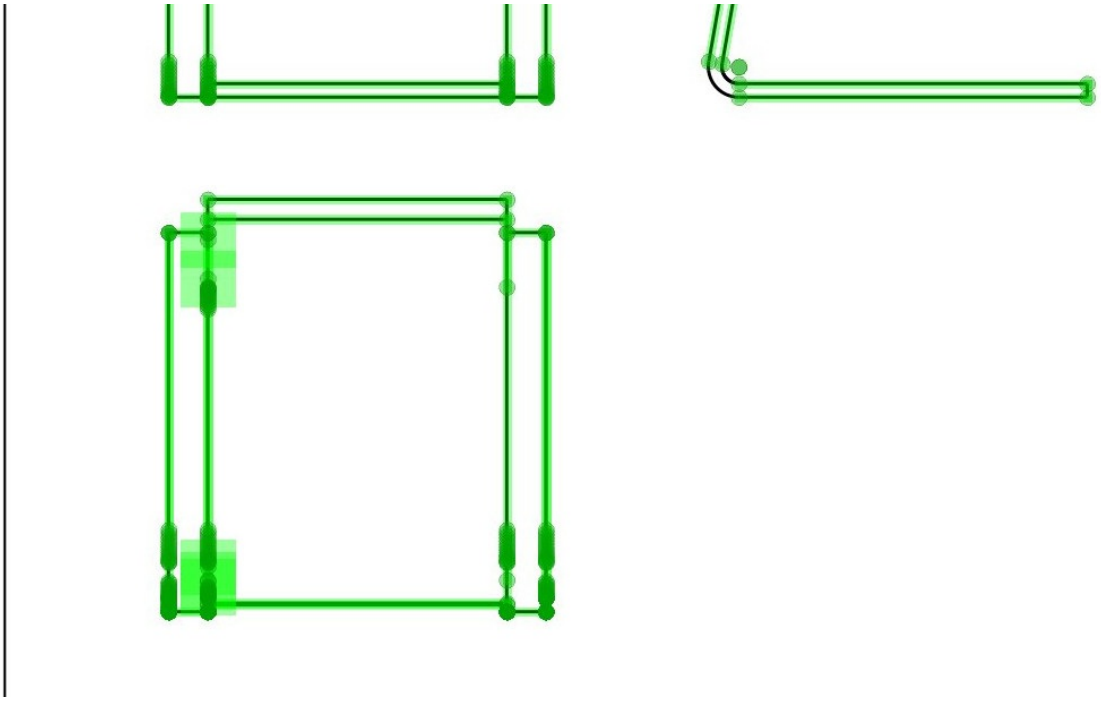
- Let's repeat the operation twice, to create two more views. We will set their X and Y values, which indicate the position of the view on the page, in order to show them apart from the top view, and their direction, to create different view orientations. Give each new view the following properties:
  - View001 (front view): X: 100, Y: 130, Scale: 0.1, Rotation: 90, Direction: (-1,0,0)
  - View002 (side view): X: 180, Y: 130, Scale: 0.1, Rotation: 90, Direction: (0,-1,0)
    - After that, we obtain the following page:



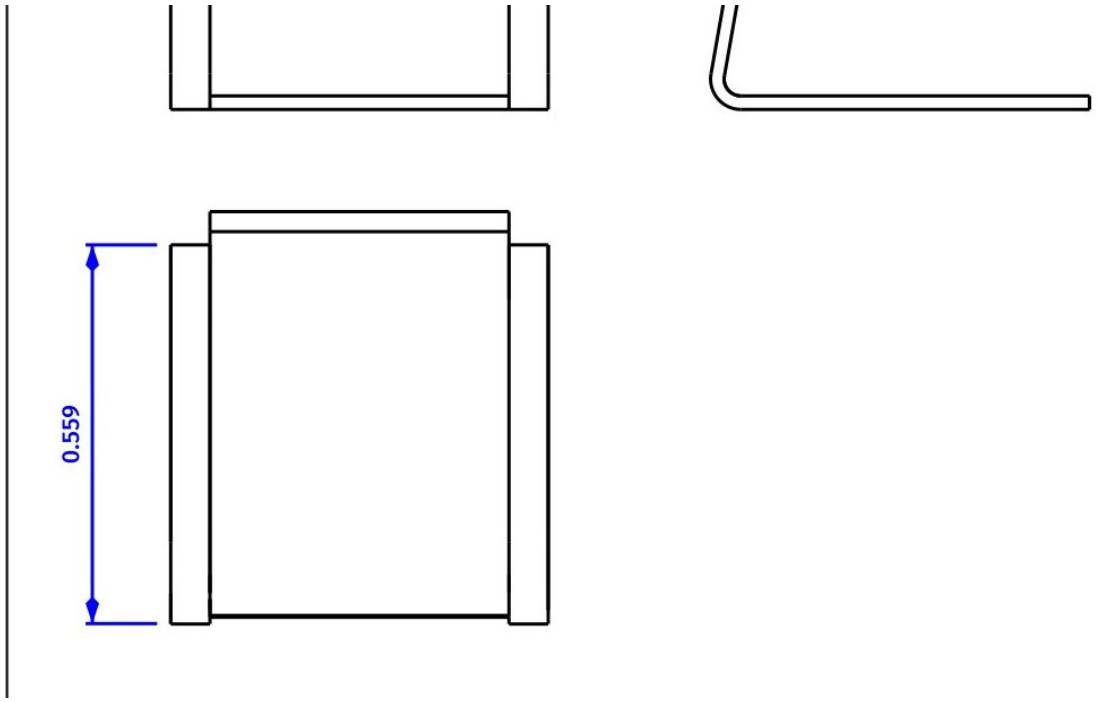
- We can tweak a bit the aspect of our views if we want, for example we can raise their **Line Width** property to 0.5.

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 place a view of these dimensions on our sheet with the Draft View tool (which can be used with a single dimension or a group containing dimensions), or we can do things directly on the Drawing sheet, using the Drawing Dimensioning Workbench, which is installable from the FreeCAD addons. We will use here this latter method.

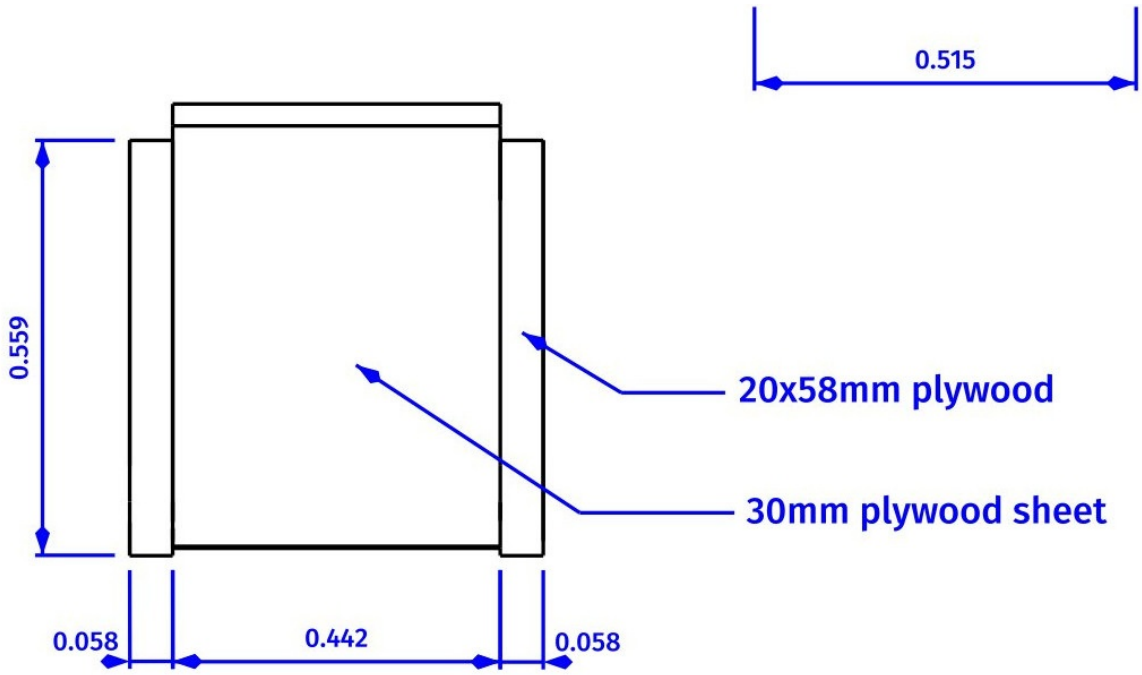
- Switch to the Drawing Dimensioning Workbench
- Press the **Add Linear Dimension** button. Available nodes are highlighted in green on the drawing page:



- Click two of these points, then click a third point to place the dimension line:



- The Linear Dimension tool, as most of the other Drawing Dimensioning tools, will not exit after you finished, allowing you to place more dimensions. When you are done, simply click the **Close** button in the Task panel.
- Repeat the operation, until all the dimensions you wish to indicate are placed. Take a minute to browse through the different options proposed in the Linear Dimension's task panel. For example, by unticking the **auto place text** option, you will be able to place the text of the dimension elsewhere, like on the image below:



- We will now place two indications, using the **Welding/Groove symbols** tool, selecting the default one (no groove symbol). Draw the two lines like on the image above.
- Now place two texts using the **Add text** tool, and change their **text** property to the contents of your likings.
- Our drawing is now complete, all that is left to do is to fill in the informations of the sheet titleblock. With most of the default FreeCAD templates, this can be done easily, by changing the **Editable Texts** property of the page.

Our page can now be exported to SVG to be worked further in graphical applications like inkscape, or to DXF by selecting menu **File -> Export**. The Drawing Dimensioning workbench also features its own **DXF export** tool, which also supports the annotations added with that workbench. The DXF format is importable in almost all existing 2D CAD applications. Drawing pages can also be directly printed or exported to PDF.

Downloads

- The chair model: https://github.com/FreeCAD/FreeCAD-library/blob/master/Furniture/Chairs/IkeaLikeChair.step
- The file created during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/drawing.FCStd
- The SVG sheet produced from that file: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/drawing.svg

Read more

- The Drawing Workbench: http://www.freecadweb.org/wiki/index.php?title=Drawing_Module
- The Drawing Dimensioning Workbench:

https://github.com/hamish2014/FreeCAD_drawing_dimensioning

- The FreeCAD library: https://github.com/FreeCAD/FreeCAD-library
- Inkscape: http://www.inkscape.org

BIM modeling

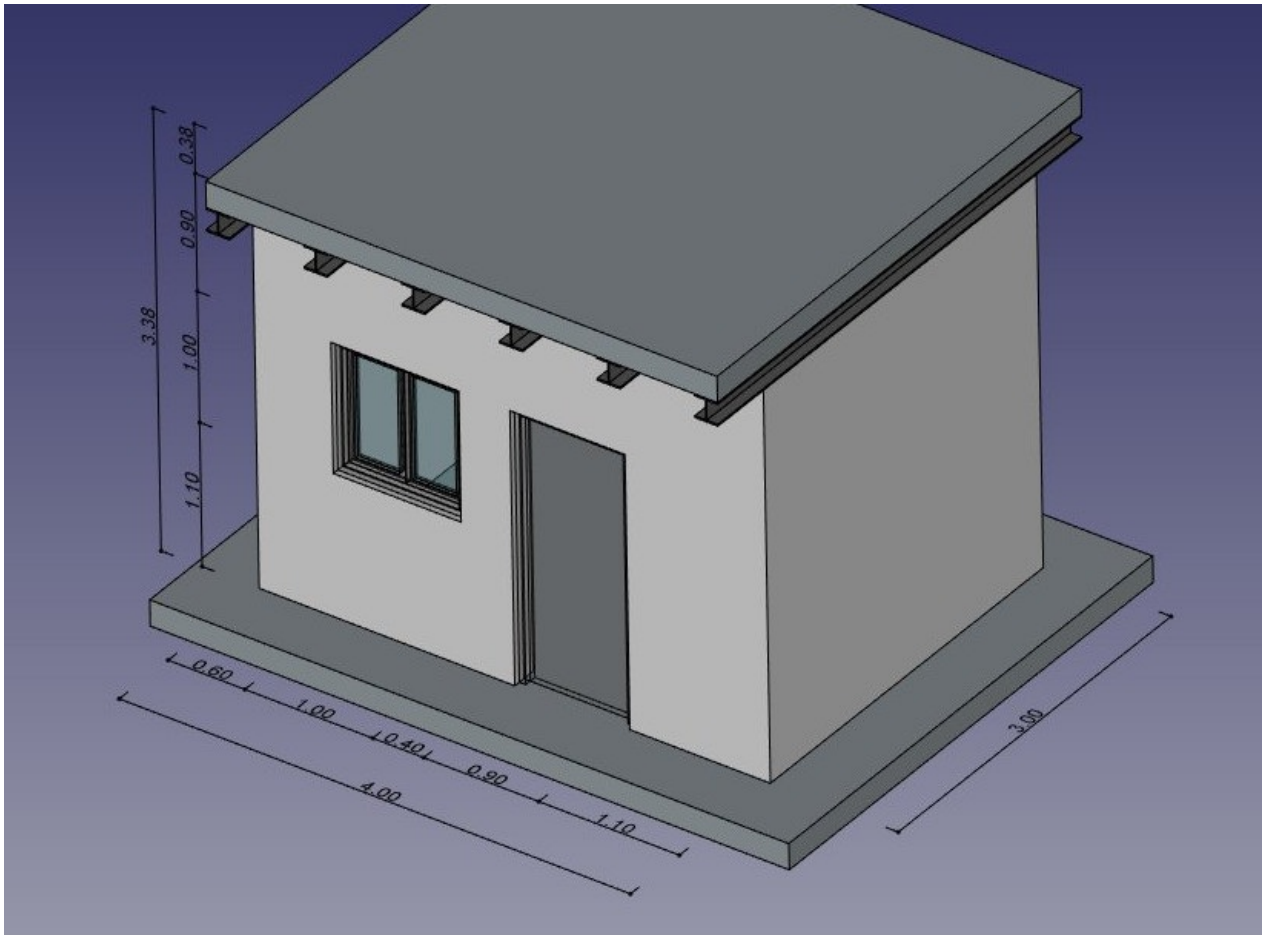
BIM stands for Building Information Modeling. The exact definition of what it is varies, but we can say simply that is how buildings and other large structures like bridges, tunnels, etc... are modeled today. BIM models are usually based on 3D models, and also include a series of additional layers of information, such as materials information, relationships to other objects or models, or special instructions for building or maintenance. This extra information permits all kinds of advanced analyses of the model, such as structural resistance, cost and construction time estimations, or calculaitons of energy consumption.

The Arch Workbench of FreeCAD implements a series of tools and facilities for BIM modeling. Although it has a different purpose, it is made to work in tight integration with the rest of FreeCAD: Anything made with any other workbench of FreeCAD can become an Arch object, or be used as a base for an Arch object.

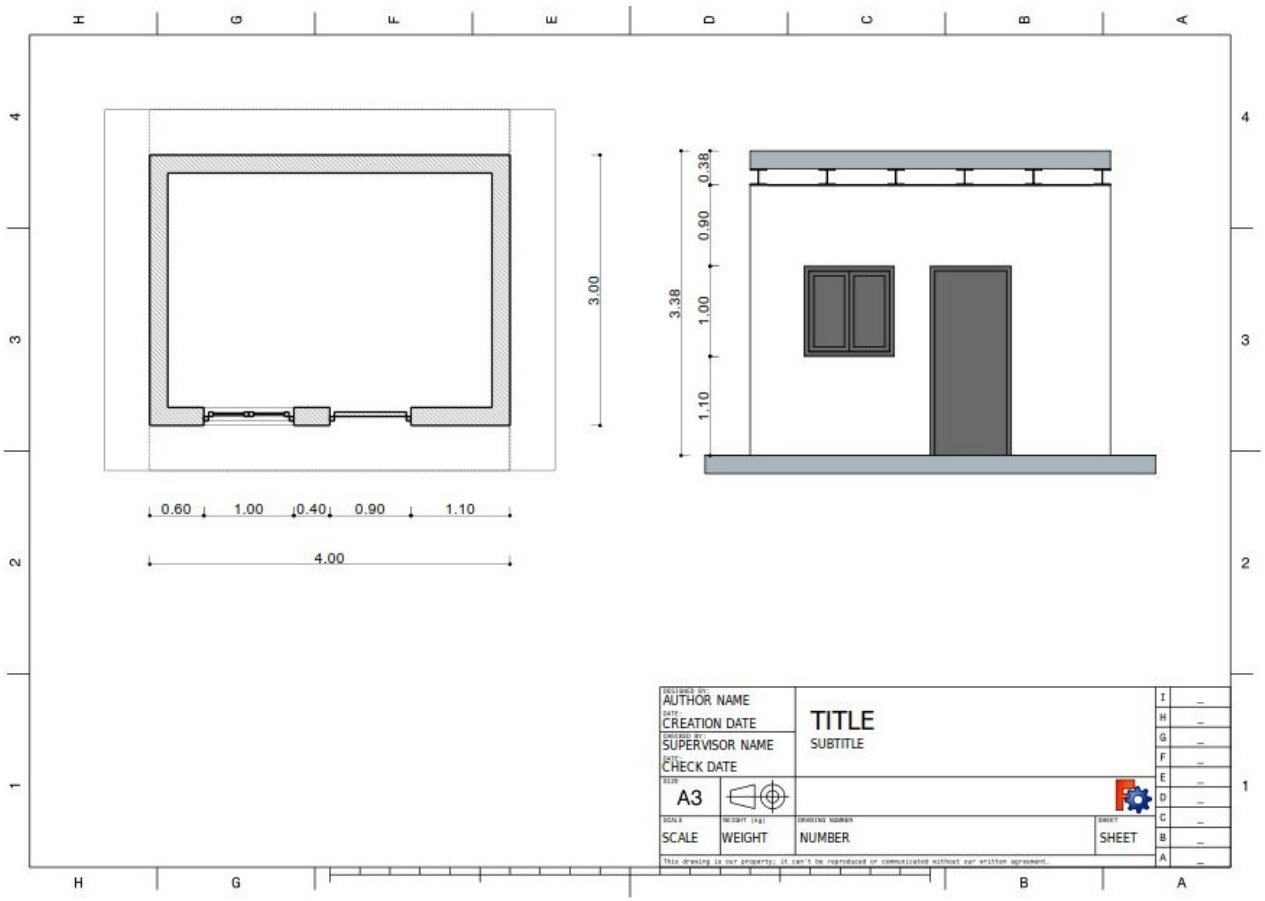
As in the PartDesign Workbench, the objects produced by the Arch Workbench are meant to be built in the real world. Therefore, they need to be **solid**. The Arch tools usually take care of that automatically, and also provide utility tools to help you check the validity of objects.

The Arch Workbench also includes all the tools from the Draft Workbench, and uses its grid and snapping system. Before beginning, it is always a good idea to browse through the preferences pages of both Draft and Arch and set the default settings to your likings.

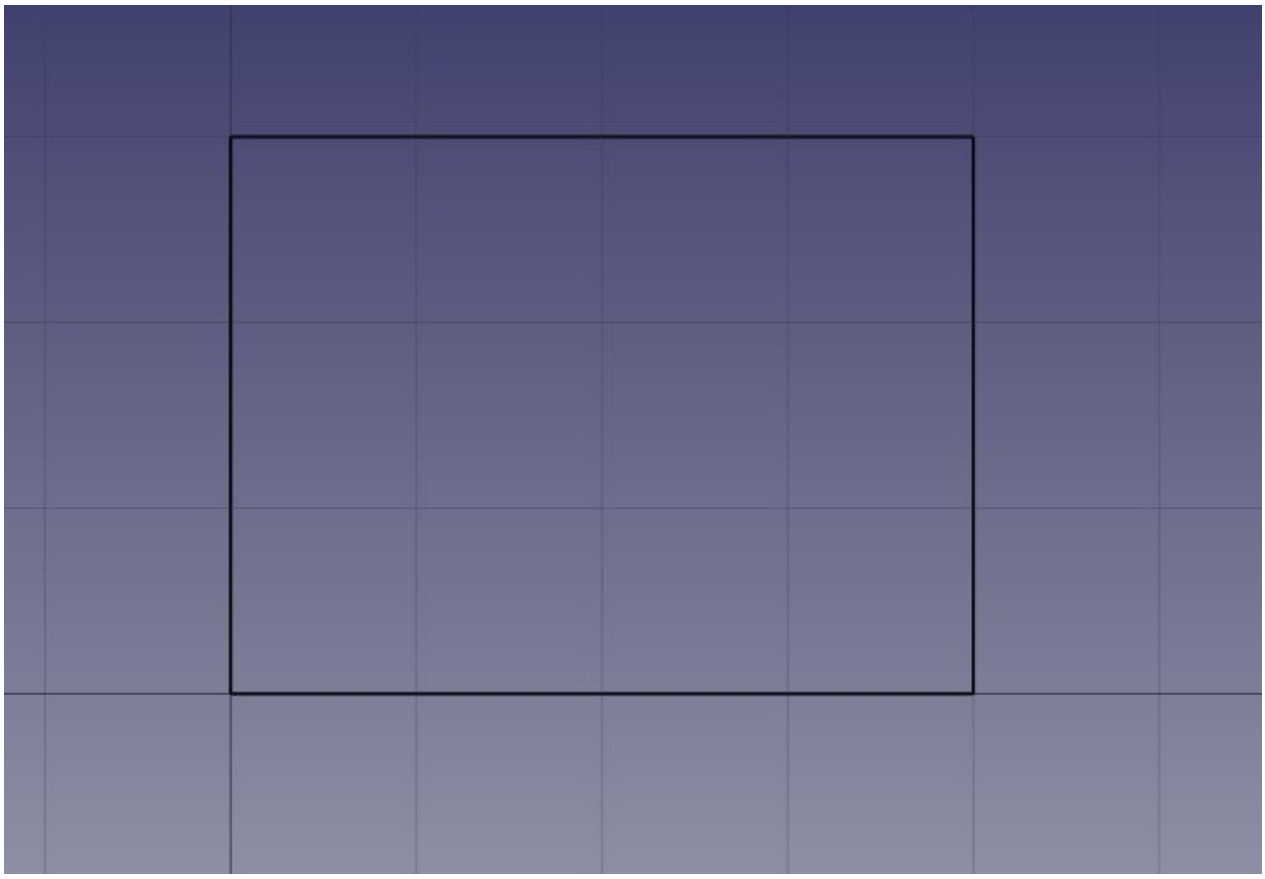
In this chapter, we will see how to model this small building:



and produce a plan and a section view from it:



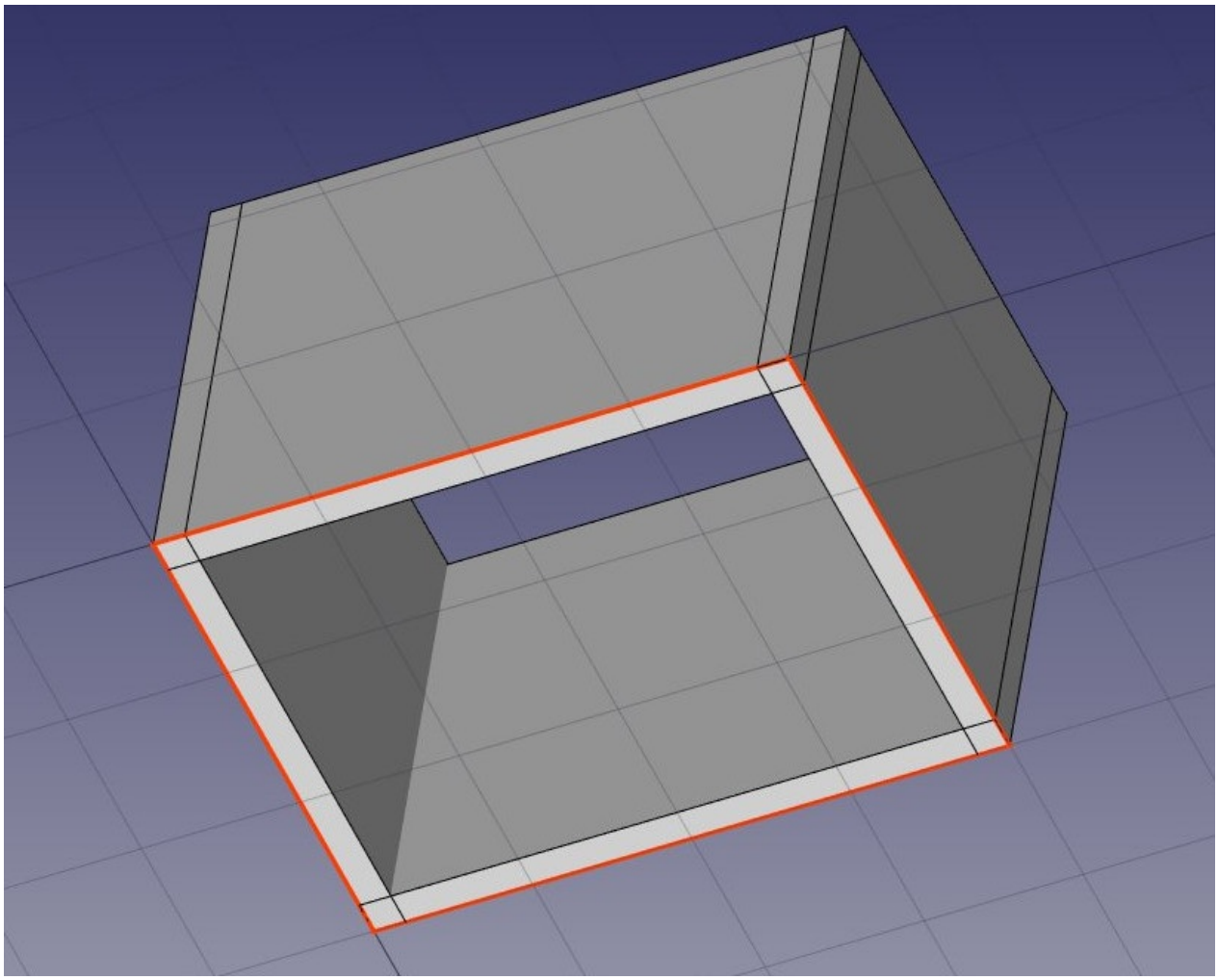
- Create a new document, and switch to the Arch Workbench.
- Open menu **Edit -> Preferences -> Draft -> Grid and Snapping** and set the **grid spacing** setting to 1000mm, so we have a one meter-based grid, which will be convenient for the size of our buiding.
- On the **snapping toolbar**, make sure the grid snap button is enabled, so we can use the grid as much as possible.
- 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:
  - 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 drew always in the same direction (clockwise). This is not necessary, but will ensure that the walls that we will build next all have the same left and right directions. You might also think we could simply have drawn a rectangle here, which is true. But the four lines will allow us to illustrate better how to add one object into another.

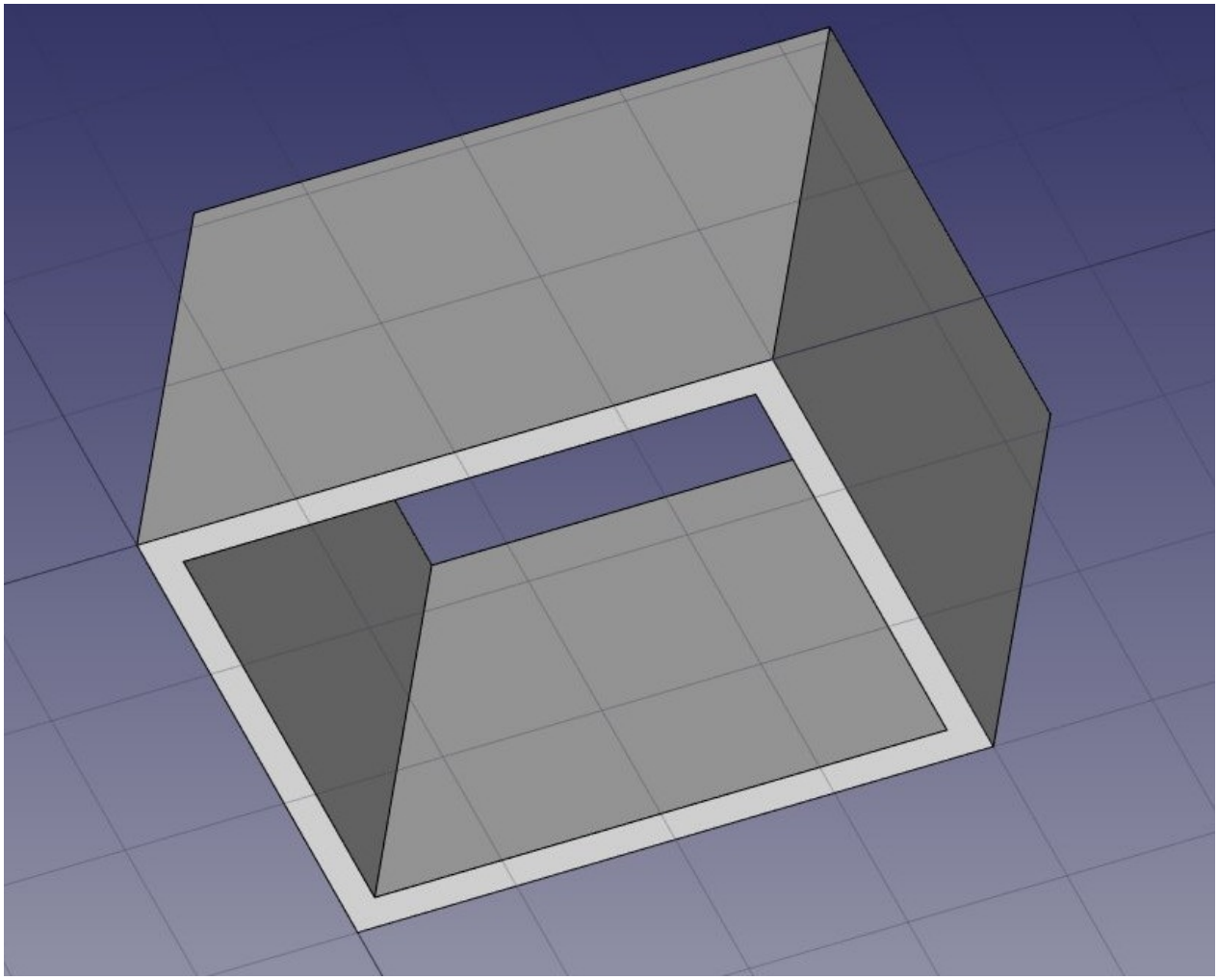
- Select the first line, then press the Arch Wall button.
- Repeat this for the 3 other lines, untilyou have 4 walls.
- Select the four walls, and set their **Height** property to **3.00m** and their **Alignment**

property to **left**. If you didn't draw the lines in the same order as we did above, some of the walls might have their left and right directions flipped, and might need to be set to **right** instead. You will obtain four intersecting walls, on the inside of the baselines:



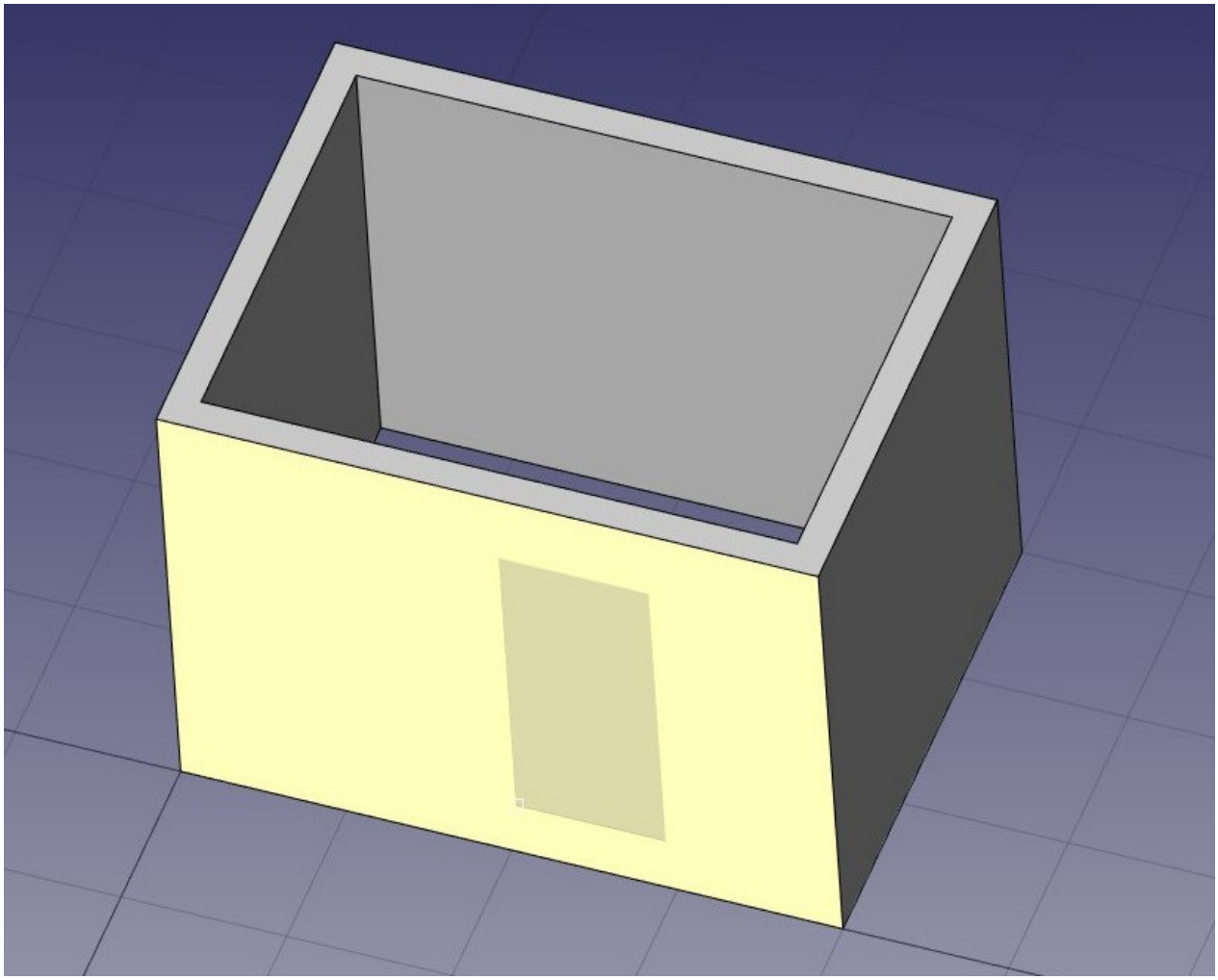
Now we need to join these walls together, so they intersect properly. This is not necessary when your walls are drawn in a way that they already connect cleanly, but here we need to, since they are intersecting. In Arch, this is done by electing one of the walls to be the "host", and adding the others to it, as "additions". All arch objects can have any number of additions (objects whose geometry will be added to the host's geometry), and subtractions (objects whose geometry will be subtracted). The additions and subtractions of an object can be managed anytime by double-clicking the object in the tree.

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

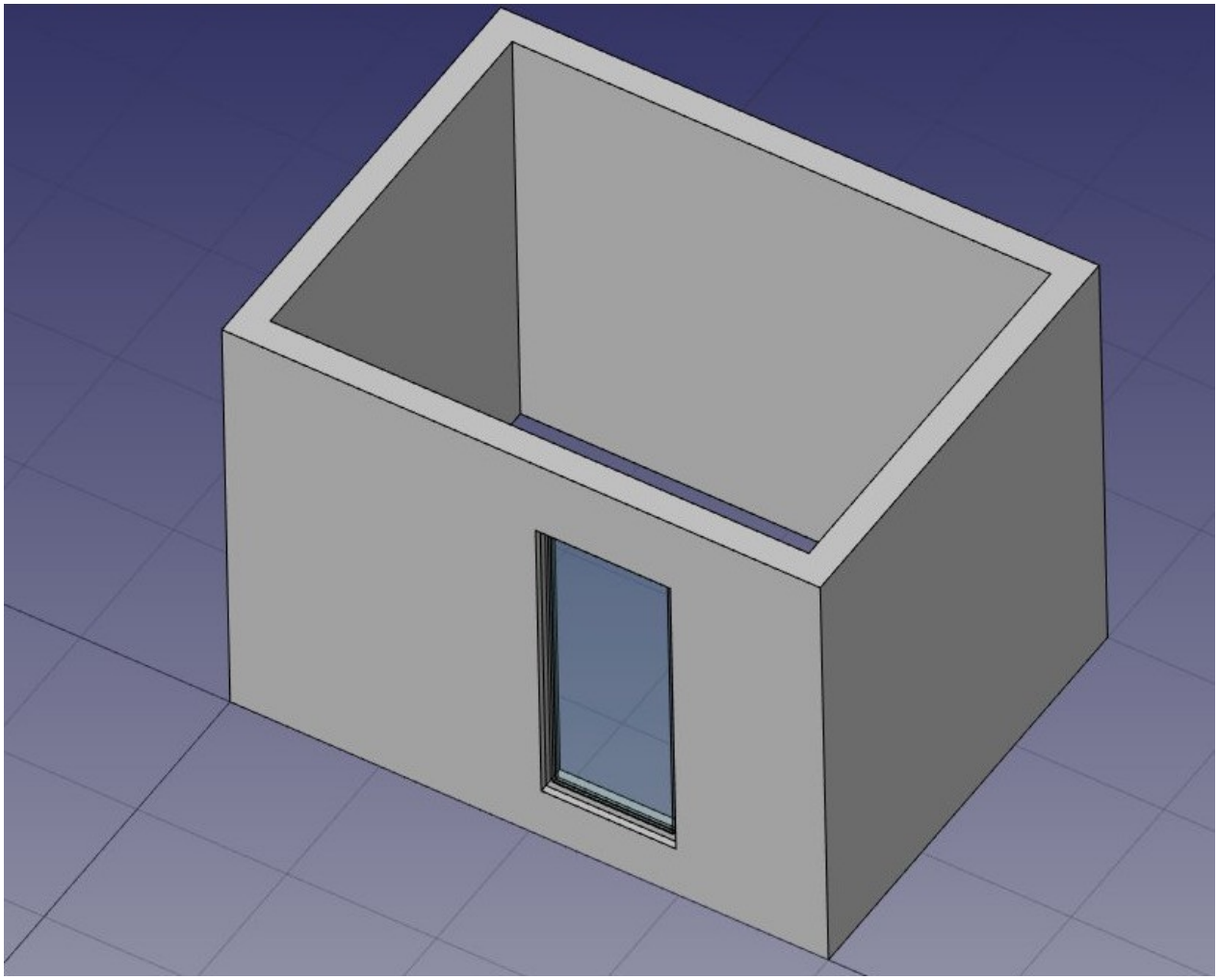


The individual walls are however still accessible, by expanding the wall in the tree view.

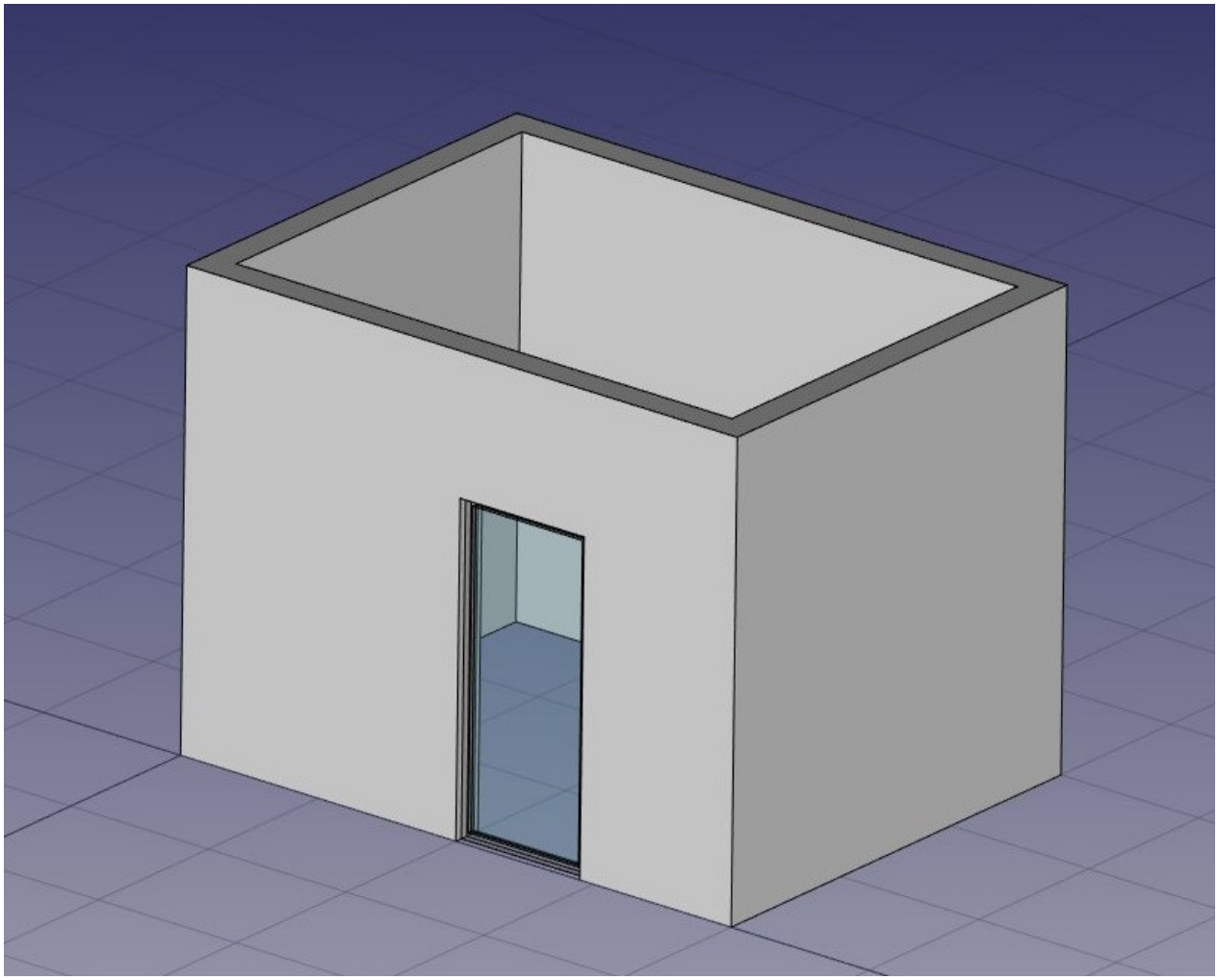
- Let's now place a door. In FreeCAD, doors are considered a special case of windows, so this is done using the Window tool.
- Start by selecting the wall. This is not necessary, but a good habit to take. If an object is selected when starting the window tool, you will force the window to be inserted in that object, even if you snap to another object.
- Set the Working Plane to **auto** so we are not restricted to the ground plane
- Press the Window button.
- In the window creation panel, select the **Simple door** preset, and set its **Width** to 0.9m and its **Height** to 2.1m
- Make sure the Near snap location is turned on, so we can snap on faces
- Place your window roughly on the middle of the front face of the wall:



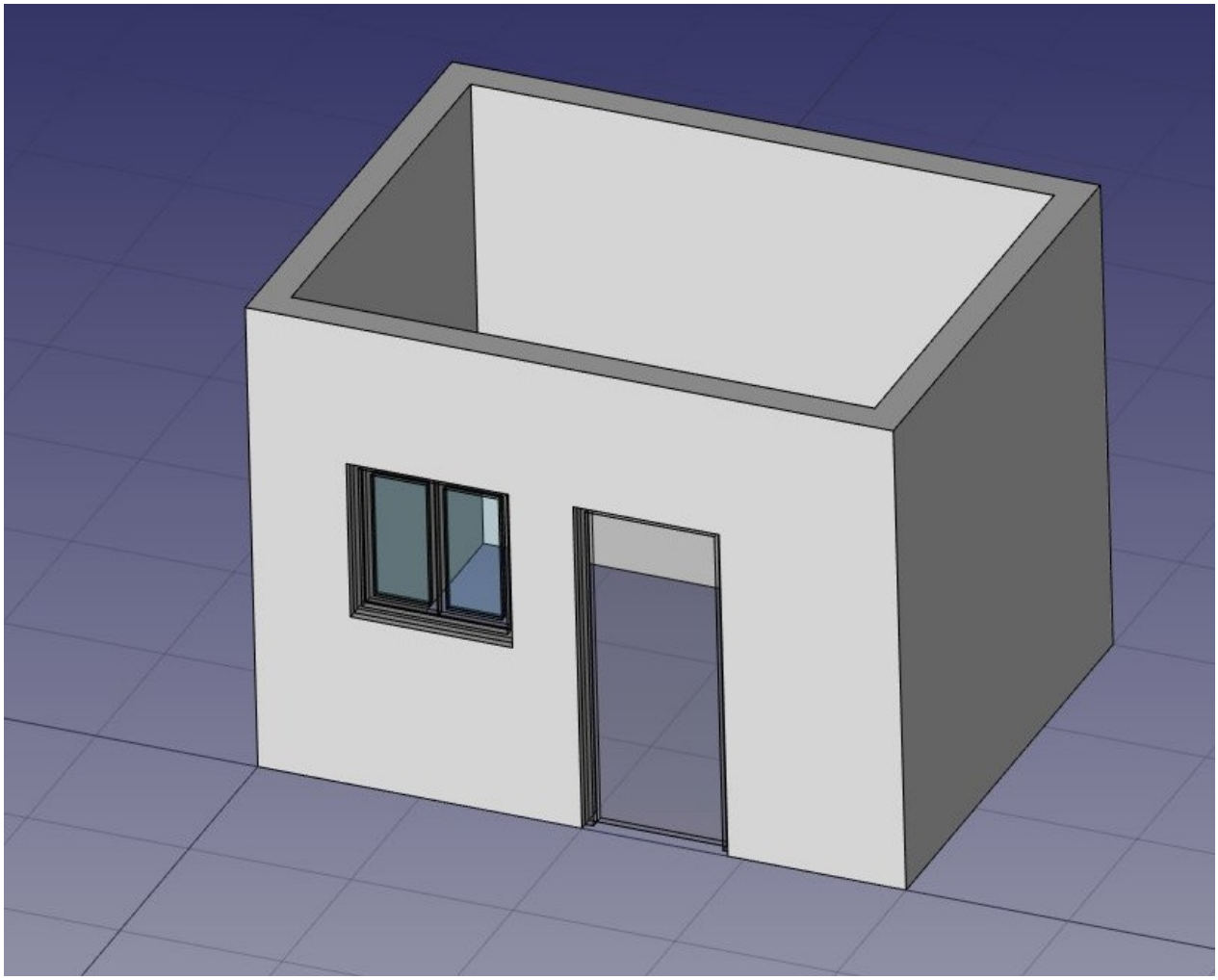
- After clicking, our window is placed on the correct face, but not exactly where we want:



- 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 = 2m, y = 0, z = 0**. Our window is now exactly where we want it:

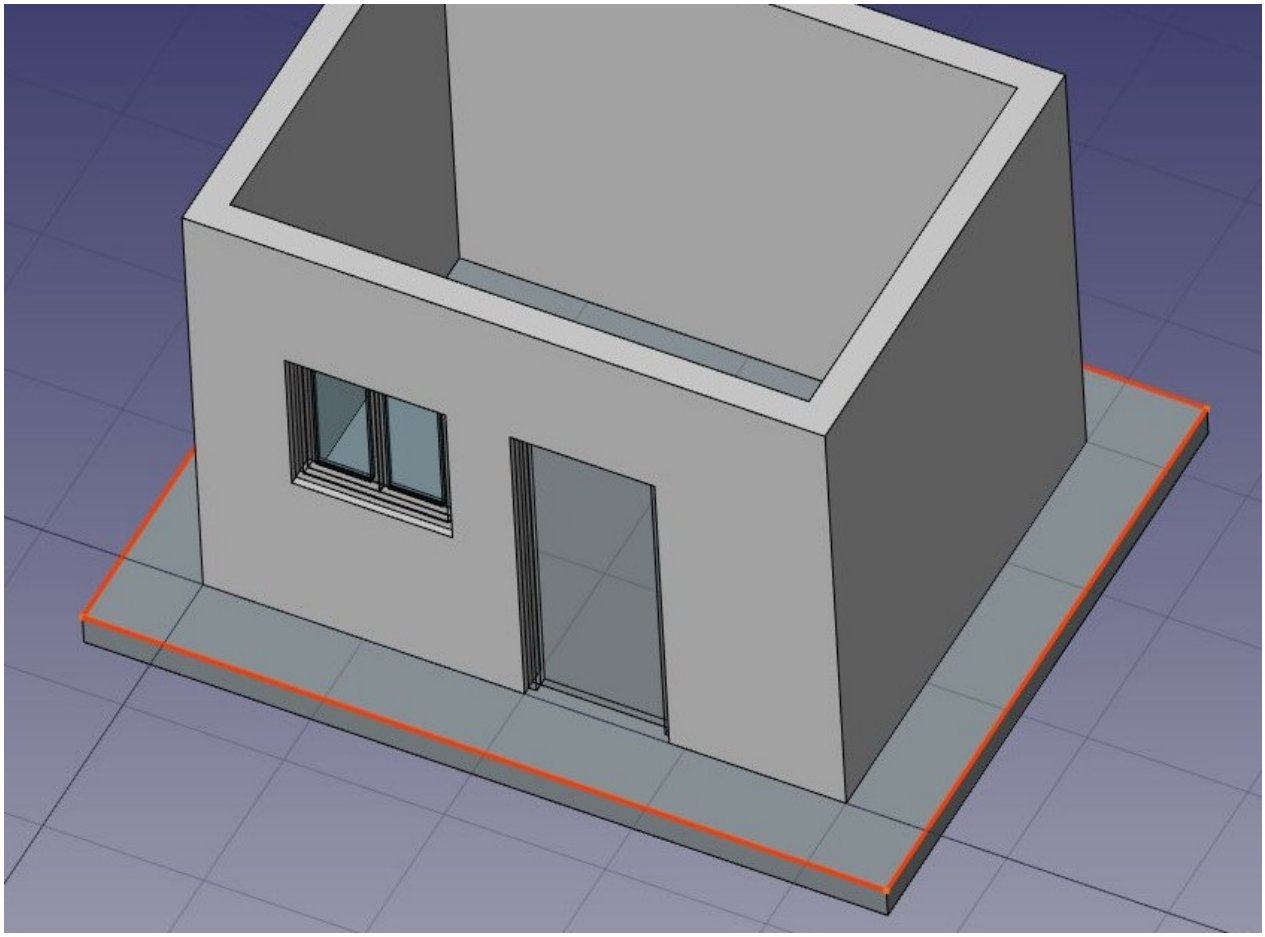


- Repeat the operation to place a window: Select the wall, press the window tool, select the **Open 2-pane** preset, and place a 1m x 1m window in the same face as the door. Set the placement of the underlying sketch to position **x = 0.6m, y = 0, z = 1.1m**, so the upper line of the window is aligned to the top of the door.

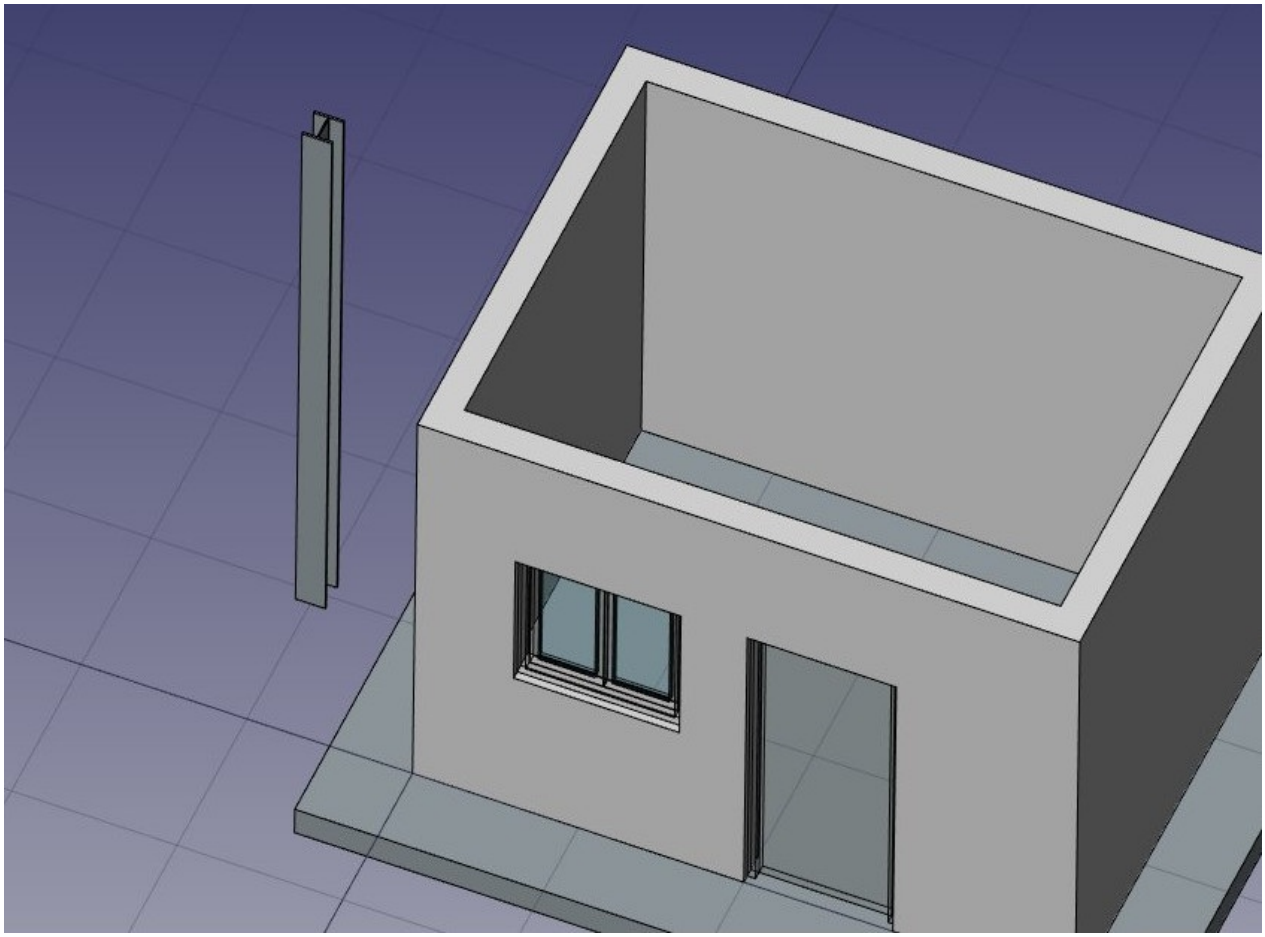


Windows are always built on sketches. It is easy to create custom windows by first creating a sketch on a face, then turning it into a window by selecting it, then pressing the window button. Then, the window creation parameters, that is, which wires of the sketch must be extruded and how much, can be defined by double-clickingthe window in the tree view. Let's now create a slab:

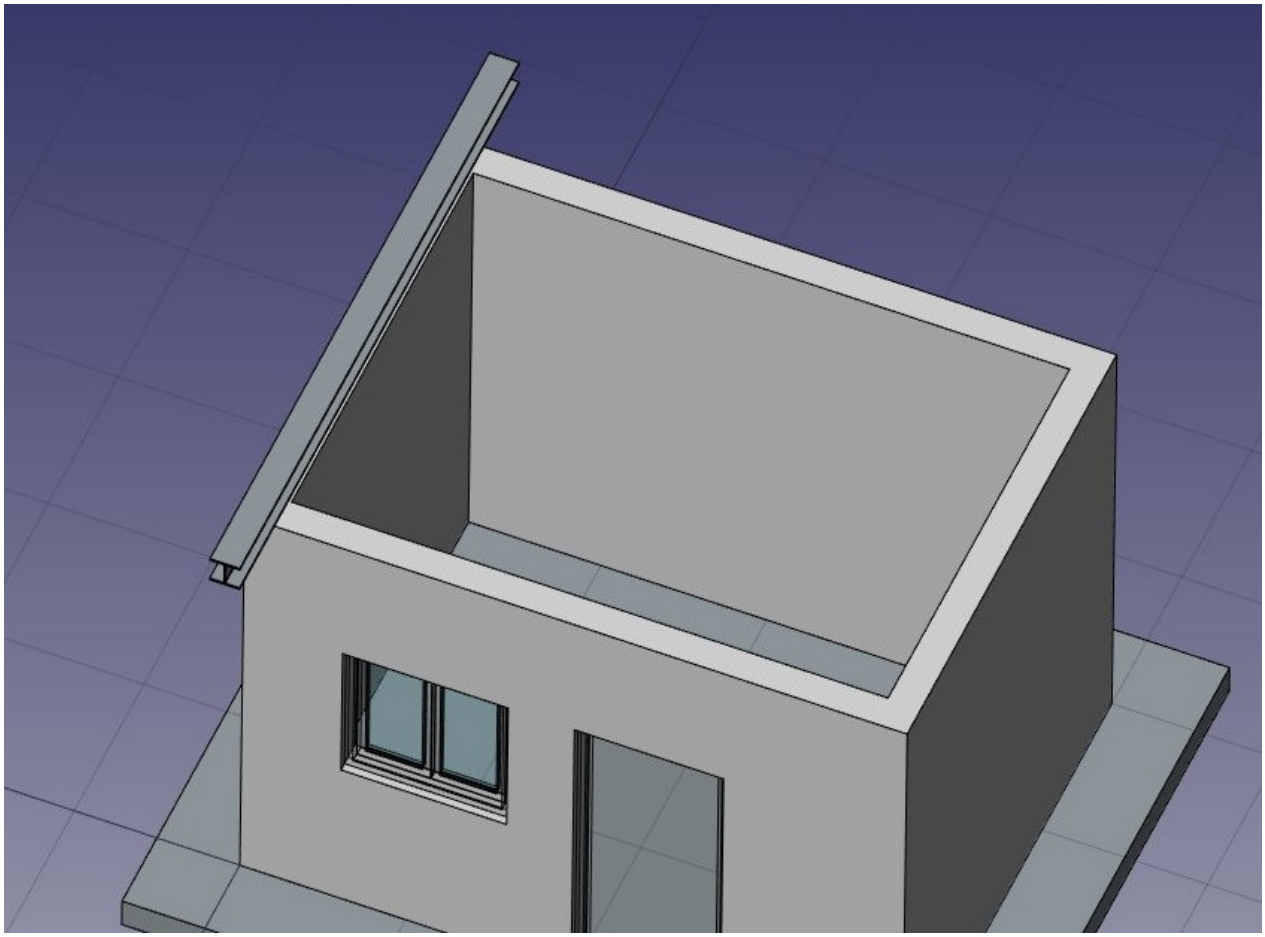
- Set the Working Plane to **XY** plane
- Create a rectangle with a **length** of 5m, a height of **4m**, and place it at position x:-0.5m, y:-0.5m, z:0.
- Select the rectangle
- Click the structure tool to create a slab from the rectangle
- Set the **height** property of the slab to 0.2m and its **normal** direction to (0,0,-1) because we want it to extrude downwards. We could also simply have moved it 0.2m down, but it is always good practice to keep extruded objects at the same place as their base profile.
- Set the **Role** 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.



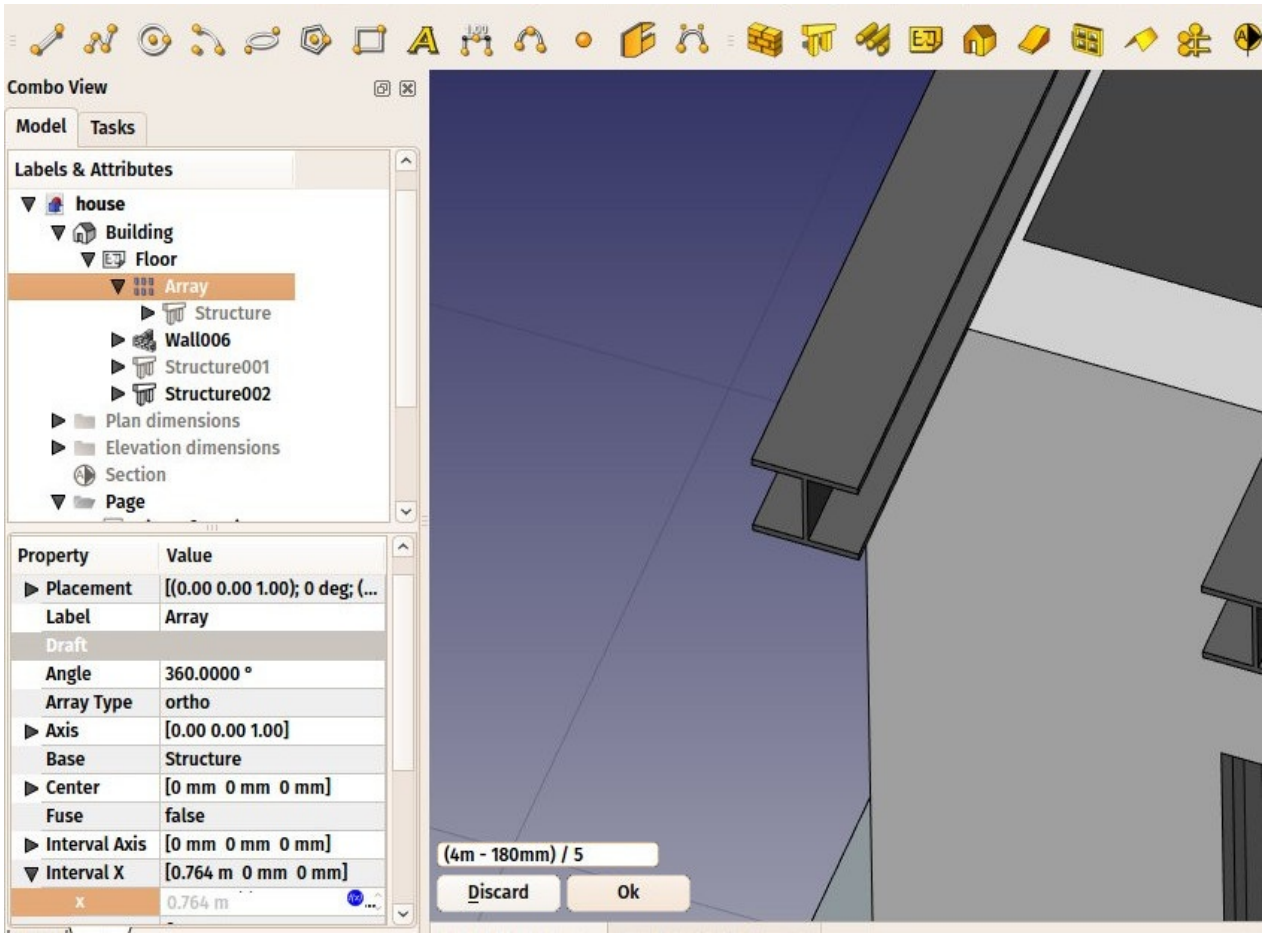
- Let's now use one of the structural presets to make a metallic beam. Click the structure button, select a **HEB 180** preset, and set its height to **4m**. Place it anywhere:



- Adjust its **placement** by setting its **rotation** to 90° in the (1,0,0) axis, and its **position** to x:90mm, y:3.5m, z:3.09m. This will position the beam exactly on one of the side walls:



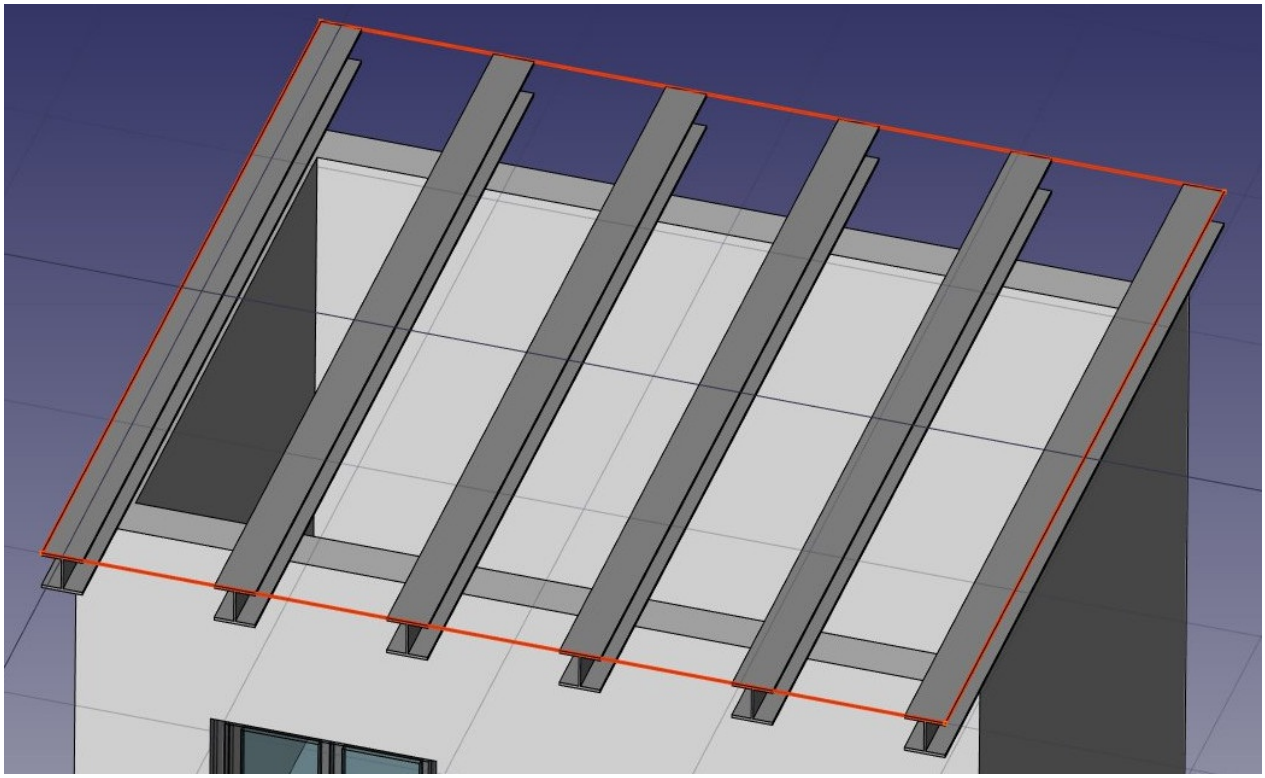
- We need now to duplicate this beam a couple of times. We could do that one by one using the clone tool, but there is a better way, to do all the copies at once using an array:
- Select the beam
- Press the Array button
- Set the **Number X** property of the array to 6, leave the Y and Z numbers to 1
- Expand the **interval X** property, and press the small **expression** icon at the right side of the X field. This will open an expressions editor:



- Write **(4m-180mm)/5** in the expression field, and press **OK**. This will set the x value to 0.764 (4m is the total length of our front wall, 180mm is the width of the beam, which is why it is called HEB180, and we want a fifth of that space as interval between each beam):

- We can now easily build a simple slab on top of them, by drawing a rectangle directly on the top plane of the beams. Select a top face of one of the beams

- Press the working plane button. The working plane is now set to that face.
- Create a rectangle, snapping to two opposite points of the border beams:

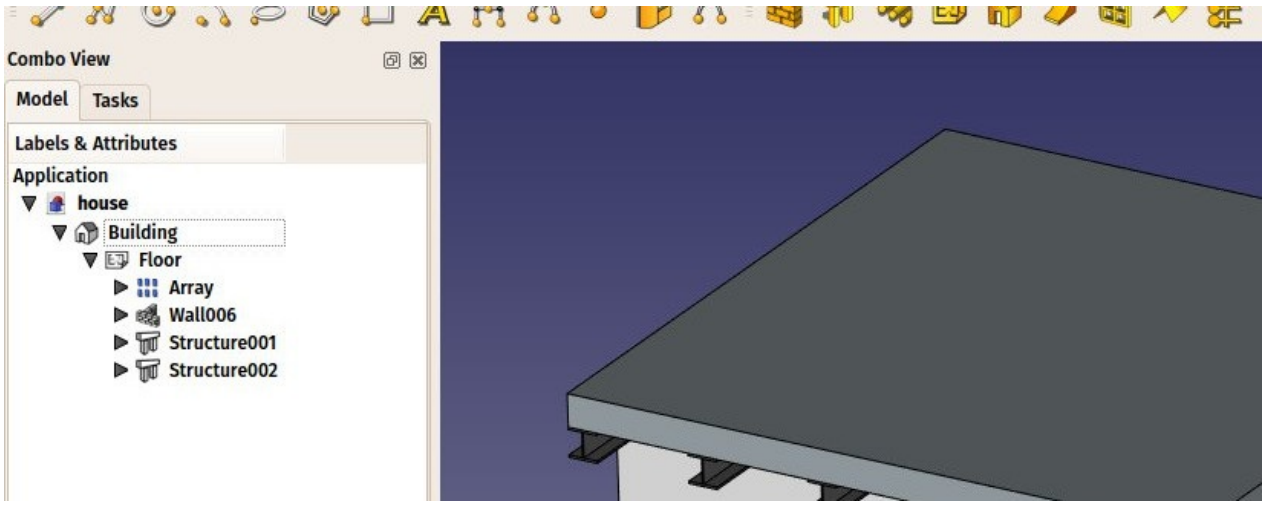


- Select the rectangle
- Click the structure button and create a slab with a height of **0.2m**.

That's it, our model is now complete. We should now organize it so it exports correctly to IFC. The IFC format requires that all objects of a building are inside a building object, and optionally, inside a storey. It also requires that all buildings are placed on a site, but the IFC exporter of FreeCAD will add a default site automatically if needed, so we don't need to add one here.

- Select the two slabs, the wall, and the array of beams
- Press the Floor button
- Select the floor we just created
- Press the Building button

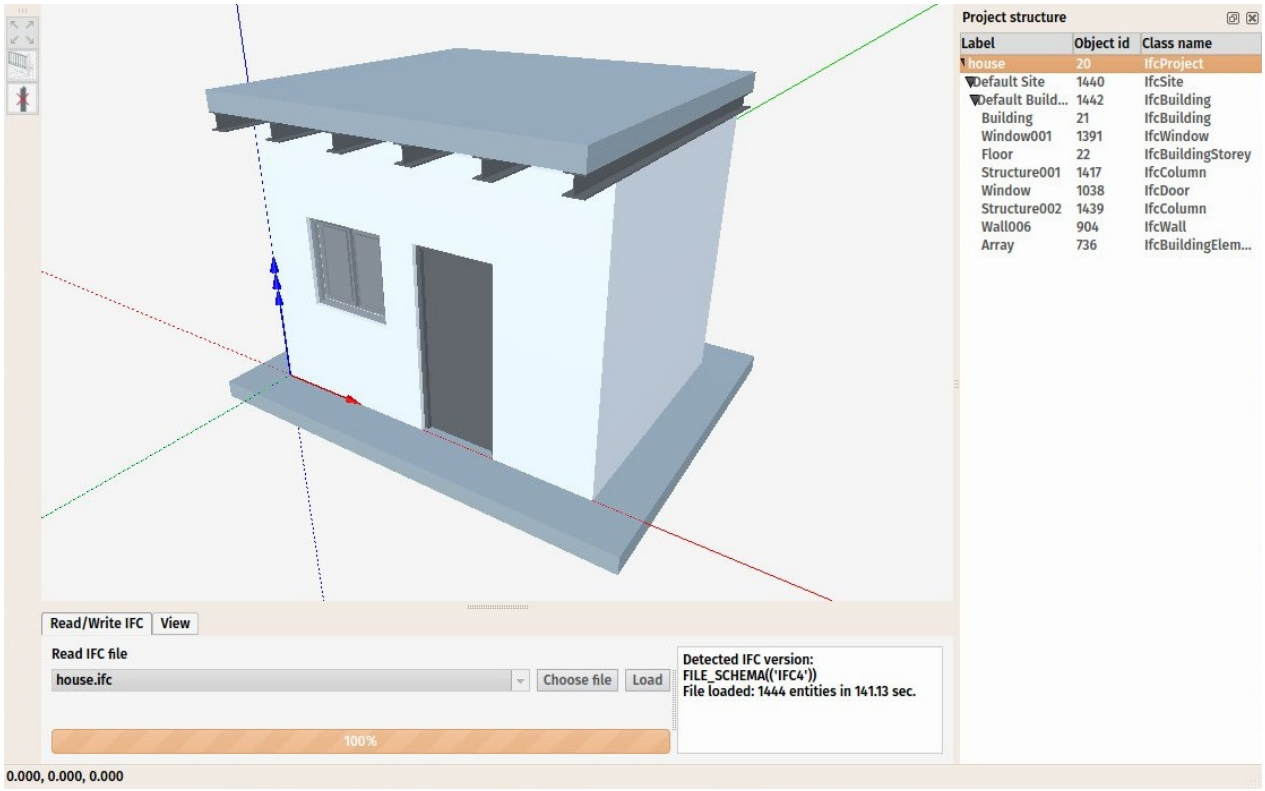
Our model is now ready to export:



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.

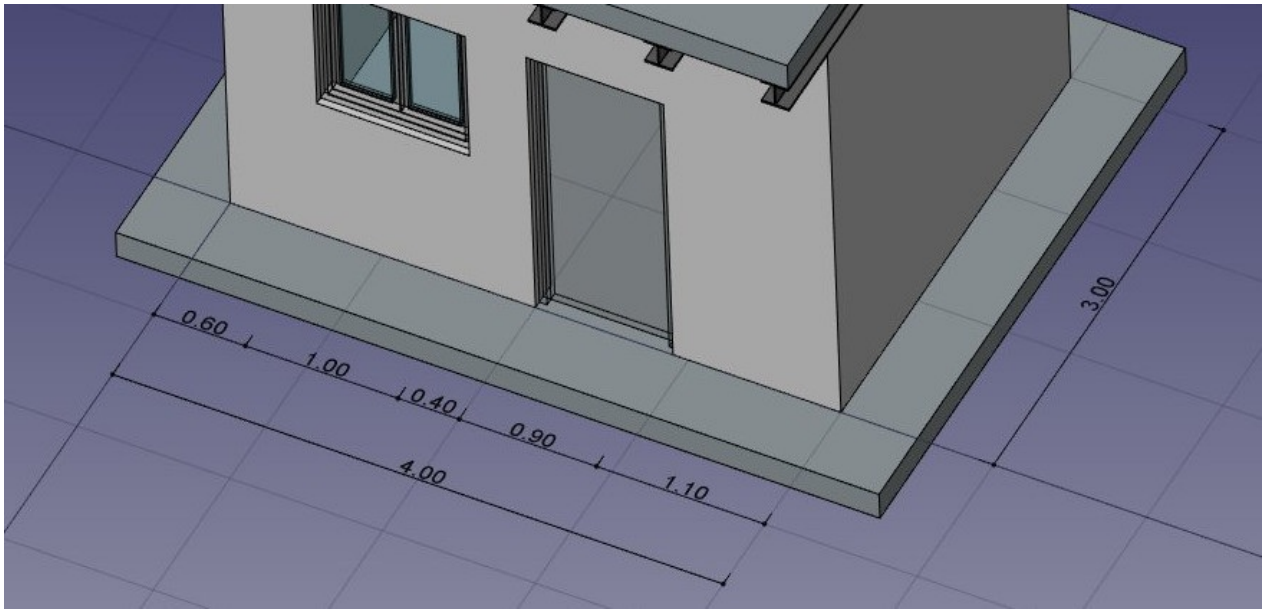
In FreeCAD, IFC import and export is done by interfacing with another piece of software, called IfcOpenShell. To be able to export to IFC from FreeCAD, the IfcOpenShell-python package must be installed on your system. Be sure to select one which uses the same python version as FreeCAD. The python version that FreeCAD uses is informed when opening the **View -> Panels -> Python console** panel in FreeCAD. When that is done, we can now export our model:

- 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.

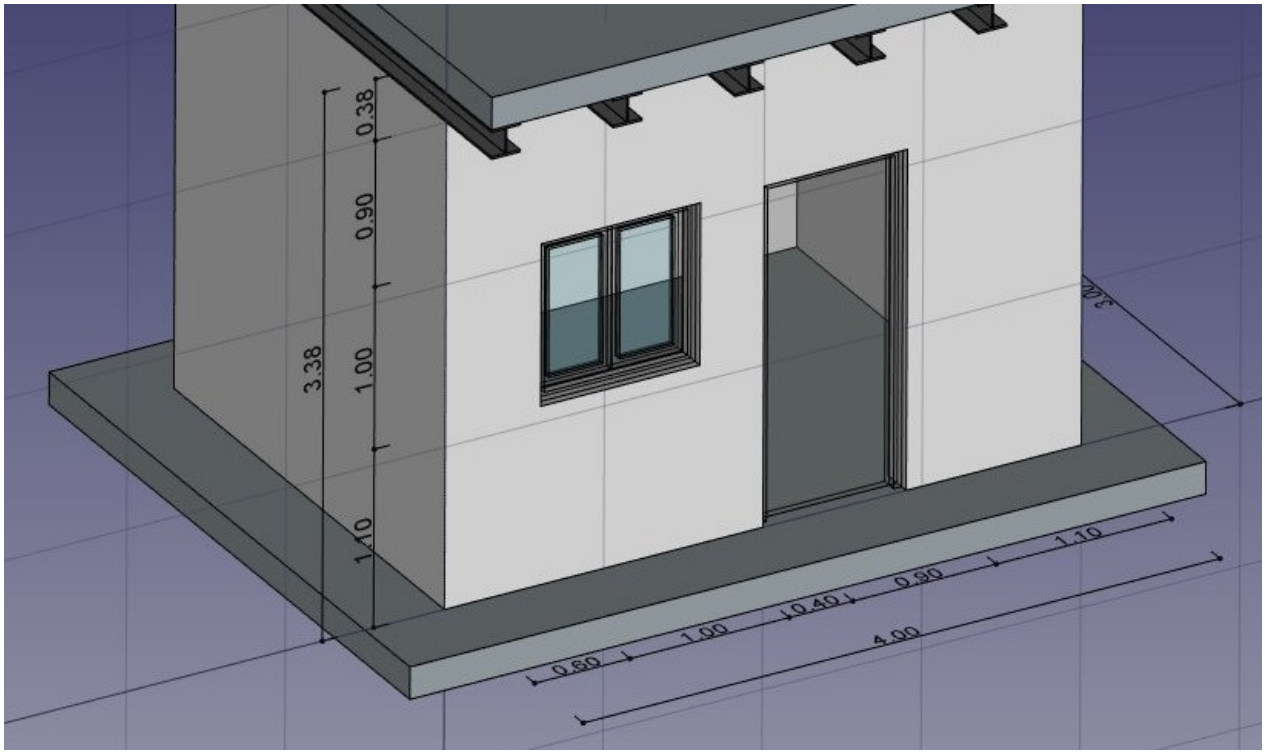


We will now place some dimensions. Unlike the previous chapter, where we drew all the dimensions directly on the Drawing sheet, we will use another method here, and place Draft dimensions directly in the 3D model. These dimensions will then be placed on the Drawing sheet. We will first make two groups for our dimensions, one for the dimensions that will appear in the plan view, and another for those that appear on the elevation.

- Right-click the "house" document in the tree view, and create two new groups: **Plan dimensions** and **Elevation dimensions**.
- Set the Working Plane to **XY** plane
- Make sure the restrict snap location is turned on, so everything you draw stays on the working plane.
- Draw a couple of dimensions, for example as on the image below. Pressing **Shift** and **Ctrl** while snapping the dimension points will give you additional options.

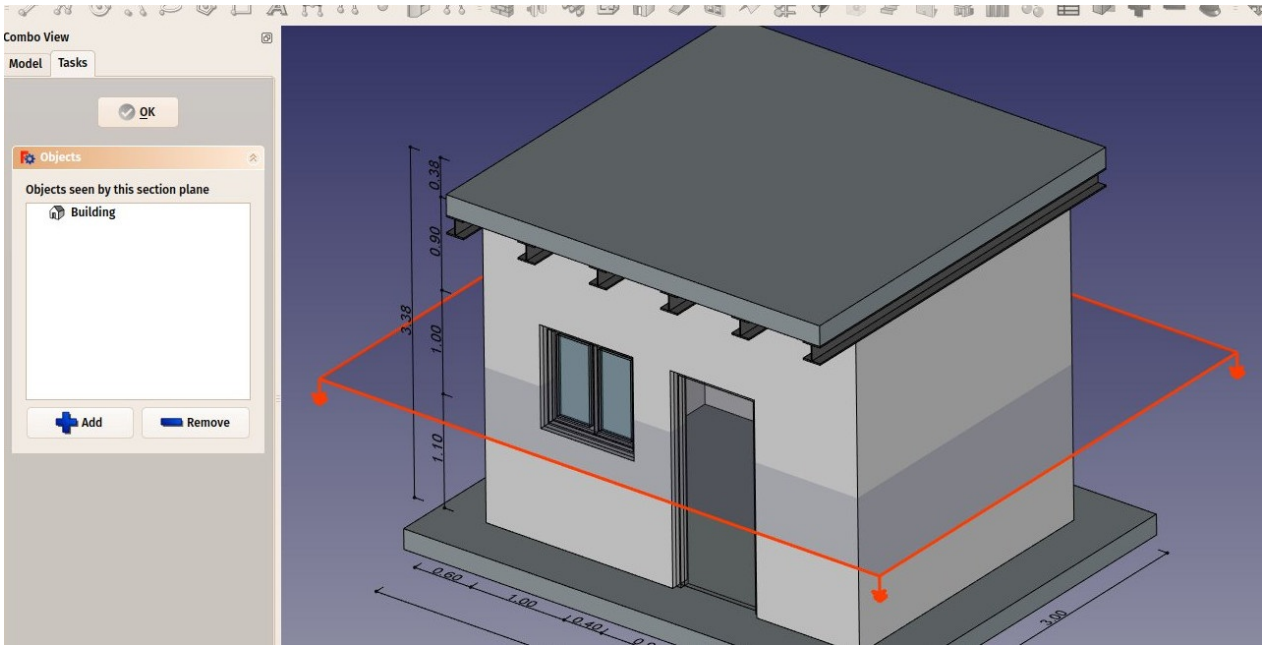


- Select all your dimensions, and drag them to the **Plan dimensions** group in the tree view
- Set the Working Plane to **XZ** plane, that is, the frontal vertical plane.
- Repeat the operation, draw a couple of dimensions, and place them in the **Elevation dimensions** group.

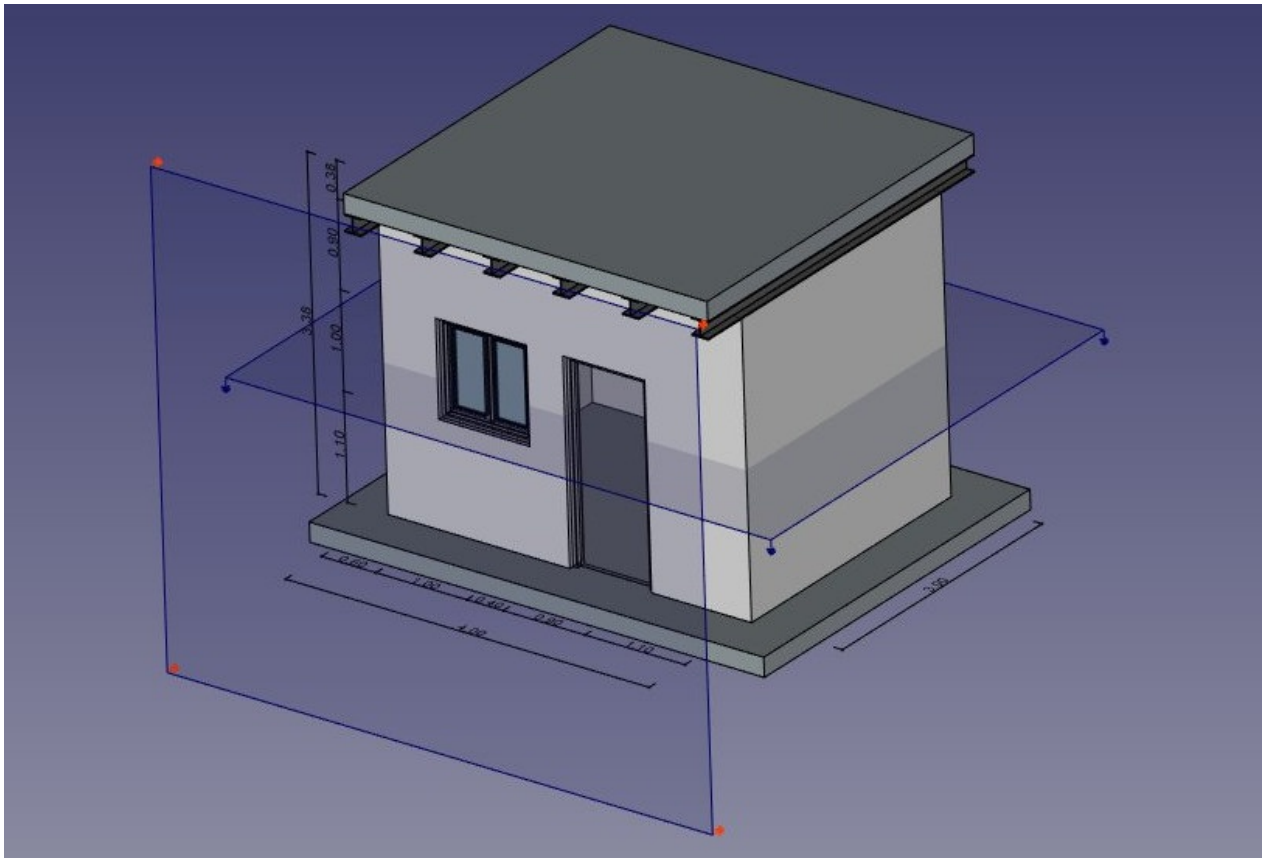


We will now prepare a set of views from our model, to be paced on a Drawing page. We can do that with the tools from the Drawing Workbench, as we have seen in the previous chapter, but the Arch Workbench also offers an all-in-one advanced tool to produce plan, section and elevation views, called Section Plane. We will now add two of these section planes, to create a plan view and an elevation view.

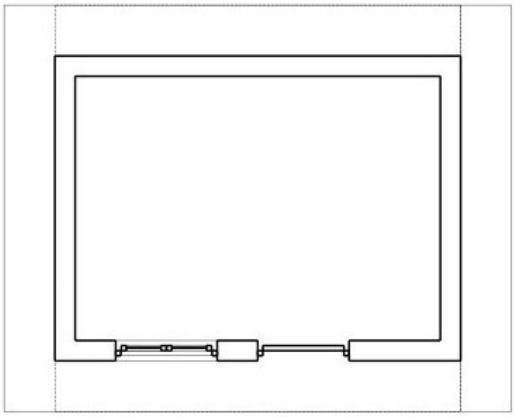
- lect the building object in the tree view
- Press the Section Plane button.
- Set its **Display Height** property to 5m, its **Display Length** to 6m, so we encompass our house (this is not needed, but will look better, as it will show naturally what it is used for), and its **Placement** position at x:2m, y:1.5m, z:1.5m.
- Check the list of objects considered by the Section Plane by double-clicking it in the tree view. Section Planes only render specified objects from the model, not all of them. The objects considered by the Section Plane can be changed here.

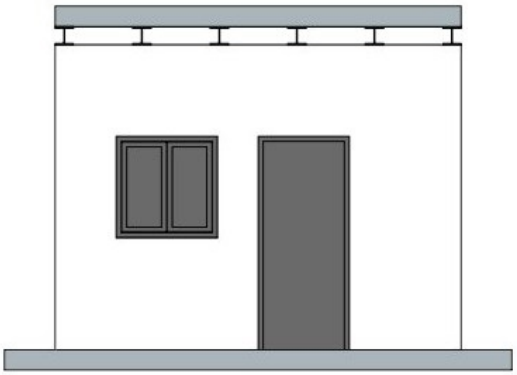


- Repeat the operation to create another section plane, give it the same display length and height, and give it the following **Placement**: position: x:2m, y:-2m, z:1.5m, angle: 90°, axis: x:1, y:0, z:0. Make sure this new section plane also considers the building object.



- Now we have everything we need, and we can create our Drawing page. Start by switching to the Drawing Workbench, and create a new default A3 page (or select another template if you wish).
- Select the first section plane, used for the plan view
- Press the Draft View button. This tool offers a couple of additional features over the standard Drawing View tool, and supports the Section Planes from the Arch Workbench.
- Give the new view the following properties:
  - X: 50
  - Y: 140
  - Scale: 0.03
  - Line width: 0.15
  - Show Cut: True
  - Show Fill: True
- Select the other section plane, and create a new Draft View, with the followng properties:
  - X: 250
  - Y: 150
  - Scale: 0.03
  - Rendering: Solid





We will now create two more Draft Views, one for each group of dimensions.

- Select the Plan dimensions group
- Press the Draft View button.
- Give the new view the following properties:
  - X: 50
  - Y: 140
  - Scale: 0.03
  - Line width: 0.15
  - Font size: 10mm
- Repeat the operation for the other group, with the following settings:
  - X: 250
  - Y: 150
  - Scale: 0.03
  - Line width: 0.15
  - Font size: 10mm
  - Direction: 0,-1,0
  - Rotation: 90°

Our page is now ready, and we can export it to SVG or DXF formats, or print it. The SVG format allows to open the file 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

- 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

Read more

- The Arch Workbench: http://www.freecadweb.org/wiki/index.php?title=Arch_Module
- The Draft working plane: http://www.freecadweb.org/wiki/index.php?title=Draft_SelectPlane
- The Draft snapping settings: http://www.freecadweb.org/wiki/index.php?title=Draft_Snap
- The expressions system: http://www.freecadweb.org/wiki/index.php?title=Expressions
- The IFC format: https://en.wikipedia.org/wiki/Industry_Foundation_Classes
- IfcOpenShell: http://ifcopenshell.org/
- IfcPlusPlus: http://ifcplusplus.com/
- Inkscape: http://www.inkscape.org

Using spreadsheets

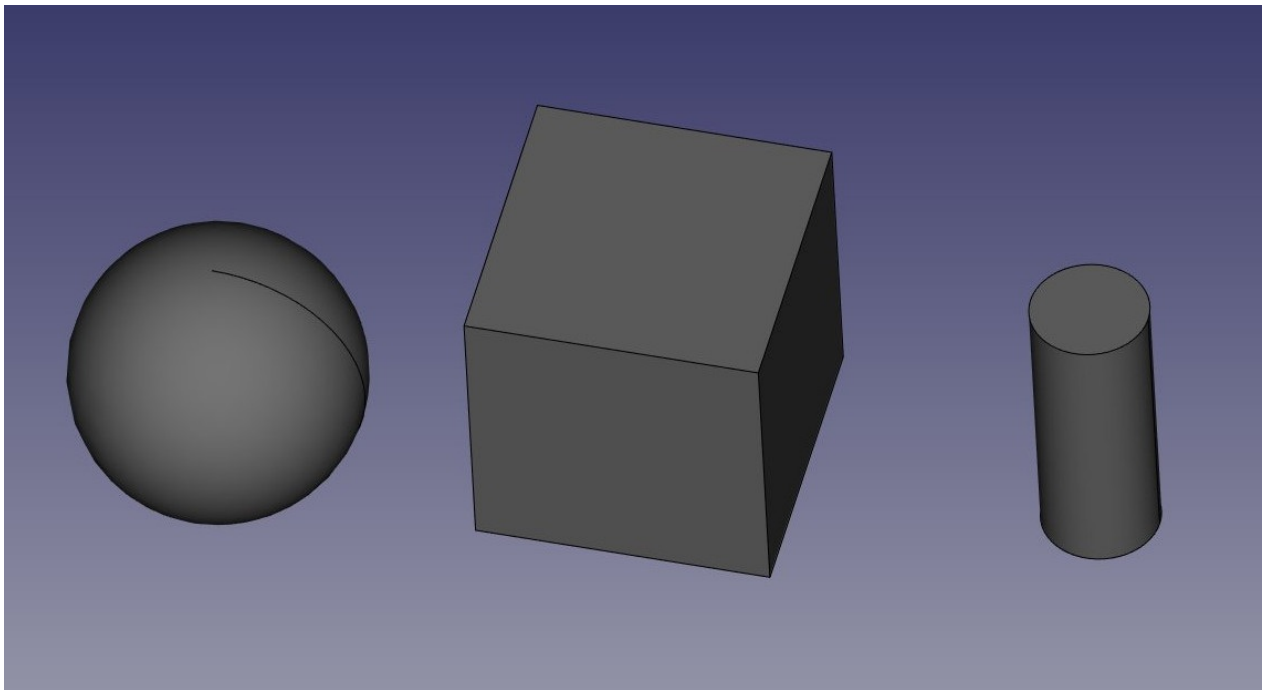
FreeCAD features another interesting workbench to explore: the Spreadsheet Workbench. This workbench allows to create spreadsheets such as those made with Excel or LibreOffice directly in FreeCAD. These spreadsheets can then be populated with data extracted from your model, and can also perform a series of calculations between values. Spreadsheets can be exported as CSV files, which can be imported in any other spreadsheet application.

In FreeCAD, however, spreadsheets have an additional utility: Their cells can receive a name, and can then be referenced by any field supported by the expressions engine. This turns spreadsheets into powerful control structures, where the values inserted in specific cells can drive dimensions of the model. There is only one thing to keep in mind, as FreeCAD prohibits circular dependencies between objects, a same spreadsheet cannot be used to set a property of an object and at the same time retrieve a property value from the same object. That would make the spreadsheet and the object depending on each other.

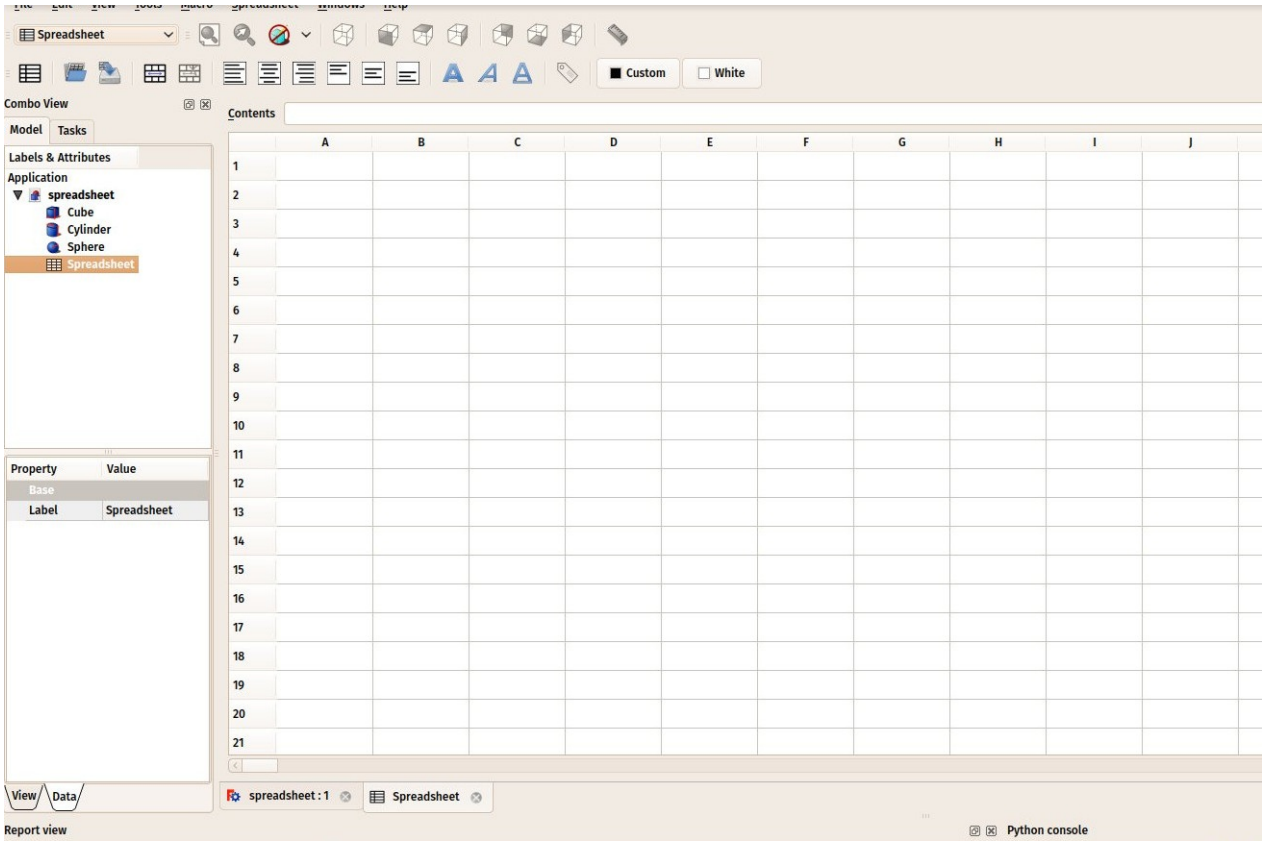
In the following example, we will create a couple of objects, retrieve some of their properties in a spreadsheet, then use the spreadsheet to directly drive properties of other objects. se

Reading properties

- 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 watch better the effects of what we'll do:



- Now, It'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:



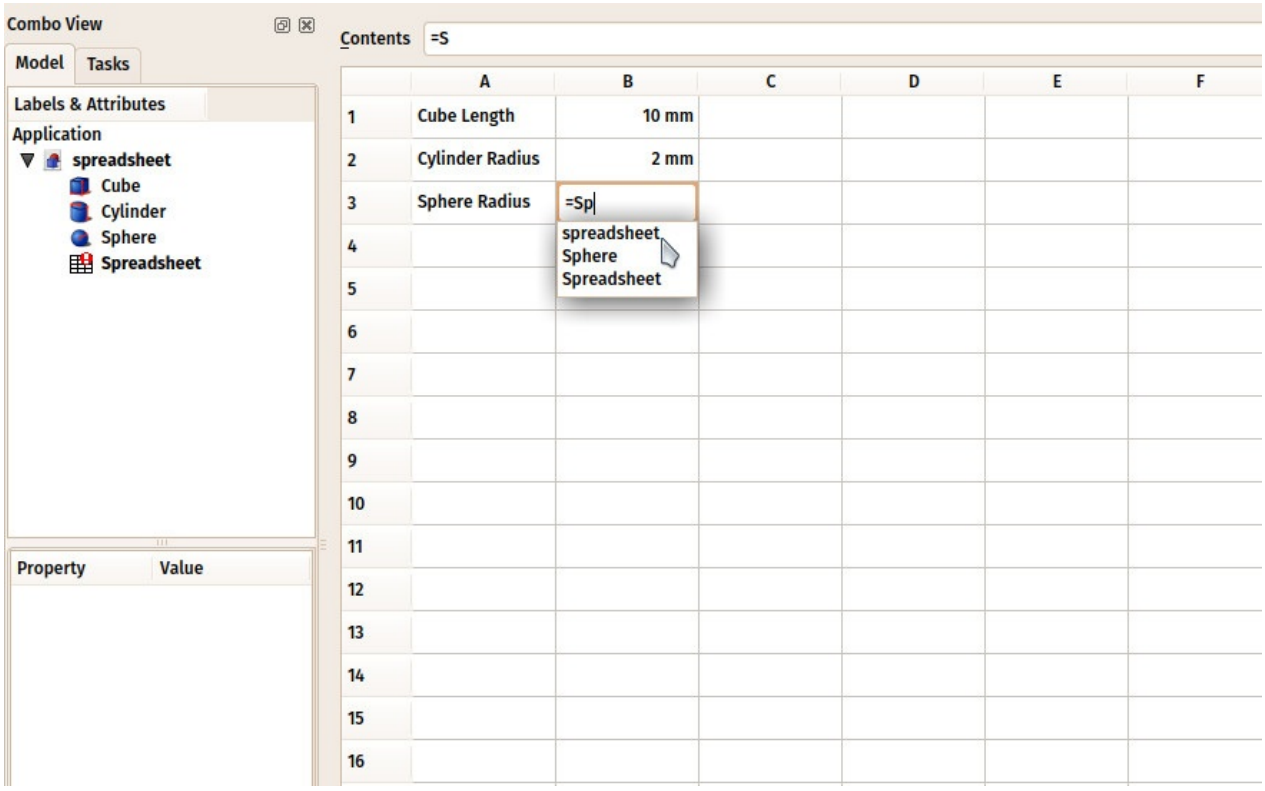
The spreadsheet editor of FreeCAD, although it is not as complete and powerful as the more complete spreadsheet applications we listed above, has nevertheless most of the basic tools and functions that are commonly used, such as the possibility to change the aspect of the cells (size, color, alignment), join and split cells, use formulas such as **=2+2**, or reference other cells with **=B1**.

In FreeCAD, to these common behaviours, has been added one very interesting: The possibility to reference not only other cells, but other objects from the document, and retrieve values from their properties. For example, let's retrieve a couple of properties from the 3 objects we created above. Properties are what we can see in the properties editor window, under the **Data** tab, when an object is selected.

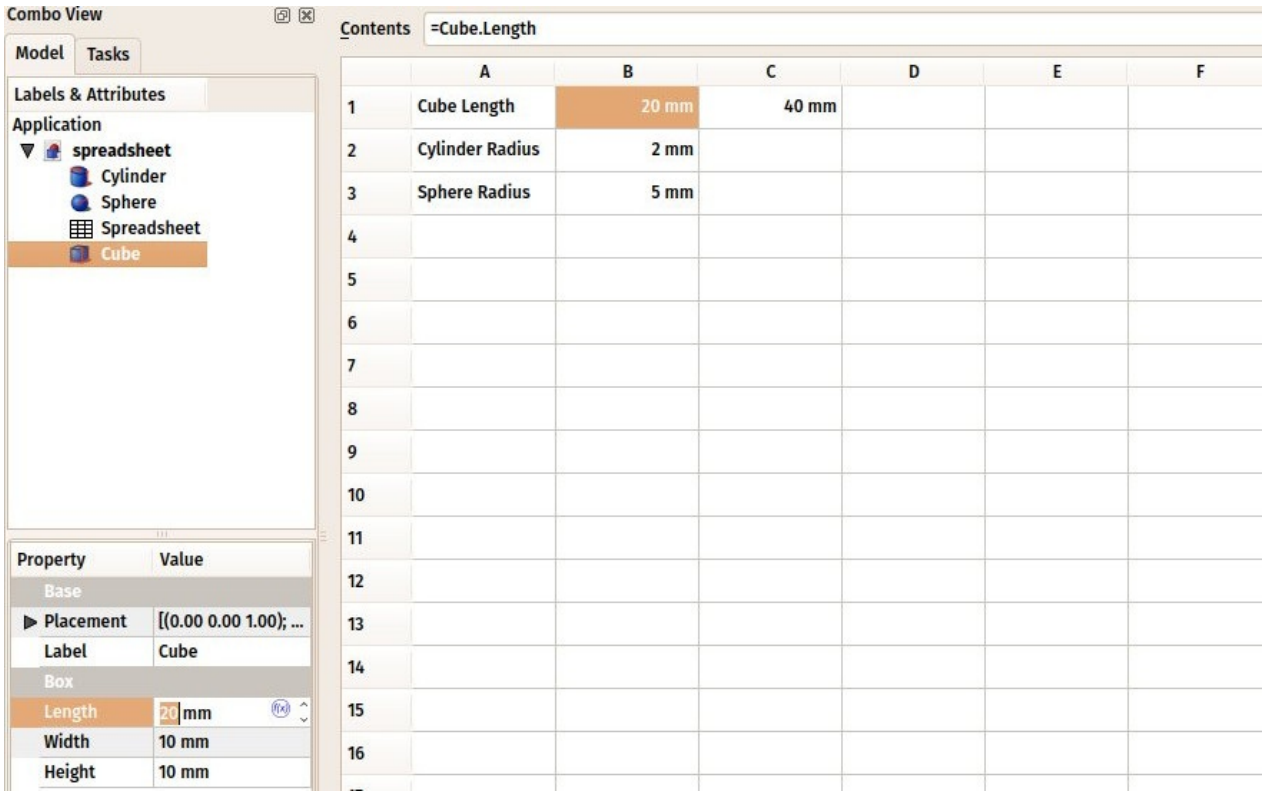
- Let's start by entering a couple of texts in the cells A1, A2 amd A3, so we remember what is what later on, for example **Cube Length**, **Cylinder Radius** and **Sphere Radius**. To enter text, just write in the "Contents" filed above the spreadsheet, or double-click a cell.
- Now let's retrieve the actual length of our cube. In cell B1, type **=Cube.Length**. You will notice that the spreadhseet has an autocompletion 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 propertties editor, and the change will be immediately reflected in the spreadsheet. For example, let's change the length of our cube to **20mm**:



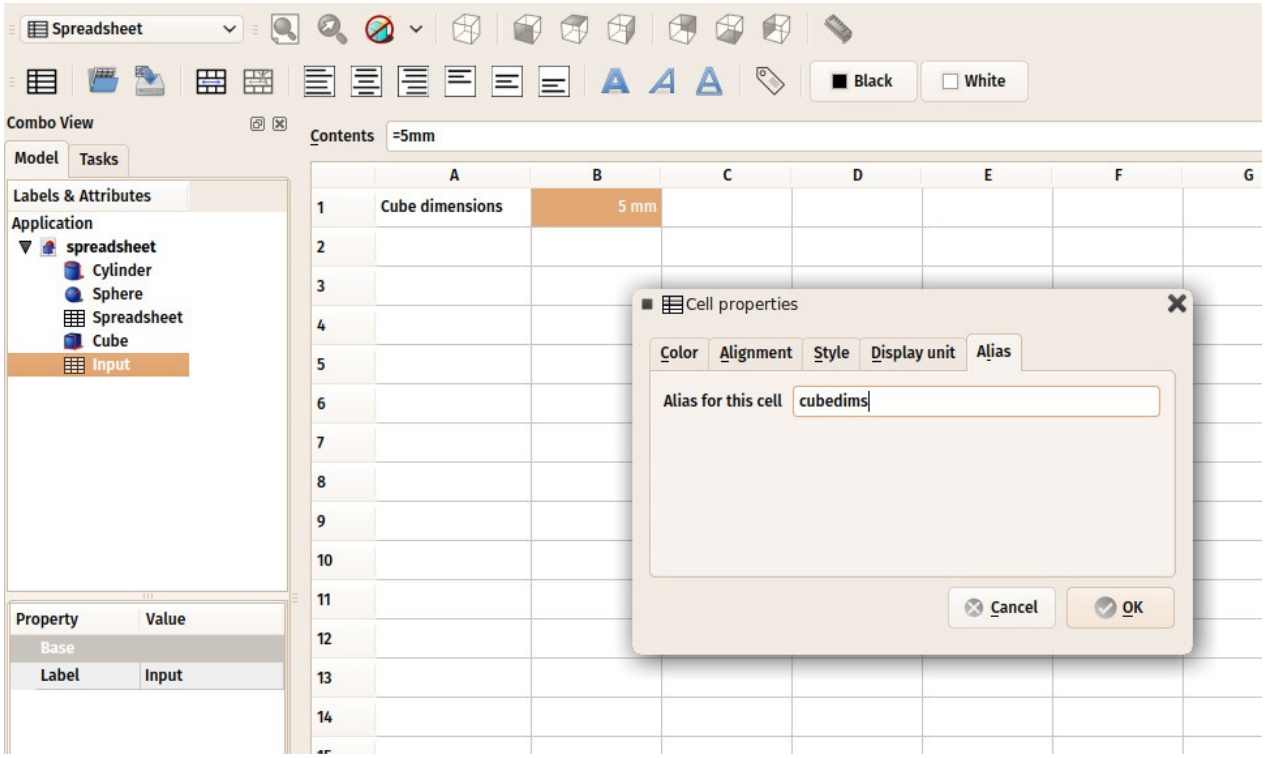
The Spreadsheet Workbench page will describe more in detail all the possible operations and functions that you can use in spreadsheets.

Writing properties

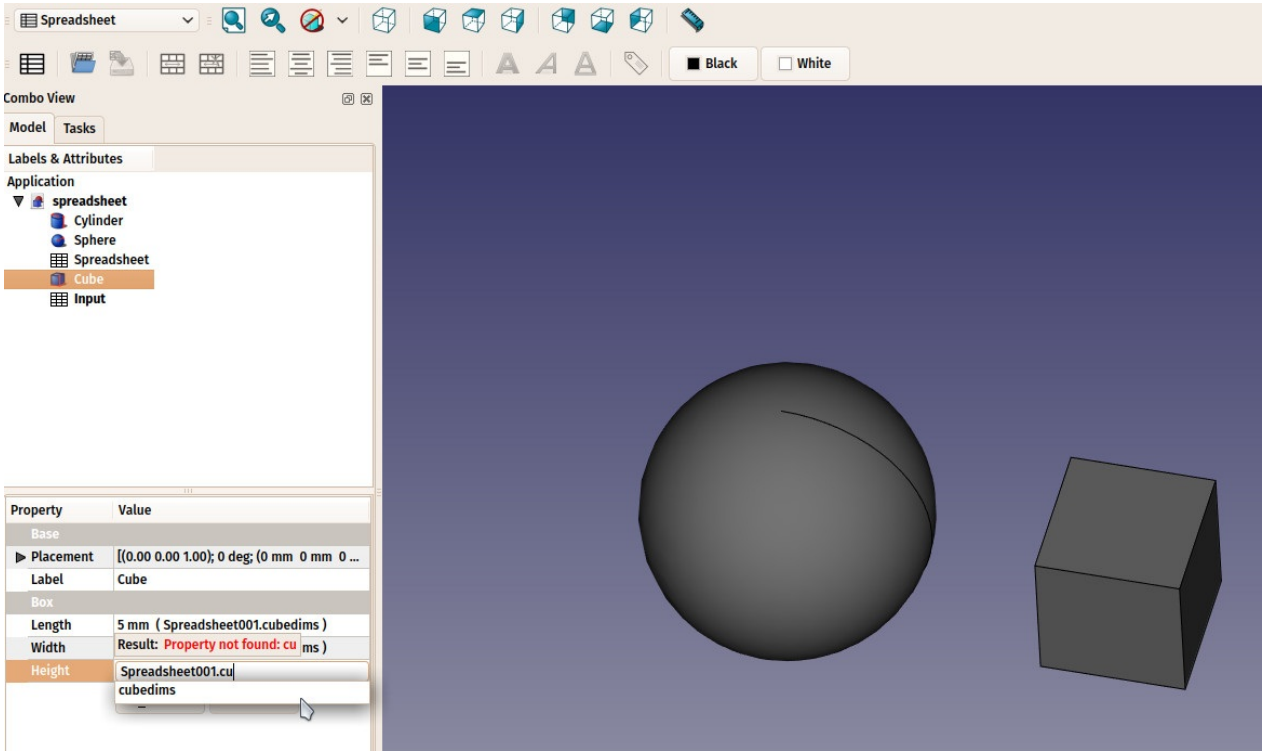
Another very interesting use of the Spreadsheet Workbench in FreeCAD is to do the contrary of what we have been doing until now: Instead of reading the values of properties of 3D objects, we can also assign values to these objects. Remember, however, one of the fundamental rules of FreeCAD: Circular dependencies are forbidden. We can therefore not use the same spreadsheet to read **and** write values to a 3D object. That would make the object depend on the spreadsheet, which would also depend on the object. Instead, we will create another spreadsheet.

- 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, suchas **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 cells, click **Properties** and select the **Alias** tab. Give

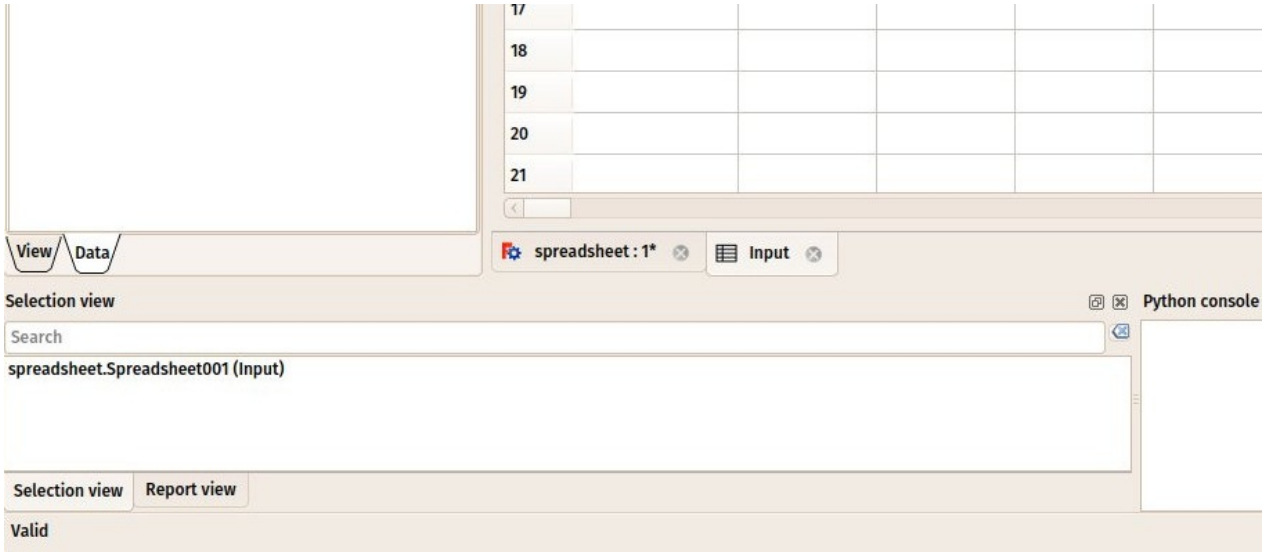
it a name, such as **cubedims**:



- 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 openthen expressions editor, where you can write **Spreadsheet001.cubedims**. Repeat this for Height and Width:

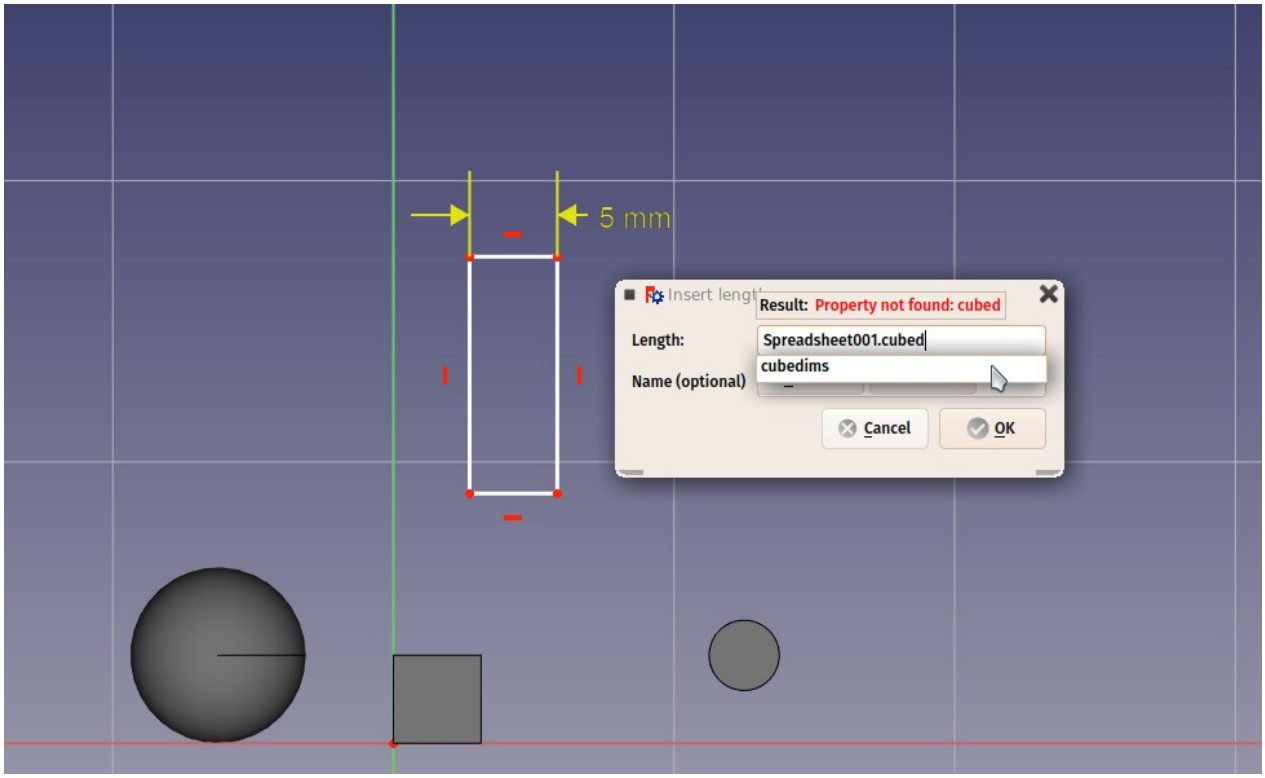


You might wonder why we had to use "Spreadsheet001" instead of "Input" in the expression above. This is beacause each object, in a FreeCAD document, has an **internal name**, which is unique in the document, and a **label**, which is what appears in the tree view. If you uncheck the appropriate option in the preferences settings, FreeCAD will allow you to give the same label to more than one object. This is why all operations that must identify an object with absolutely no doubt, will use the internal name instead of the label, which could designate more than one object. The easiest way to know the internal name of an object is by keeping the **selection panel** (menu Edit->Panels) open, it will always indicate the internal name of a selected object:

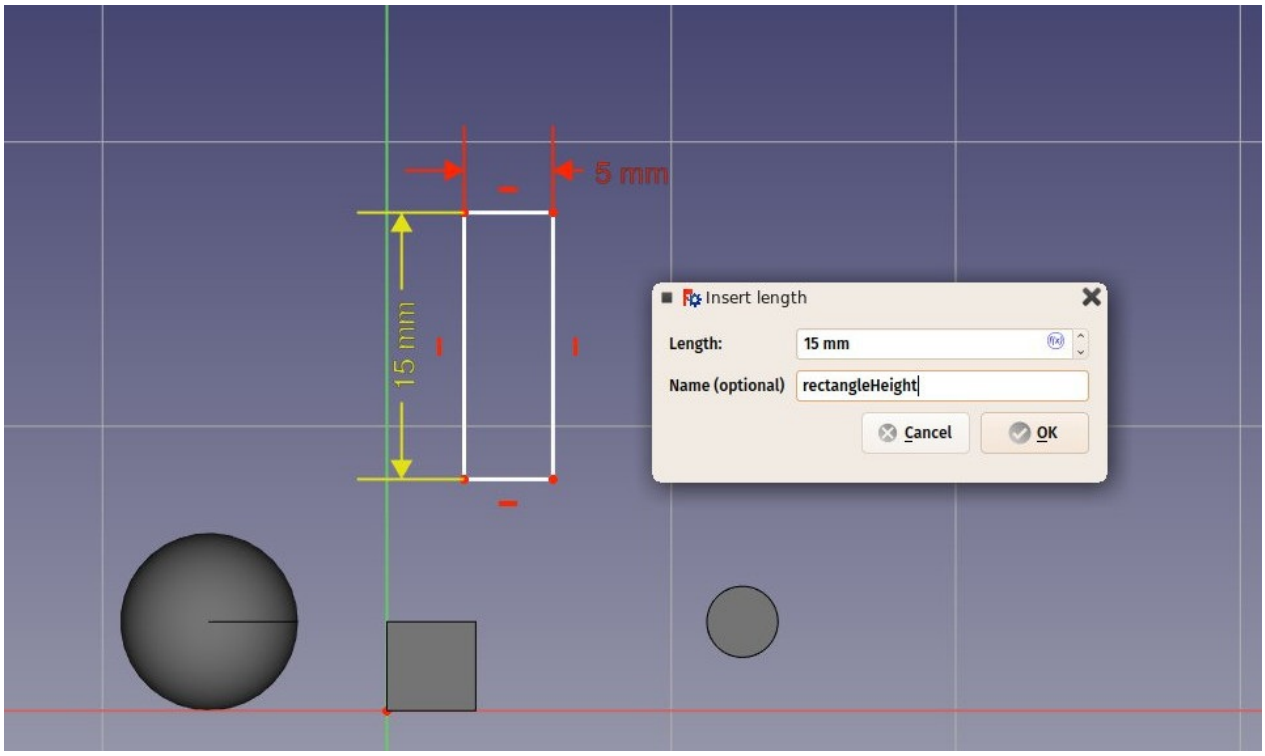


By using cell aliases in spreadsheets, we are able to use a spreadsheet to store "master values" in a FreeCAD document. This can be used, for example, to have a model of a piece of certain dimensions, and to store these dimensions in a spreadsheet. It becomes then very easy to produce another model with different dimensions, it is just a matter of opening the file and changing a couple of dimensions in the spreadsheet.

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

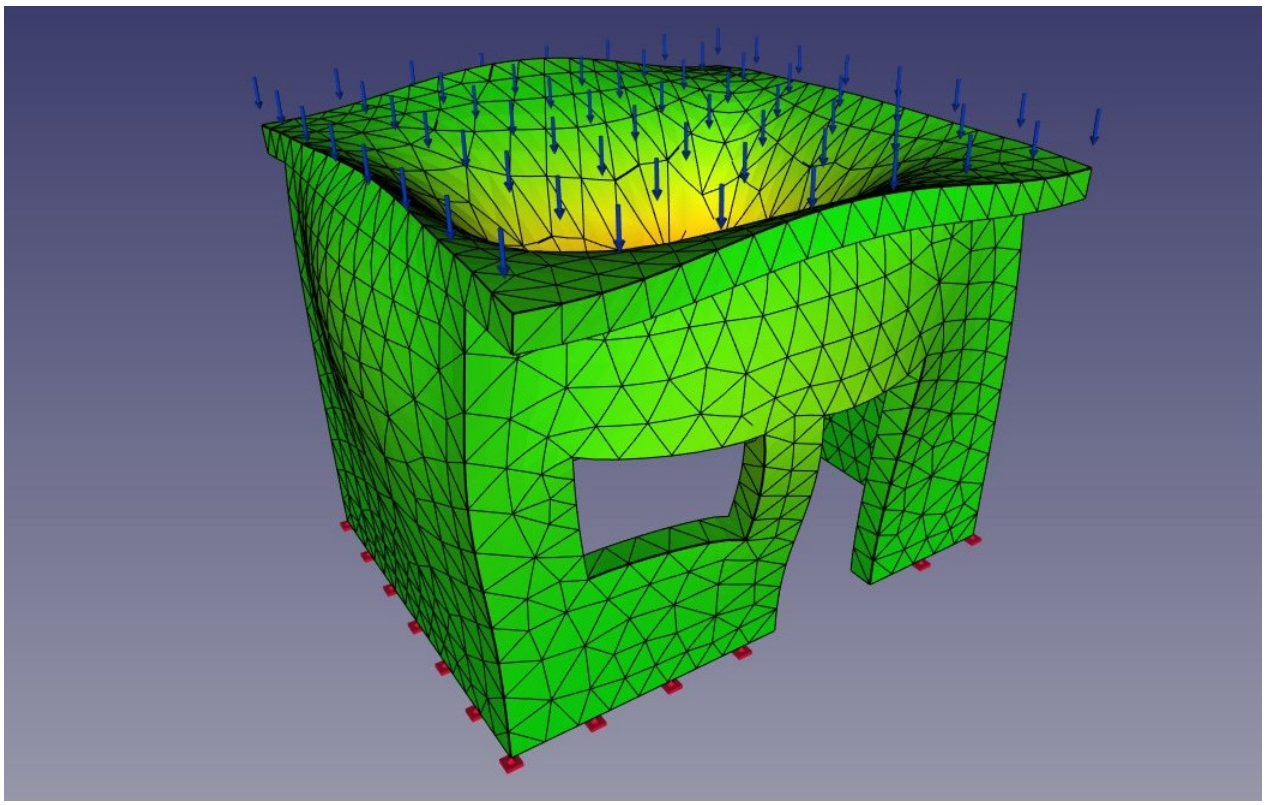
Read more

- The Spreadsheet Workbench: http://www.freecadweb.org/wiki/index.php?title=Spreadsheet_Module
- The Expressions engine: http://www.freecadweb.org/wiki/index.php?title=Expressions

Creating FEM analyses

FEM stands for Finite Element Method. It is a vast mathematical subject, but in FreeCAD we can resume it as a way to calculate propagations inside a 3D object, by cutting it into small pieces, and analyzing the impact of each small piece over its neighbours. This has several uses in the engineering and electomagnetism fields, but we will look here more in depth at one use that is already well developed in FreeCAD, which is simulating deformations in objects which are submitted to forces and weights.

Obtaining such simulation is done in FreeCAD with the FEM Workbench. It involves different steps: Preparing the geometry, setting its material, performing the meshing (division into smaller parts, like we did in the Preparing objects for 3D printing chapter), and finally calculating the simulation.



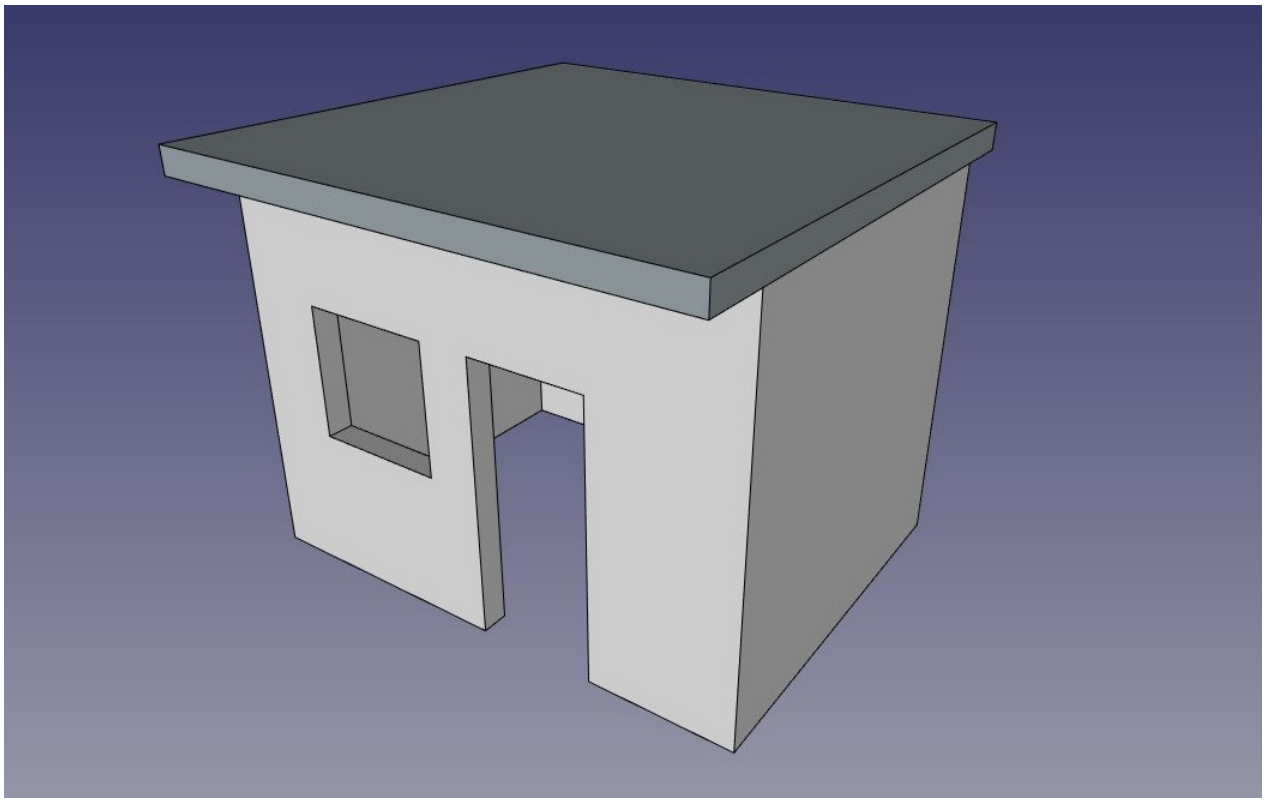
Preparing FreeCAD

The simulation itself is done by another piece of software, that is used by FreeCAD to obtain the results. As there are several interesting open-source FEM simulation applications available, the FEM Workbench has been made to be able to use more than one. However, currently only CalculiX is fully implemented. Another piece of software, called NetGen, which is responsible for generating the subdivision mesh, is also required. Detailed instructions to install these two components are provided in the FreeCAD documentation.

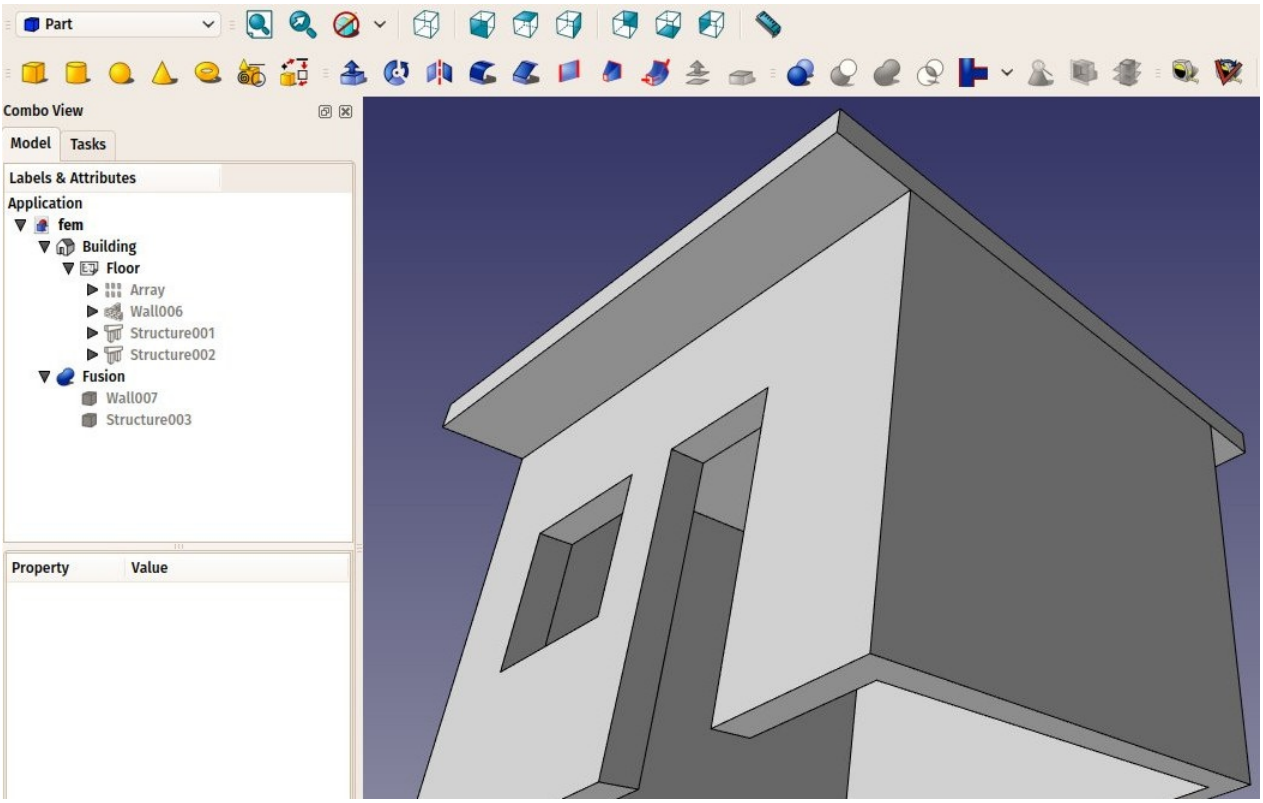
Preparing the geometry

We will start with the house we modelled in the BIM modeling chapter. However, some changes have to be made to make the model suitable for FEM calculations. This involves, basically, discarding the objects that we don't want to include in the calculaiton, such as the door and window, and joining all the remaining objects into one.

- Load the house model we modeled earlier
- Delete or hide the page object, the section planes and the dimensions, so we stay only with our model
- Hide the window, the door and the ground slab
- Also hide the metal beams from the roof. Since they are very different objects from the rest of the house, we will simplify our calculation by not including it. Instead, we will consider that the roof slab is directly placed on top of the wall.
- Now move the roof slab down so it rests on top of the wall: Edit the **Rectangle** object that we used as a base of the roof slab, and change it's **Placement->Position->X** value from 3.18m to 3.00m
- Our model is now clean:

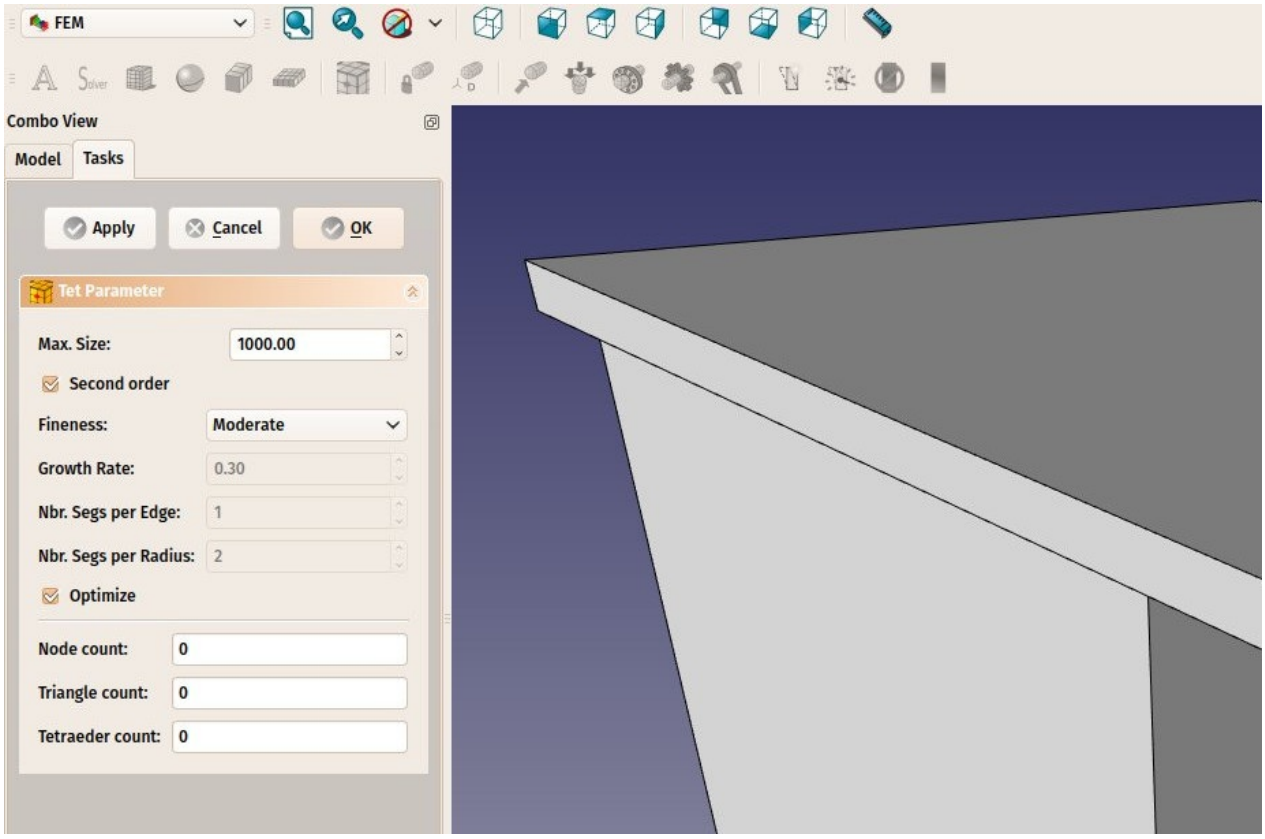


- The FEM Workbench can currently calculate deformations on one single object only. Therefore, we need to join our two objects (the wall and the slab). Switch to the Part Workbench, select the two objects, and press the Fuse. We now have obtained one fused object:

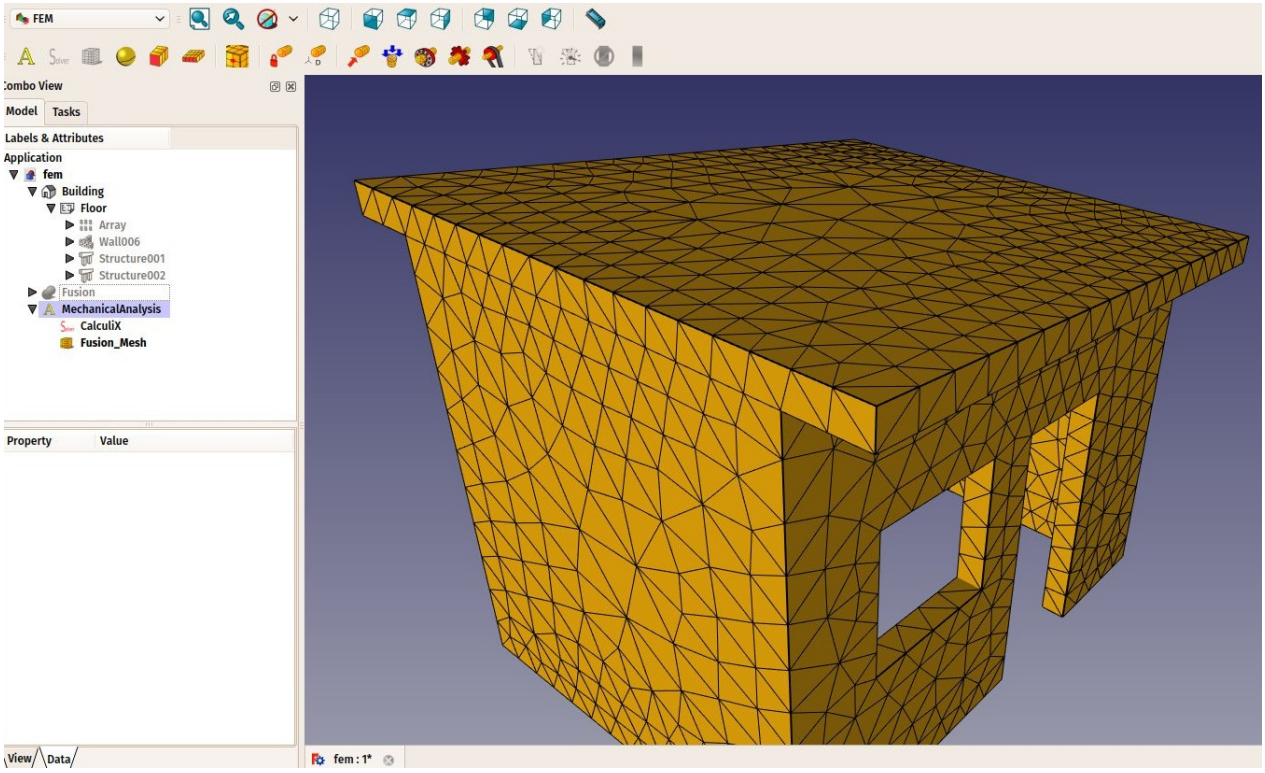


Creating the analysis

- We are now ready to start a FEM analysis. Let's switch to the FEM Workbench
- Select the fusion object
- Press the New Analysis button
- A new analysis will be created and a settings panels opened. Here you can define the meshing parameters to be used to produce the FEM mesh. The main setting to edit is the **Max Size** which defines the maximum size (in millimeters) of each piece of the mesh. For now, we can leave the default value of 1000:

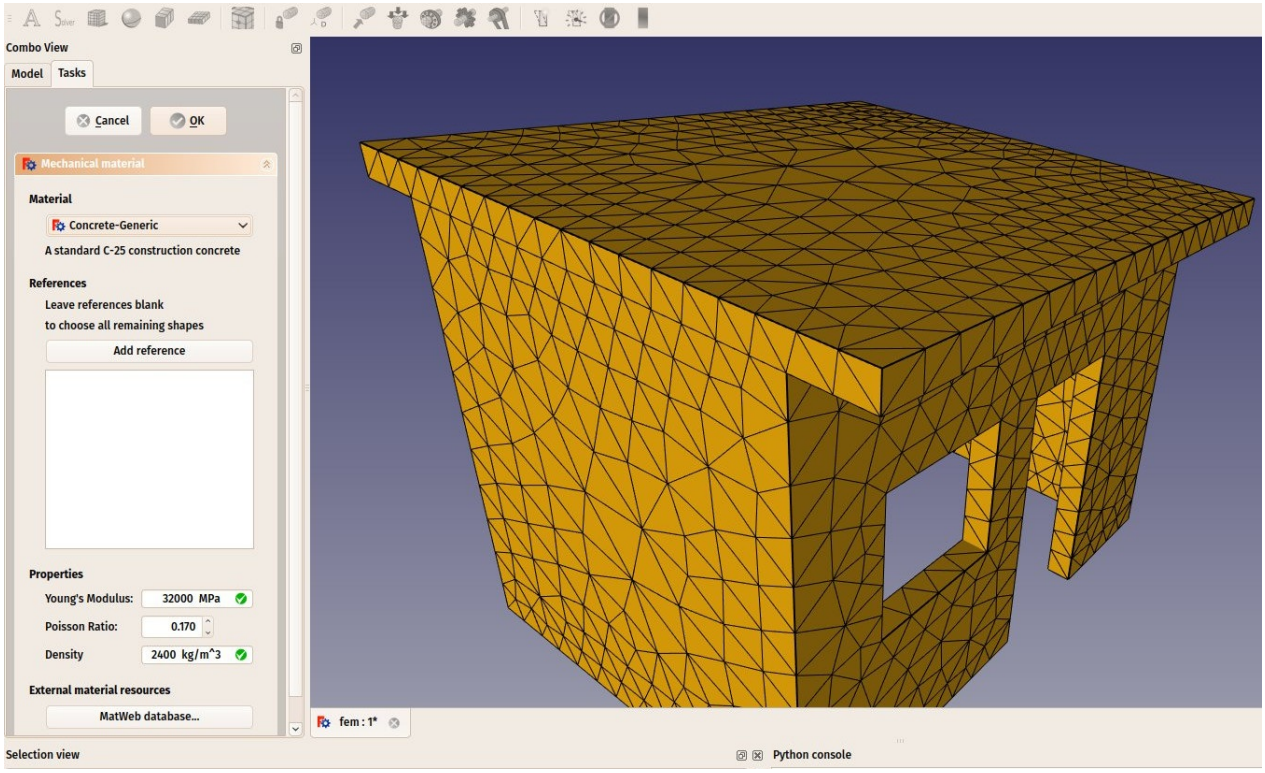


- After pressing OK and a few seconds of calculaiton, our FEM mesh is now ready:

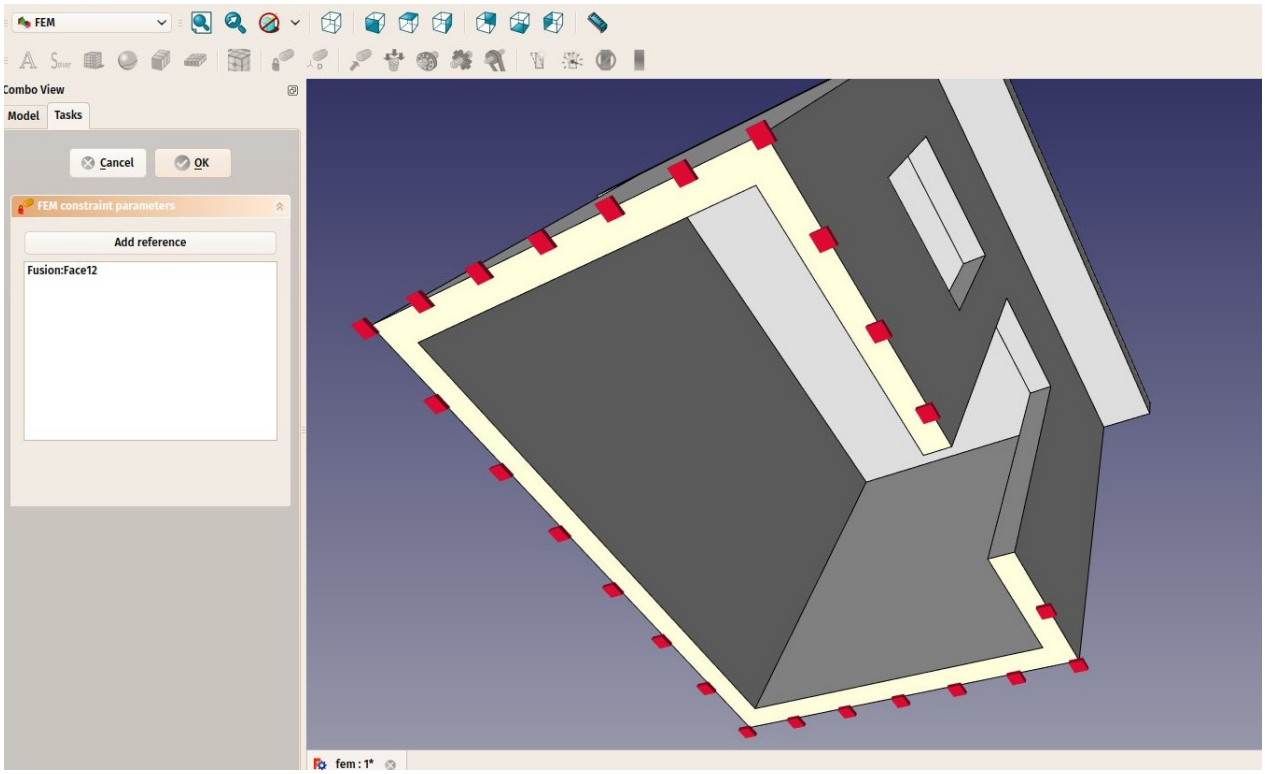


- We can now define the material to be applied to our mesh. This is important because depending on the material strength, our object will react differently to forces applied to it. Select the analysis object, and press the New Material button.
- A task panel will open to allow us to choose a material. In the Material drop-down list,

choose the **Concrete-generic** material, and press OK.



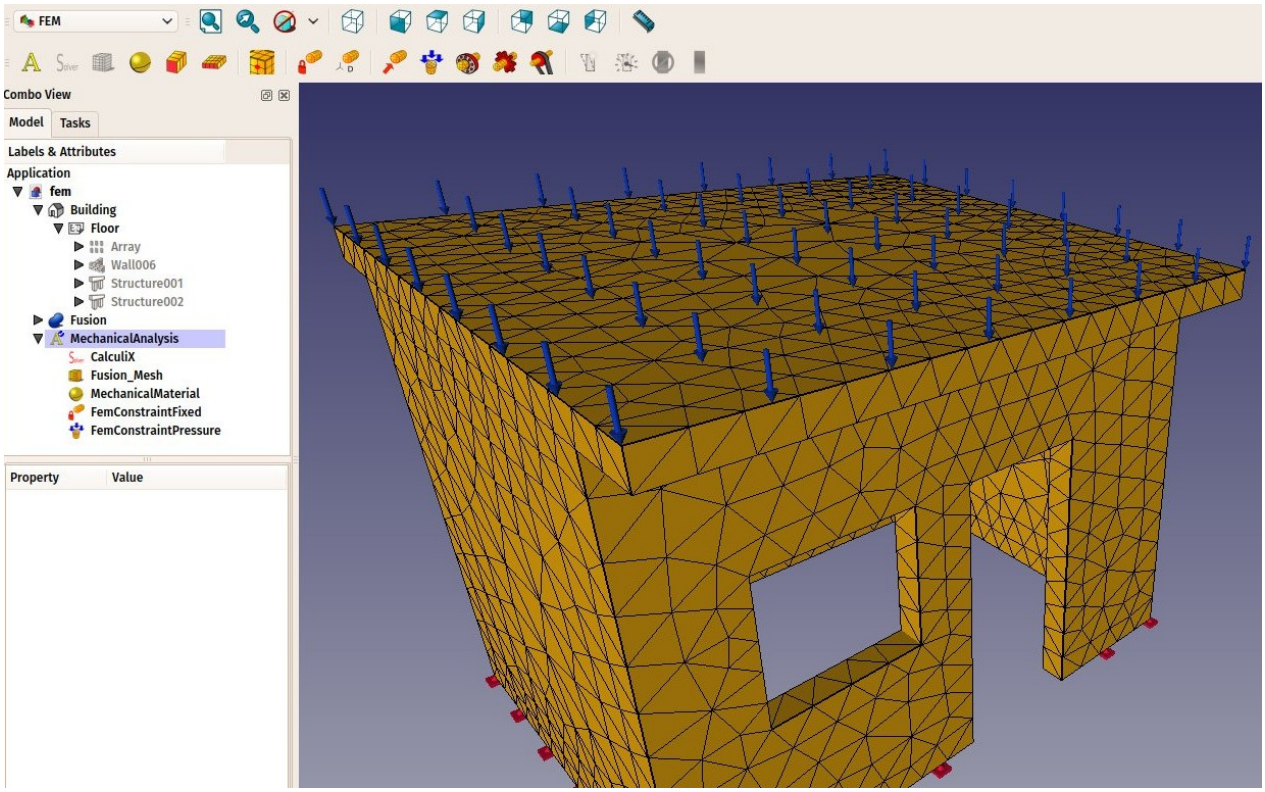
- We are now ready to apply forces. Let's start by specifying which faces are fixed into the ground and can therefore not move. Press the Fixed Constraint button.
- Click on the bottom face of our building and press OK. The bottom face is now indicated as unmovable:



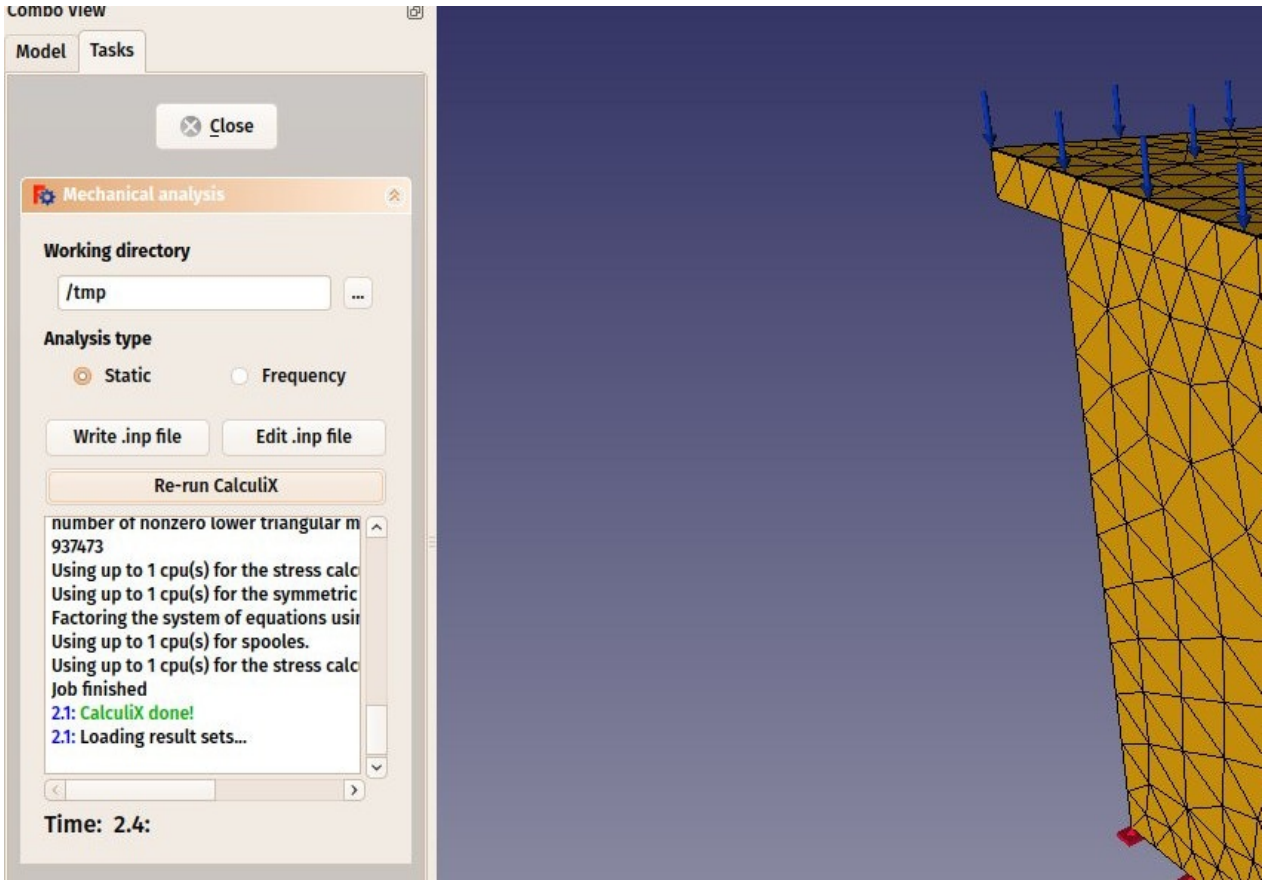
- We will now add a load on the top face, that could represente, for example, a massive

weight being spread on the roof. For this we will use a pressure constraint. Press the Pressure Constraint button.

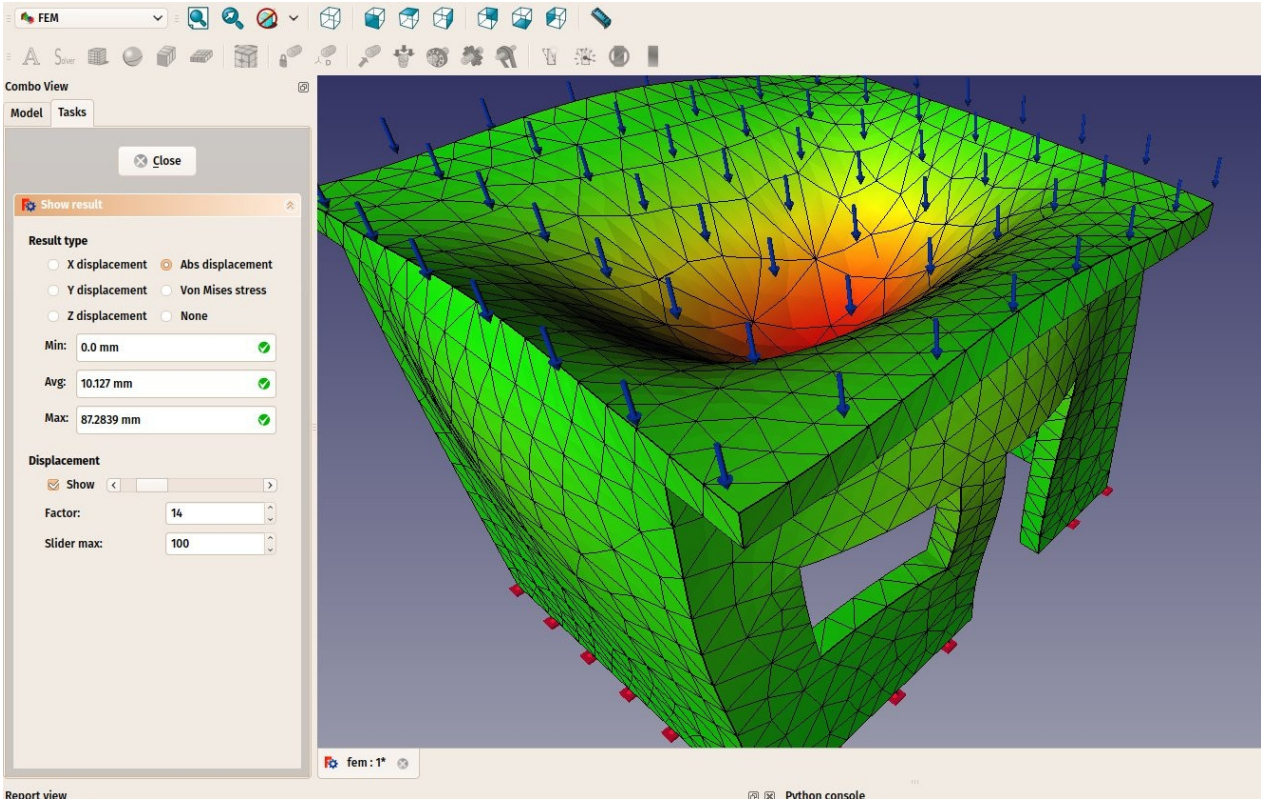
- Click the top face of the roof, set the pressure to **10MPa** (the pressure is applied by square millimeter) and click the OK button. Our force is now applied:



- We are now ready to start the calculation. Select the **CalculiX** object in the tree view, and press the Start Calculation button.
- In the task panel that will open, click first the **Write .inp file** button to create the input file for CalculiX, then the **Run CalculiX** button. A few moments later, the calculation will be done:



- We can now look at the results. Close the task panel, and see that a new **Results** object has been added to our analysis.
- Double-click the Results object
- Set the type of result that you want to see on the mesh, for example "absolute displacement", tick the **show** checkbox under **Displacement**, and 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 dimensionning 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 bear the more stress.

Downloads

- The file created during this exercise: https://github.com/yorikvanhavre/FreeCAD-manual/blob/master/files/fem.FCStd

Read more

- The FEM Workbench: http://www.freecadweb.org/wiki/index.php?title=Fem_Workbench
- Installing required FEM components: http://www.freecadweb.org/wiki/index.php?title=FEM_Install
- CalculiX: http://www.calculix.de/
- NetGen: https://sourceforge.net/projects/netgen-mesher/

Creating renderings

In computer talk, a rendering is a word used to describe a nice image produced from a 3D model. Of course, we could say that what se see in the FreeCAD 3D view is already nice. But anybody who saw a recent Hollywood movie knows that it is possible to produce images with a computer that are almost undistinguishable from a photograph.

Of course, producing such photo-realistic images requires a lot of work, and a 3D application that offers specific tools for that, such as precise controls for materials and lighting. FreeCAD being an application more geared towards technical modeling, it doesn't feature any advanced rendering tool.

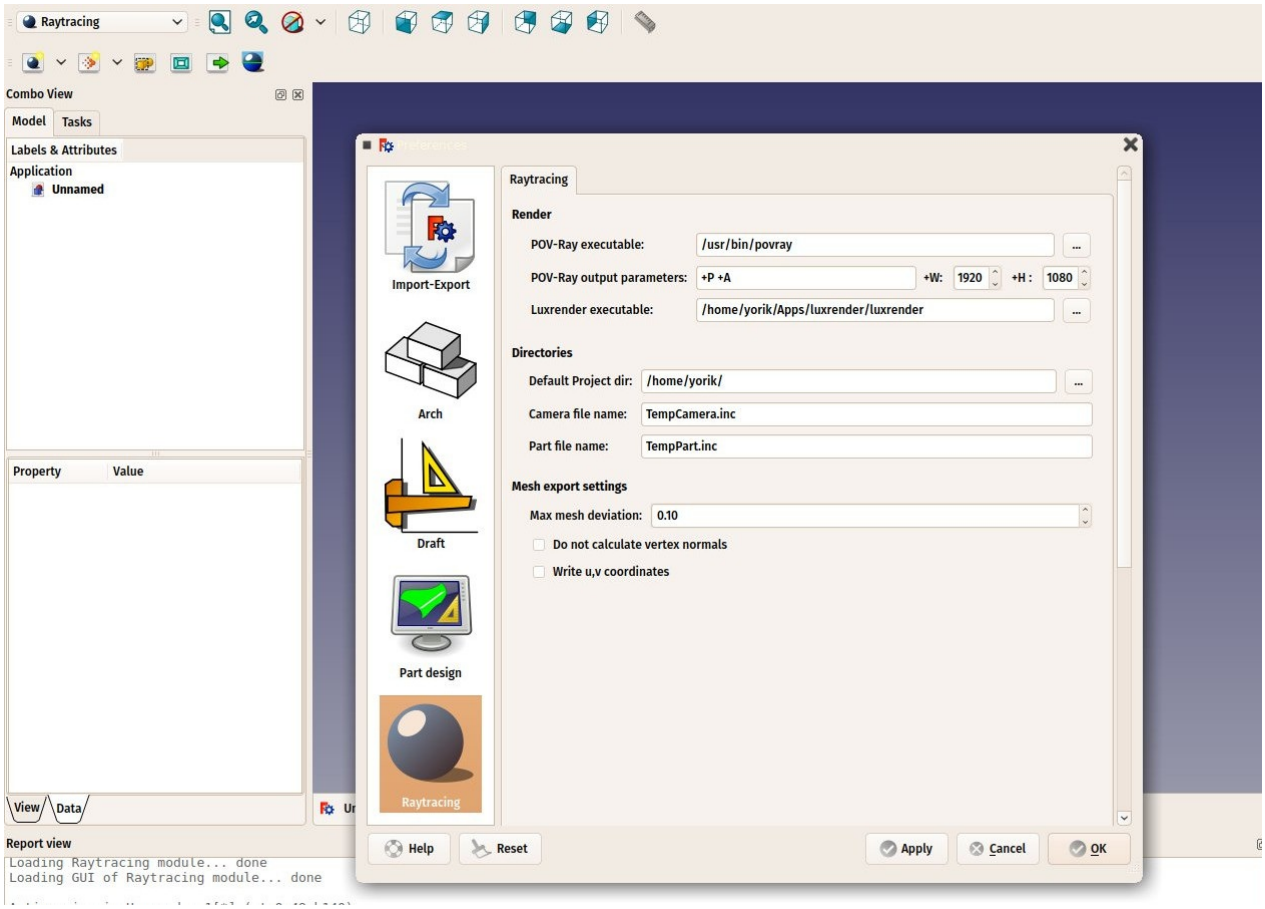
Fortunately, the opens-source world offers many applications to produce realistic images. Probably the most famous one is Blender, which is very popular and widely used in the movies and gaming industries. 3D models can very easily and faithfully be exported from FreeCAD and imported into Blender, where you can add realistic materials and illumination, and produce the final images or even animations.

Some other open-source rendering tools are made to be used inside another application, and will take care of doing the complex calculations to produce realistic images. Through its Raytracing Workbench, FreeCAD can use two of these rendering tools: POV-Ray and Luxrender. POV-Ray is a very old project, and is considered a classical raytracing engine, while Luxrender is much newer, and is categorized as an unbiased renderer. Both have their strengths and weaknesses, depending on the type of image one wants to render. The best way to know is to look at examples on both engines websites.

Installation

Before being able to use the Raytracing Workbench in FreeCAD, one of these two rendering applications needs to be installed on your system. This is usually very straightforward, both provide installers for many platforms or are usually included in the software repositories of most Linux distributions.

Once POV-Ray or Luxrender 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 simply searching your system for files named povray (or povray.exe on Windows) and luxrender (or luxrender.exe on Windows).

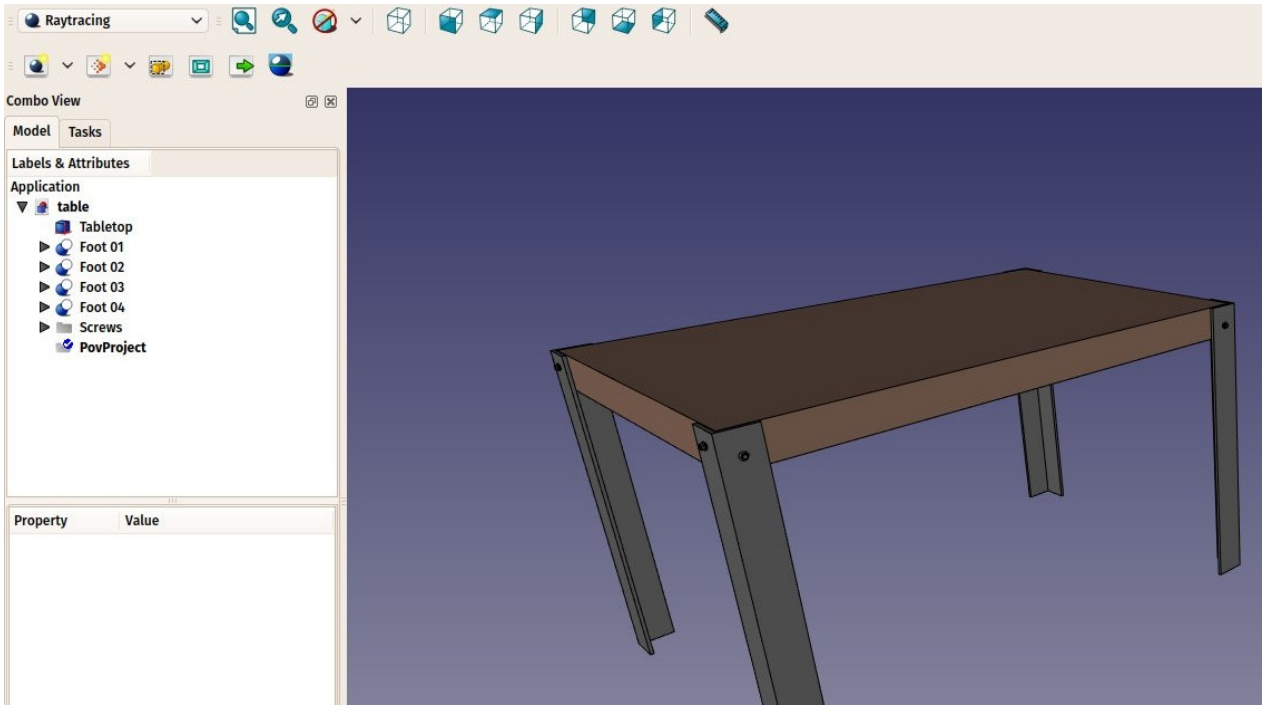


In this preferences screen we can also set the desired image size we want to produce.

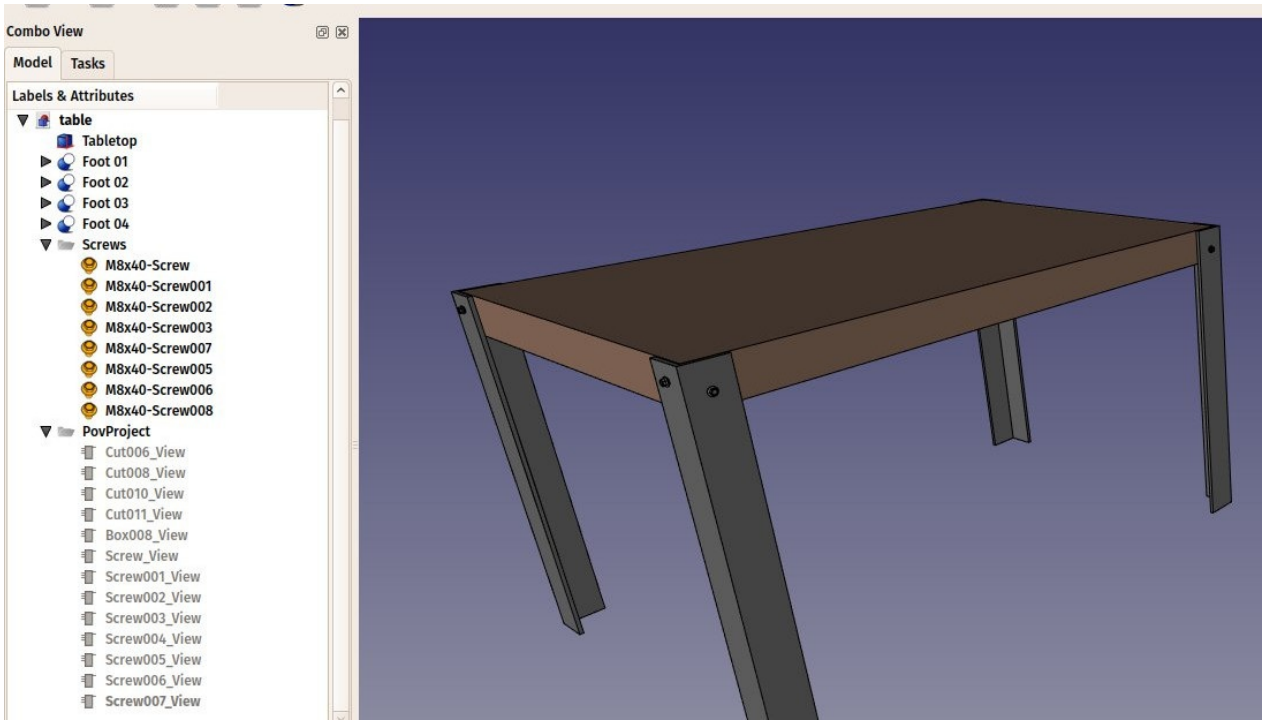
Rendering with PovRay

We will use the table we have been modeling in the tradtional modeling chapter to produce renderings with PovRay and Luxrender.

- Start by loading the table.FCStd file that we modeled earlier or from the link at the bottom of this chapter.
- Press the small down arrow next to the New Povray project button, and choose the **RadiosityNormal** template
- A warning message might appear telling you that the current 3D view is not in perspective mode and the rendering will therefore differ. Correct this by choosing **No**, choosing menu **View->Perspective view** and choosing the RadiosityNormal template again.
- You might also try other templates after you created a new project, simply by editing its **Template** property.
- A new project has now been created:

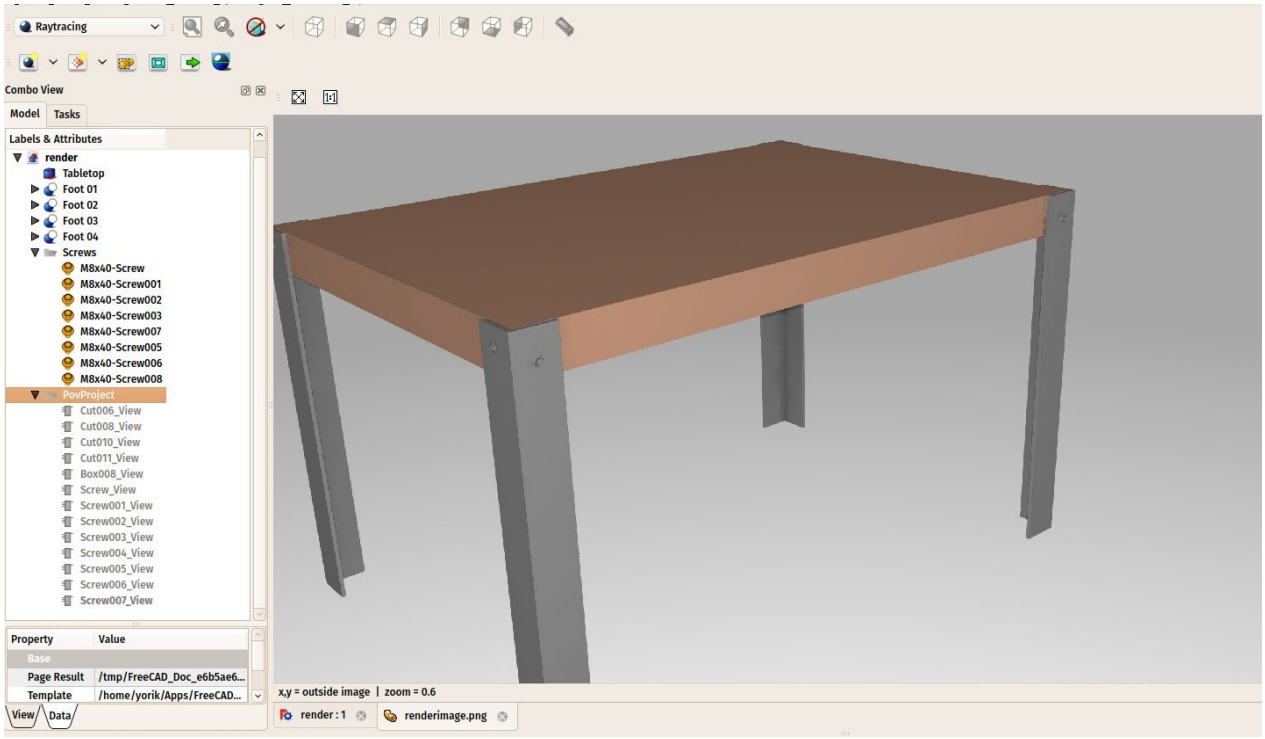


- The new project has adopted the point of view of the 3D view as it was at the moment we pressed the button. We can change the view, and update the view position stored in the povray project anytime, by pressing the Reset camera button.
- The Raytracing Workbench works the same way as the Drawing Workbench: Once a project folder is created, we must add **Views** of our objects to it. We can now do that by selecting all the objects that compose the table, and press the Insert part button:



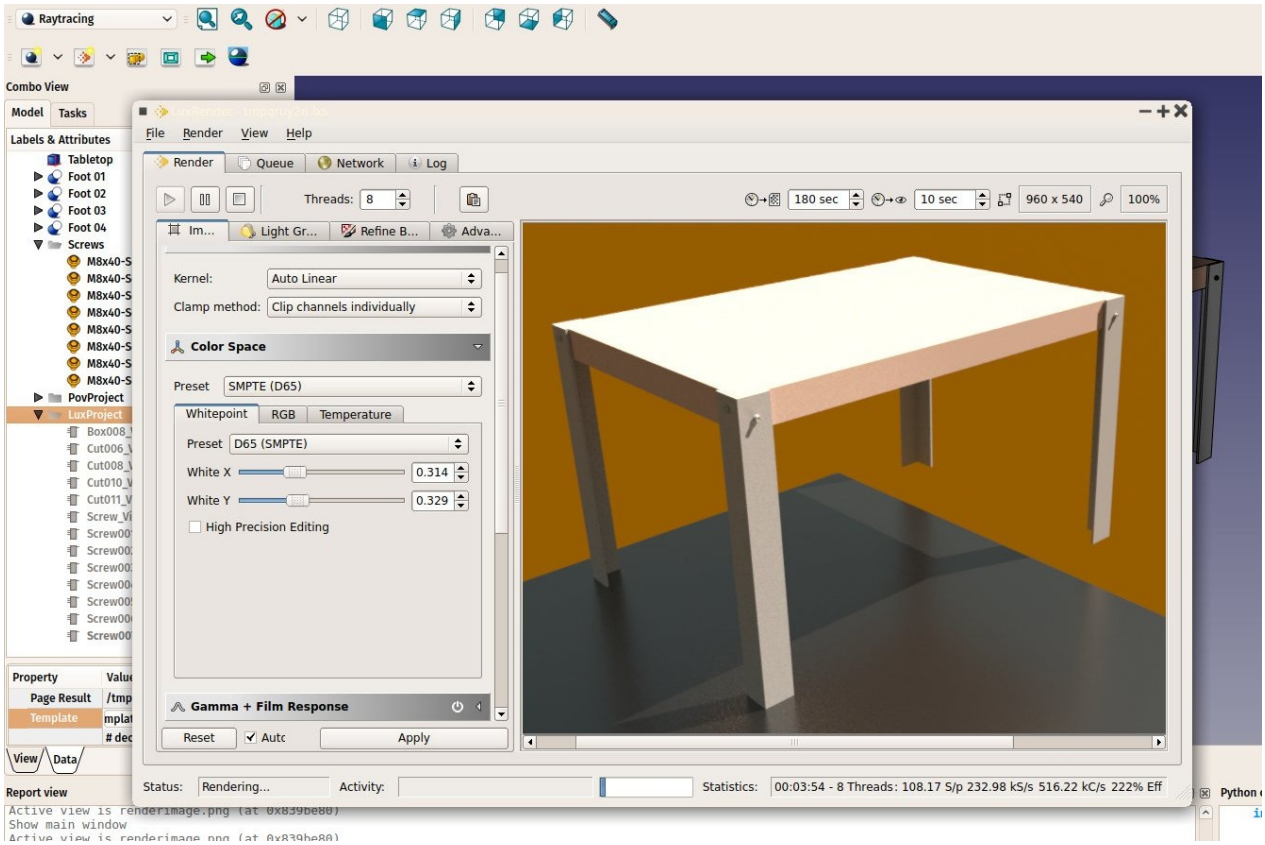
- The views have taken the color and transparency values from their original parts, but you can change that in the properties of each individual view if you wish.
- We are now ready to produce our first povray render. Press the Render button.

- You will be asked to give a file name and path for the .png image that will be saved by povray.
- The povray will then open and calculate the image.
- When this is done, simply click the image to close the povray window. The resulting image will be loaded in FreeCAD:



Rendering with LuxRender

- Rendering with Luxrender works almost the same way. We can leave our file open and create a new Luxrender project in the same file, or reload it to start from scratch.
- Press the little down arrow next to the New Luxrender project button and choose the **LuxOutdoor** template.
- Select all the components of the table. If you still have the povray project in your document, be sure to also select the lux project itself, so the views created in the next step won't go in the wrong project by mistake.
- Press the Insert part button.
- Select the luxrender project, and press the Render button.
- Luxrender works differently than povray. When you start the render, the luxrender application will open and immediately start rendering:



- If you leave that window open,Luxrender will continue calculating and rendering forever, progressively refining the image. It is up to you to decide when the image has reach a sufficient quality for your needs, and stop the render.
- There are also many controls to play with, on the left panel. All these controls will change the aspect of the image being rendered on the fly, without stopping the rendering.
- When you feel the quality is good enough, simply press **Render->stop**, and then **File->Export to image->Tonemapped low dynamic range** to save the rendered image to a png file.

You can extend greatly the render possibilities of FreeCAD by creating new templates for povray or luxrender. This is explained in the Raytracing Workbench documentation.

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

- The Raytracing Workbench: http://www.freecadweb.org/wiki/index.php?title=Raytracing_Module
- Blender: http://www.blender.org

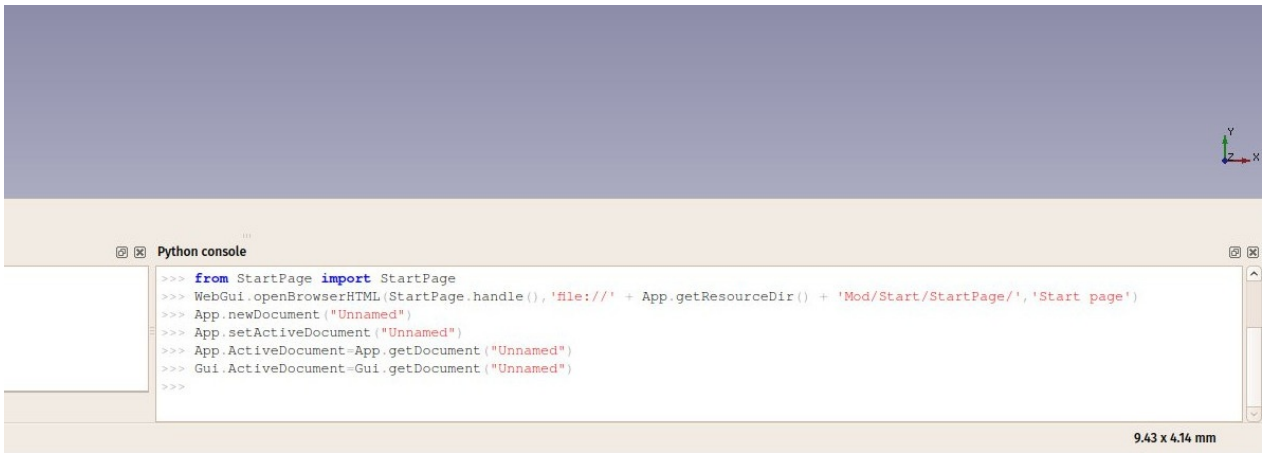
- POV-Ray: http://www.povray.org
- Luxrender: http://www.luxrender.net

Python scripting

A gentle introduction

Python is a widely popular, open-source programming language, very often used as a scripting language, embedded in applications, as this is the case with FreeCAD. It also has a series of features that makes it specially interesting for us FreeCAD users: It is very easy to learn, specially for people who had never programmed before, and it is embedded in many other applications, which makes it a very valuable tool to learn, as you will be able to use it in many other applications, such as Blender, Inkscape or GRASS.

FreeCAD makes an extensive use of Python. With it, you can access and control almost any feature of FreeCAD. You can for example create new objects, modify their geometry, analyze their contents, or even create new interface controls, tools and panels. Some workbenches of FreeCAD and most of the addon workbenches are fully programmed in python. FreeCAD has an advanced python console, available from menu **View->Panels->Python console**. It is often useful to perform operations for which there is no toolbar button yet, or to check shapes for problems, or to perform repetitive tasks:



But the python console also has another very important use: Everytime you press a toolbar button, or perform other operations in FreeCAD, some python code is printed in the console and executed. By leaving the Python console open, you can litterally see the python code unfold as you work, and in no time, almost without knowing it, you will be learning some python language.

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

There are two easy ways to write python code in FreeCAD: From the python console (menu **View -> Panels -> Python Console**), or from the Macro editor (menu **Tools -> Macros -> New**). In the console, you write python commands one by one, which are executed when you press return, while the macros can contain a more complex script made of several lines, which is executed only when the macro is launched from the same Macros window.

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 error you could do while typing.

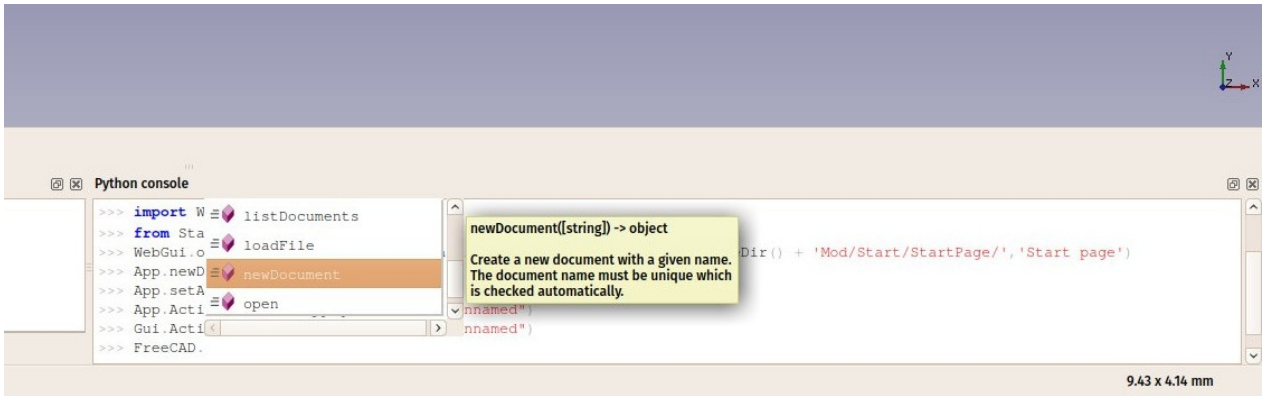
If this is the first time you are doing Python coding, consider reading this short introduction to Python programming before going further, it will make the basic concepts of Python clearer.

Manipulating FreeCAD objects

Let's start by creating a new empty document:

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

If you type this in the FreeCAD python console, you will notice that as soon as you type "FreeCAD." (the word FreeCAD followed by a dot), a windows pops up, allowing to quickly autocomplete the rest of your line. Even better, each entry in the autocomplete list has a tooltip explaining what it does. This makes it very easy to explore the functionality available. Before choosing "newDocument", have a look at the other options available.



As soon as you press **Enter** our new document will be created. This is similar to pressing the "new document" button on the toolbar. In Python, the dot is used to indicate something that is contained inside something else (newDocument is a function that is inside the FreeCAD module). The window that pops up therefore shows you everything that is contained inside "FreeCAD". If you would add a dot after newDocument, instead of the parentheses, it would show you everything that is contained inside the newDocument function. The parentheses are mandatory when you are calling a Python function, such as this one. We will illustrate that better below.

Now let's get back to our document. Let's see what we can do with it:

```
doc.
```

Explore the available options. Usually names that begin with a capital letter are attributes, they contain a value, while names that begin with small letter are functions (also called methods), they "do something". Names that begin with an underscore are usually there for the internal working of the module, and you shouldn't care about them. Let's use one of the methods to add a new object to our document:

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

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 we need:

```
doc.recompute()
```

Now our box appeared in the 3D view. Many of the toolbar buttons that add objects in FreeCAD actually do two things: add the object, and recompute. If you turned on the "show script commands in python console" option above, 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
```

View properties are stored inside a **ViewObject**. Each FreeCAd object possesses a ViewObject, which stores the vieual properties of the object. When running FreeCAD without its Graphical Interface (for example when launching it from a terminal with the -c command line option, or using it from another Python script), the ViewObject is not available, since there is no visual at all.

For example, to access the line color of our box:

```
box.ViewObject.LineColor
```

Vectors and Placements

Vectors are a very fundamental concept in any 3D application. It is a list of 3 numbers (x, y and z), describing a point or position in the 3D space. A lot of things can be done with vectors, such as additions, subtractions, projections and much more. In FreeCAD vectors work like this:

```
myvec = FreeCAD.Vector(2,0,0)
print(myvec)
prnimarkdownt(myvec.x)
print(myvec.y)
othervec = FreeCAD.Vector(0,3,0)
sumvec = myvec.add(othervec)
```

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. It is 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
```

Read more

- Python: https://www.python.org/
- Working with Macros: http://www.freecadweb.org/wiki/index.php?title=Macros
- Introduction to Python scripting: http://www.freecadweb.org/wiki/index.php?title=Introduction_to_Python
- Using Python in FreeCAD: http://www.freecadweb.org/wiki/index.php?title=Python_scripting_tutorial

- The Python scripting wiki hub: http://www.freecadweb.org/wiki/index.php?title=Power_users_hub

Creating and manipulating geometry

In the previous chapters, we learned about the different workbenches of FreeCAD, and that each of them implements its own tools and geometry types. The same concepts applies when working from Python code.

We also saw that the big majority of the FreeCAD workbenches depend on a very fundamental one: the Part Workbench. In fact, may other workbenches, such as Draft or Arch, do exactly what we will do in this chapter: They use Python code to create and manipulate Part geometry.

So the first thing we need to do to work with Part geometry, is to do the Python equivalent to switching to the Part Workbench: import the Part module:

```
import Part
```

Take a minute to explore the contents of the Part module, by typing `Part.` and browsing through the different methods offered there. The Part module offers several convenience fuctions such as makeBox, makeCircle, etc... which will instantly build an object for you. Try this, for example:

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

When you press Enter after typing the line above, nothing will appear in the 3D view, but something like this will be printed on the Python Console:

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

This is where an important concept takes place. What we created here is a Part Shape. It is not a FreeCAD document object (yet). In FreeCAD, objects and their geometry are independent. Think of a FreeCAD document object as a container, that will host a shape. Parametric objects will also have properties such as Length and Width, and will **recalculate** their Shape on-the-fly, whenever one of the properties changes.What we did here is calculate a shape manually.

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 we just made:

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

Note how we handled `myObj.Shape`, see that it is done exactly like we did in the previous chapter, when we changed other properties of an object, such as `box.Height = 5`. In fact, **Shape** is also a property, just like **Height**. Only it takes a Part Shape, not a number. In next chapter we will have a deeper look at how those parametric objects are constructed.

For now, let's explore our Part Shapes more in detail. At the end of the chapter about traditional modeling with the Part Workbench we showed a table that explains how Part Shapes are constructed, and their different components (Vertices, edges, faces, etc). The exact same components exist here and can be retrieved from Python. All Part Shape always have the following attributes:Vertexes, Edges, Wires, Faces, Shells and Solids. All of them are lists, that can contain any number of elements or be empty:

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

For example, let's find the area of each face of our box shape above:

```
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)
```

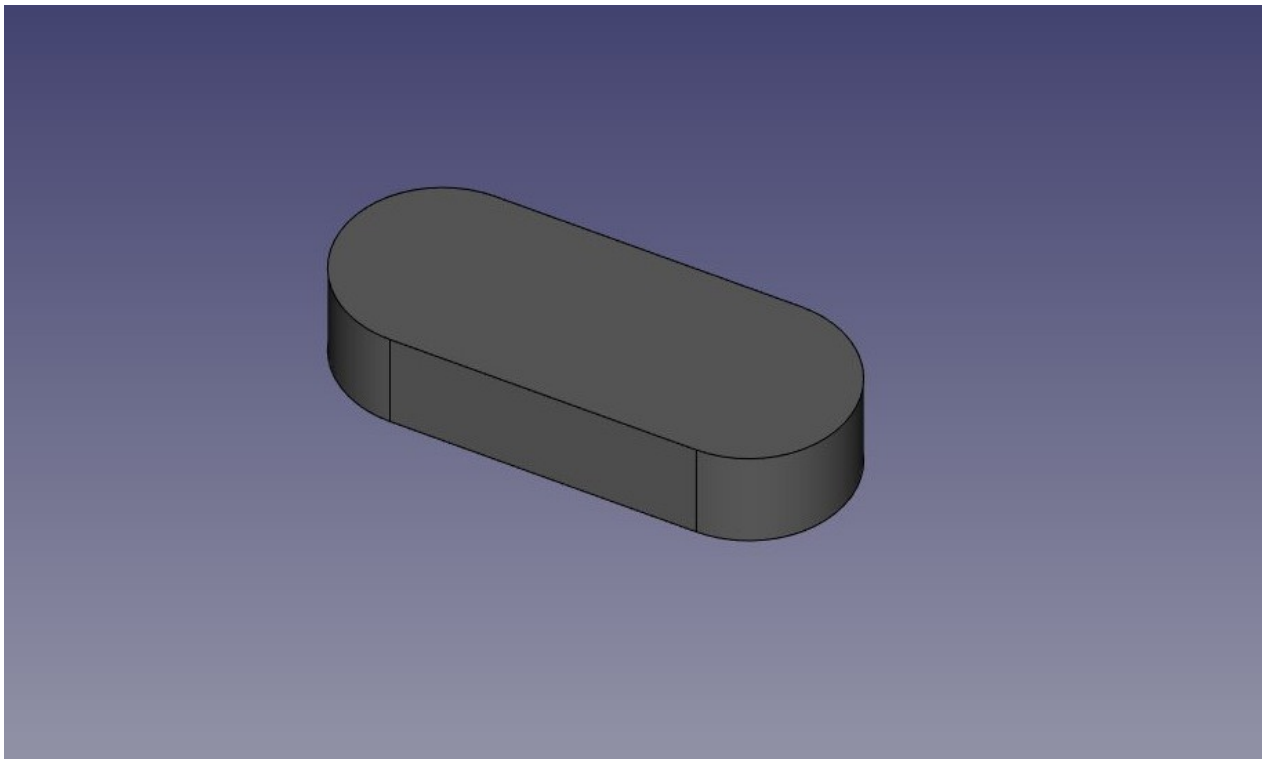
As you see, if our boxShape has a "Vertexes" attribute, each Edge of the boxShape also has a "Vertexes" attribute. As we can expect, the boxShape will have 8 vertices, while the edge will only have 2, which are both part of the list of 8.

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

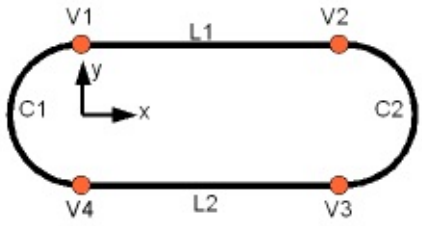
```
print(boxShape.ShapeType)
print(boxShape.Faces[0].ShapeType)
print (boxShape.Vertexes[2].ShapeType)
```

So to resume the whole diagram of Part Shapes: Everything starts with Vertices. With one or two vertices, you form an Edge (full circles have only one vertex). With one or more Edges, you form a Wire. With one or more closed Wires, you form a Face (the additional Wires become "holes" in the Face). With one or more Faces, you form a Shell. When a Shell is fully closed (watertight), you can form a Solid from it. And finally, you can join any number of Shapes of any types together, which is then called a Compound.

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:



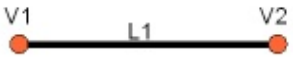
We will start by creating a planar shape like this:



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:

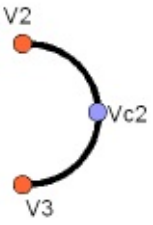



```
L1 = Part.Line(V1,V2)
L2 = Part.Line(V4,V3)
```

Note that we didn't need to create Vertices? We could immediately create Part.Lines from FreeCAD Vectors. This is because here we haven't created Edges yet. A Part.Line (as well as Part.Circle, Part.Arc, Part.Ellipse or PArt.BSpline) does not create an Edge, but rather a base geometry on which an Edge will be created. Edges are always made from such a base geometry, which is stored its Curve attribute. So if you have an Edge, doing:

`print(Edge.Curve)`

will show you what kind of Edge this is, that is, if it is based on a line, an arc, etc... But let's come back to our exercise, and build the arc segments. For this, we will need a third point, so we can use the convenient Part.Arc, which takes 3 points:




```
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 do 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 don't need to take care of the order. OpenCasCade, the geometry "engine" of FreeCAD, is extraordinarily tolerant to unordered geometry. It will sort out what to do:

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

And we can check if our Wire was correctly understood, and that it is correclty 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 had 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)
```

Because when extruding 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 attribute it our new Solid:

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

Altenatively, 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: http://www.freecadweb.org/wiki/index.php?title=Part_Workbench
- Part scripting: http://www.freecadweb.org/wiki/index.php?title=Topological_data_scripting

Creating parametric objects

In the previous chapter, we saw how to create Part geometry, and how to display it on screen, by attaching it to a "dumb" (non-parametric) document object. This is tedious when we want to change the shape of that object. We would need to create a new shape, then attribute it again to our object.

However, we also saw in all the preceding chapters of this manual how parametric objects are powerful. We only need to change one property, and the shape is recalculated on-the-fly.

Internally, parametric objects don't do anything different than we just did: They recalculate the contents of their Shape property, over and over, each time another property has changed.

FreeCAD provides a very convenient system to build such parametric objects fully in Python. They consist of a simple Python class, which defines all the properties that the object needs, and what will happen when one of these properties changes. The structure of such 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 usually have an **init** method. What is inside that method is executed when that class is instantiated (which means, in programming slang, that a Python Object is created from that class. Think of a class as a "template" to create live copies of it). In our **init** function here, we do two important things: 1) store our class itself into the "Proxy" attribute of our FreeCAD documetn object, that is, the FreeCAD document object will carry this code, inside itself, and 2) create all the properties our object needs. There are many types of properties available, you can get the full list by typing this code:

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

Then, the second important part is the execute method. Any code in this method will be executed when the object is marked to be recomputed, which will happen when a property has been changed. That is all there is to it. Inside execute, you need to do all that needs tobe done, that is, calculating a new shape, and attributing to the object itself with something like `obj.Shape = myNewShape`. That is why the execute method takes an "obj" argument, which will be the FreeCAD document object itself, so we can manipulate it inside our python code.

One last thing is important to remember: When you create such parametric objects in a FreeCAD document, when you save the file, the python code above is not stored inside the file. This is for security reasons, if a FreeCAD file contained code, it would be possible for someone to distribute FreeCAD files containing malicious code that could harm other people's computers. So, if you distribute a file that contains objects made with the above code, such code must also be present on the computer that will open the file. The easiest way to achieve that is usually to save the code above in a macro, and distribute the macro together with your FreeCAD file, or share your macro on the FreeCAD macros repository where anybody can download it.

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 object 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 o
f the Console
    # (in a macro, for example) where the FreeCAD module has not been imported automat
ically
    import Part,FreeCAD

    # first we need to make sure the values of Length and Width are not 0
    # otherwise the Part.Line 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.Line(v1,v2).toShape()
    e2 = Part.Line(v2,v3).toShape()
    e3 = Part.Line(v3,v4).toShape()
    e4 = Part.Line(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 Tools -> 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):

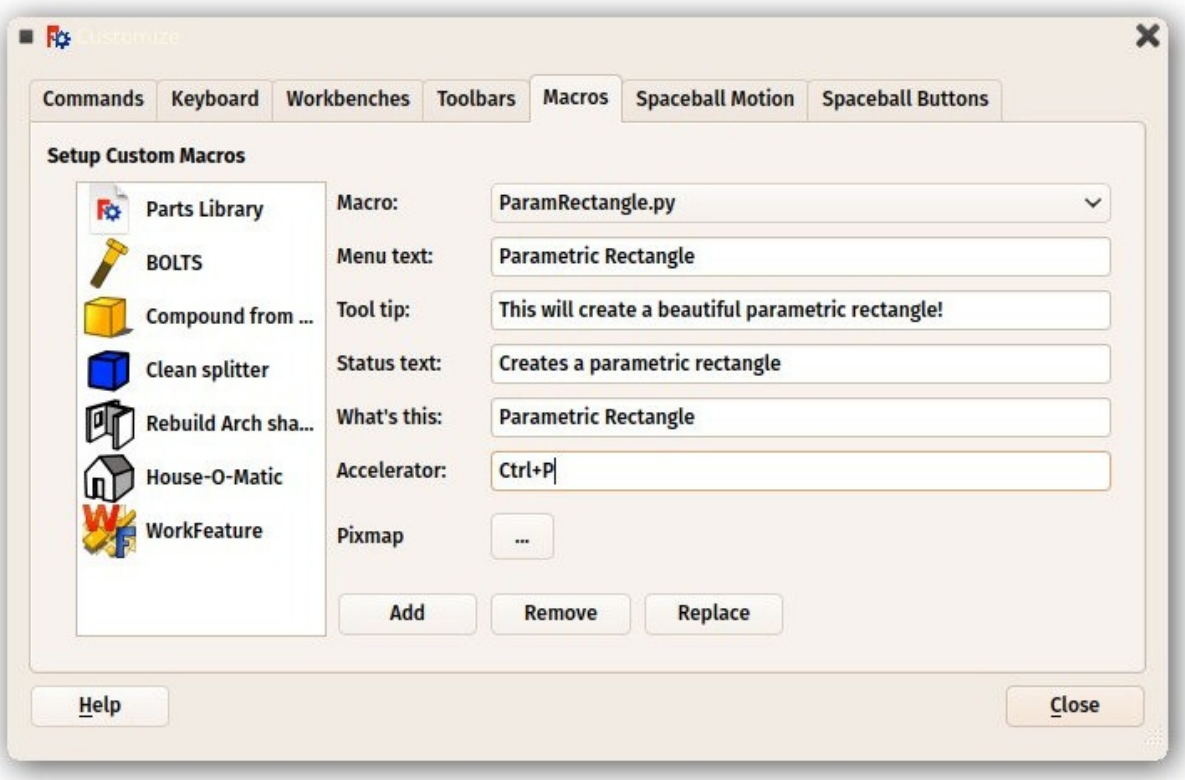
```
myObj = FreeCAD.ActiveDocument.addObject("Part::FeaturePython","Rectangle")
ParamRectangle.ParametricRectangle(myObj)
myObj.ViewObject.Proxy = 0 # this is mandatory unless we code the ViewProvider too
FreeCAD.ActiveDocument.recompute()
```

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.

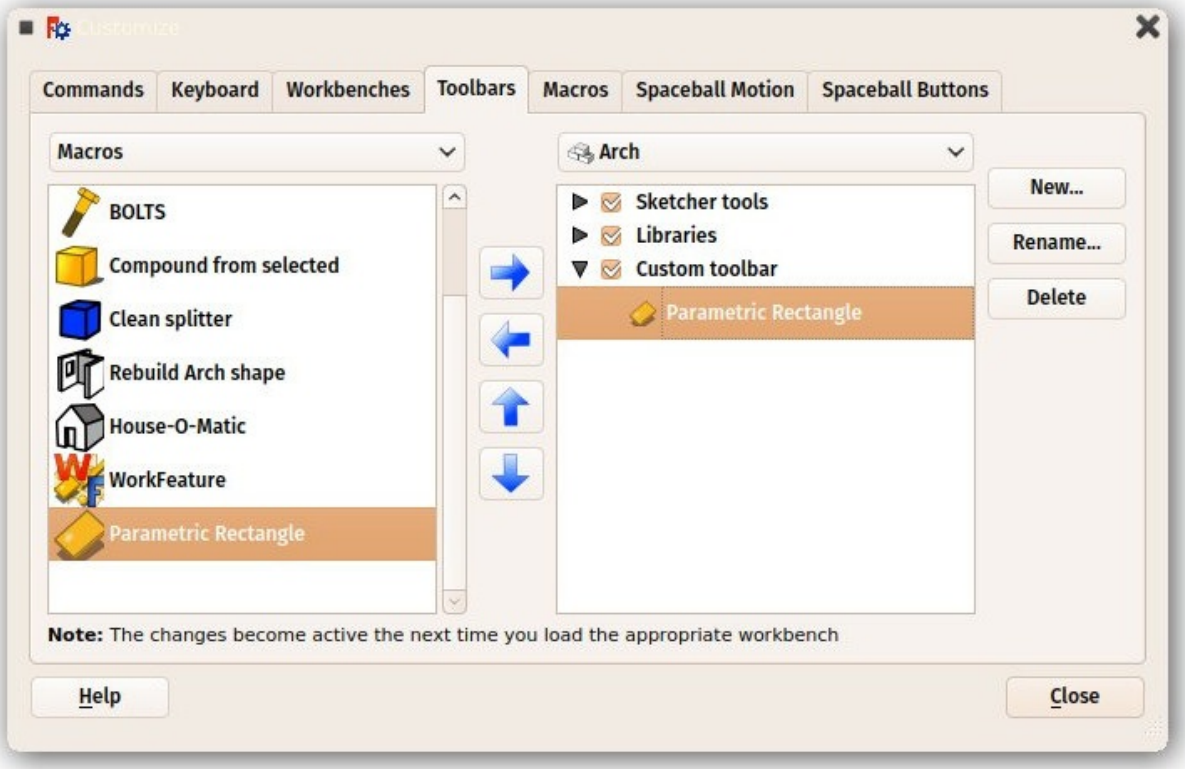
Of course it would be tedious to have to type these 4 lines of Python code each time we want to create a new parametric rectangle. A very simple way to solve this is placing the 4 lines above inside our ParamRectangle.py file, at the end, after the end of the ParametricRectange class (We can do this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

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

Read more

- The FreeCAD macros repository: http://www.freecadweb.org/wiki/index.php?title=Macros_recipes
- Parametric object example: http://www.freecadweb.org/wiki/index.php?title=Scripted_objects
- More examples in the FreeCAD code: https://github.com/FreeCAD/FreeCAD/blob/master/src/Mod/TemplatePyMod/FeaturePython.py

Creating interface tools

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 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, or GUI, that is, all the part of FreeCAD that is displayed on your screen (the menu, toolbars, 3D view, etc), is all there for that purpose: interact with the user. FreeCAD's interface is built with Qt, a very common open-source GUI toolkit that offers a big range of tools such as dialog boxes, buttons, labels, text input boxes or pull-down menus (all these are generically called "widgets").

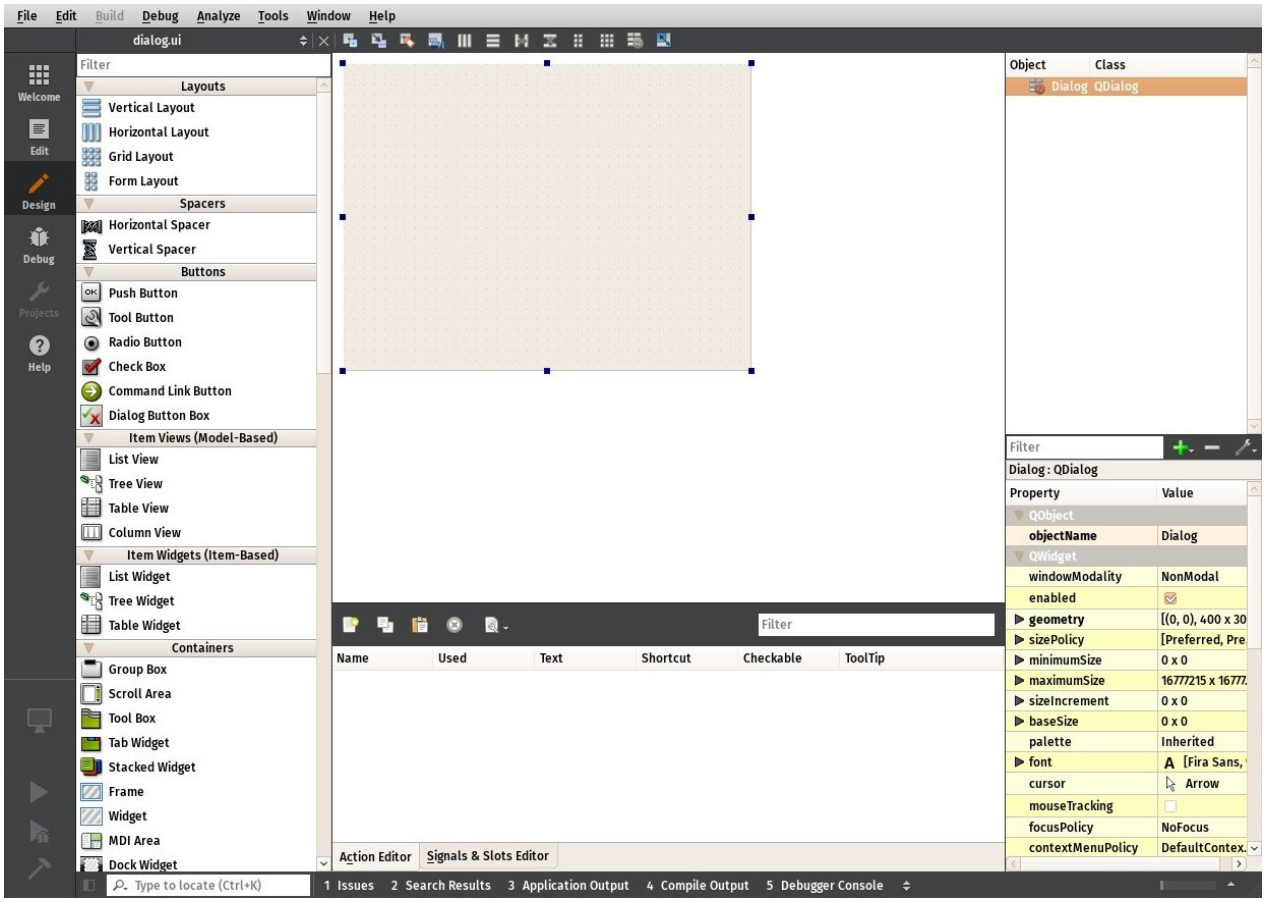
The Qt tools are very easy to use from Python, thanks to a Python module called Pyside (there are several other Python modules to communicate with Qt from Python too). This module allows you to create and interact with widgets, read what the user did with them (what was filled in text boxes) or do things when, for example, a button was pressed.

Qt also provides another interesting tool called Qt Designer, which is today embedded inside a bigger application called Qt Creator. It allows to design dialog boxes and interface panels graphically, instead of having to code them manually. In this chapter, we will use Qt Creator to desig 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).

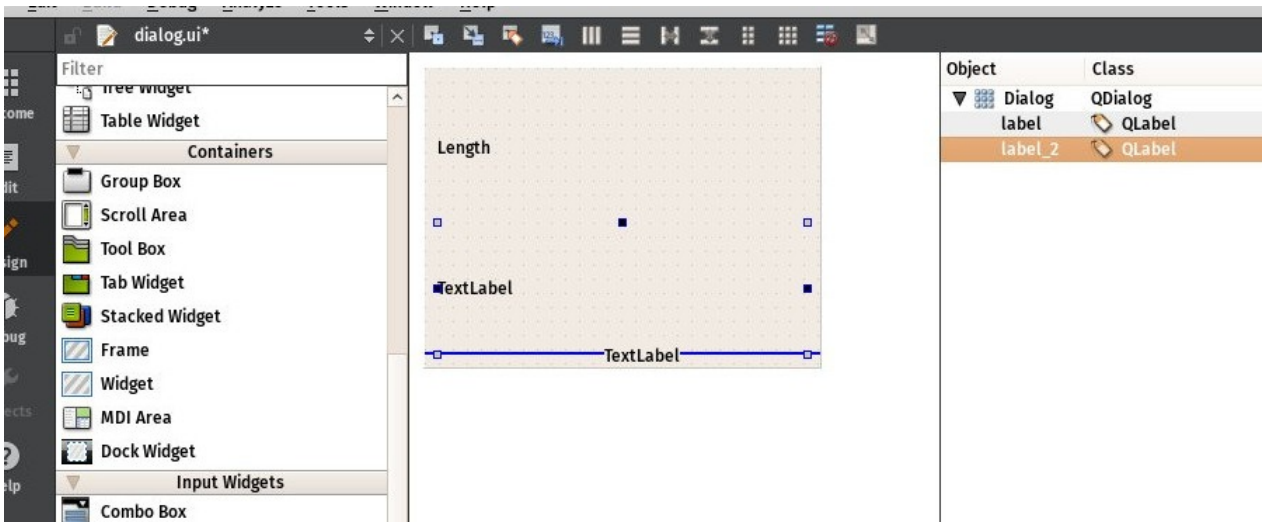
In the following exercise, we will first create a panel with Qt Creator that asks for length, width and height values, then we will create a Python class around it, that will read the values entered by the user from the panel, and create a box with the given dimensions. This

Python class will then be used by FreeCAD to display and control the task panel:

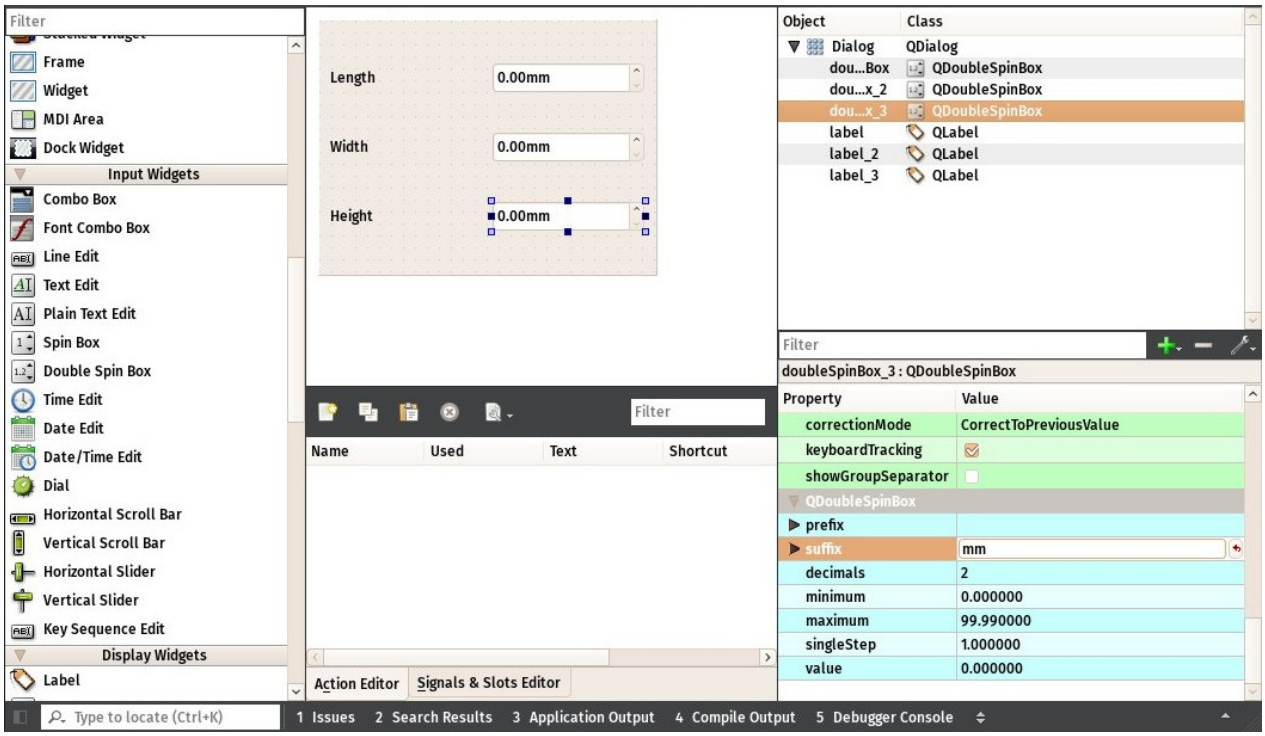
Let's start by creating the widget. Start Qt Creator, then menu **File -> New File or Project -> Files and Classes -> Qt -> Qt Designer Form -> Dialog without buttons**. Click **Next**, give it a filename to save, click **Next**, leave all project fields to their default value (""), and **Create**. FreeCAD's Task system will automatically add OK/Cancel buttons, that's why we chose here a dialog without buttons.



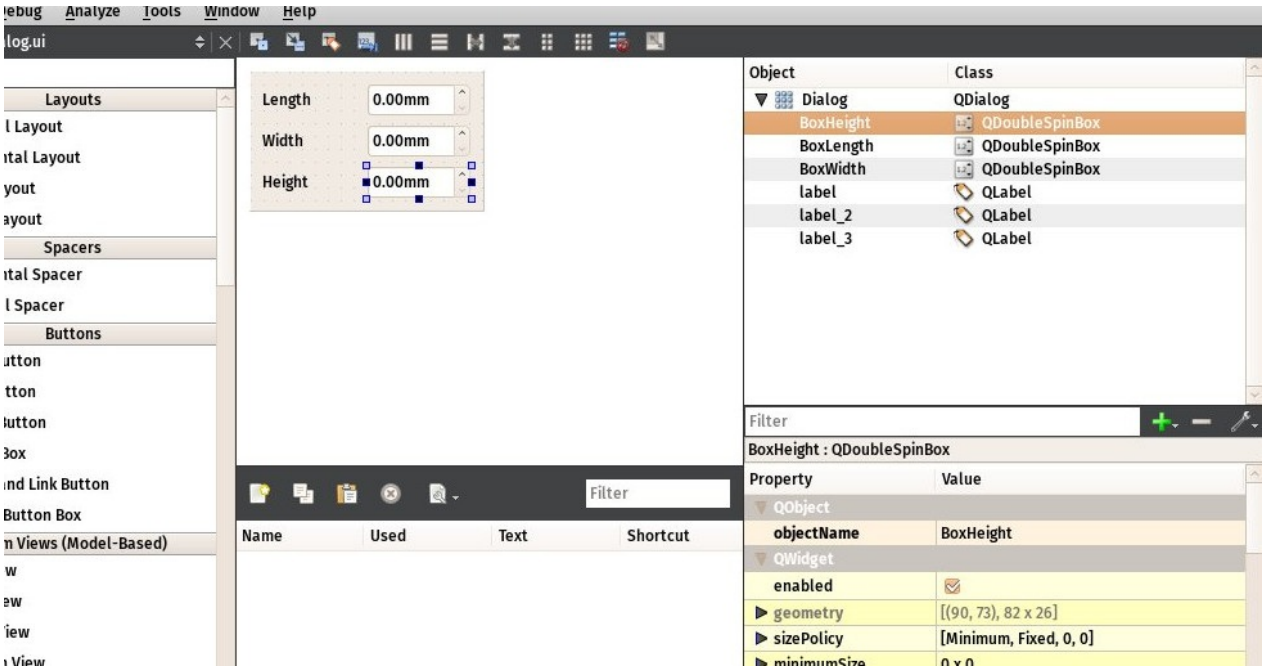
- Find the **Label** in the list in the left panel, and drag it onto the canvas of our widget. Double-click the recent placed Label, and change its text to **Length**.
- Right-click the widget canvas, and choose **Lay out-> Lay out in a Grid**. This will turn our widget into a grid with currently only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



- Now place 3 **Double Spin Box** widgets next to our Length, Width and Height labels. For each of them, in the lower left 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 retrive 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 a "OK" button in a dialog (which is the case by default when using the FreeCAD Tasks panel), any funcion named "accept" will automatically be executed when the "OK" button is pressed. Similarily, you can also add a "reject" function which gets executed when the "Cancel" button is pressed. In our case, we ommitted 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 which 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" button 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.

Read more

- Qt Creator: https://en.wikipedia.org/wiki/Qt_Creator
- Installing Qt Creator: https://www.qt.io/ide/
- Python scripting documentation: http://www.freecadweb.org/wiki/index.php?title=Power_users_hub
- Dialog creation tutorial: http://www.freecadweb.org/wiki/index.php?title=Dialog_creation
- PySide tutorials: http://www.freecadweb.org/wiki/index.php?title=PySide
- PySide documentation: http://srinikom.github.io/pyside-docs/index.html

The Community

No manual dealing with free and open-source software would be complete without a chapter about the community. As the vast majority of free and open-source software projects, FreeCAD is made by a community, and maintained by that community. Instead of the opaque, unknown, impersonal and inaccessable firm that is more than often 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 to 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 or Google+.

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, for example:

- **Help to spread the word**: Many people would get huge benefit from using a free, open-

source 3D modeler like FreeCAD, but simply don't know its existence. Publishing the work you do with FreeCAD, talking about it on social networks, etc... helps these people to discover FreeCAD.

- **Help newcomers**: The vast majority of discussions on the forum are questions asked by new users. You might know good anwers to give them.
- **Help reporting bugs**: The stablility 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, it is important that users 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 to write documentation**: The FreeCAD documentation wiki is also written by community members. Some sections of it are still incomplete, or their information incorrect or obsolete. You might be able to help to 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 that you work on something that someone else is already working on too, and ensures that others agree with the way you are doing it, so 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: http://forum.freecadweb.org

- The source code of FreeCAD: https://github.com/FreeCAD/FreeCAD
- The Facebook FreeCAD community: https://www.facebook.com/FreeCAD
- The Google+ FreeCAD community: https://plus.google.com/u/0/communities/103183769032333474646
- The FreeCAD documentation wiki: http://www.freecadweb.org/wiki
- Translating FreeCAD on crowdin: https://crowdin.com/project/freecad
- The FreeCAD bug tracker: http://www.freecadweb.org/tracker