



Full Circle

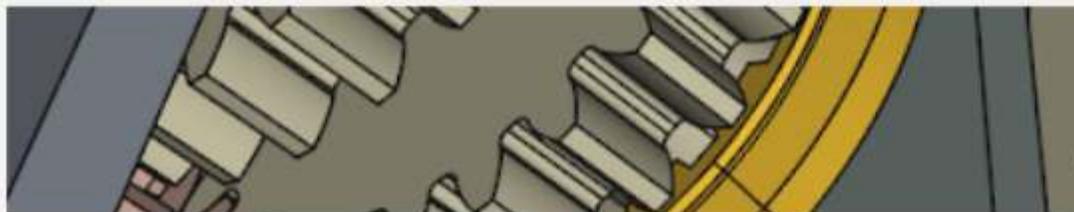
THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

UBUNTU GAMES SPECIAL EDITION

SPECIAL EDITION



FreeCAD



FreeCAD

Volume One 1 - 10

Full Circle Magazine is neither affiliated, with nor endorsed by, Canonical Ltd.



Affordable Computer Assisted Design (CAD) and its complement, Computer Assisted Manufacturing (CAM), have revolutionized many professional workflows in the last several years. There was a time when professional-grade software such as Dassault's CATIA – and the hardware necessary to run it – was out of reach of most small businesses and the occasional hobbyist. Nowadays, the advent of 3D printing using plastic extrusion has made physical prototyping a viable proposition, meaning, in turn, that a larger segment of computer users actually has a need for usable software to design their pieces.

Another group of users includes people designing virtual 3D environments on computers. Many of the same principles apply as when building 3D objects, since working with spatial coordinate systems presents the same challenges in both scenarios, though virtual world designers and ray-tracing artists must additionally contend with object

surface qualities and the behavior of light when interacting with the object.

Luckily, CAD software for the open-source software user has gone a long way from its (rather timid) beginnings. In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that it still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories.

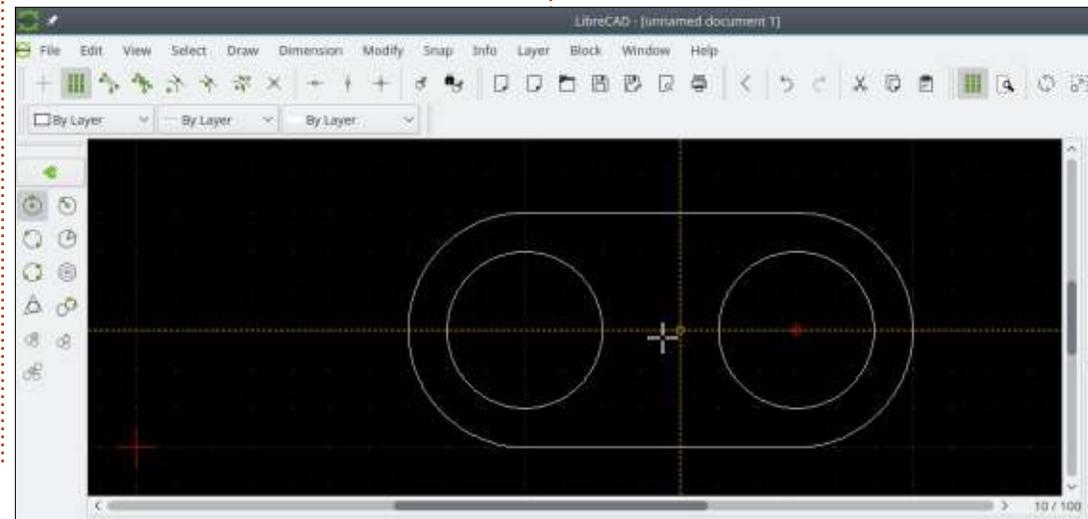
A CHOICE OF PROGRAMS

Industrial drawing and design software has historically been an area with a small number of offerings. Even in the world of commercial applications, until very recent years, one single name used to arise, repeatedly, enjoying a dominant position as well as defining file formats. This is perhaps understandable, since it does take some time to correctly operate what can be rather complex pieces of software. Once one has come to dominate a

specific application, facing a similar and protracted process to learn another can be something of a challenge, even if one is not starting at the very beginning of the learning curve. So, it comes as a little surprise that the situation was even worse concerning open source CAD software. Ten years ago, perhaps the only application that ran on Ubuntu was qcad (<http://qcad.org>), still in its infancy back then but available in the Canonical repositories.

Nowadays, things have changed for the better, and there is a fair offering of programs available that can both read and produce DXF file

format drawings. Both qcad, and its fork, LibreCAD, (<http://librecad.org>) are open-source design programs that focus on 2D, and can, within some limits, be seen as viable alternatives for the popular but non-free AutoCAD (<http://www.autodesk.com>) series. The price for both qcad and LibreCAD is right (as in free), and availability is for GNU/Linux, Apple's Mac OS, and Microsoft Windows. As can be seen in the screenshot, the LibreCAD interface is very similar to AutoCAD's original user interface, which can make conversion from one program to the other easier for the experienced user.



In this series, however, we would prefer the software to more easily create designs in 3D. This is both for ease of learning - correctly "seeing" a 3D object from flat plans can be somewhat difficult for beginners - and because our design will then be exported for printing in a suitable 3D printer, thus creating a physical object that we can compare to our original idea. In this scenario, a piece of software that has gained wide acceptance is SketchUp (formerly Google SketchUp, <http://www.sketchup.com>) for many reasons, including its ease of use. However, this program has been made available over the years exclusively for the Windows and Mac OS platforms. Though there have been reports of people installing and using it successfully from inside the Wine emulator, an open-source piece of software (as opposed to being merely free for use) that is better integrated into Ubuntu can be seen as preferable.

OpenSCAD (www.openscad.org) is another option for designing 3D objects suitable for 3D printing, though its use-case is focussed on Constructive Solid Geometry (CSG) and is thus perhaps a bit more

limited than other applications. However, object creation can easily be conceived as a metalanguage or script, which may have its attraction for users of the Povray raytracer that uses a similar conceptual model.

Another option would be TinkerCAD (<http://www.tinkercad.com>), an online program that can be used for simple projects. However, it can only be used through a web browser and is closed-source, which can present both practical and philosophical inconveniences.

Finally, FreeCAD (<http://www.freecadweb.org>) is the application we will be focusing on in this series. There are several reasons for this choice, including a relative ease of use, being open source, and available for GNU/Linux but also Windows and Mac OS. It should be said that FreeCAD has modules for both 2D and 3D drawing, though its target seems to be mainly the latter. Interaction between 2D and 3D design is also possible as, for example, when building a 3D model from an initial 2D floor plan, or when exporting the 2D plans from a 3D model. Though the

project does caution us on their website that "FreeCAD is under heavy development and might not be ready for production use" - which is coherent with their current 0.16 version number - in actual fact the software does seem to work quite well - at least well enough, in fact, to make this software a viable option for the enthusiast and for learning purposes. Professional users may wish to evaluate the application thoroughly before making a decision, to ensure it fits in well for their own particular needs.

There is a large corpus of user documentation available for this project on the site, and also on Youtube. As often with software under heavy development, the documentation often is not quite at the same level as the software and some discrepancies can be seen between versions in the documentation and on your computer, though it is usually not too complicated to figure out how to make things work out. This series of articles is, obviously, not meant to replace the official documentation and tutorials. What it is aimed at is to provide a practical introduction to the use of this software by someone who has

not participated in its creation and who, for this reason, may have a slightly different point of view and priorities: those of an ordinary user.

INSTALLATION

```
sudo apt update ; sudo apt
install freecad
```

Or use your favorite software manager in any version of Ubuntu; 'nough said. At the time of writing, version 0.15 is to be found in the Xenial repositories, which is a stable version. Developer version 0.17_pre can be downloaded for Windows and Mac OS, while stable 0.16 can be downloaded for GNU/Linux from the project's Github page (<http://github.com/FreeCAD/FreeCAD/releases>). Version 0.16 can be installed under Ubuntu as well by adding the project maintainer's PPA repository:

```
ppa:freecad-
maintainers/freecad-stable
```

It must be said, however, that, with a project such as this one that is moving along quite quickly, it may be best staying with the version in Ubuntu's repositories -



even if it is slightly older than the one in the repositories. This more conservative choice means more bugs will have been ironed out and will not come down to bite us.

FreeCAD itself will take up only about 68 MBytes of disk space on our system, which can be rather impressive for users who are used to installing commercial CAD applications. It does come with several dependencies on other packages, such as the Python language it has been developed in, and other graphical libraries such as Boost. However, the sum total of software packages that are (automatically) downloaded and

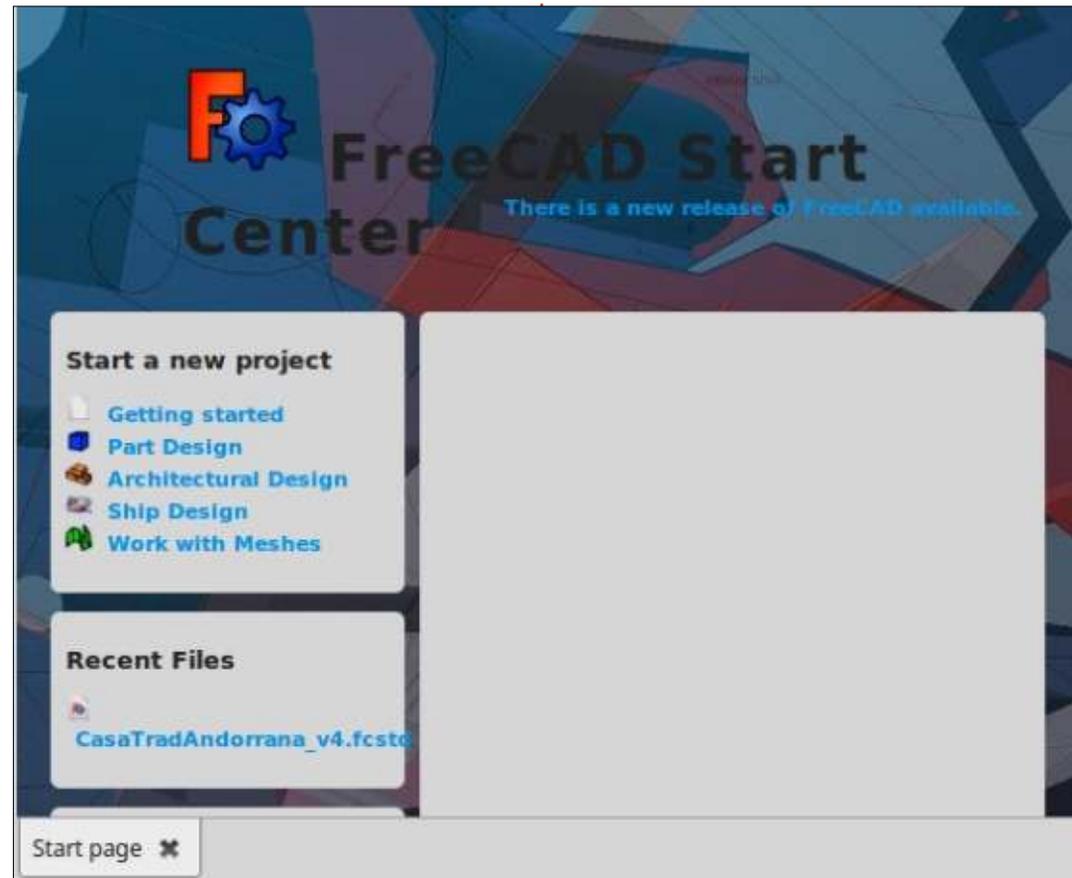
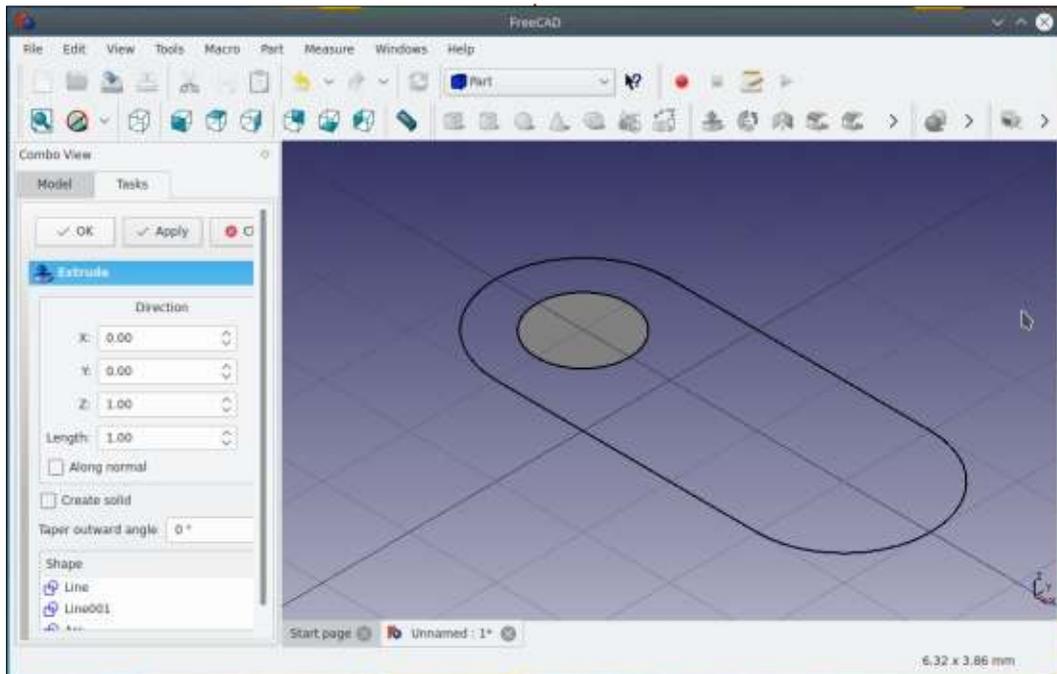
installed is well within the bounds of reason, even for systems with a low amount of available disk space. Not-so-recent processing hardware can be usable. The following screenshot was captured running FreeCAD - installed to RAM - within a Neon 5.9 live session on a laptop equipped with an AMD Athlon X2 processor and 2 GigaBytes of memory, but whose hard drive has been scrapped several years ago. This would clearly not be optimal for production use, but can be envisioned to work on simple projects.

THE PROGRAM INTERFACE

Designing a user interface for a CAD program is always complex, since there is a rather large amount of information to be displayed. Toolbars can include drawing tools for two- and three-dimensional objects, operations on objects such as scaling and duplication, operations combining objects, and different layers may be displayed or hidden. A program such as FreeCAD that works with an internal tree representation of the scene includes object

inspection, thus allowing the user to edit object parameters (such as length and coordinates) directly. But this makes further demands on user interface space since these options must be displayed at some time.

As can be seen in the screenshots, the FreeCAD user interface has condensed all these elements into three main areas. The main space is at the bottom right. In this, we will see the start



page or "Start Center" when the application is started up. This contains some rather handy links to various simple tasks that can be of help to the novice. Recent projects can also be opened directly from this pane, though they can also be opened from a more traditional File > Open menu option. This area has a system of panes, in which the different projects we are working on will be displayed one at a time. We can switch to one or another at any moment, making it easy to work on several projects, or several different pieces for a single final object.

On the left, we have a column that usually contains a dialog with contextual information on a specific object, either the project as a whole or the element selected at the time. This is also where the parameters relating to that object can be inspected, and altered manually, if needed.

Finally, the top of the screen is populated with toolbars that contain the different tools and other options. Herein lies the specificity of the FreeCAD user interface. Toolbar visualization is controlled by a system of

"Workbenches". Within each workbench - with titles such as "Drawing", "Draft", "Part" or "Arch" - specific toolbars are activated. The paradigm is similar to a physical fabricator's workplace. In most shops, different working areas are disposed along the walls. Each bench will have nearby a set of tools, grouped according to the type of work being performed in that area in a way to minimize movement. A (physical) project may then be transferred to a workbench dedicated to soldering, or another specializing in electronic instruments, as the need arises.

In a similar fashion, the FreeCAD user will activate one or another workbench inside the user

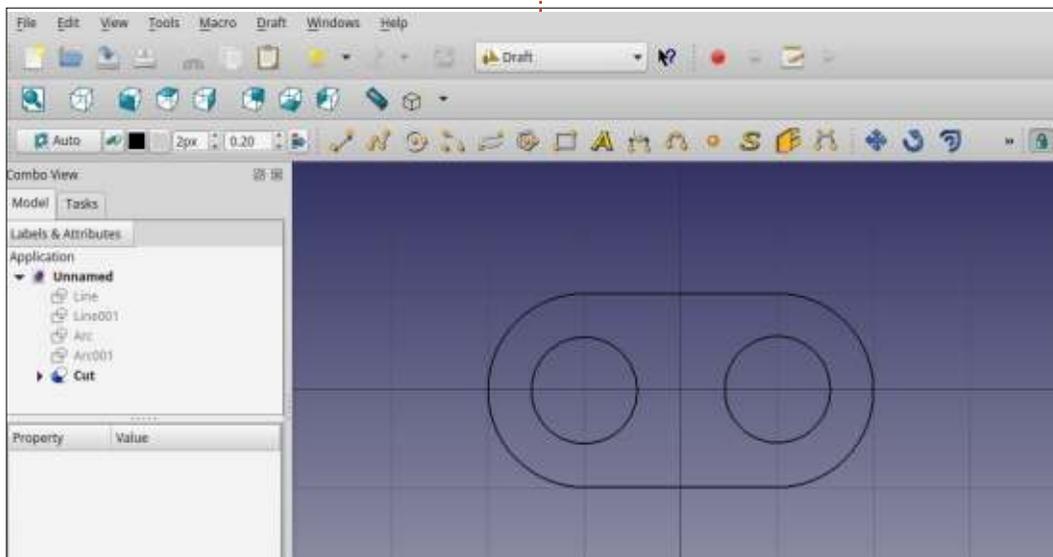
interface as the project evolves from one stage to another. In each workbench, only the toolbars with the most pertinent tools will be seen at any one time, thus reducing visual clutter on-screen. However, it should be noted that all tools within FreeCAD can be accessed from the menu system, even if they are not promoted within the active Workbench.

It should also be noted that windows and toolbars are fully floatable, and can be tailored to the user's specific needs (and the screen's available space), much in the way many modern word processing applications work. However, since there are very many different options available, it may be best for beginners to leave

tools and toolbars in their default positions, at least while starting to become familiar with the application.

WHAT NEXT?

In this first article on using FreeCAD, we went over the basics of choosing and installing a CAD application for Ubuntu or GNU/Linux, and reviewed some salient points of the FreeCAD user interface. In the next part, we will be creating a simple planar object to illustrate the use of the main workspaces, drawing, and extrusion tools. Constructive Solid Geometry will also be demonstrated, to punch holes in an unsuspecting piece of plain material.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



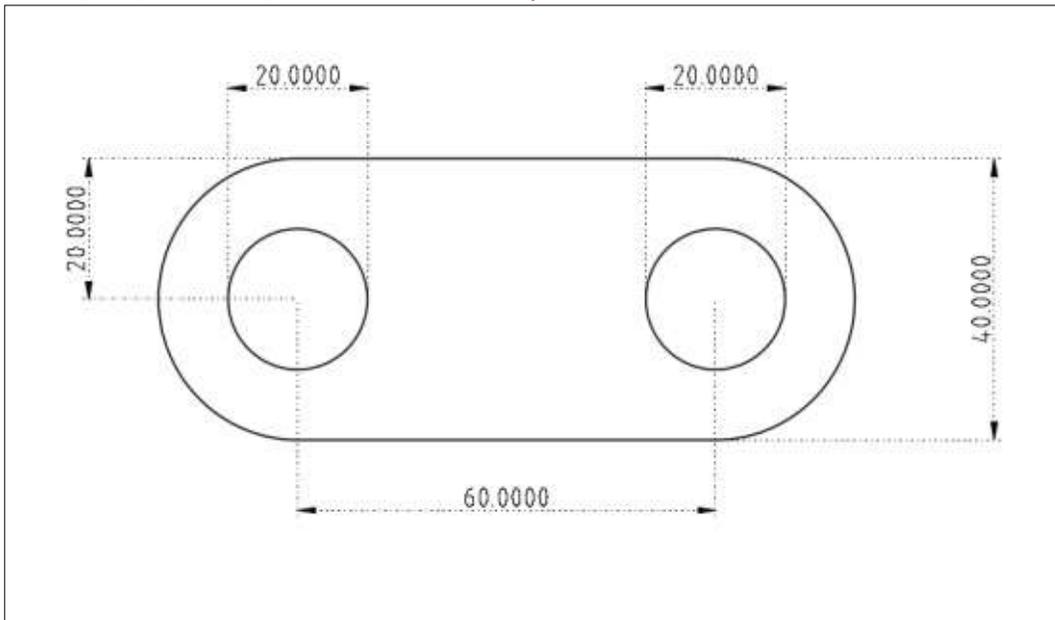
HOW-TO

Written by Alan Ward

In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the first article on using FreeCAD, we went over the basics of choosing and installing a CAD application for Ubuntu or GNU/Linux, and reviewed some salient points of the FreeCAD user interface.

In this part, we will be creating a simple planar object to illustrate

the use of the main workspaces, drawing and extrusion tools. Constructive Solid Geometry will also be demonstrated, to punch holes in an unsuspecting piece of plain material. Here is a quick sketch of our new object: basically, it will be a flat piece of material 2mm thick, with an external shape made out of straight lines and arcs, and two circular pieces cut out from the inside. Since the author is a European, all dimensions are in millimeters, though the reader can easily convert them into the units of his or her choice.



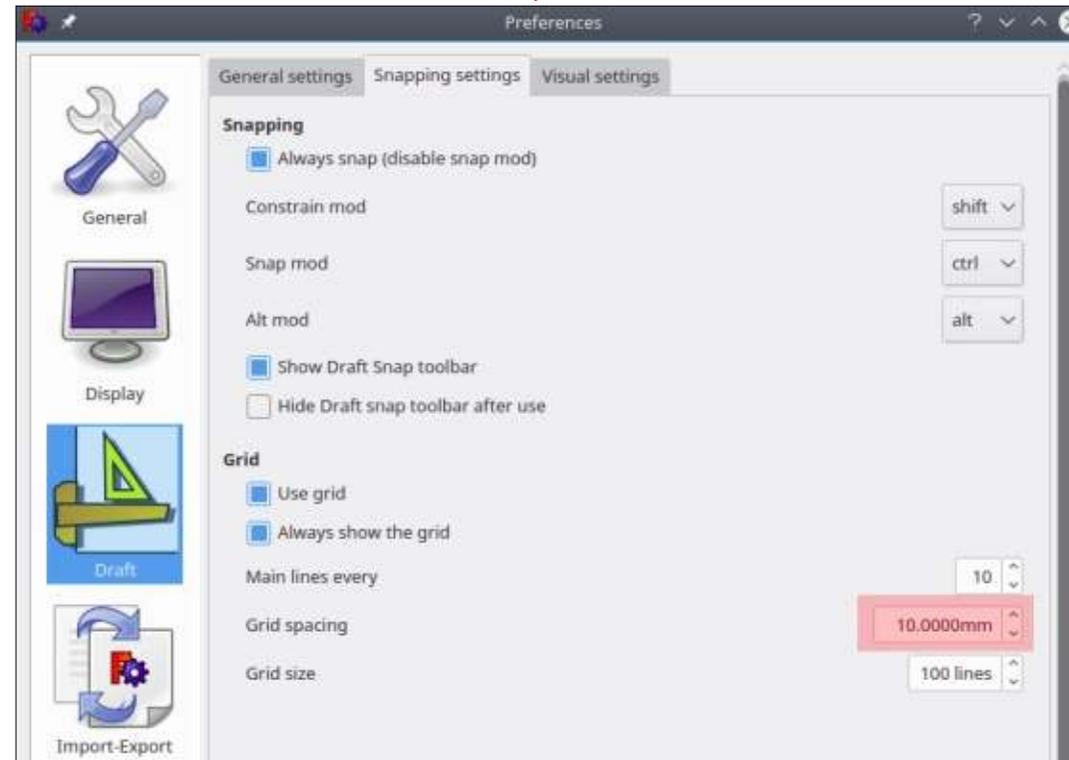
USING THE DRAWING WORKBENCH

Once inside FreeCAD, to begin a new project we can head over to menu option File > New. Alternatively, we can choose the appropriate tool from the default bar (the leftmost icon), or even use the keyboard shortcut Ctrl+N.

As discussed in the previous article, the FreeCAD user interface

has a series of workbenches, each with a specific selection of toolbars. In order to start a new project, one of the most useful is "Draft". As its name suggests, the preset toolbars in this bench contain the tools most usually used to draw up the main characteristics of the object quickly, which can then be refined with the tools in other benches.

Once inside Draft mode, there



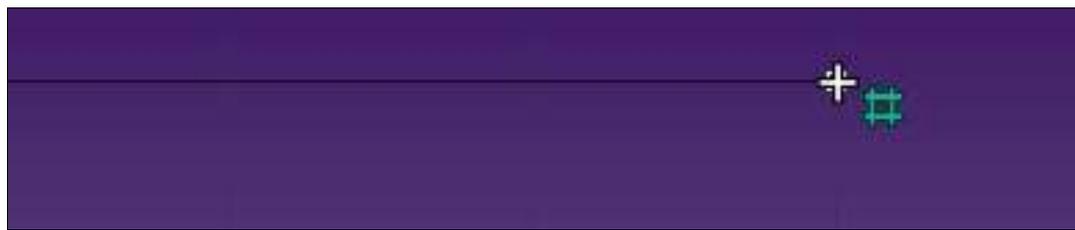
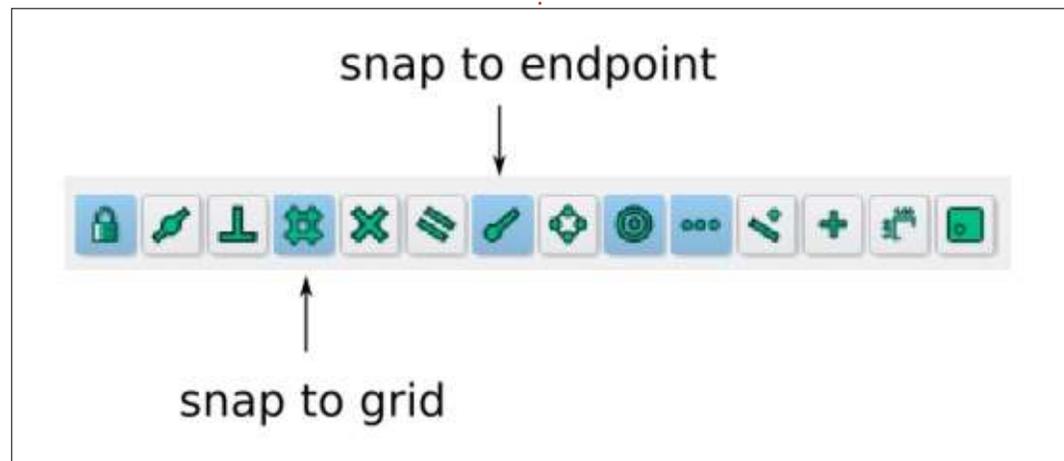
HOWTO - FREECAD

are several interface items than can be set up to facilitate working with our object. In the first place, this workbench shows us by default a grid pattern set up within the X-Y plane, that we are watching from above (down the Z axis). This grid has lines set up, with spacing of one millimeter, which may be a bit fiddly with the dimensions of our piece. So the first thing to do may be to head over to menu option Edit > Preferences. Here, we can configure the Draft workbench to our specific needs. One of the option panes, "Snapping settings", allows us to specify grid spacing. I set this to 10 millimeters, which makes it easier to get a clear sense of the dimensions of each element in our drawing.

We can now adjust the zoom

factor (e.g. with the mouse wheel, or swiping vertically on a laptop's touchpad) so as to see at least four vertical grid divisions spanning 40 mm - note the view's visible dimensions in the window's lower right corner.

As for the toolbars, for some reason the bar related to snapping points to the grid is usually collapsed. Snapping, or letting the user interface guide the point indicated with the mouse, is one of the most powerful features that is shared by many CAD programs. In this toolbar, we can configure snapping options to help up easily draw clean diagrams with object elements precisely aligned. As with all toolbars in FreeCAD, we can move the snapping toolbar to a place where we can examine its contents better.

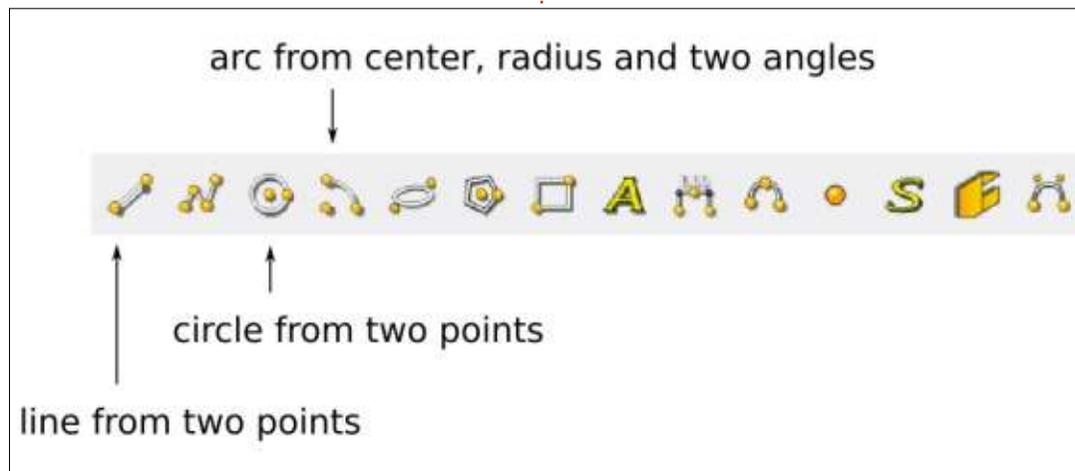


As a matter of personal choice, I tend to activate the options for snapping to the grid, which in this case will ensure the endpoints of our lines, for example, get coordinates that are integer multiples of 10 mm. I also tend to activate snapping to endpoints, which helps when drawing the last element of a multiple-segment closed path.

As for the drawing tools themselves, they are grouped into another toolbar. FreeCAD does show commendable consistency across toolbar icons, so while the previous (snapping) toolbar had

icons all in the same shade of green, drawing tools are all shown with icons in yellow and black. In this project, we will be using the tool to draw a straight line from two points, the tool to draw a full circle from its center and a point, and the tool to draw a circular arc.

Let us begin (above) by drawing the top line of our shape, from coordinates (-30, 20) to (30, 20). Since we have snapping to the grid activated, we will see the mouse icon change to show a green grid icon whenever it detects we are close to a grid intersection, and thinks we may wish to place this



point at this place.

We can then go on to draw the bottom segment, from coordinates (-30, -20) to (30, -20). Once we have the two horizontal lines set up, let us change to the arc drawing tool. To draw the circular arc closing the right end of our piece, first select the center of the arc at coordinates (30, 0). Then click on the end of the top horizontal line at coordinates (30, 20) once to indicate the radius of our arc, and a second time to indicate the point where we are beginning our arc. Finally, click on the end of the lower horizontal line at (30, -20) to give the endpoint of our arc. When clicking on the end of the horizontal lines, our mouse cursor

should change to the appropriate green icon to show us FreeCAD has detected a previous line and is placing the new point at these precise coordinates. The drawing, so far, should resemble the following capture.

We can now proceed to draw the arc on the other side of the piece, closing the outside shape. With the tool to draw circles, we can put in the two circles at coordinates (-30, 0) and (30, 0), both with radius 10 mm.

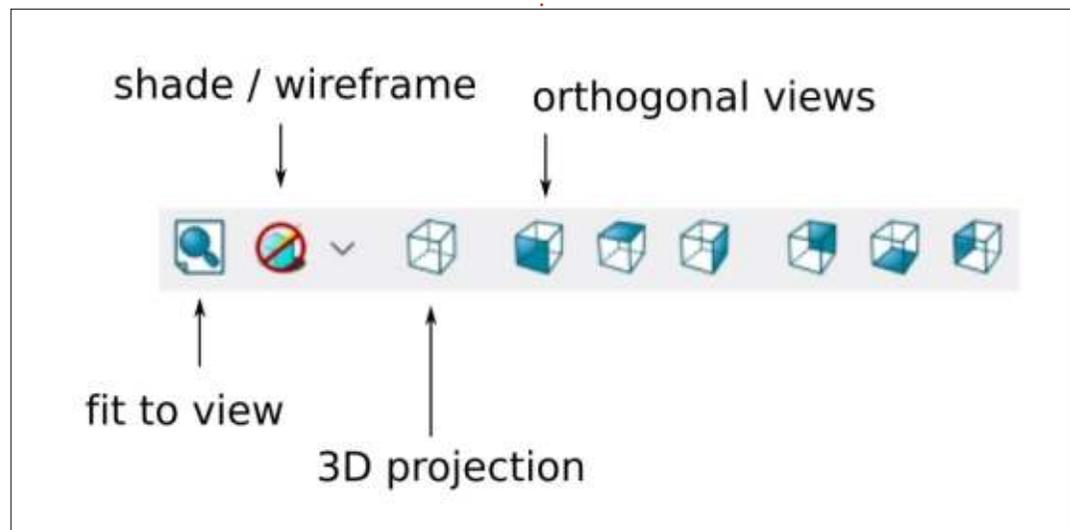
One final point is that the outside of our piece is, for the moment, a collection of four different segments: two Line objects, and two Arc objects. This



can clearly be seen in the “Combo View” window at the left hand side of the screen. Further on, we will need to convert this object into a 3D object, and for this reason we must convert the collection of four segments into a single path. This is done with the “join objects” tool in the modification toolbar. Select all four segments, either in the combo view or in the drawing itself - holding down to Ctrl key to select multiple objects - and use this tool. In the combo view, we will see the four segments disappear, to be replaced by a single Wire object.

IN THE PART WORKBENCH

Once we have the planar part of our project set up - in essence, a horizontal projection of the final piece - we can switch workbenches and choose “Part”. This is where we will give the piece its 3D touches. To begin with, let us use the views toolbar to switch to a 3D projection view, to see the piece in its current shape as a flat drawing contained within the X-Y plane. Depending on the current zoom factor, it may be useful to also choose the “fit to view” button to



HOWTO - FREECAD

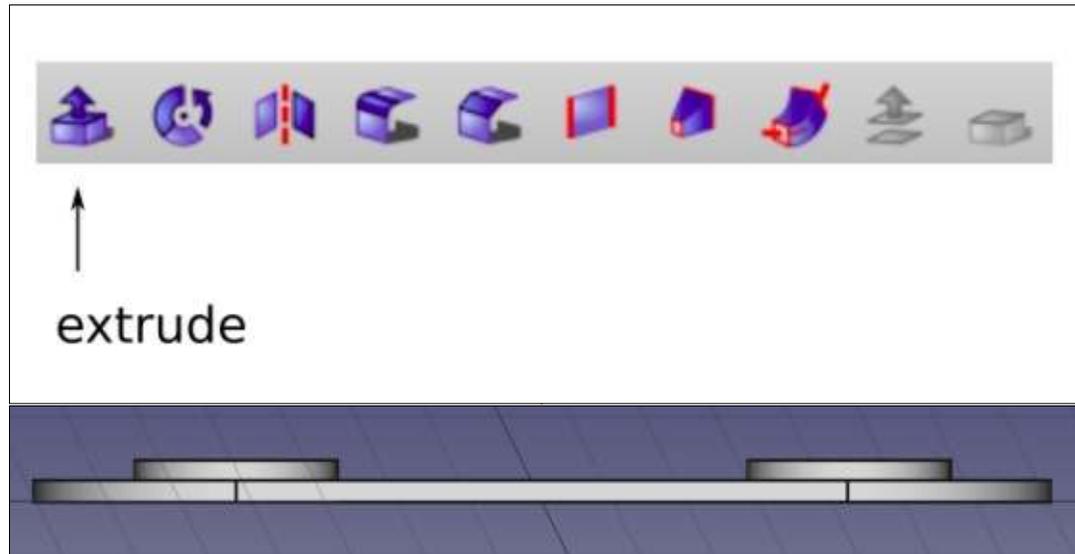
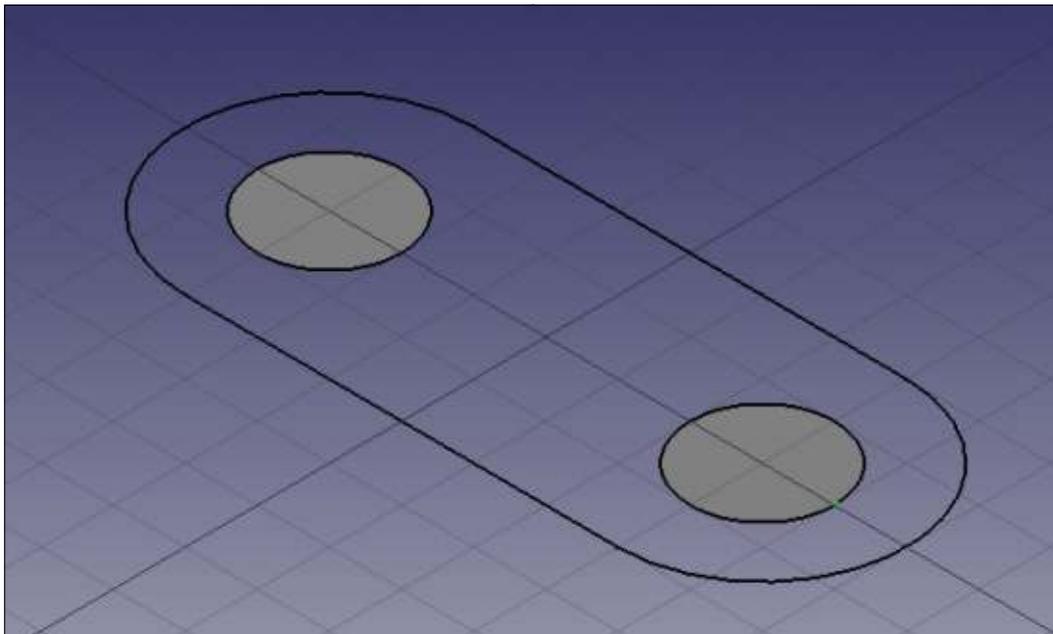
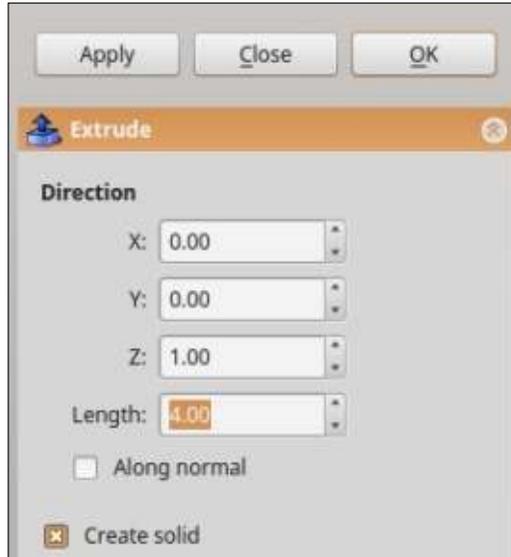
get the whole part nicely centered in the view window.

Below is what we should see at this point: the outer Wire and the two inner Circles.

One of the toolbars that come by default with the Part workbench contains tools to transform flat parts into volumes. Begin by choosing one of the circles, and then use the “extrude” tool.

Since the circle is contained within the X-Y plane, extrusion will take place along the Z axis. Increment the length of extrusion to 4 mm, so the final piece will be a

cylinder 4 mm in length. Also make sure the “Create solid” option is checked, as otherwise only the walls of the cylinder would be created.

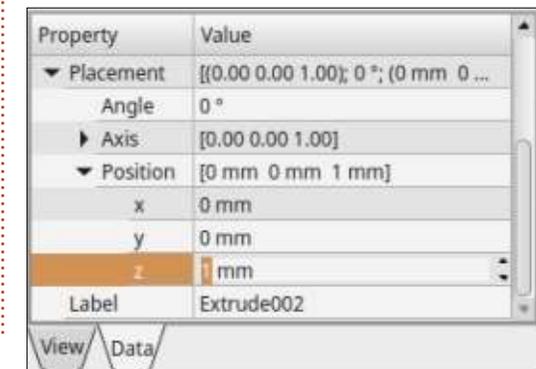


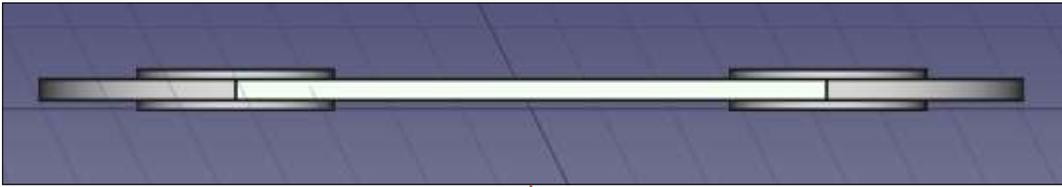
Now do the same with the other circle, converting it also into a cylinder 4 mm high. Finally, let us make an extrusion from the Wire piece, but this time only 2 mm high.

By the end of this process, if we choose a lateral orthogonal view in the view toolbar, we should see the two cylinders protruding from the main part. However, their bases are all on the same plane. This is not suitable for us, since the next operation will be to subtract the cylinders from the main part, thus creating two holes. If we leave things as they are, there may be some confusion at the lower face of each hole. It is best to make sure the cylinders protrude both

above and below the main piece.

To do so, we will simply displace the main part upwards by one millimeter. This is done by clicking on the part in “Combo view”, where it will probably be labeled as “Extrude002” or something similar. Then click on the tab marked “Data” at the bottom of the Combo view, unfold option Placement, then Position, and





increase the value for “z” from 0 mm to 1 mm.

At the same time, one should see the main part going upwards in the main view, giving this result:

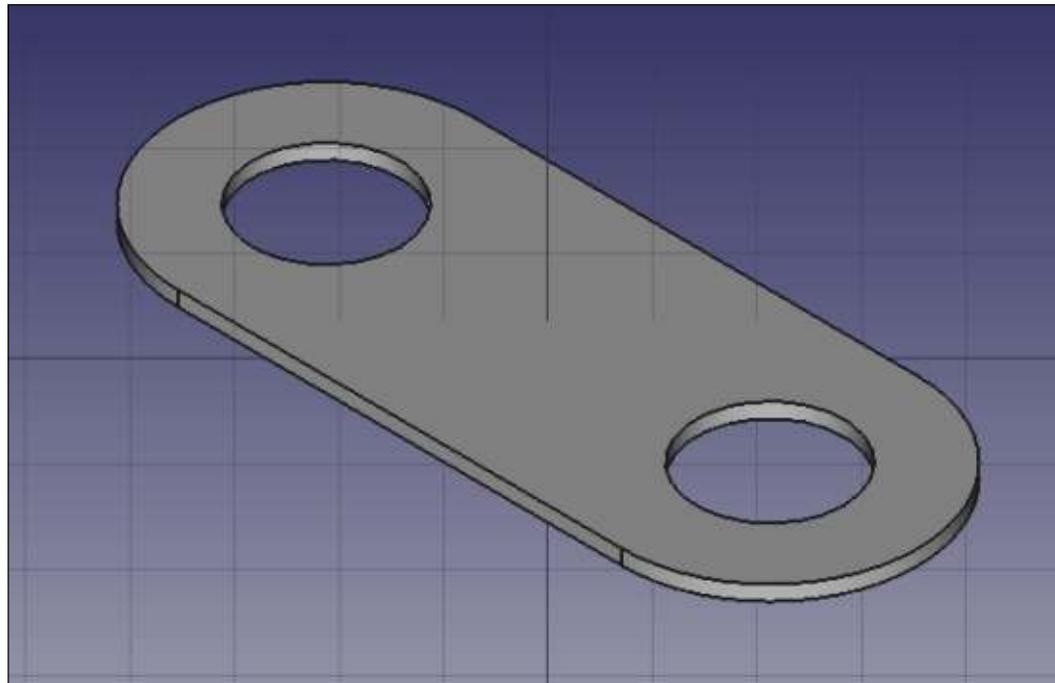
Finally, we can make the holes in the main part. To do so, start by selecting both cylinders in the Combo view, and then choose menu option Part > Boolean > Union. This should make the cylinders disappear from the Combo view, and be replaced by a single Fusion object. Inside the Combo view, choose first the main part Extrude002, and then (with the Ctrl key), also choose Fusion. Then choose Part > Boolean > Cut. Voilà, we have cut out the two holes from the main part.

In the last operation, the order of choosing the main part, and then the fusion of the two cylinders, is important. If one proceeds otherwise, FreeCAD will try to cut the main part out from the two cylinders, giving four very thin cylinders - definitely not what

we expected!

WHAT NEXT?

In this article on using FreeCAD, we created a simple planar object to illustrate the use of the main workspaces (Draft and Part), drawing tools, and extrusion. Constructive Solid Geometry was used to unite two cylinders, and the resulting Fusion object was used to cut two holes in the main



piece thus creating the final object. In the next part of the series, we will use further tools to create a more complex 3D object, representing a Y-junction between two pipes of different diameters.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).

to drawing objects in two dimensions, which is why FreeCAD had been preferred as a base tool for this series. However, LibreCAD does have an option to export drawings in the widely-used DXF (AutoCAD) file format, which can then be imported into FreeCAD and used as a basis to work with. Since the lines are already correctly positioned, it may be advantageous to use this scheme to set up our FreeCAD object.

In FreeCAD, begin by starting a new project. Now choose menu option File > Import

Now, we have several more elements than are needed for the tube wall. The axis needs to be removed, as well as the (previously dotted) lines denoting rotated edges. The lower copy of the wall section will also be erased, leaving just the higher copy, ready to be

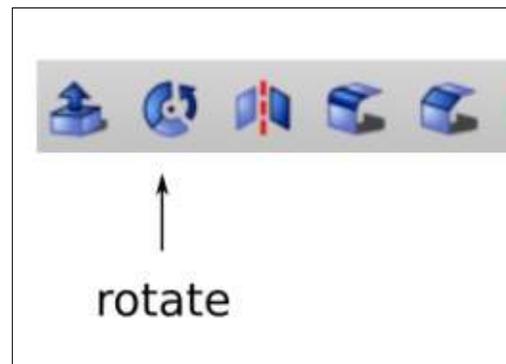
rotated. In the Draft workbench, examine the Combo view on your left, and you will see each individual line from the DXF file, that can then be erased, moved, or modified, as needed.

When we are satisfied, as before, we will need to combine the collection of Lines into a single Wire object with the "join objects" tool in the modification toolbar (whose icon is the blue up arrow). Select all segments, either in the combo view or in the drawing itself - holding down the Ctrl-key to



select multiple objects - and use the join tool. In the combo view, we will see the four segments disappear, to be replaced by a single Wire object.

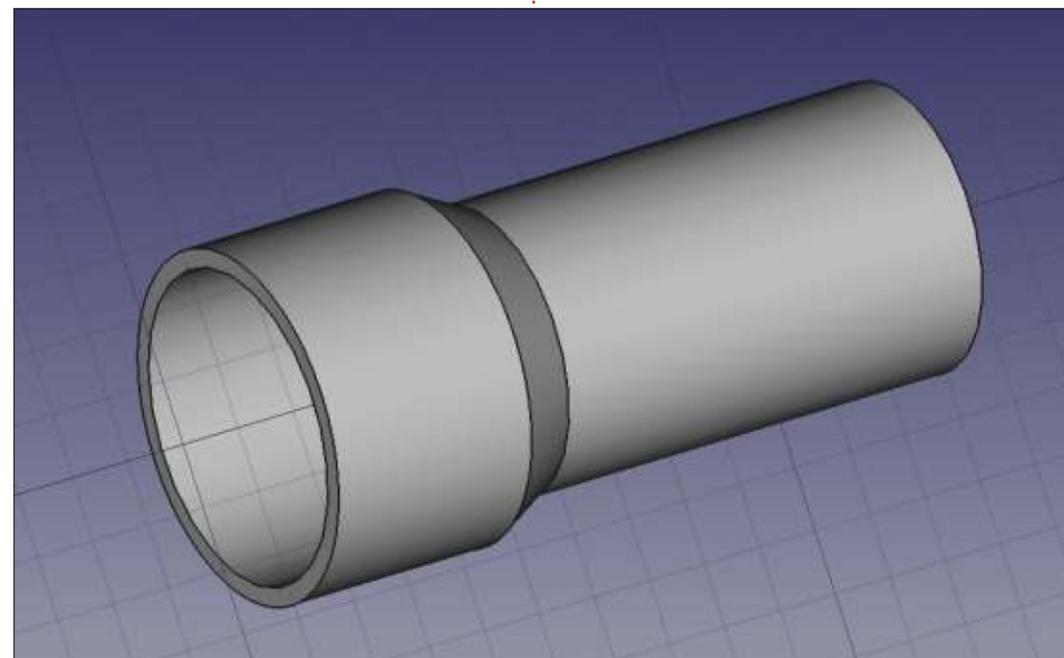
To create the actual 3D tube, let us move to the Part workbench. Now choose the "Rotate object" tool from the 3D toolbar.



Be sure to select the X axis to rotate around, since the tube object is symmetrical about this horizontal axis. Also, activate the "Solid object" checkbox. This will allow us to use this part as a complete object further on, and to make holes in it to allow branching with the lateral piece of pipe.

ADDING A LATERAL PIECE OF PIPE IN A Y-JUNCTION

We are now going to add a small piece of pipe, with inner diameter 20 mm and outer diameter 26 mm, attacking our existing tube at an angle of 45 degrees. The axis of



the new pipe will intersect our existing axis at 100 mm along its length.

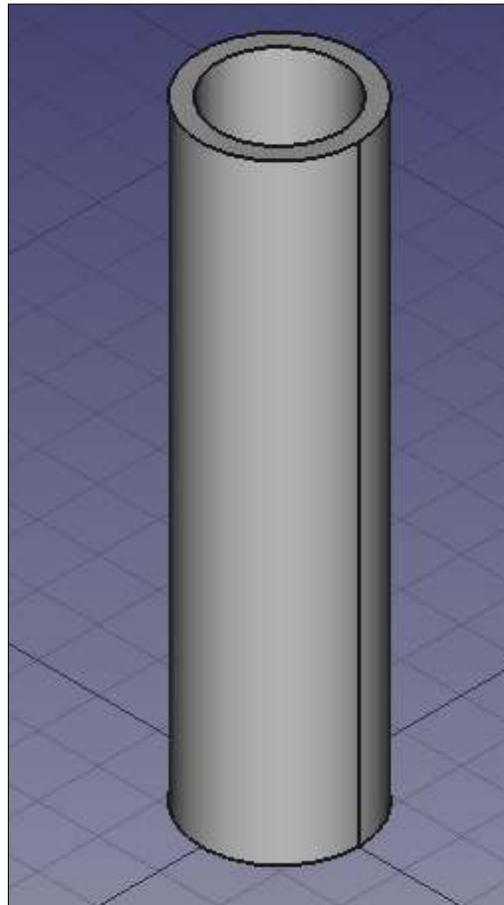
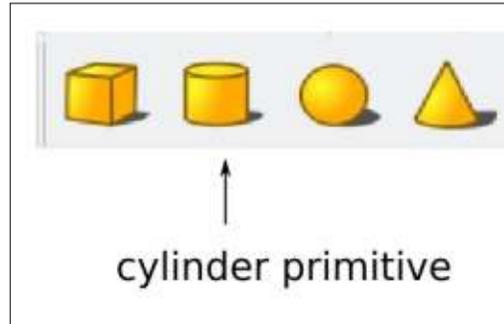
There are several ways of creating this new piece of pipe. We could proceed as before, drawing the outline with Line segments and then rotate the piece to create a 3D revolution solid. This can be created aligned with one of the standard axes, for example the Y axis, and then moved into position as required. However, since the shape of the new piece of pipe is rather straightforward, it may be easier to use pre-existing primitive

shapes to form it.

Let us begin by hiding the existing tube. This is done by going to the Combo view on the left, selecting the Revolve object, and hitting the space key on the keyboard. It will not remove the tube from our project, but simply make it invisible - thus making the construction of the smaller tube easier to visualize.

Now, using the Primitives toolbar visible in the Part workbench, let us create two cylinders. For the time being, they

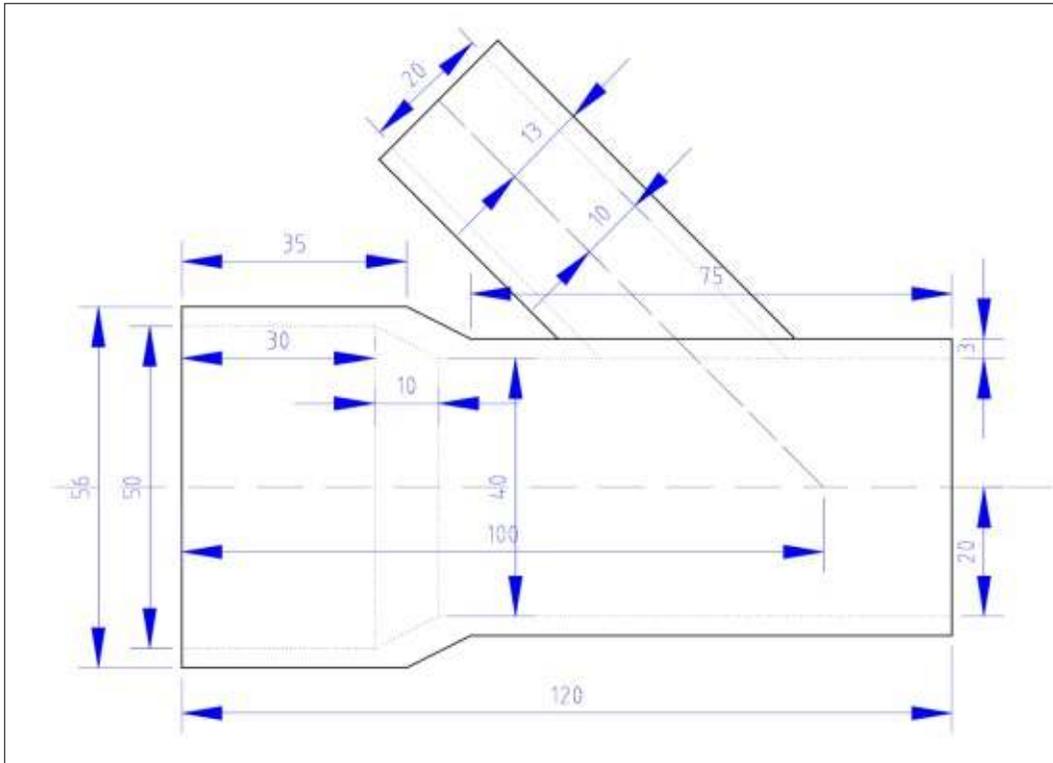
will both appear in the same position, as "Cylinder" and "Cylinder001". Default characteristics are a radius of 2 mm and height of 10 mm.



Now, using the Combo view on the left, choose the first Cylinder and go to the bottom tab marked "Data". Here, change the radius to 13 mm and the height to 100 mm. This will be the solid material for our new tube.

Choose the second Cylinder, and do the same, giving it radius 10 mm and height 104 mm. This is the shape we will cut out from the first cylinder, making it hollow. However, as in the part created in the last article of this series, both cylinders end at the same Z-coordinate. This means FreeCAD can have difficulties calculating exactly where the cutout is to end. To make things more clear, let us move the second cylinder downwards by 2 mm, making it protrude slightly from each end of the first cylinder. This is done by editing Placement > Position > z in the same tab, and giving it negative value -2 mm.

We will now use the second cylinder to cut a hole in the first. In the Combo view, choose in sequence the first cylinder, then the second, holding down the Ctrl-key when choosing the second object. Then, choose menu option Part > Boolean > Cut, and the two



cylinders should be combined into a single hollow tube using Constructive Solid Geometry.

We are ready to assemble the two parts of our project. Go back to the Combo view, choose the Revolution object - the main tube - and press the space key. Both objects should become visible at once, with a bit of zooming. However, the new Cut object is still

vertical and needs to be tilted to 45 degrees, while the larger Revolution object has been placed at some distance from the coordinate system's origin for some reason, perhaps related to the use of a DXF file to import its basic shape.

To tilt the Cut object, go to the Combo view, select this object, choose the Data tab as before, and

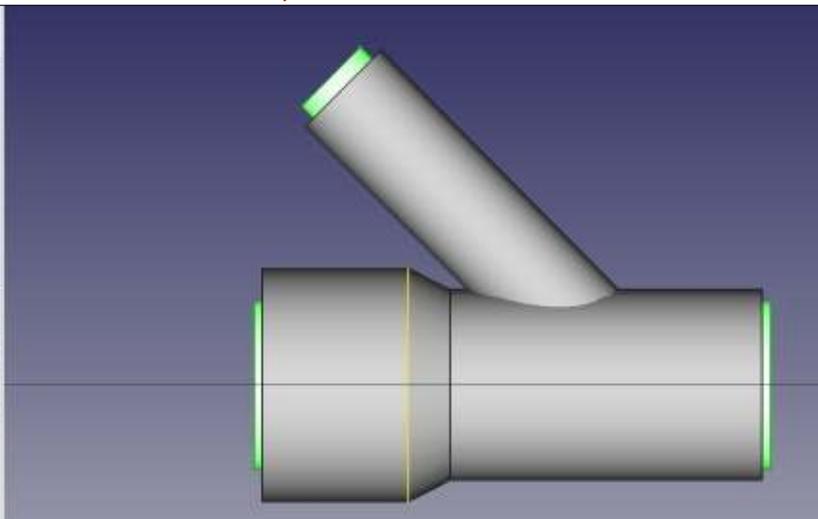
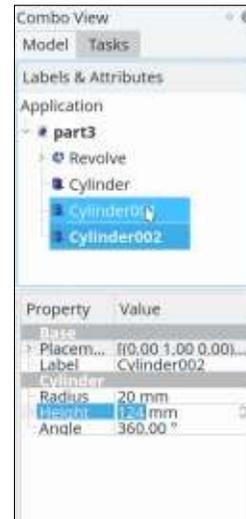
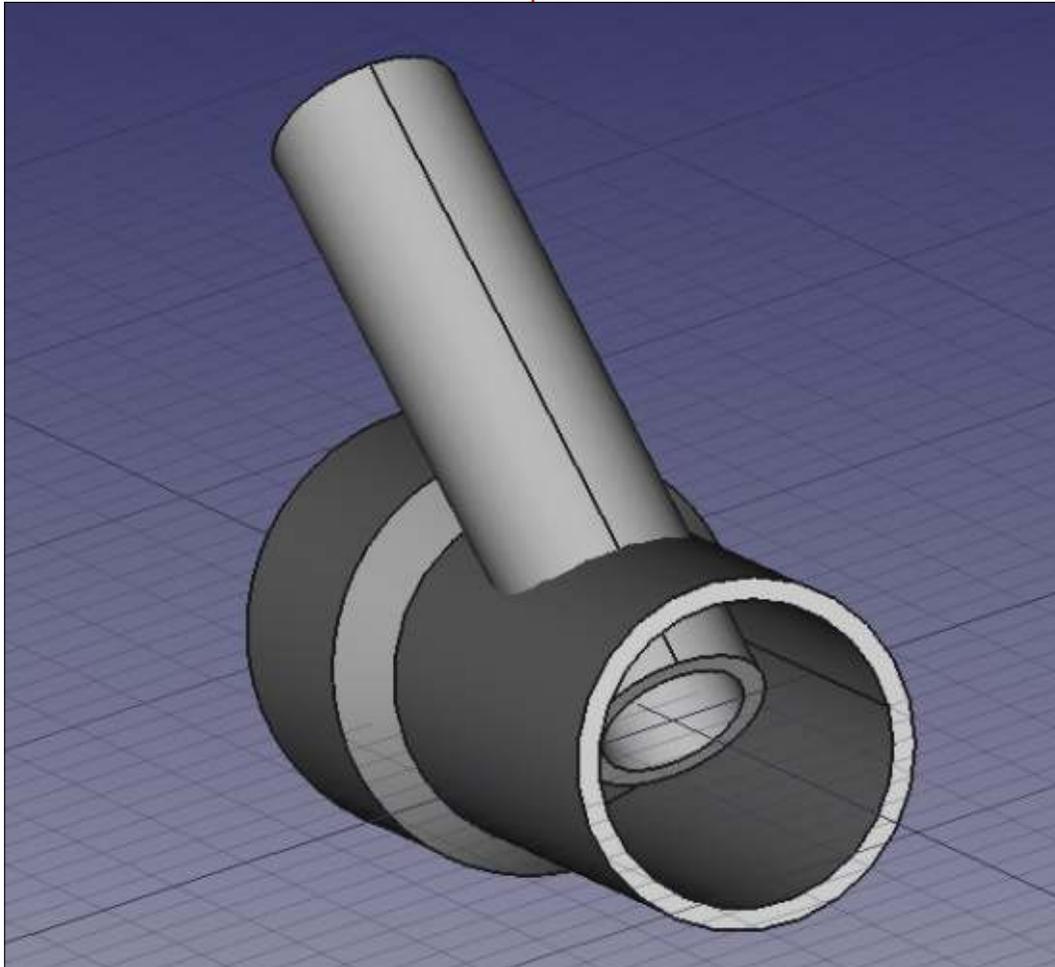
unscroll Placement. We need to change the Axis values to [0.00, 1.00, 0.00] since we will be rotating around the Y axis, and then change Angle to -45 degrees to tilt the Cut object backwards to our left.

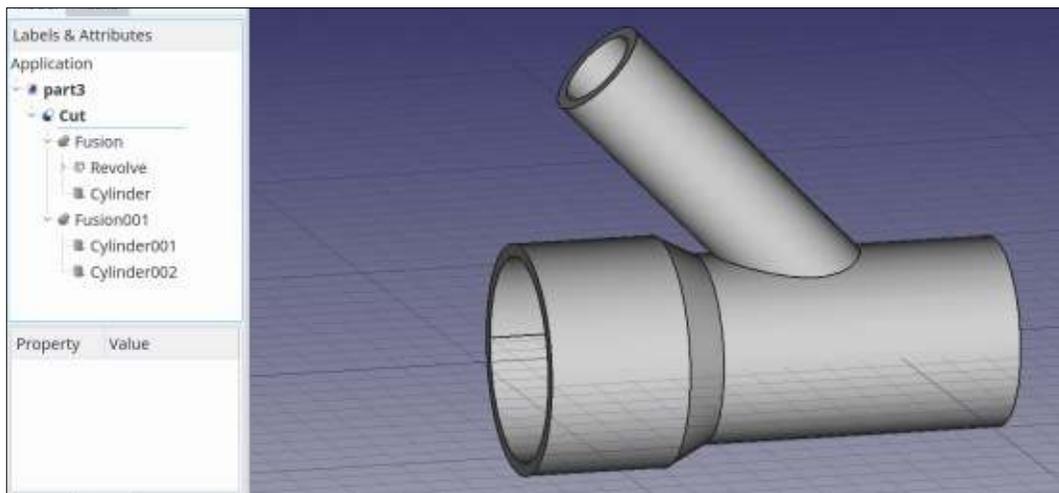
Now select the Revolution object, and modify Placement > Position > x value until the two parts are assembled in their correct position. I needed to specify -200 mm for this to happen.

We still have a couple of problems, however. The small pipe is protruding through the wall of the main tube, but it is not yet cutting a hole in that wall. On the other hand, there is a piece of the small pipe within the large one that also needs to be cut off.

The easiest solution for the first bit is simply to undo our boolean cut operation by simply clicking on the Cut object in Combo view, and deleting it. We should now have three objects in our project: the Revolve object representing the large tube, Cylinder as the outer shape of the small pipe, and Cylinder001 the inner shape of the cutout. The two Cylinder objects will need to be rotated once more to -45 degrees along the Y axis, since the rotation we had applied previously concerned the combined Cut object which we have since erased.

Now, add a third Cylinder object - labelled Cylinder002 - to represent the interior of the large tube. It should have radius 20 mm,





height 124 mm, be rotated on Axis [0.00, 1.00, 0.00] (the Y axis) by +90 degrees, and then placed to the left by 92 mm by changing Position > x value to -92 .

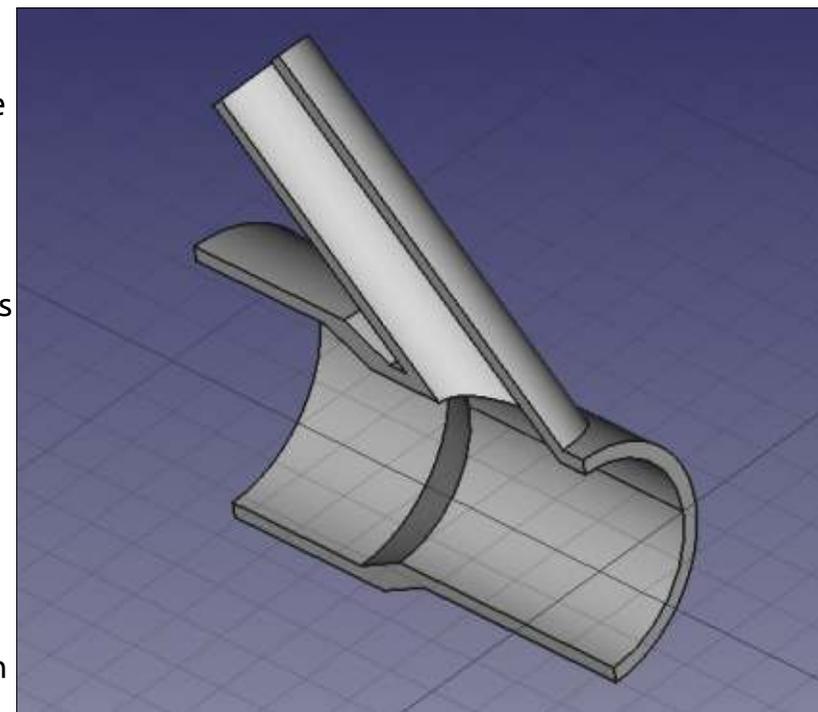
Finally, we can join the two external objects, Revolve and Cylinder, in a single Fusion object: choose Revolve, then Cylinder with the Ctrl-key pressed, and finally menu option Part > Boolean > Union. This should create a new object called "Fusion". Now, do the same with Cylinder001 and Cylinder002, creating "Fusion001". At this point, we should see only the two Fusion objects, one representing the material we are adding to our project, and the second representing the cutout or material we will be subtracting. Now choose these two objects in

order: Fusion, then Fusion001, and choose Part > Boolean > Cut. We should end up with our final, finished tube Y-union. We can note the shape of the union between the two tubes.

This technique of combining all the bits of our project that add material into a single Fusion object, and all cutouts into another before making the final Cut, can come in useful when designing parts with complex CSG geometries. The general workflow is similar to that used in Sketchup, which should mean making the transition from one program to the other is rather painless.

Once completed, our object can be cut open as needed to examine the interior geometry and the

section of each piece, for instance to make sure there is enough material to support any structural or pressure stresses that the finished part may encounter. This cutting open is left as an exercise to the reader. Hint: try Part > Boolean > Intersection with another object.



WHAT NEXT?

In this article on using FreeCAD, we used several techniques to create a complex 3D object, representing a Y-junction between two pipes of different diameters. The Revolution tool was used to create the form of a tube in 3D, from a plane section of the tube wall. The Cylinder primitive was used to create the form of the small pipe attached to the main tube, and then to create cutouts to empty out both forms. In the next part of the series, we will

investigate the use of sketches to implement constraints on segment placing.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the third article on using FreeCAD, we created a complex 3D object representing a Y-junction between two pipes of different diameters.

In this part, we will go back to basics and examine how constraints can be used to draw complex planar shapes, that can then be used as a basis to create figures in 3D.

Users of traditional CAD software such as AutoCAD or LibreCAD are well accustomed to two sets of techniques that help draw complex shapes from individual elementary shapes such as lines or arcs. The first set of techniques is the different ways in which an elementary shape can be defined in these applications. For instance, a straight-line segment may be defined by indicating both

ends of the segment. But it could also be defined as the tangent to a circle at a certain point, and with a specific length. A further possibility is to define a line segment as being parallel to a previous segment of the same length and offset to a specific distance. As for circles, they may be defined from a center and a radius, or from three points that are not on the same straight line, and so forth.

A second set of techniques that may be considered quite basic in traditional applications is the use of layers. In a program such as LibreCAD, default line width, color and style (dashed, dotted, ...) may be defined for each layer. A handy technique is thus to place the main elements of each drawing in one layer, while another is used to indicate dimensions, and a third to draw auxiliary items to help construction. When exporting the drawing, individual layers may be hidden with a single mouse click.

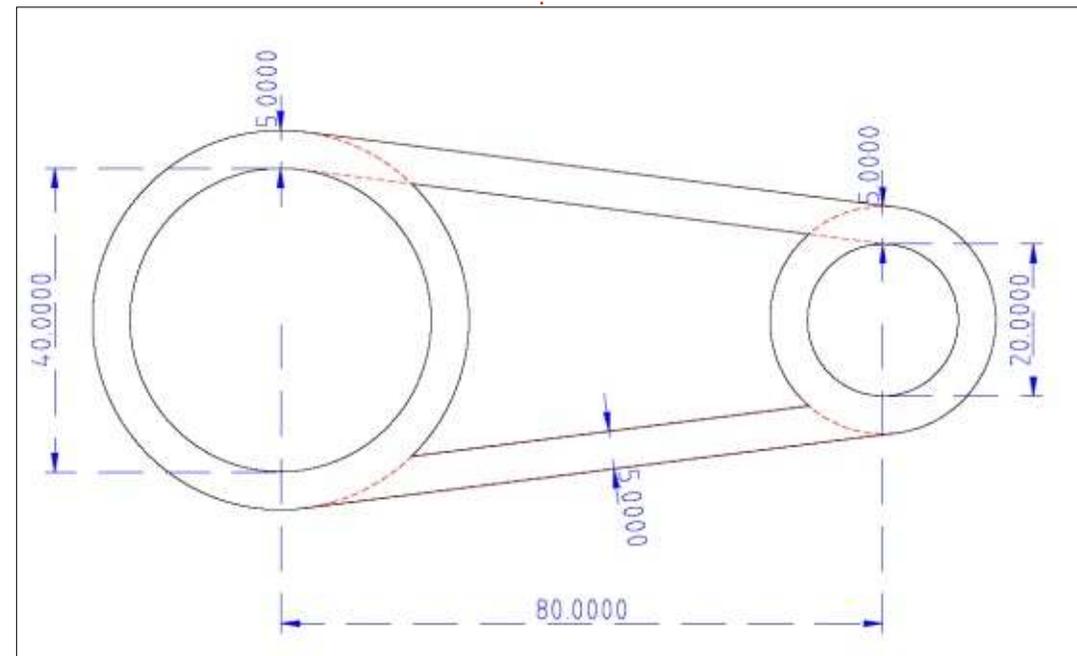
At this point in time, the use of such techniques is perhaps not as

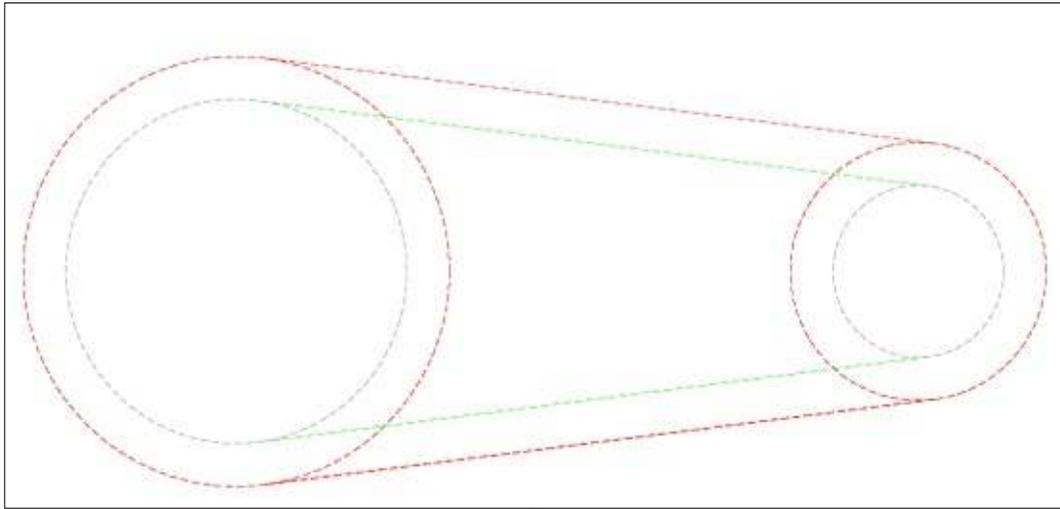
easily visible in FreeCAD version 0.15 as could be desired. As has been pointed out in a previous part of this series, this application is known to still be very much in development, so there is hope that such features may be made more accessible as the application evolves. Version 0.16 - in the repositories for Ubuntu 17.04 - already hints at the presence of layers within a sketch. In the meantime, users of version 0.15 - in the repositories for Ubuntu 16.04 LTS and Linux Mint 18 - can today work around these limitations by

using other features that are more clearly available in the program. This is the subject of this article.

AN EXAMPLE

To visualize the problem, let us begin by drawing up a simple flat piece, with a geometry similar to that used in an engine connecting rod. This part is basically made up of two rings, one at each end, connected with two rectangular spars. The center of the rod has been removed, possibly to lighten





the part.

To draw this part in a traditional manner, the first step would be to create a layer that holds only auxiliary lines – that will not be part of the finished drawing. For instance, one could begin by drawing all the circles.

Once the circles are defined, the external edge of the spars can be placed (in red). The easiest way in a traditional CAD application is to specify a line segment as tangent to both red circles, once for the upper spar and once for the lower. Then, the inner edge of each spar needs to be drawn (in green). There are several ways of doing this. The same procedure may be used, with each segment defined

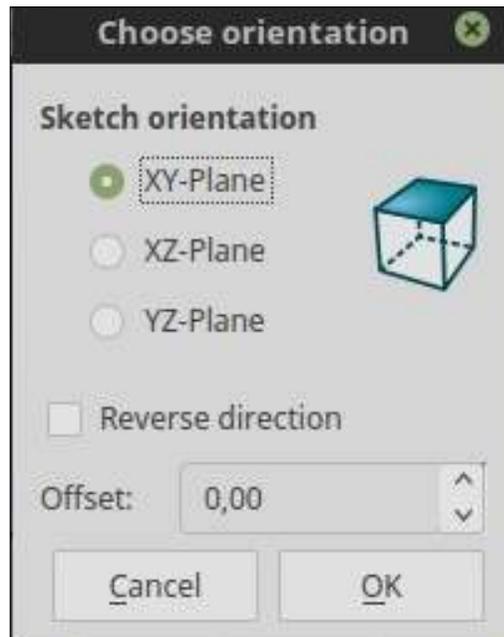
as tangent to the grey circles. An alternative way of doing it is to define the green segments as parallel to the red segments that have already been placed, while specifying an offset - in this case, 5mm.

We can then create a second - main - layer, and draw the segments and arcs of the final part. Selecting snap-to-intersection instead of the more usual snap-to-grid allows us to carefully terminate each element precisely at the intersections between lines and circles.

ON TO FREECAD

To create the same part in FreeCAD, let us begin by creating a

new project. Now, we could proceed as in the previous episodes, by going into the Draft workbench and setting up the elements of the drawing using the tools available there for drawing lines, circles and arcs. However, though we can snap element vertices to the grid, or even to another element, there is no way of ensuring that a line stays tangent to a circle or an arc. For this reason, we will go into another workbench, the Sketcher. Here, we will create a new Sketch object, within the X-Y (horizontal) plane. This type of object represents a flat drawing, considered as a separate entity from the rest of our project.



Once editing the new Sketch, we can begin by changing the default grid size from 10 to 5mm, since the dimensions of our part are all multiples of 5mm. Let us draw the two circles that define the left ring of our part, with respective radii of 20 and 25mm. It is interesting to note how the shape of the mouse pointer changes when creating a new point. In a general situation, the shape is a red circle, with the new point's coordinates in blue beside it. However, if we click on an existing point, a red dot shows up beside the circle. If we click in this situation, we can choose to link both points. This is ideal when we are drawing two circles with the same center. If, later on, we move one of the center vertices, both will move at the same time - and both circles will be displaced an equal distance.

Other options include placing the new point on a segment of an arc, thus linking the point to the arc. If we then displace one of the two objects, the movement of the other will equally be constrained. Likewise, when a line segment is drawn and one of the vertices has already been placed, a horizontal



Just creating a simple point



Snapping onto an existing point



Snapping onto an arc

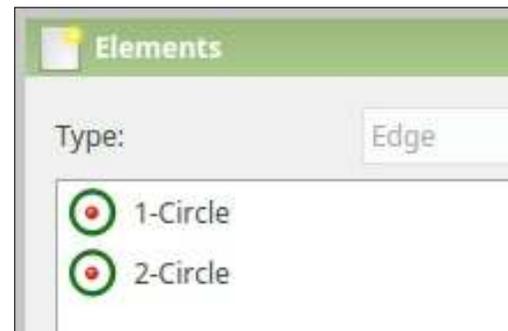


One vertex of a segment with a horizontal constraint



or a vertical constraint can be placed on the segment by placing the second vertex when the mouse cursor contains the corresponding red horizontal or vertical bar.

Once the two circles have been



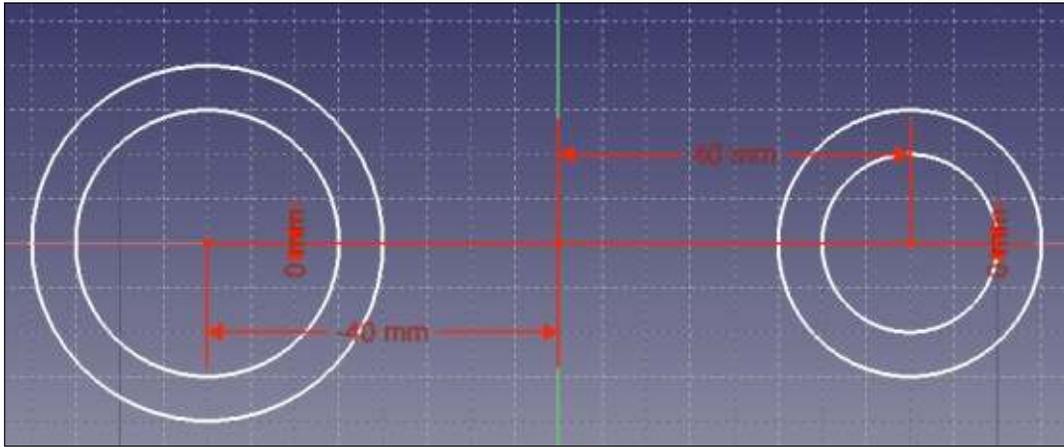
created, we can inspect the number of elements created in the "Elements" window, and select each element by clicking on it.

Once selected, each element can be moved around using the mouse. This is fine for the time being, but will introduce a difficulty when the connecting segments are placed to create the connecting rod itself: altering the position of a segment may very well end up by moving one of the circles that is connected to it, thus

making sure the two elements remain in connection. We do not want this to happen; on the contrary, we want the circles to determine the position of the segments. So let us place a constraint on the position of each of our circles. Click on the center of the circles - which should be a single round dot - and then use the constraint toolbar to choose the "lock" constraint which has an icon shaped, rather appropriately, like a padlock.

Let us continue by drawing the remaining circles, and locking them into place. We should now see four constraints, two for each center (one horizontal and one vertical), within the "Constraints" window at the left of the screen. The four circles should look approximately like the image on the next page (top left).

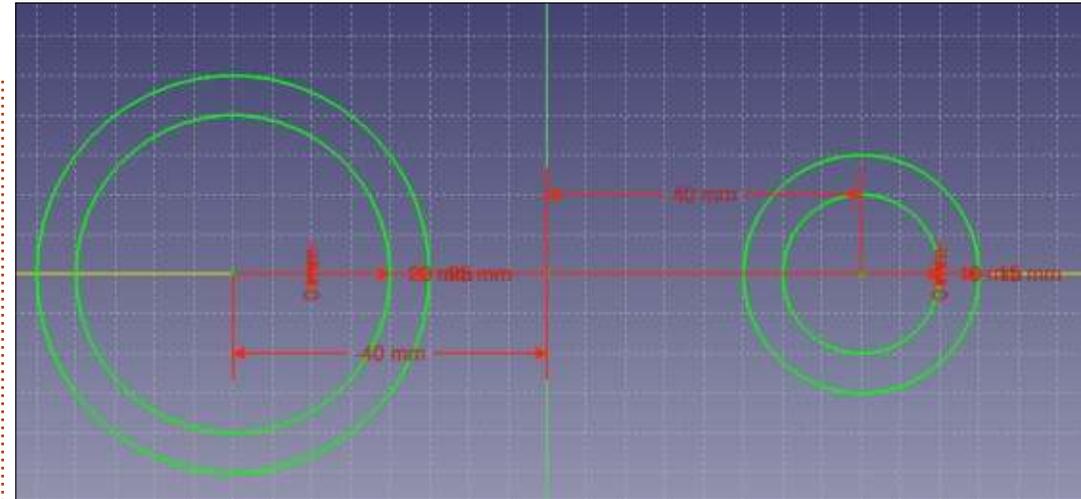
We have constrained (fixed) the position of the center of each circle. However, their radii are not yet constrained, and could be altered when connecting segments are added. To fix their size, select



each circle in turn and select the "Fix de radius" constraint, a red circle with a bar on it, from the constraints toolbar. Our sketch should now change aspect, with all elements changing color to become green. This indicates our sketch is fully constrained: existing elements cannot be further moved, unless at least one of the constraints is lifted.

We are now ready to place the connecting segments on our - now immobile - circles. Start by drawing a line segment from one of the external circles, to the other. By carefully placing the mouse, we should be able to constrain the segment's vertices so that each remain on a circle - this is the symbol of a red arc with a dot in its center. However, it should rapidly become clear that this segment is

not yet necessarily tangent to each circle. To impose this new set of constraints, click on the dot representing the segment terminal vertex, then click on the circle. Finally, choose the "Create a tangent" constraint. The operation will need to be repeated several times, once for each intersection between a segment and a circle. It may also be necessary to remove spurious horizontal constraints on our segments, if they should appear during construction.



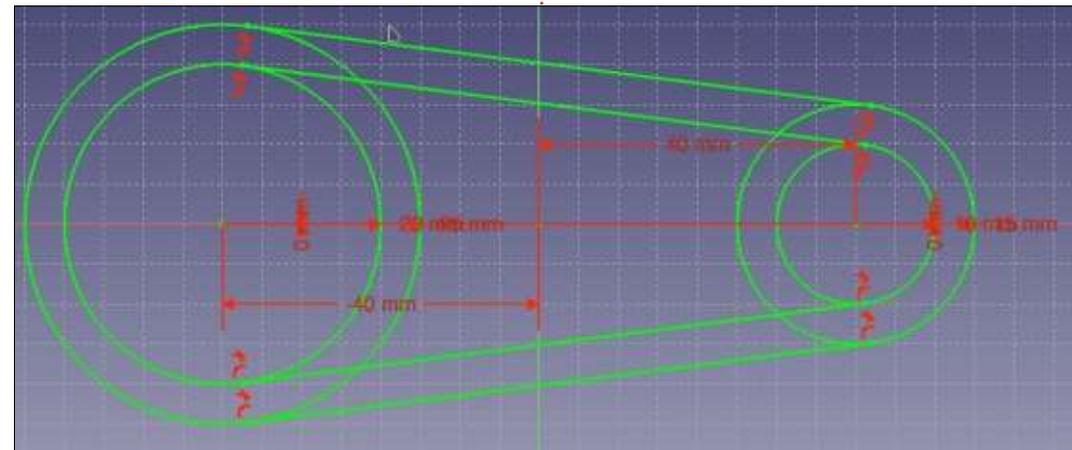
The final result, with all four segments placed and the elements completely constrained, should appear like that shown below.

Up until this point, we have been actively editing our Sketch object. We can now close this object, returning to the standard FreeCAD view, and examine our handiwork. All constraints have

disappeared in the normal view, and we are left with our shape's individual elements - all grouped together in a single planar Sketch object.

BUILDING THE FINAL SHAPE

What we have obtained so far is, in fact, just the auxiliary, constructive aides to help us place



our final arcs and segments. It would be nice to see them in a different color and style, to help us distinguish between helper traces and elements belonging to the final drawing. There are two ways of going about this. The first is to exit Sketch edition. By clicking once on the Sketch, we can proceed down to the "Property" window, and change both "Line Color" and "Draw Style" for all lines in the Sketch in a single action.

Property	Value
Base	
Bounding ...	false
Deviation	0,50
Display Mo...	Wireframe
Draw Style	Dashed
Lighting	Two side
Line Color	 [255, 0, 0]
Line Width	2,00
Point Color	[255, 255, 255]

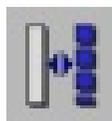
Then, we can proceed to the "Draft" workbench. Once here, we can use the drawing tools from the Draft toolbar (with yellow/black icons) to draw on top of the Sketch.

Intersections between Sketch elements can be made easily detectable if a point is placed at each intersection from within the

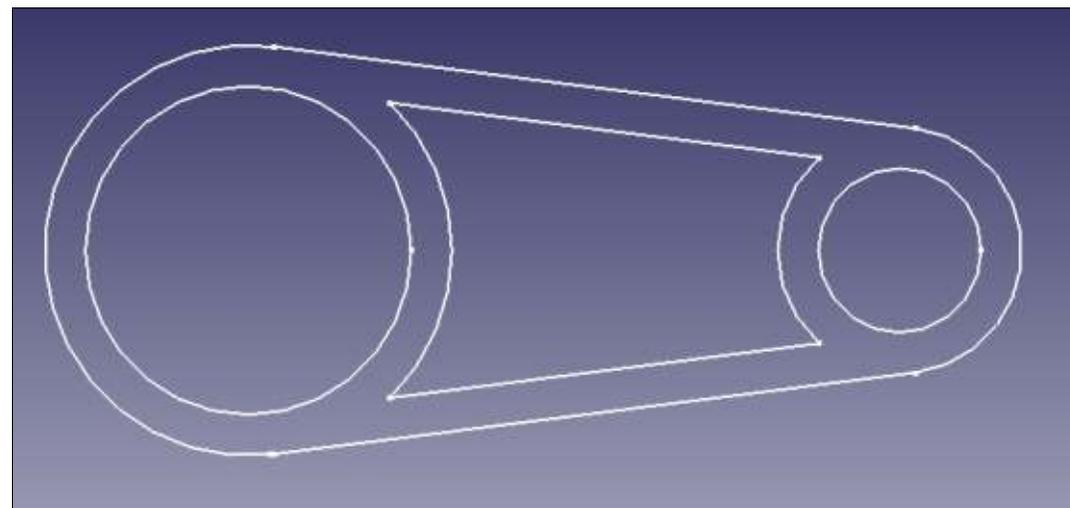
Sketch. To do so, create a point using the tools with the icon with the red dot. When placing the point, snap to one of the elements creating the intersection, for example to the circle. Then click on the new point, click on the other element of the intersection - for example, a line segment - and create a new constraint of type "Fix a point onto an object". The point should then be fixed by being constrained twice, once to each element, and thus should stay in place at the intersection.



Once the points at the intersections have been created, lines in the Draft workbench and can be snapped to these points if the "Snap to intersection" option is chosen (the icon with a green 'X' from the snap toolbar).



A second way of drawing the final shape of our part is to do it directly inside the

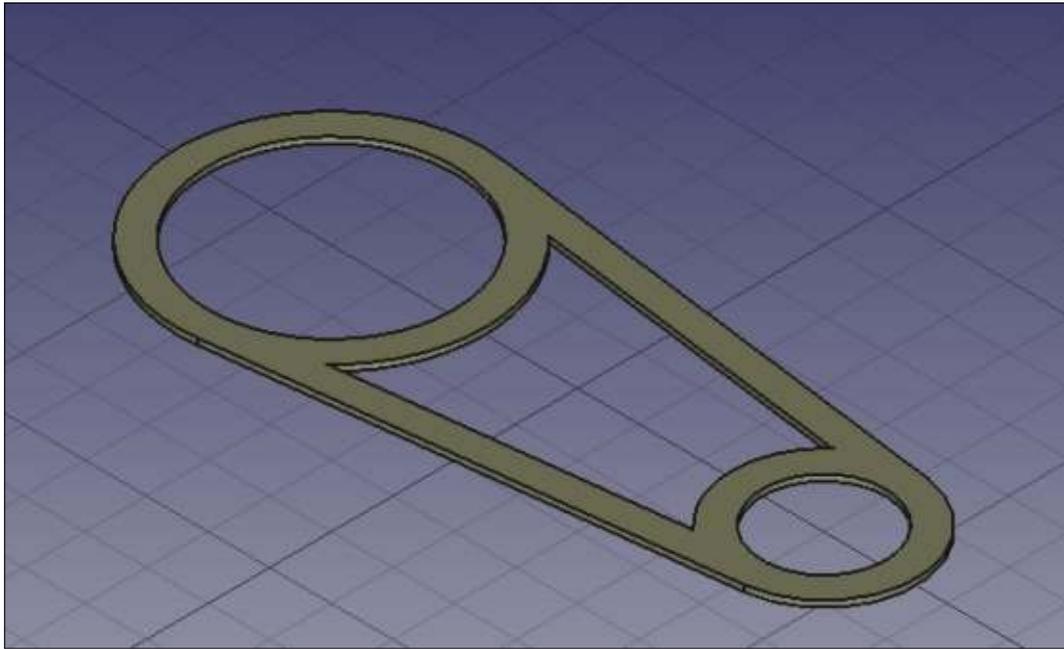


Sketch. Go back into editing the Sketch object (double-click on the object), selecting all the lines and switch them all to "Construction mode" using tool . This will change their color to blue - and make them disappear from the Sketch when this is not being viewed from within edition mode. We can then proceed by creating the points at intersections, as needed. Continue by adding further elements to the Sketch object, taking these constructive elements as a guide. These elements, drawn in normal mode, will appear when not editing the Sketch. Since they are based upon other elements in constructive mode, some elements may need to be converted to normal mode using the same tool: .

At this point, we can use the Sketch object to print a diagram in two dimensions. However, we cannot use it directly to create a three-dimensional part. To do so, we must begin by going to workbench Draft, and there convert the Sketch object into a collection of drawing elements using the appropriate conversion tool: .



Once this has been done, we can go to the Part workbench, extrude each object (the two circles and the two outline shapes) individually, and then combine them using boolean operations (Fusion and Cut out) to create the final piece in 3D, as described in Part 2 of this series.



WHAT NEXT?

In this article on using FreeCAD, we created a Sketch object, to place individual drawing elements such as lines, arcs and points, in a precise relationship to each other using constraints. We noted the use of Construction mode elements within the Sketch object, to aid construction of the complete diagram while not appearing in the final drawing. In the next part of the series, we will change scale altogether and work on an architectural project.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



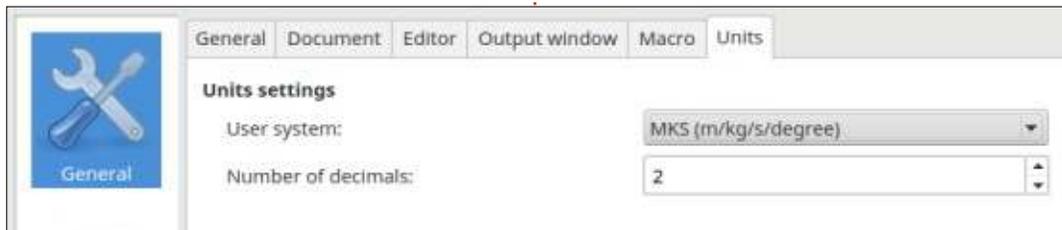
HOW-TO

Written by Alan Ward

In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the fourth article on using FreeCAD, we created a Sketch object, to place individual drawing elements such as lines, arcs, and points, in a precise relationship to each other using constraints. We noted the use of Construction mode elements within the Sketch object, to aid construction of the complete diagram while not appearing in the final drawing. In this part of the series, we will change scale altogether and work on an architectural project.

SETTING UP OUR UNITS

The main difference between



the small technical parts we have drawn up so far and an architectural project lies in the units used. Small parts tend to be easily measured in millimeters. However, in the case of a building it would make sense to use larger units such as meters (in the metric system). To set up the environment, let us begin by starting up FreeCAD, and choose a new Project. Then go to menu option Edit, and choose Preferences. Within the General pane, choose the Units sub-pane and change "User system" settings from millimeters to meters - or the imperial system if that is your preference.

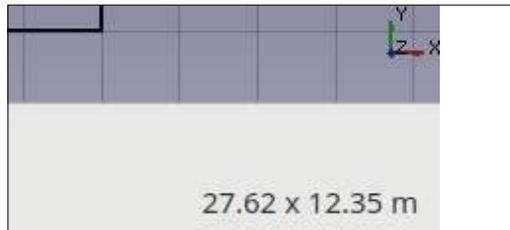
Though we have changed the main units, the auxiliary grid that comes up to help us place elements will still use the former aperture value between grid lines. Having a 1x1 m grid with lines

Intro To FreeCAD - Pt5



every millimeter may not be of much use to build a house. So, continue within the Preferences dialog box, and go to the Draft pane. Here, choose the Grids and Spacing sub-pane, and change the value for "Grid spacing".

As a quirk, it may be necessary to save your project, close FreeCAD and open it once more to actually see the grid with its new size. Remember to use the mouse wheel to zoom out to see a sufficiently large amount of the X-Y plane. Remember the visible area

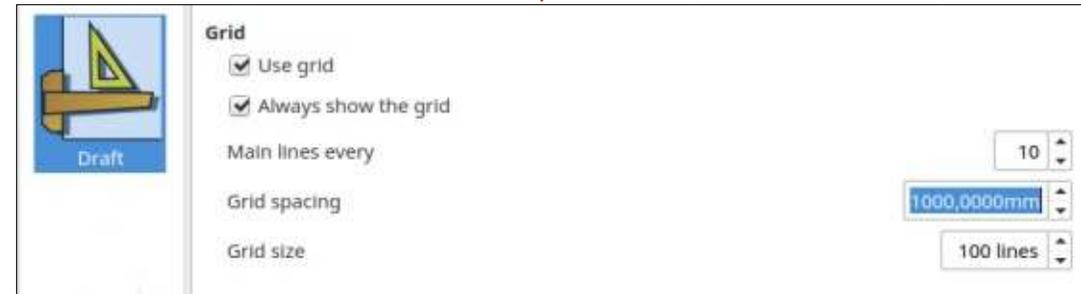


is always indicated in the lower right corner of the display.

USING THE ARCH WORKBENCH

Let us begin actual design by noting the existence of a specific "Arch" workbench. This toolset has been specifically conceived to draw designs of buildings as a Building Integrated Model, or BIM. In this concept, building elements are labelled as such: walls become a Wall object, openings such as windows or doors become Window objects, and objects such as a Support, a Roof or a complete Building object can be specified using the appropriate toolbar.

To begin a new building, one



HOWTO - FREECAD

could start in the Draft workbench, by drawing out the general floorplan. Simple lines are sufficient to indicate the position of each wall. For the time being, no mention needs to be made of wall widths and of the placement of openings. For instance, one could draw up the following building, representing a small school module with two classrooms or laboratories, one preparation room between them, and an exterior corridor.

Once the floorplan has been drawn in this fashion, move to the Arch workbench, and select all lines. Then select the Wall tool, and all lines magically become proper Wall objects. A default value of 3m is used for wall heights, and 0.2m for wall

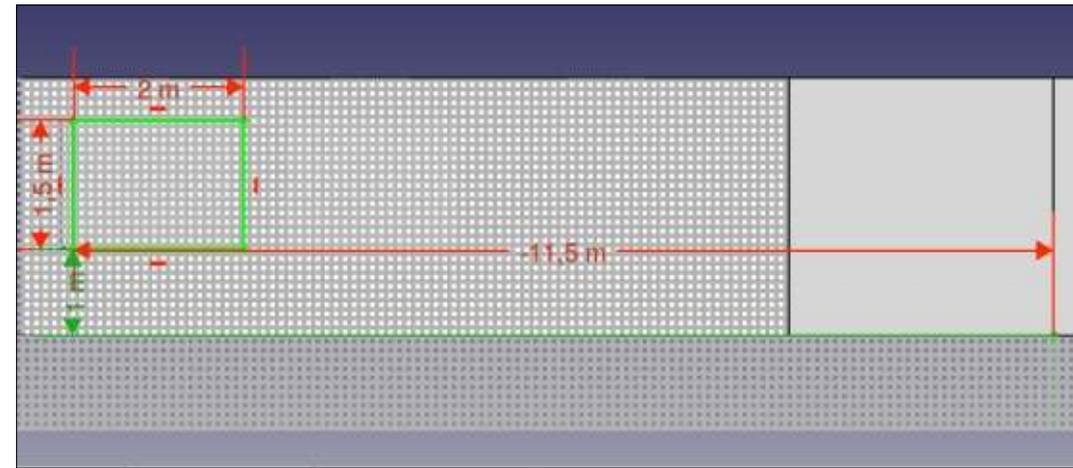


thicknesses.

To continue, one would need to place openings. To do so, start by creating a Sketch object attached to the corresponding wall object. This Sketch object needs to contain a closed form that represents the shape of the opening.

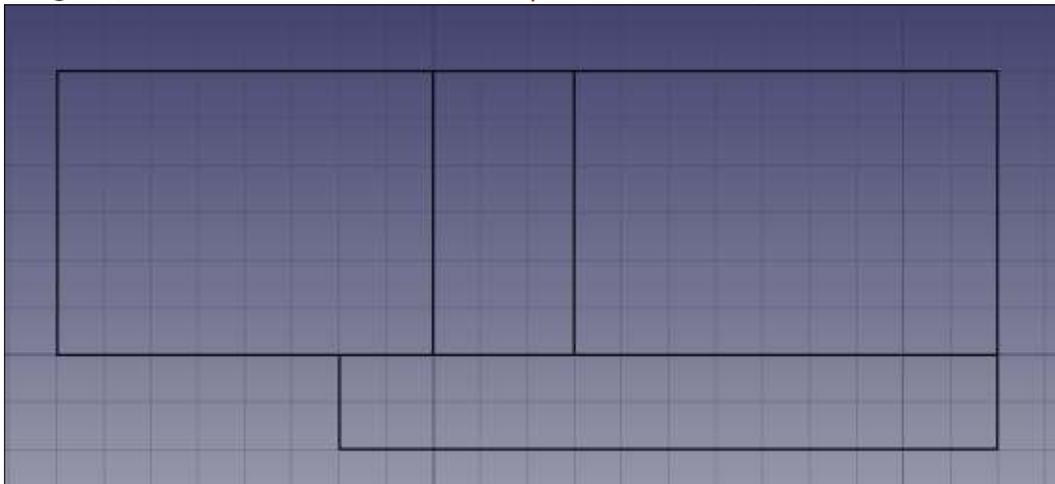
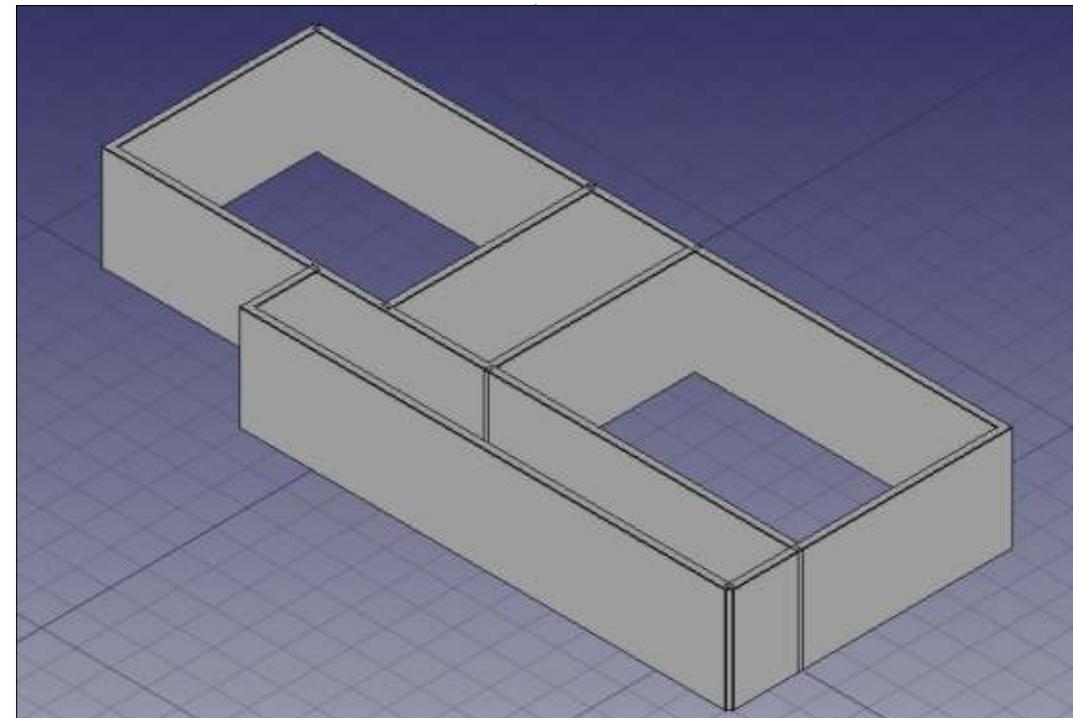
For instance, one could create a simple rectangular window for one of the classrooms. Choose the wall object on which to draw, go to the Sketch workbench, and create a new Sketch object. Using the Sketch constraint system described in the previous part of this series, the four lines that describe the position of the new window can be placed with some precision.

Once finished, close Sketch edition mode, and make sure the

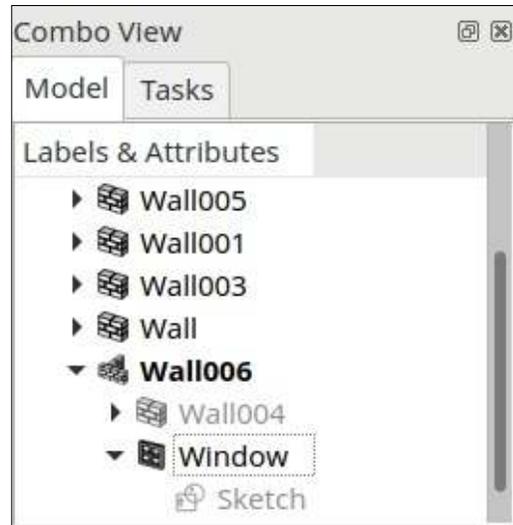


new Sketch object is actually associated to and within the Wall object. Then go to the Arch workbench and transform the Sketch into a Window object. Within the Combo View to our left,

we should see how the original Wall object (Wall004) has been replaced by a new Wall (Wall006), that contains both the original shape and a new Window object. This latter object in turn contains



the Sketch. Each internal object defines the shape of the external one that contains it.



The Arch workbench and the BIM approach to building an architectural project has several advantages. One is ease of use, specially when working with a rapid prototyping approach, under which several models may be tested to explore the possibilities of a new site or construction idea, before settling on a particular solution. Just lay down the lines, and build up your walls. A second advantage is that, once the new building's elements have been defined as such, the information contained within the model can be used to automate calculations, such as surface area or building volume.

On the other hand, this part of the FreeCAD software is possibly the least mature for the time being. As work in progress, developers seem to be exploring the possibilities of the BIM approach, and there are some rough edges. Element placement is still rather fiddly, and much care must be given by the user to place objects correctly within their containing objects. Roof creation is an art by itself. Object conception is made using regular parts that may, or may not, correspond to the real world. For instance, working with a building interior floor that is not completely flat may become a challenge, as may be working with walls that have varying levels of thickness along their length. Solving corner intersections between walls may also become an issue.

For this reason, the choice of using the Arch workbench must be left entirely to each individual user. Some people may hate the limits this approach places on their workflow, while others may revel in the ease of creation of modern-looking buildings. In any case, one of the designers of FreeCAD has created a rather good tutorial on

its features, that may be of interest as further reading: https://www.freecadweb.org/wiki/Arch_tutorial

A MORE TRADITIONAL APPROACH

Users who do not need or do not care for the features of BIM may feel more comfortable using the more standard tools in the

Draft and Sketch workbenches to draw traditional views of our construction. However, we should always bear in mind that FreeCAD is a 3D-oriented computer design program. For this reason, limiting ourselves to the more traditional conception of architectural drawings - planar projections and perspectives - can certainly work, but we would lose out on the capabilities of visualizing the building in the third dimension



Photo credit: Wikipedia user Zarateman (<https://commons.wikimedia.org/wiki/User:Zarateman>).
Original link: https://ca.wikipedia.org/wiki/Catedral_de_Tarragona#/media/File:Tarragona_-_Catedral,_claustro_01.JPG

that FreeCAD allows us.

To see how this could work out, let us project something that would be rather difficult to draw up using the Arch workbench: an arcaded cloister. This is typically a square or rectangular space that consists in a covered walkway surrounding a central open area with vegetation, an arrangement that has been used both in a certain type of Islamic garden (e.g. Patio de los Leones, Alhambra Palace in Granada, Spain) and in Christian churches, both in Romanic and Gothic styles. It is this

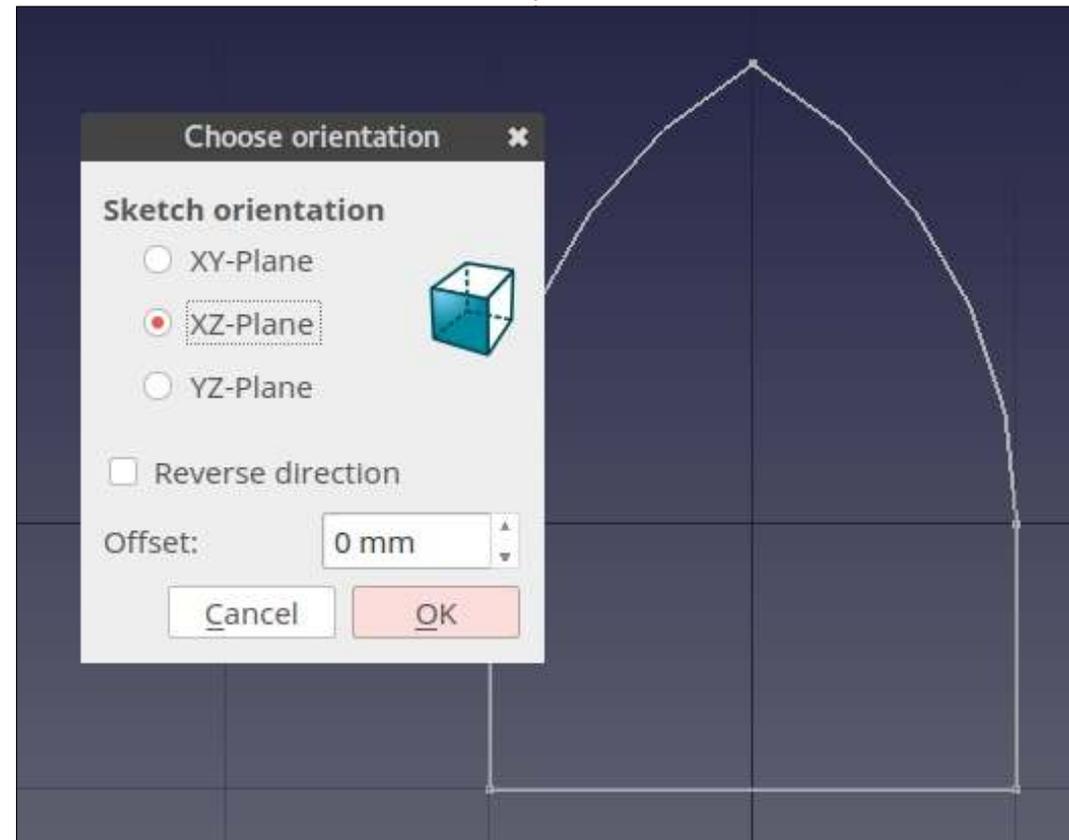
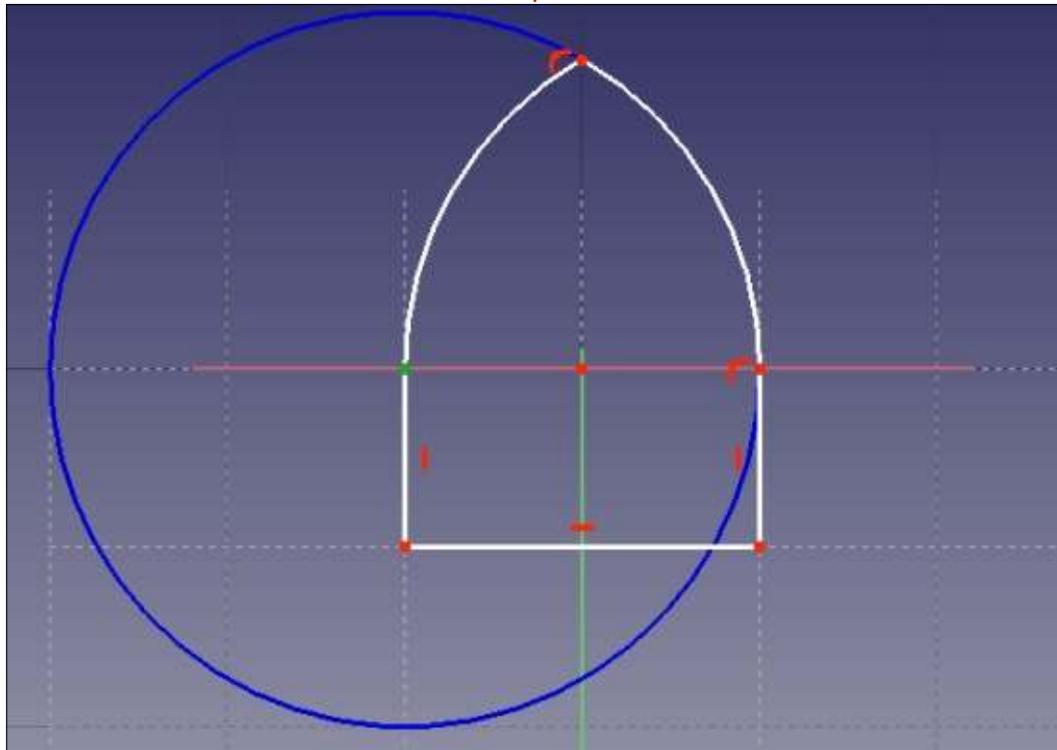
latter that will inspire us to design the cloister arches, such as those found in Tarragona Cathedral, Catalonia.

Let us begin by drawing a simple Gothic arch. In FreeCAD (bottom left), go to the Draft workbench and set the grid aperture to 500mm. Then go to the Sketcher workbench, and start a new Sketch object. Here, we will probably need to alter the grid aperture a second time, to 0.5 m (same value, different units). We can then begin by drawing the base of our arch, using the grid as a

support. Using simple values, I drew two vertical lines 0.5m high and set 1m apart. I was then able to draw the arched top part of our figure. In its most traditional form, this is the combination of two circular arcs. In the figure below, a construction circle has been drawn in blue: centered on the top of the leftmost vertical segment (green point), it passes through the top of the right post, giving it a radius of 1m. Using this circle, draw an arc of a circle from the rightmost post up towards the centerline of the

figure (snap to the vertical grid line), giving the right part of the arch. Now draw a similar circle centered on the top of the right post, and draw the left part of the arch.

Once we exit editing mode, we obtain a nice outline of our Gothic arch, which would in this case be called an equilateral arch. Since real architectural elements have volume, we now need to find a way of transforming this into a volume, keeping as much in character with



real Gothic building techniques as possible. However, when one studies carefully the stone parts that go into making up real Gothic arches, one almost always finds that the same profile has been cut all along soft pieces of white sandstone. Therefore, if we can create such a profile and somehow sweep it along our arch outline, we should end up with a perfect three dimensional arch frame.

Please do make sure this shape is complete, i.e. that all lines connect well to each other. The penalty for not doing so would be a final arch shape that is built up only in part - with another part missing.

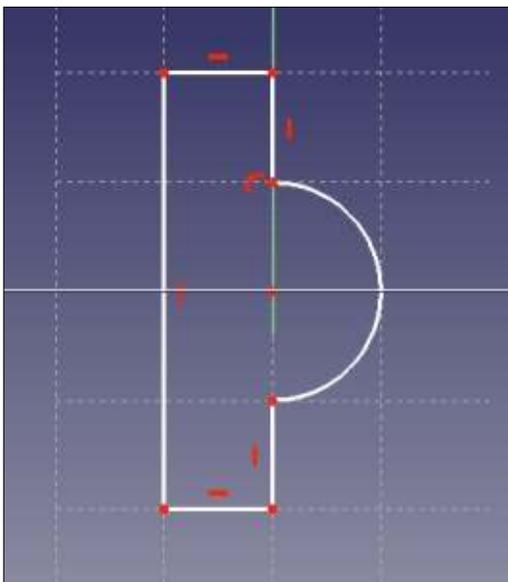
I drew my arch sketch (previous page, bottom right) in the XY plane. I will now create a second Sketch object, but at right angles within the XZ plane: the X-axis is in the left-right direction, Y is top-bottom, and Z goes along the arch depth.

Within this new Sketch

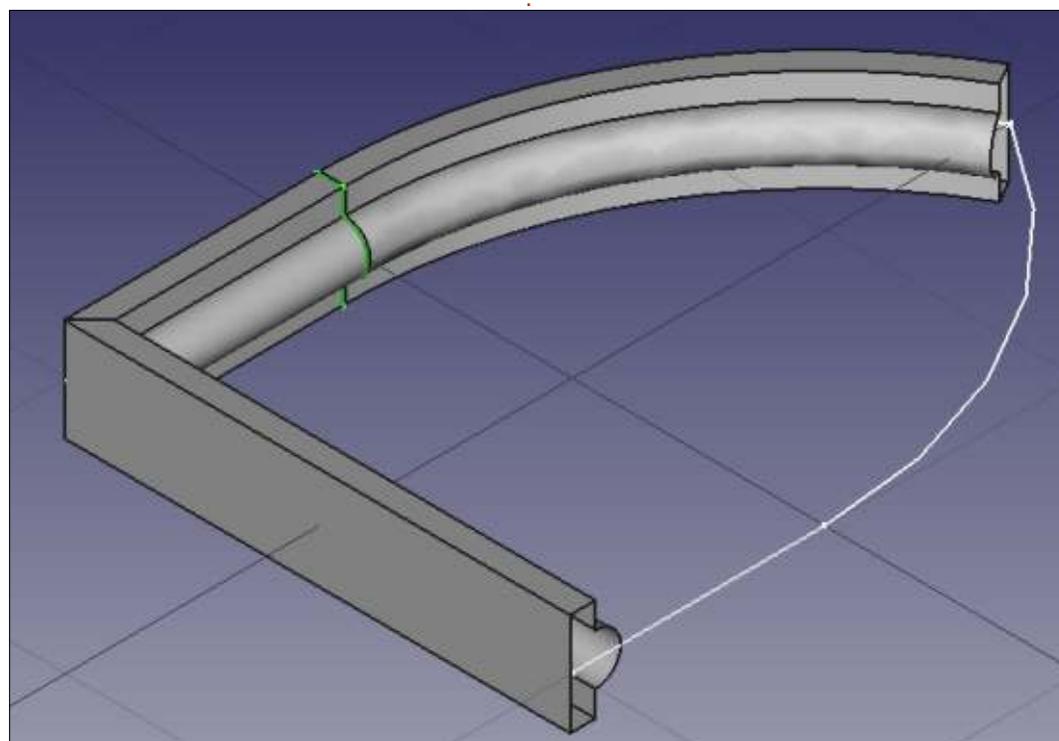
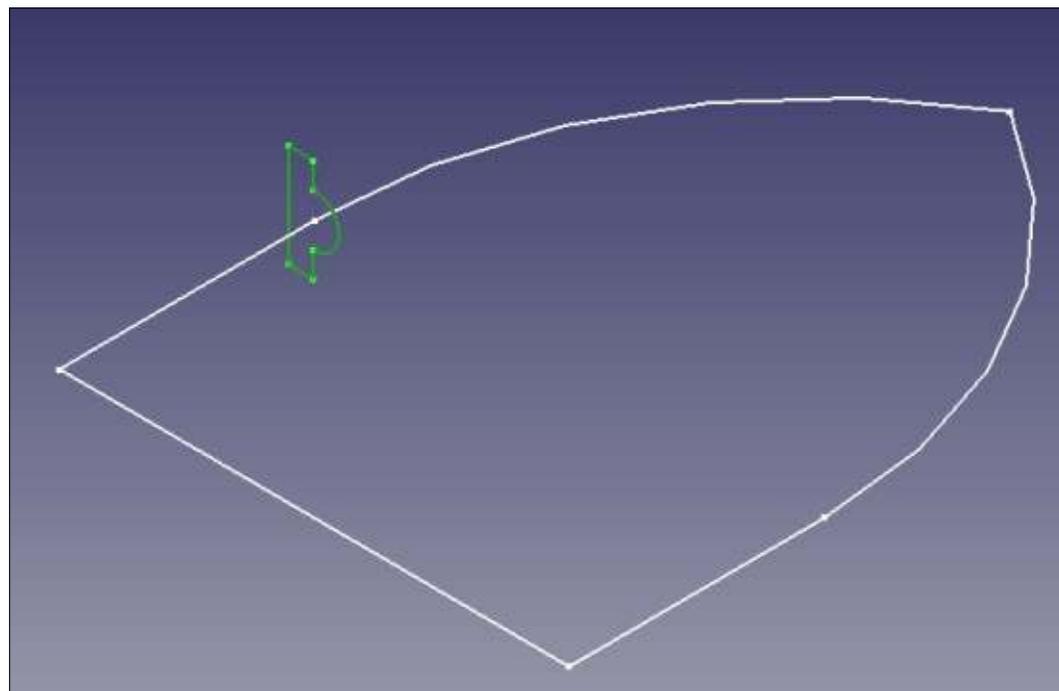
Please note that the grid dimensions have been changed. We are now working with an aperture of 5 cm, giving an

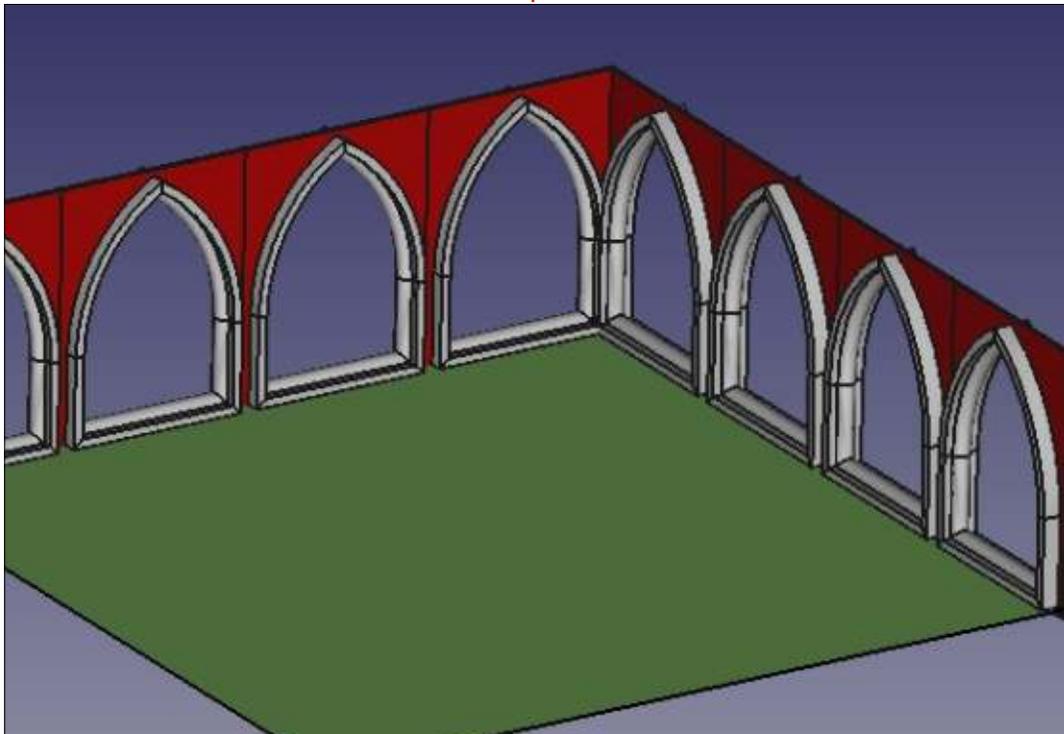
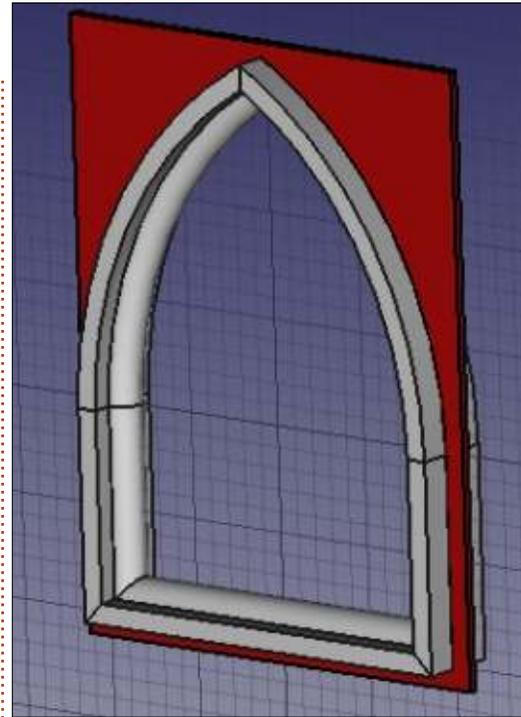
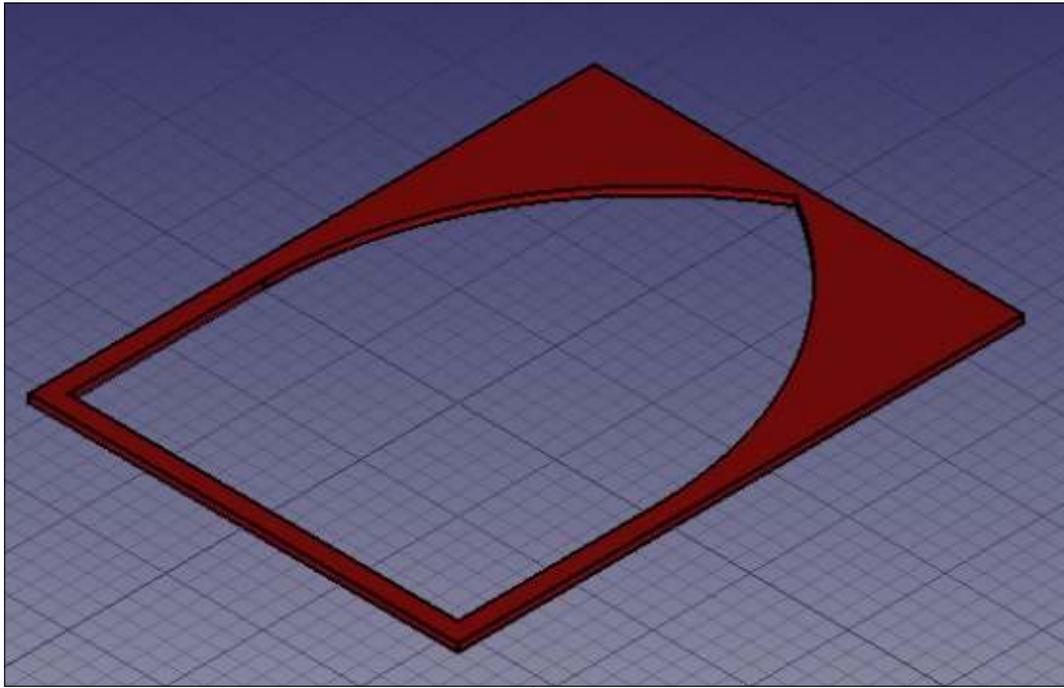
element profile that fits within a 10x20 cm rectangle. Close this second sketch, and we can now proceed to the Part workbench. In this, select the Sweep tool.

In this tool, we select one of the sketches as the



form to be swept, and the other as the path to sweep it along. However, there are some caveats. The main point is the relative placement of each sketch. Their relative positions will be respected when one is swept along the other. It is best to ensure the swept form is actually placed upon the path before proceeding. In my case, I needed to displace it some 0.5 m leftward of the place where it had been created, in the vicinity of the





prepared, what we have actually achieved is a computer model of the elegant carved stone frame of the arch. However, this arch would not work well in architecture by itself, but must be part of a wall or a complete structure.

To create this wall, I draw a simple rectangle 1.1 m wide by 2.1 m high in the XY plane, overlapping the arch by several centimeters on all sides. I then extruded this flat piece as a rectangular volume - as in Part II of this series, which I then dyed red. The final touch is to make a Pocket indentation in this rectangle, to accommodate our arch. This done by selecting one of the outward faces of the rectangle, and creating a Sketch on this face, with the same shape as our original arch. A copy of the arch sketch could also be made, and then attached to the face of the rectangle. Then use the Pocket tool in the Part design workbench to push the sketch form "inwards". If sufficient depth is given to it, it will end up by making a hole straight through our rectangle, in the correct shape to place our arch volume in.

coordinate system's origin. For some reason, I also needed to move the second sketch downward a small amount - some 25 mm, presumably to ensure it was not in the vicinity of a connection point between segments of our arch sketch.

In this partial view, one can see how the wireframe arch sketch is acting as a support for the second, profile, sketch, as it goes around using the first sketch as its path.

Once the final arch tridimensional shape has been

The arch Sweep volume and the

HOWTO - FREECAD

red Pocket can then be selected, and united into a single Component representing a complete architectural module.

This module can then be copied and pasted several times, to form a series of arches. Each element will need to be displaced and perhaps also rotated into its final place, working with the object's Data pane in the lower left toolbox. This series of arches can be further replicated, until a complete architectural ensemble is formed.

WHAT NEXT?

In this article on using FreeCAD, we worked on an architectural project in two different ways. In the first place, we used the Arch workbench to create a modern architectural project, in which supplementary information is given to the computer, so using FreeCAD to create a Building Integrated Model (BIM). Since this approach is in an early stage of development, and is limited to simple forms, we then used a more traditional approach to create volumes in the same way as in previous projects, but on a larger scale. The sweeping technique

allowed us to create an element with the shape of an arch by sweeping one sketch (a profile) around another sketch (the outline of an arch).

In the next part of this series, we will extend FreeCAD's possibilities using a little Python programming, to create a helicoidal surface in the shape of a gear wheel.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that it still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the fifth article on using FreeCAD, we worked on an architectural project in two different ways. In the first place, we used the Arch workbench to create a modern architectural project, in which supplementary information is given to the computer, so using FreeCAD to create a Building Integrated Model (BIM). Since this approach is in an early stage of development, and is limited to simple forms, we then used a more traditional approach to create volumes in the same way as in previous projects, but on a larger scale. The sweeping technique allowed us to create an element with the shape of an arch by sweeping one sketch (a profile) around another sketch (the outline of an arch).

In today's part of this series, we will extend FreeCAD's possibilities

using a little Python programming to create a helicoidal surface in the shape of a mechanical gear wheel.

PROGRAMMING FREECAD? AND WHY PYTHON?

As many readers will be aware, the world of program language implementation is divided into two main categories. There are programming languages such as C or Fortran for which the source code needs to be compiled (into our computer's machine language) in order to be executed. There are also interpreted languages that do not need to be compiled (or "translated", to give a mental picture of what is happening when using a compiler), but may be interpreted directly by a special program on the user's computer, called an interpreter. This is the

case of many programming languages with a wide acceptance in our days, such as PHP on servers or Python on users' computers. As a side-note, the Java language tends to pertain mostly to the former, compiled category (though with caveats), while the very similarly termed Javascript is actually quite a different beast and is mostly used interpreted by web browsers.

FreeCAD has been built in Python, thus an interpreted language. This is quite convenient for several reasons. In the first place, it makes the application more easy to transport to other computer architectures and operating systems, as long as a Python interpreter is available for the desired platform - and Python is getting quite ubiquitous, indeed.

In the second place, we can open a console view of FreeCAD's inner workings by simply choosing menu option View > Views > Python console. Each action we carry out through the User interface is actually converted into Python commands the program's core logic - and we can see it in real time in this console. If, for instance, I create a new project, switch to the Draft workbench, and draw a line, Below is what actually happens.

This is quite neat, since one can learn about the different commands used in an interactive way. Naturally, once one has a grasp of the fundamentals, they can be used to write one's own scripts, and have them executed by FreeCAD.

```
>>> import WebGui
>>> from StartPage import StartPage
>>> WebGui.openBrowserHTML(StartPage.handle(), 'file:/// + App.getResourceDir() + 'Mod/Start/StartPage/', 'Start page')
>>> App.newDocument("Unnamed")
>>> App.setActiveDocument("Unnamed")
>>> App.ActiveDocument=App.getDocument("Unnamed")
>>> Gui.ActiveDocument=Gui.getDocument("Unnamed")
>>> Gui.activateWorkbench("DraftWorkbench")
>>> import Draft
>>> points=[FreeCAD.Vector(-5.26563731752, -5.3714927212, 0.0), FreeCAD.Vector(3.9703203574, -0.340363797945, 0.0)]
>>> Draft.makeWire(points, closed=False, face=True, support=None)
```

To take an example, let us write a simple Python script that will create a simple box shape. Create a new file called "test1.py", and copy in the code shown top right.

The Part library contains the tools from the Part workbench. We begin by creating a new project, called "Box Model". We make this the active document (window), and add a new object based on the "Part::Box" prototype, naming it "box1". We set its dimensions, and have the document recalculate itself. We then tell the user interface ("Gui") to zoom the view to fit the new object, and select the Axonometric (3D) view.

To execute our script, switch to a terminal window in the same directory we have the .py file, and

issue command:

```
freecad test1.py
```

We will see FreeCAD start up and execute our script line-by-line, giving the final result shown below.

As a second example, let us build something slightly more complex: the shape of a tin that consists in a flat shape (two arcs connected by straight segments), that will then be extruded to form a volume. Bottom right is the script, in file "test2.py".

Let us comment on the differences with the previous example. In this case, we begin our new object by creating four vectors V1 to V4, that indicate the

```
import Part

doc = FreeCAD.newDocument("Box Model")
doc = App.ActiveDocument
box1 = doc.addObject("Part::Box", "box1")
box1.Height = 40
box1.Width = 30
box1.Length = 50

doc.recompute()

Gui.SendMsgToActiveView("ViewFit")
Gui.activeDocument().activeView().viewAxometric()
```

```
import Part

doc = FreeCAD.newDocument("Tin")

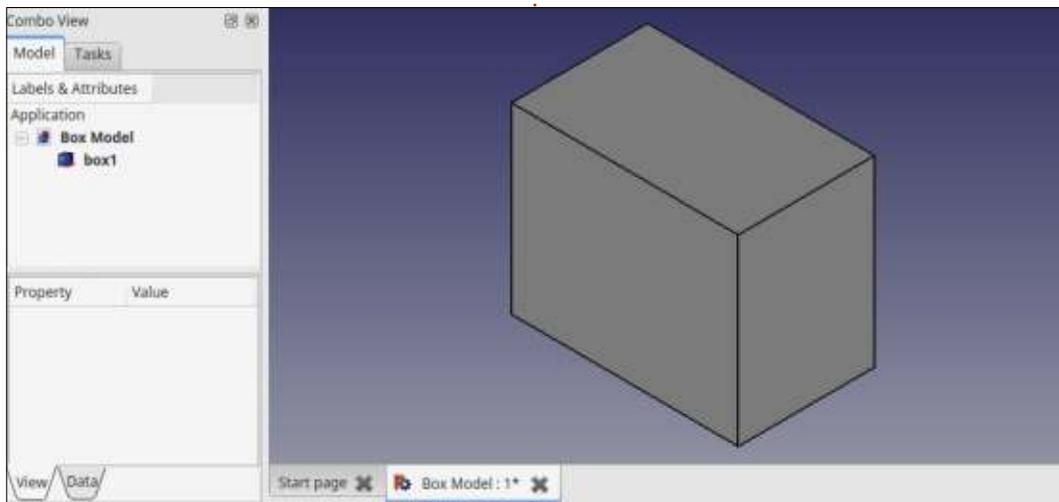
V1 = FreeCAD.Vector(0, 10, 0)
V2 = FreeCAD.Vector(30, 10, 0)
V3 = FreeCAD.Vector(30, -10, 0)
V4 = FreeCAD.Vector(0, -10, 0)

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

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

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

W = Part.Wire([E1, E2, E3, E4])
F = Part.Face(W)
P = F.extrude(FreeCAD.Vector(0, 0, 10))
tin = doc.addObject("Part::Feature", "tin_solid")
tin.Shape = P
doc.recompute()
Gui.SendMsgToActiveView("ViewFit")
Gui.activeDocument().activeView().viewAxometric()
doc.saveAs("tin.fcstd")
```



positions of the connection points between the arcs and the straight lines. We then create the two straight segments, L1 and L2, and finally the two arcs C1 and C2. We then need to convert these four items into Edge objects, E1 to E4, which are then connected into a Wire object W. This is the outline of our tin's top. Please ensure lines and arcs are set up in the correct order, otherwise connectivity errors may ensue. Finally, the Wire is converted into a bidimensional Face object F, which is then extruded into a Shape P. A generic volume is derived from "Part::Feature", and is given P as its shape.

As a final note, the finished project can be saved directly from our script, by calling the doc.saveAs procedure. When

executed from the terminal, shown below is the result of our script.

LETS DRAW SOME GEARS

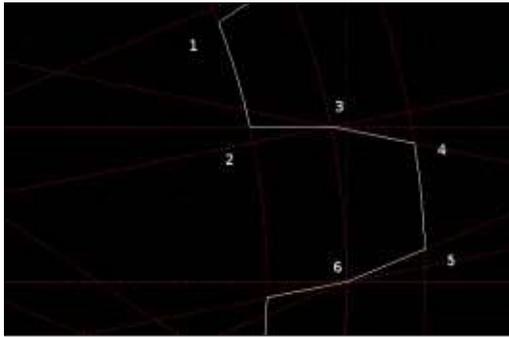
Scripting can come in useful when we need a shape that contains many similar, but different and calculable parts. In such cases, writing a program to iterate over our shapes can be cost-effective in terms of our time. A typical case is when creating a mechanical gear wheel. Such a wheel can be seen as a combination of an exterior shape, formed by a certain number of teeth or cogs, and an inner shape that delimits the axis. This can be either smooth or splined so that the axis can transfer torque to and from the wheel. Both the interior and the exterior wheel forms are formed of a basic shape or motif,

that is iterated at a fixed angular offset between each individual tooth.

Let us concentrate on the exterior shape, slightly more complex than the interior splines. Each individual tooth is centered at a certain radius from the wheel

center, or pitch surface. In our case, we specify a 100mm radius. Simplifying somewhat, this is the point at which the other gears connected to this one will transfer their force. Points 3 and 6 in our schematic are on this surface. Going outwards, teeth extend to an outer limit, the top flat. Points 4





and 5 are on this radius, in our case taken at 110 mm. Finally, we need to “make some space” inwards, to accommodate the other wheel’s teeth. So we go inwards to the bottom flat, at radius 90 mm in our case, and holding points 1 and 2. We will iterate this basic shape over the 16 teeth of our wheel.

Our script (right) will simply draw a series of lines from vertices 1 through 6 of each cog, and then on to vertex 1 of the next cog. Since Python has a mathematical library “math”, sine and cosine functions can be used to calculate a pair of X and Y-axis coordinates for each vertex. The script itself is quite straightforward. As a note, in our example we are using only straight segments. However, in a real gear wheel, the bottom and top flats, and the contact faces, would more usually be drawn with arcs.

```
import Part, math
radius = 100 # wheel pitch surface radius (mm)
bottom = 90 # bottom land radius (mm)
top = 110 # top land radius (mm)
teeth = 16 # number of teeth
doc = FreeCAD.newDocument("Cog")
vertex1 = []
for i in range(0, teeth):
    x = bottom * math.cos(2 * math.pi * (i - 0.45) / teeth)
    y = bottom * math.sin(2 * math.pi * (i - 0.45) / teeth)
    vertex1.append(FreeCAD.Vector(x, y, 0))

vertex2 = []
for i in range(0, teeth):
    x = bottom * math.cos(2 * math.pi * (i - 0.05) / teeth)
    y = bottom * math.sin(2 * math.pi * (i - 0.05) / teeth)
    vertex2.append(FreeCAD.Vector(x, y, 0))

vertex3 = []
for i in range(0, teeth):
    x = radius * math.cos(2 * math.pi * i / teeth)
    y = radius * math.sin(2 * math.pi * i / teeth)
    vertex3.append(FreeCAD.Vector(x, y, 0))

vertex4 = []
for i in range(0, teeth):
    x = top * math.cos(2 * math.pi * (i + 0.1) / teeth)
    y = top * math.sin(2 * math.pi * (i + 0.1) / teeth)
    vertex4.append(FreeCAD.Vector(x, y, 0))

vertex5 = []
for i in range(0, teeth):
    x = top * math.cos(2 * math.pi * (i + 0.4) / teeth)
    y = top * math.sin(2 * math.pi * (i + 0.4) / teeth)
    vertex5.append(FreeCAD.Vector(x, y, 0))

vertex6 = []
for i in range(0, teeth):
    x = radius * math.cos(2 * math.pi * (i + 0.5) / teeth)
    y = radius * math.sin(2 * math.pi * (i + 0.5) / teeth)
    vertex6.append(FreeCAD.Vector(x, y, 0))

edges = []
for i in range(0, teeth):
    nexti = (i + 1) % teeth
    L1 = Part.Line(vertex1[i], vertex2[i])
    edges.append(Part.Edge(L1))
    L2 = Part.Line(vertex2[i], vertex3[i])
    edges.append(Part.Edge(L2))
    L3 = Part.Line(vertex3[i], vertex4[i])
    edges.append(Part.Edge(L3))
    L4 = Part.Line(vertex4[i], vertex5[i])
    edges.append(Part.Edge(L4))
    L5 = Part.Line(vertex5[i], vertex6[i])
    edges.append(Part.Edge(L5))
    L6 = Part.Line(vertex6[i], vertex1[nexti])
    edges.append(Part.Edge(L6))

W = Part.Wire(edges)
F = Part.Face(W)
wheel = doc.addObject("Part::Feature", "cog")
wheel.Shape = F
doc.recompute()
```

Once we have the cog shape in FreeCAD, we can continue using it “by hand” within the normal user interface, as a basis for an extrusion or any other operation we wish. With a similar script to draw the central, crenellated shape for the splines, we can extrude both the external volume of the gear wheel and the volume of the inner axle, and then use a boolean operation to cut one out of the other. We can thus obtain the shape of a traditional straight gear that would be found in many traditional applications such as a

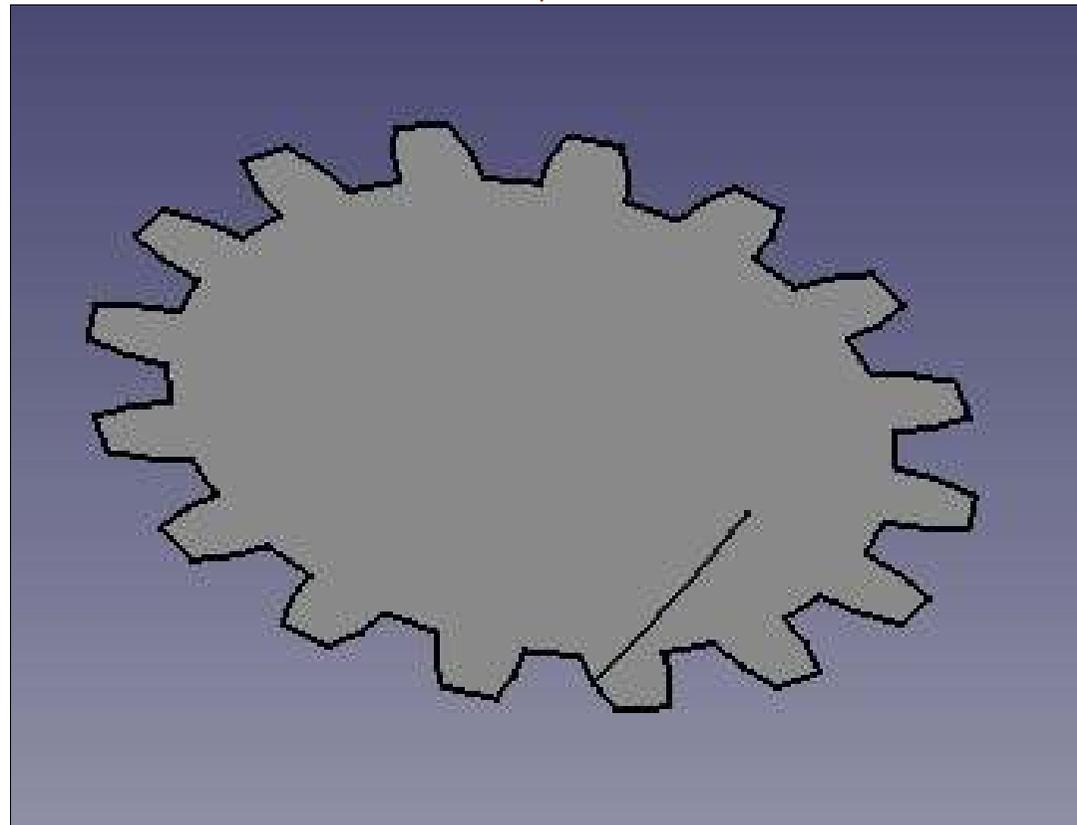
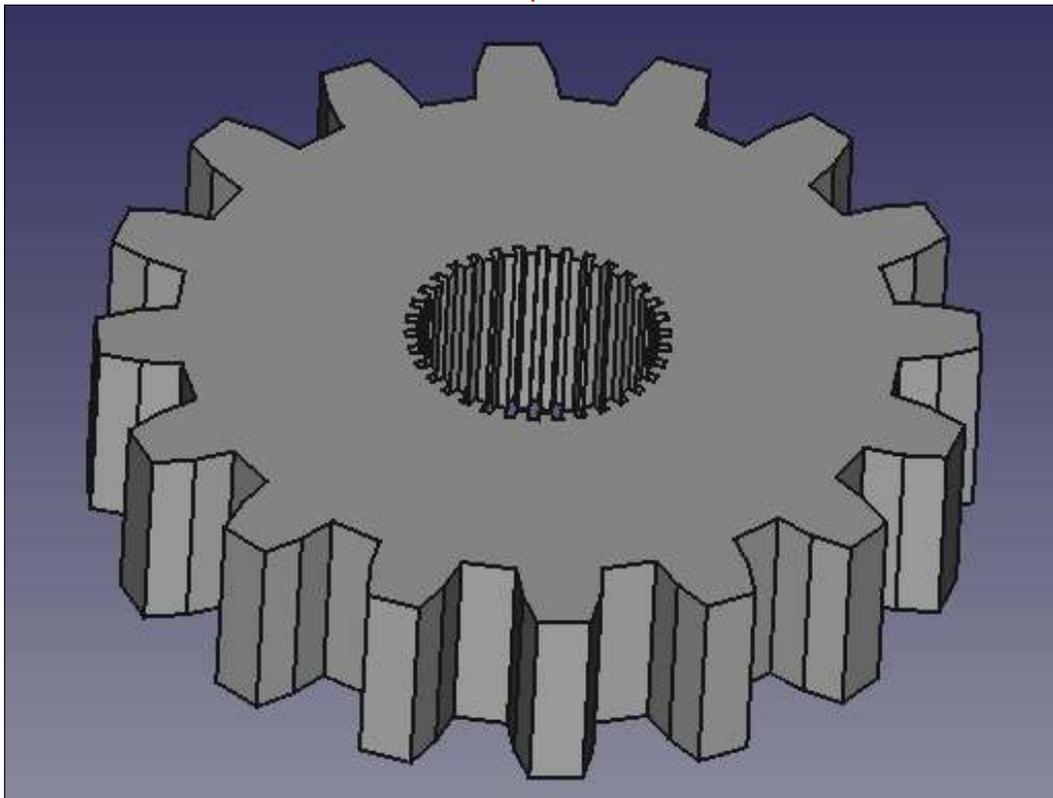
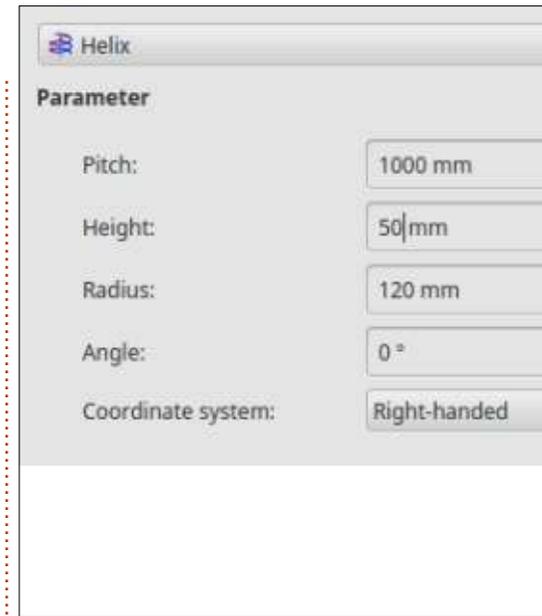
four-by-four vehicle’s transfer box.

However, straight teeth do have the disadvantage of creating quite a bit of noise during operation, since, during each rotation, each tooth engages immediately with the other gear wheel’s corresponding tooth along the complete width of the tooth. This produces the typical whining noise that can be heard from some mechanical setups. In most modern applications where smoothness of operation and low noise emissions are valued, such as

vehicle gear boxes, helicoidal gears can be preferred.

To draw such a gear wheel, the same tooth pattern can be used, but swept along a helicoidal path instead of using the simple linear extrusion tool.

Start, in the Part workbench, by selecting the Part > Create Primitive menu option. Here, we can select the Helix shape. Since I wished to create a gear wheel 50 mm in width, with teeth sloped at



approximately 1:20 across the width of the wheel, I chose a helix height of 50 mm, but a pitch of 1000 mm between helix spires. The external radius of the helix should correspond to the point at which it will be created. Both right-handed and left-handed helices can be used, as needed.

Once the helicoidal line is drawn, it can be used as a path along which to sweep the cog's external face, using the same tools as when creating a Gothic arch in the previous part of this series. The internal surface will be created using a linear extrusion as before, since even a helicoidal gear wheel's internal splines tend to be straight so as to facilitate assembly of the wheel on its supporting axle. The resulting wheel is actually a fair approximation of an actual gear. Some aspects would be made better, though, such as bevelling external edges to make them less aggressive, or cutting out part of the gear wheel's material to make it lighter and use less material in fabrication. These operations can also be done in FreeCAD, and are left as an exercise to the reader (hint: use a revolution surface to create cutouts for each face).

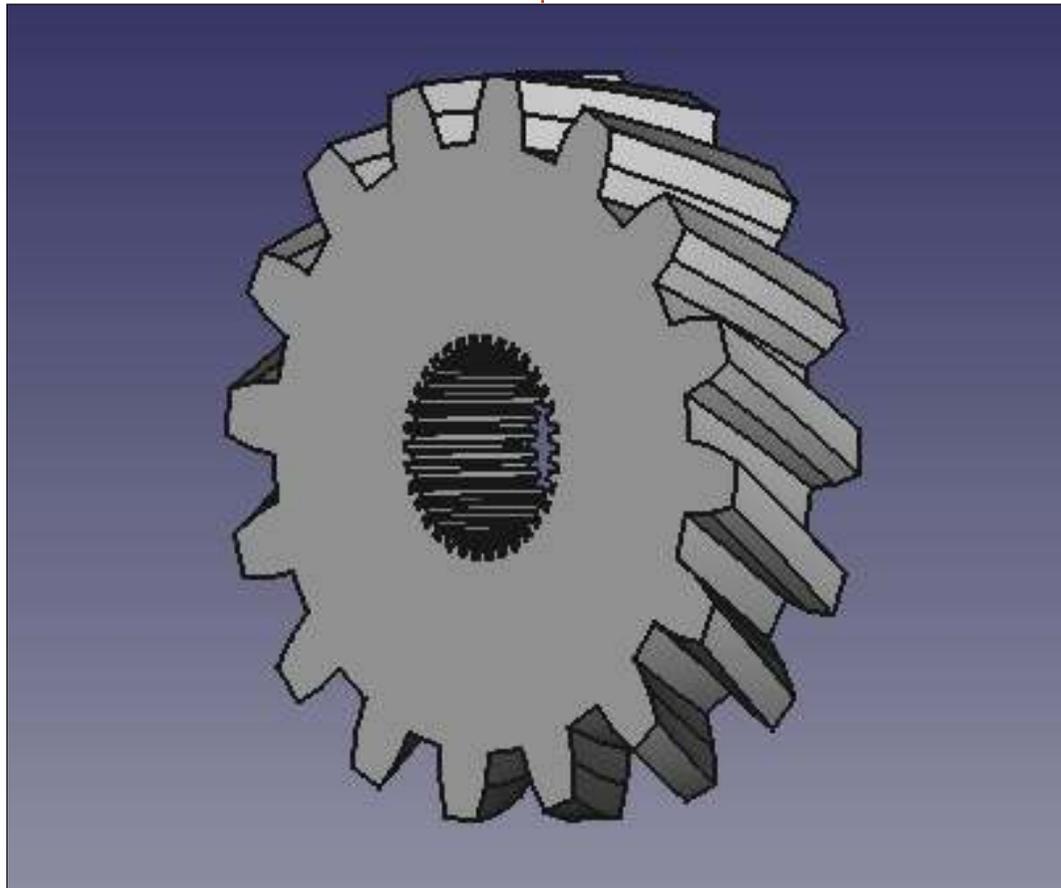
WHAT NEXT?

In this article on using FreeCAD, we extended FreeCAD's possibilities using a little Python programming to create a helicoidal surface in the shape of a gear wheel. Once the basic shape has been drawn using a script, it can be used in combination with any other of the techniques available from the graphical interface to create

the final object. Python is a fully-developed programming suite, containing many different libraries both mathematical and for other purposes, that can be used in combination with FreeCAD to create objects that do not exist within the initial library of basic shapes.

In the next part of this series, we will concentrate on a more complex primitive object that

allows us to create forms and volumes with less regularity, the mesh.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the last article on using FreeCAD, we worked on an architectural project in two different ways. In the first place, we used the Arch workbench to create a modern architectural project, in which supplementary information is given to the computer, so using FreeCAD to create a Building Integrated Model (BIM). Since this approach is in an early stage of development, and is limited to simple forms, we then used a more traditional approach to create volumes in the same way as in previous projects, but on a larger scale. The sweeping technique allowed us to create an element with the shape of an arch by sweeping one sketch (a profile) around another sketch (the outline of an arch).

In today's edition, we will concentrate on a more complex

primitive object that allows us to create forms and volumes with less regularity, the mesh.

WHAT IS A MESH?

A mesh can be taken as a representation of a two-dimensional object (a surface), situated within tridimensional space. Mesh objects can be made up of very many types of elementary elements, some of which can be rather complex such as Non-Uniform Rational B-Splines (NURBS). However, the most common varieties are simple triangles and flat four-sided elements. This is for several reasons, including the fact that most complex surfaces can be approximated by triangles with a reasonable level of precision - much in the same way that the plots of simple mathematical functions are often represented on-screen with an array of straight segments, when in reality some of these functions have no straight bits all along their length. Another aspect of the equation is that many computer meta-languages

describing scenes in 3D -such as OpenGL- have primitives for such triangles.

According to the specific application, however, 3D scene file formats can hold more, or less, information about the mesh. One of the file formats commonly used in 3D printing, the STereoLithography (STL) format, merely contains a list of triangles. Vertices are repeated as needed, and no further information is recorded about the actual structure of the underlying object. In a more complex case such as Computer Fluid Dynamics (CFD), toolkits such as OpenFOAM (<https://openfoam.org/>) have a file format that draws up the mesh using a list of vertices, then a list of faces through referral to the vertices, and finally the complete mesh as a list of faces with their relative positions and associated variables. Fluid pressure, velocity and temperature are often used, and must be stored for several points in time in auxiliary structures that hinge on the mesh.

FreeCAD already knows how to build several types of basic meshes, such as the simple shapes (cylinder, cone, sphere) defined in the Part workbench. These meshes can be exported to several file formats, among them STL. Simply choose the part, then switch to the Mesh workbench and choose menu option Mesh > Create mesh form shape. A new part, with a meshed version of the original, will be inserted into the project. Also within the Mesh workbench, tools are available to export this mesh to a file (tool on the right).



Once a STL file has been saved, this can be used with most 3D printers to print a physical copy of our original shape.

IMPORTING AND USING MESHES

Another useful feature of the Mesh workbench is its capacity to import a mesh from a file, and create a new Part element from

HOWTO - FREECAD

the data imported. I downloaded a test mesh named DAVID-Angel from 3D scanner producer DAVID (<http://www.david-3d.com/en/support/downloads>). I then used the Mesh tool (the leftmost of the pair) to import this mesh into a new FreeCAD project. The result was quite good, and one can navigate around the digital model and examine the statue's admittedly rather plump arms from up close – if so inclined.

Other parts can be added to the scene within FreeCAD, allowing us to modify the model and then export our modified version, if needed. One specific use for this could be to add supports or other auxiliary features to a model,

before printing in 3D. To take an example, I added a circular base to the angel statue.



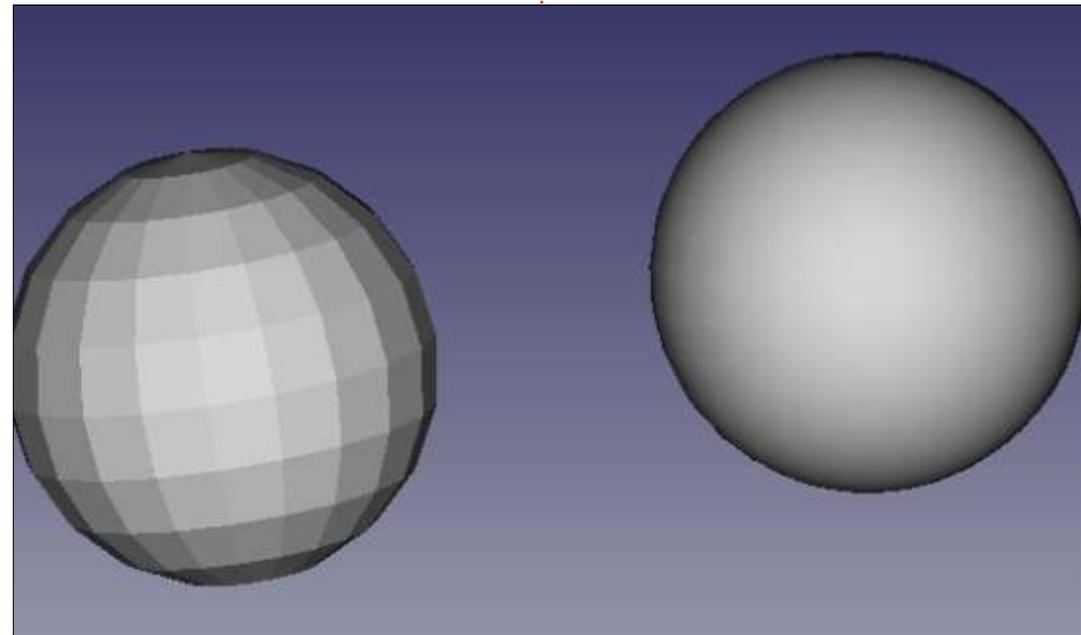
However, some care needs to be exercised when working on models with very many triangles. The angel sample mesh used above is already quite capable of exhausting FreeCAD's memory management, so it may be judicious to save our work every few steps.

CREATING OUR OWN MESHES

The STL file format is basically just a text file with a very simple internal structure. For instance, to create a mesh that contains just one single square facet, we could use the following code:

```
solid Square (Meshed)
  facet normal 0.0 0.0 1.0
    outer loop
      vertex 1.0 1.0 0.0
      vertex -1.0 1.0 0.0
      vertex -1.0 -1.0 0.0
      vertex 1.0 -1.0 0.0
    endloop
  endfacet
endsolid Mesh
```

Most indications should be self-explanatory. The “normal” keyword gives the facet's normal vector, basically telling us which side of our facet is to be considered “outward” or “inward” in respect to the complete object. If a triangular facet is required, just use three vertices to define it. If several facets are needed, iterate the facet...endfacet sequence.



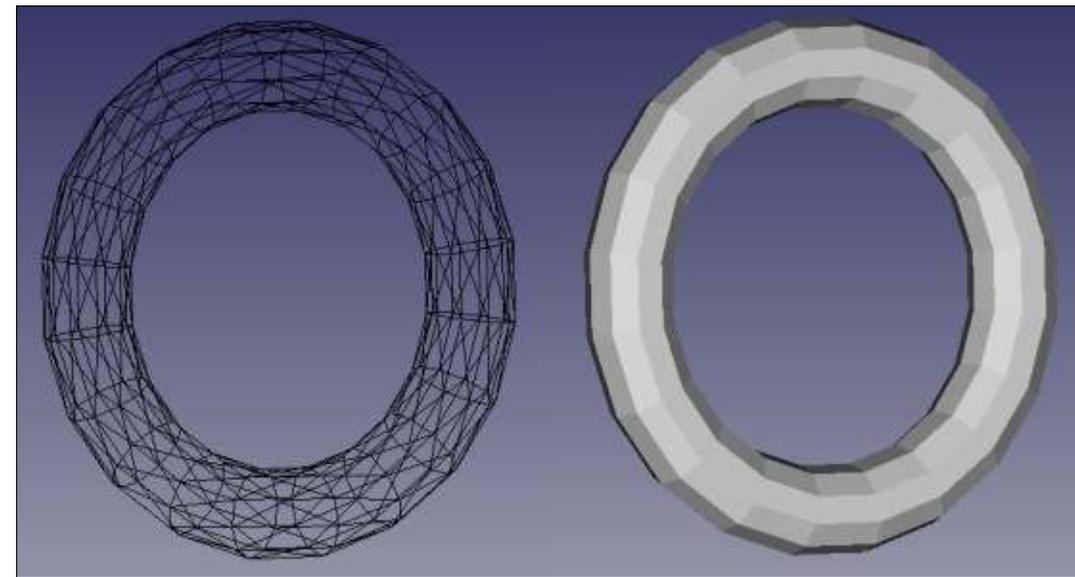
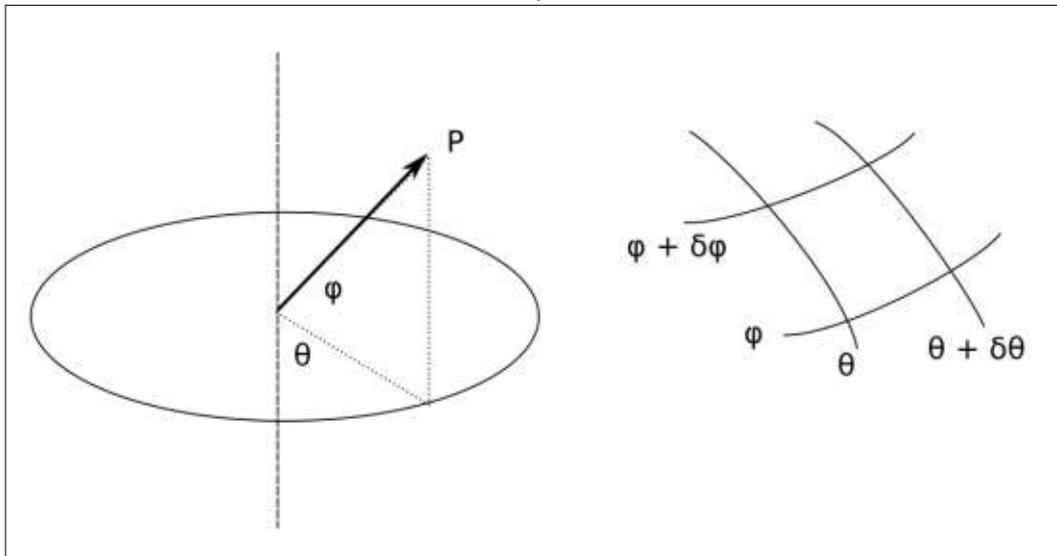
This very simple structure makes writing our own programs to create a mesh file automatically an easy proposition. It could be done in just about any programming language such as Pascal, C, Java, JavaScript with Node.js, and many others, but my personal preference will go to Python - in keeping with the fact that FreeCAD is written in this language. Let us start with a simple sphere. In the following screenshot, the object to the right - seen from within FreeCAD - is an instance of the application's inbuilt Sphere object. The object to the left, however, is a mesh that has been generated with a simple Python script.

Any point P on the surface of a sphere can be defined using horizontal angle theta (θ) within the equatorial plane, and then vertical angle phi (ϕ) to give its height above the plane. In essence, this is what we do when using latitudes and longitudes to give the position of an object or place on the Earth's surface. So our program simply needs to calculate a series of coordinates, while varying θ from zero to 2π radians, and ϕ from $-\pi/2$ to $\pi/2$. Radians are

our angular measurement unit of choice, since this is what computer programs use to calculate sines and cosines.

Once we have our double for loop set-up, we need to transform the more or less rectangular shapes we obtain between θ and $\theta + \delta\theta$ horizontally, and between ϕ and $\phi + \delta\phi$ vertically - where the deltas are the difference between successive values of each respective angle. The easiest course is to cover this area with two triangles. The complete Python program is simple, but a tad longer than could be acceptable for this publication. For this reason I put it up on Pastebin at the following address:

<https://pastebin.com/jvv35AgZ>.

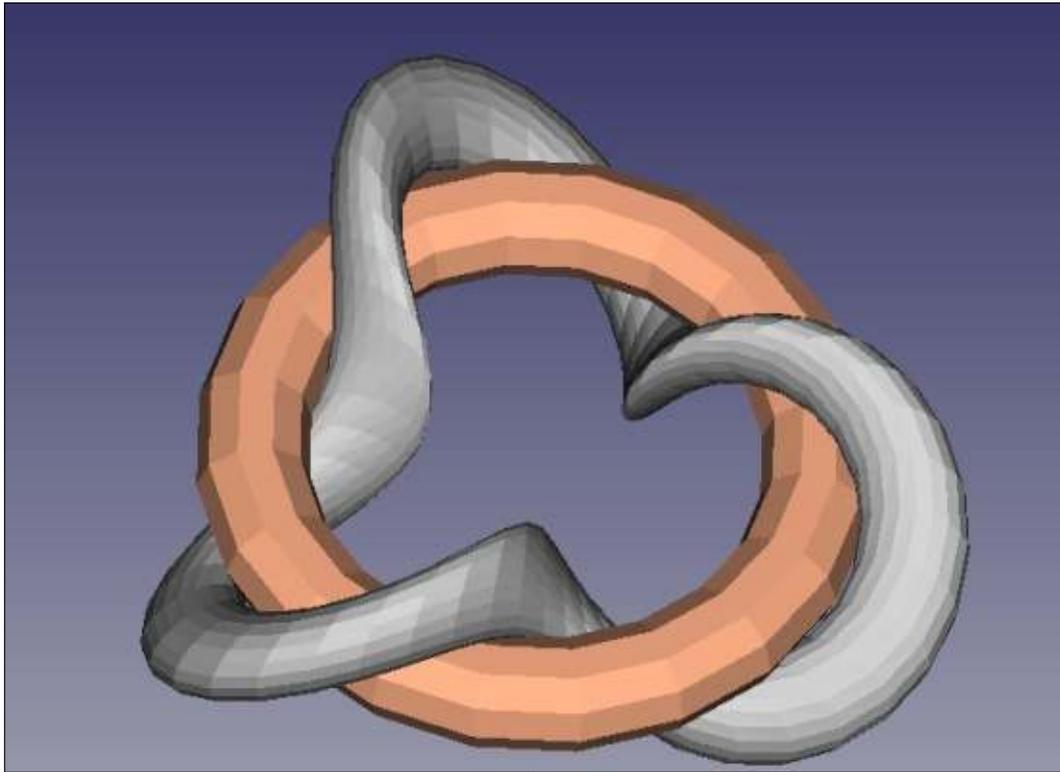


Please do not hesitate to use it - and to experiment.

Going on to more complex objects, a ring -or, in mathematical terms, a torus- is an object that has two radii: the main ring radius in

one place, and a secondary radius that defines the thickness of the object, in a plane set off at right angles to the main plane. In the following capture, we can see two copies of the mesh as imported into FreeCAD, one on the left with mesh edges apparent, and the second on the right all built up. In this way, we can see that what seem to be flat four-sided facets are in fact each a combination of two triangles.

The Python program to create this mesh file is actually rather similar to the previous code. However, in this case ϕ needs to iterate over a full circumference (from $-\pi$ to π) to complete the ring's tube shape along the smaller



circles. As before, θ iterates over the ring's main circle. The code can be found at:

<https://pastebin.com/BNxPztFP> .

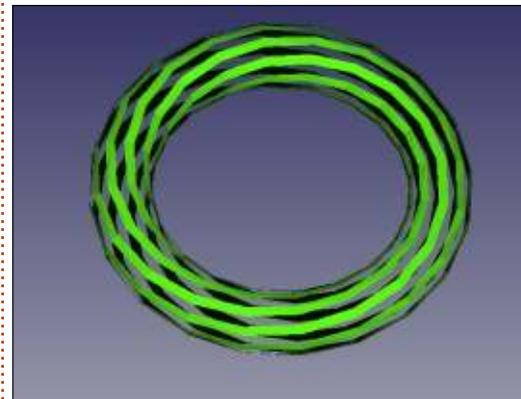
Please note the use of r_1 , the outer radius, here set at 5 units, and r_2 , the smaller radius, here set at 1 unit.

Once we have the basic code setup, we can have some mathematical fun with it. For instance, we can have our ring material twist about the main ring, by giving it a further (third) radius to offset it from its "normal"

position and have that turn around a number of times while we iterate over θ . We could, for instance, use $\cos(3\theta)$ and $\sin(3\theta)$ to calculate its radial and vertical coordinates to have the ring "wobble" three times along the main circumference. If our resulting object is quite flat, and the number of turns is odd, it can even resemble a Moebius strip. In the next screenshot, we can see our original ring in copper, combined with the new twisted shape in grey. The Python code to create this mesh file is, as always, on Pastebin:

<https://pastebin.com/ZvnDdLTX> .

One advantage of writing our own programs is that we can then go on to modify our objects as desired. A simple alteration in the value of $\delta\phi$ can make our triangles cover only half the surface of our object. If, at the same time, we give it a single twist while iterating along θ , the final appearance can resemble not a single shape, but a collection of interwoven rings. In the following screenshot, note how each ring twists once around ϕ while making its circuit of the main ring.



WHAT NEXT?

In this article on using FreeCAD, we concentrated on a more complex primitive object that allows us to create forms and

volumes with less regularity, the mesh. Using the widely accepted STL file format, a mesh or collection of simple triangular or four-sided facets can be retrieved either from a physical 3D scanning device, from other people's work, or created using ad hoc programs. With a bit of mathematical expertise, the objects created can vary from the very simple to rather more complex objects.

In the next part, we will use this technique in combination with other, more standard FreeCAD tools, to build a 3D representation of a modern building with a lattice roof structure.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



HOW-TO

Written by Alan Ward

Intro To FreeCAD - Pt8

In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the previous (seventh) article on using FreeCAD, we concentrated on the mesh as a complex primitive object that allows us to create forms and volumes, either from scanned data or by using simple programming techniques.

In this part, we will use this technique in combination with other, more standard FreeCAD tools, to build a 3D representation of a modern building with a lattice roof structure.

ARCHITECTURAL MESHES

Some of the inspiration for this article comes from buildings such as the Esplanade Theatres on the Bay, Singapore (DP Architects), and the Rihke Park music theatre in Tbilisi, Georgia (Studio Fuksas). In all cases, a lattice arrangement has

been used for the exterior of the building. From an architectural standpoint, this technique has at least two salient points:

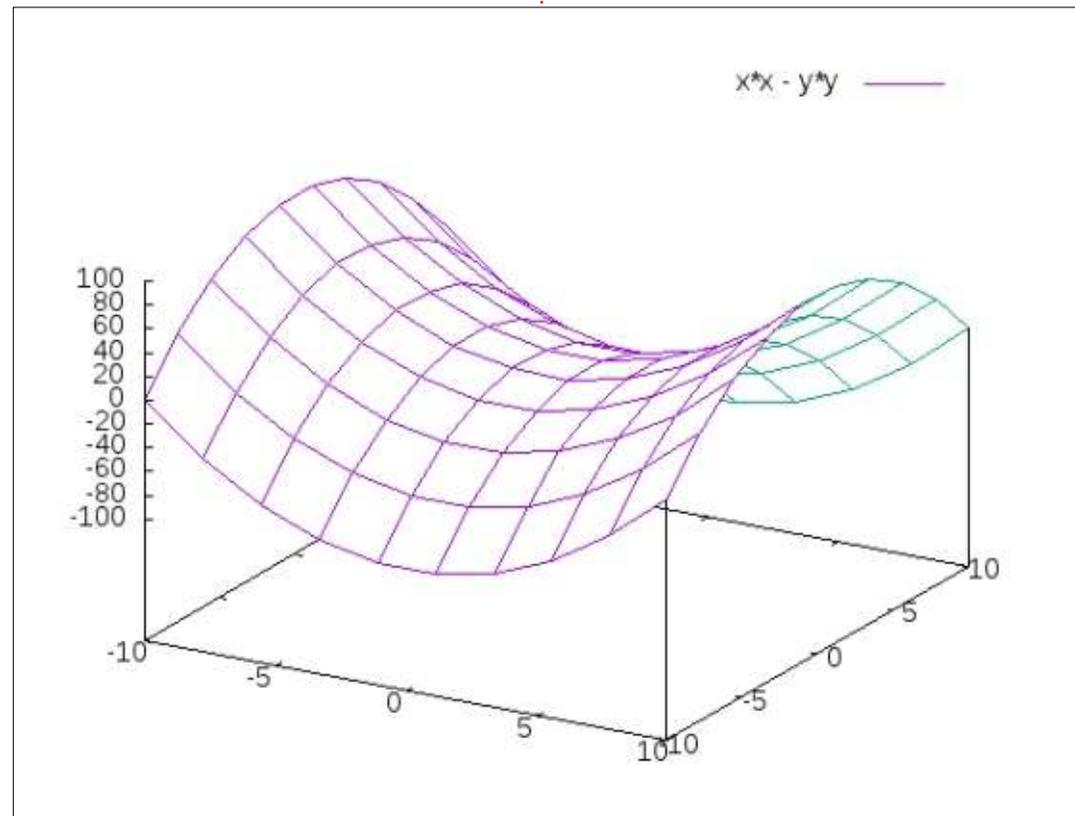
- On the one hand, the lattice can support a large proportion of its own weight, which allows the architect to cover a large area without using internal supports such as columns. This is perfect for large open spaces such as concert halls or sports venues, or even the open concourses inside airport buildings, such as Hong Kong International Airport at Chek Lap Kok.
- On the other hand, using a lattice implies that part of the structural strength comes from working with curved surfaces that share some of their properties with the arch. The final result is a building that eschews the flat, regular shapes that have become so common in urban architecture during the last century.

Some simple lattices can be drawn "by hand", as long as the overall shapes remain planar or use a single curvature. However, once the final shape contains double

curvatures - along two intersecting axes at once - things do tend to get a little complicated. This is when a computer comes in as an essential tool to calculate the position of each lattice point, and from there to calculate the constraints expected within the physical structure.

For instance, let us consider the height function $h(x, y) = x*x - y*y$. A

simple surface plot shows us that, in the vicinity of coordinate origin point (0, 0, 0), this surface shows a concave curvature in the upwards direction, along axis X. However, curvature is convex along axis Y, also towards the top. This very simple function shows a double curvature that is easy to calculate, but not easy to draw with precision without the help of a computer.

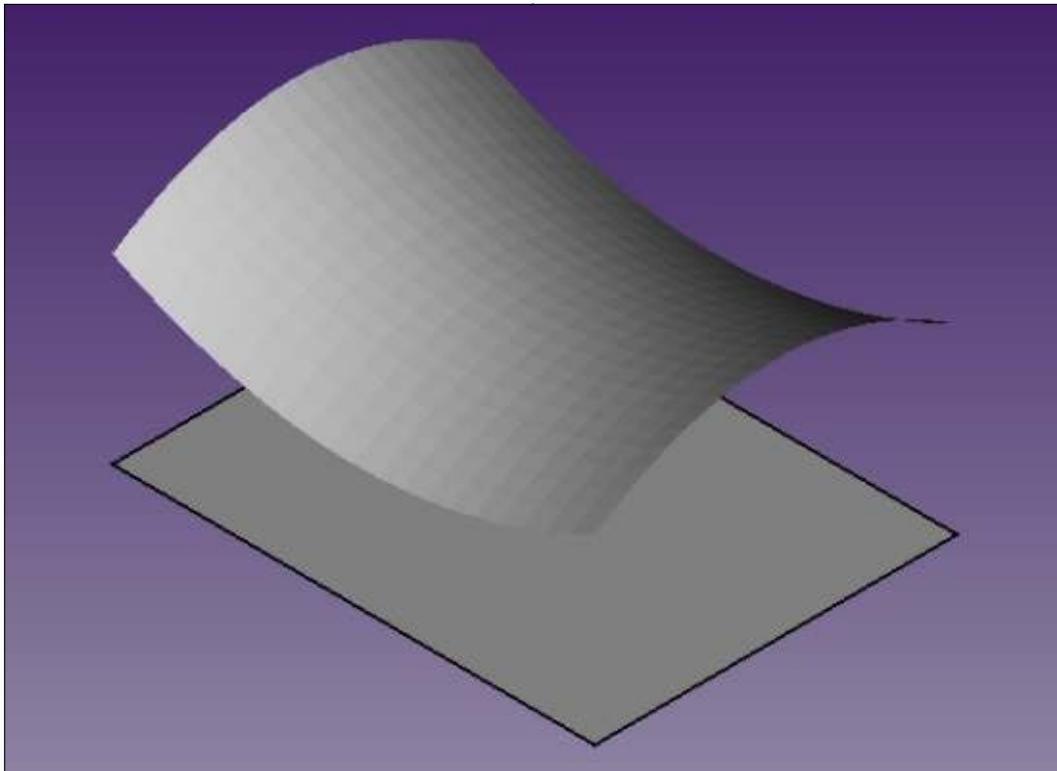


INTEGRATING A LATTICE INTO A BUILDING

In the following discussion, we will create a medium-sized pavilion using a lattice defined with the above function. Naturally, the reader is not encouraged to actually build such a structure in real life - at least, not unless considerable architectural and engineering experience is available to ensure materials are correctly chosen and dimensioned, local building codes are followed, and the construction has some chance

of holding up to its intended use. There are some tricky aspects to consider with this type of construction, not least of which is the fact that the lattice would be stressed in compression along axis Y, and in tension along axis X, all by its own weight. Factors such as wind pressure, and the weight of rainfall, ice, snowfall, etc, would also need to be accounted for.

Let us begin by writing a short Python program to create an STL file with the mesh. The complete code can be found at:



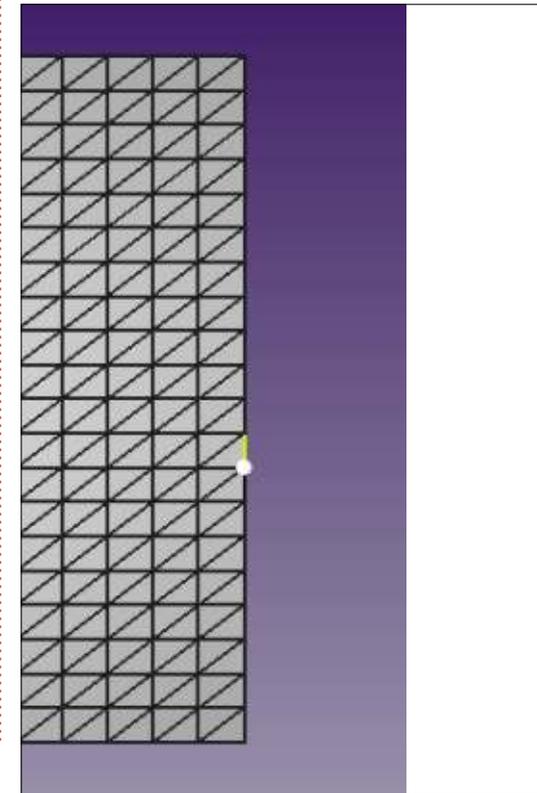
<https://pastebin.com/tsi5dbLw>. Working on a floor plan of 40 x 30 m in the XY plane, we will calculate a value for height along the Z axis. Minimum and maximum heights have been taken, for this example, at 10 and 20 m above ground level respectively, though this could easily be scaled and adjusted to suit a particular implementation. Finally, 20 separate mesh separations have been taken, along both the X and Y directions.

The end result of this program is an STL file that can be imported into a new FreeCAD project using the technique discussed in the previous part of this series. To better visualize proportions, a 40x30 m rectangle has been drawn at ground level, under the roof.

Closing off the walls on all four sides of this building will be no easy task, since all four walls will have three straight edges, but the fourth (upper) edge needs to follow a parabolic line. There are several solutions for this problem. One would be to write further programs in Python to create mesh files to suit. However, FreeCAD does offer alternatives. In this case, I made mesh lines visible. Select the mesh object, and, in the

Property box, select "Display Mode" and switch from the original "Shaded" mode to "Flat Lines".

We can then go into the Draft workbench, and, working carefully, draw a new Wire - selecting, point by point, all the vertices along one edge of the roof. Then, close the shape by selecting the corners of the ground rectangle beneath this edge. This flat shape can then be promoted to a plane?plain? DWire object. Then proceed in the same way for the other walls.



Once the walls have been defined, the roof mesh can either be left with its mesh structure apparent, or the Display Mode can be switched back to "Surface", as desired.

One can then add columns to hold up the roof mesh. However, it then becomes apparent that the mesh has no thickness. Even if much care is taken with column heights, some discrepancy appears since column ends are horizontal disks, but the part of the mesh that is in contact with them is not flat. Intersections with the

building's four exterior walls also become apparent.

One relatively realistic solution is to give the roof mesh some thickness. Simply select the mesh, and, in the Part workbench, create a new Extrusion. Top and bottom surfaces will have the shape of the mesh, while the vertical depth of the object will be the same along its surface. A value of one meter seems appropriate for this building.

Finally, the colors and transparency levels of each object can be adjusted in the Properties

box. For instance, to represent glass walls - allowing light to enter the building between columns - the four walls can be left with their default color ([204, 204, 204]), but with a transparency of 40.

WHAT NEXT?

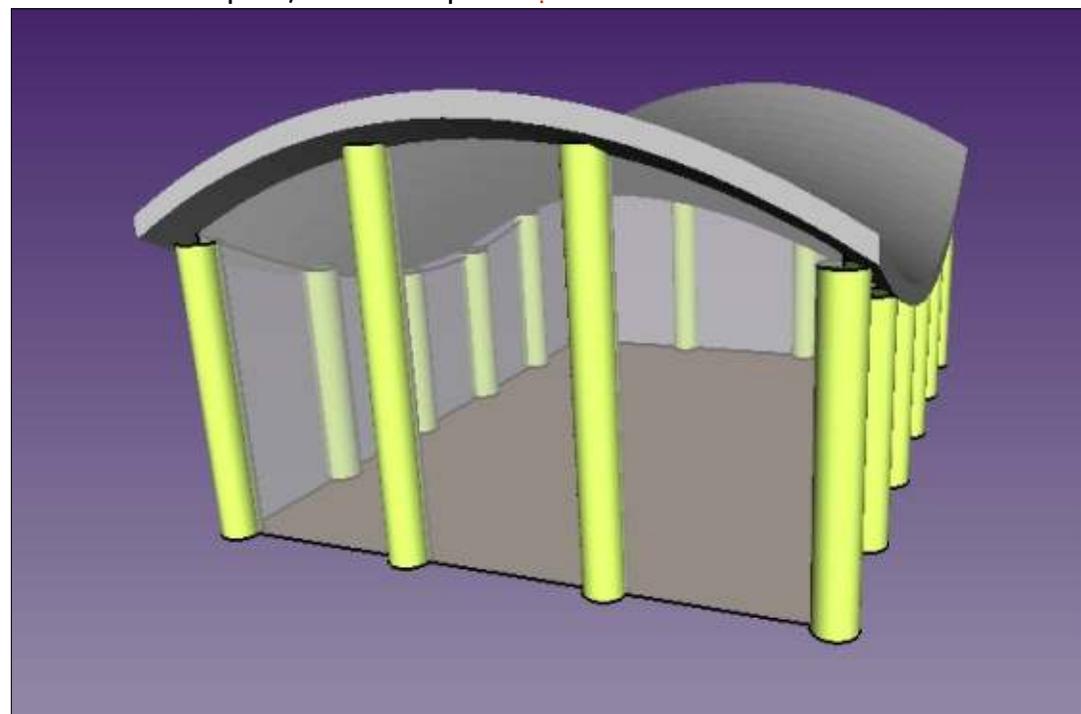
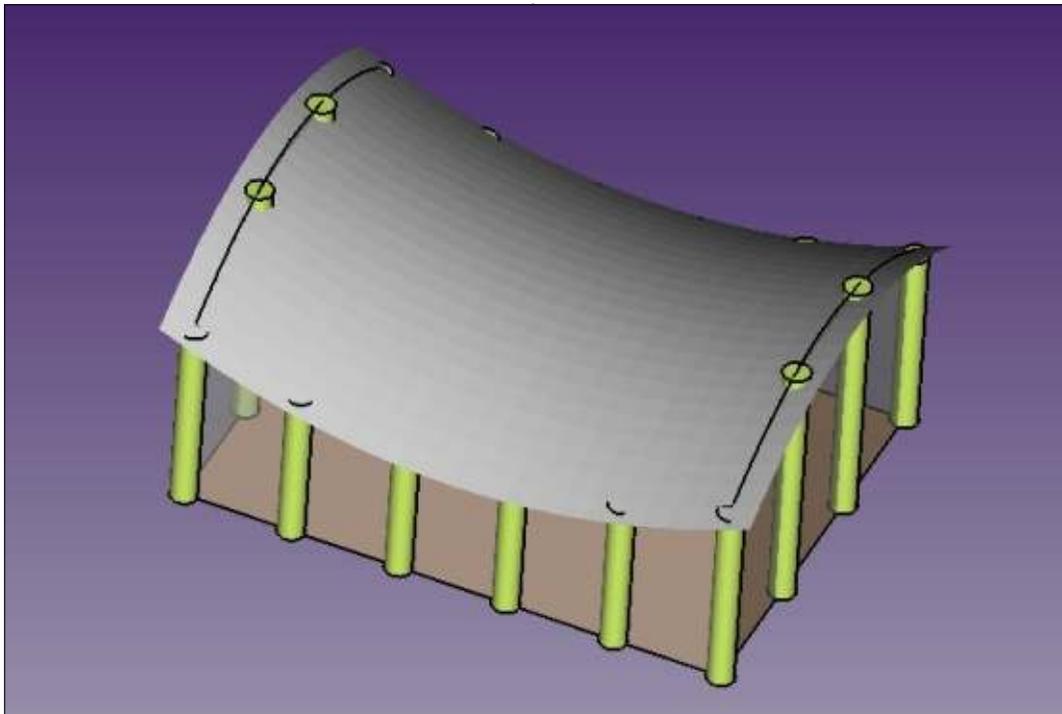
In this article on using FreeCAD, we used a mesh in combination with other, more standard FreeCAD tools, to build a 3D representation of a modern building with a lattice roof structure.

In the next part, we will explore

going from a computer model to something physical, by exporting a finished project and printing it using a 3D printer.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).





In this series, we will be examining the world of FreeCAD, an open-source CAD modelling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the last (eighth) article on using FreeCAD, we used a mesh in combination with other, more standard, FreeCAD tools, to build a 3D representation of a modern building with a lattice roof structure.

In this part, we will go from a computer model to the physical world, using a 3D printer to create a physical representation of our construction.

SOME NOTES ON 3D PRINTING

It will come as no surprise that 3D printing has become something of a fad in the last few years. Starting out as a bit of a hobbyist activity, it has found its practical application in many rather different fields, such as art and

crafts, design, engineering, and even some medical fields. Relatively cheap printers that come fully assembled and ready to print are making the technique more accessible to a large variety of users. However, it must be said that 3D printing is not yet quite as mature as traditional printing on flat pieces of paper, and some practical inclination is still very much a necessity for users. Dealing with platform placement calibration, nozzle stoppages, or other mechanical issues, may not be within everybody's comfort zone.

There are many techniques of 3D printing. They are usually seen as some form of additive construction, where the resulting part is built up progressively. This is in contrast to machining - for instance using a computer controlled lathe - where an existing block of material is cut down to the final desired shape by removing excess material. Some materials such as plastics lend themselves best to additive processes, while others such as metals are more

often than not best handled with subtractive methods.

Even within the domain of 3D printing, there are many variants. Some of the more expensive, such as sintering, involve heating small particles of the material with a laser to fuse them together and form the object being built. In others, a solution of material is locally heated, transforming the liquid solution into a solid layer. In the vast majority of commercial 3D printers that would be in the price range of the enthusiast or a small business, a plastic extrusion process is used. In this, a plastic filament is slowly extruded through a heated nozzle. The plastic melts when going through the nozzle, and fine points or lines of material are deposited in layers to build up the object from bottom to top.

This system has its quirks. The first main point to take into account is that very fine object volumes or parts may not come out as expected. Details of less than 2-3 mm thickness may be very brittle

once printed and, in fact, may easily be broken off when removing the printed part from the supporting plate. Naturally, the details depend on the actual printer used, and on the level of detail dialed into the printer. With thinner layers (0.1 mm instead of the more common 0.2 mm), finer details will come out better, but at the expense of a much longer print run. Time spans of 2-3 hours are not uncommon for small objects (1-2 cm tall), and can go up from there for larger objects.

The second point is that the upper layers of plastic are laid down on top of the lower layers. However, the plastic is quite liquid when leaving the nozzle, and so needs a stable base to rest on while solidifying. Structures such as overhangs or arches in the model will not come out well, if left unsupported.

Many printer control applications alter our model adding supportive structures. These are printed together with the model itself, and must be

removed after printing. In the accompanying image, a model of a wheel rim has been printed. Part of the mat laid down by the printer to fix the part to the supporting plate is still attached to the bottom of the part. The interior of the recess along the rim has been filled in with vertical column-like shapes by the printing software, in an effort to ensure the top edge does not fold down while still hot. These shapes are quite ungainly, but are also thin walls and may easily be pared off with a sharp knife (but do be careful with your fingers).

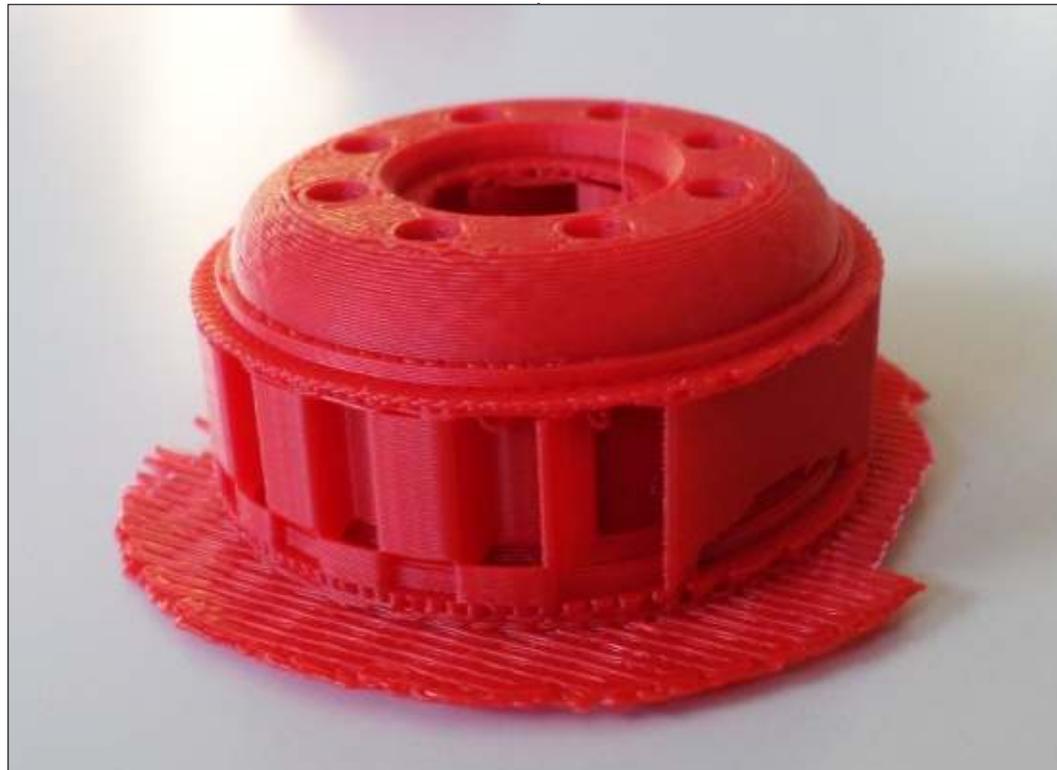
Depending on the shape of the model, cleaning up may be quite involved. In a recent project, a 4x4 link chainmail assembly took one hour of printing time, but then required two hours of manual cleaning up and surfacing. Material loss would also be a concern in an industrial environment: in this case, 3.3 g of the final object required a total of 7.2 g of printed material. A material efficiency of less than 50% can be seen as far from ideal.

BUILDING AND PRINTING A SIMPLE OBJECT

The actual details of our workflow can vary, depending on which program set we choose to use. However, the main steps will be as follows:

- Build the computer model, using volumes. Thin, flat parts must be rendered as volumes, with a thickness that for best results should not go below 1 mm. In this series, we will naturally use FreeCAD for this stage. However, other options such as Blender are also quite suitable, as long as they can export object meshes in the STL file format.

- Use a slicer program to convert the object into a series of flat slices. These slices are then converted into a sequence of G-code commands, that in essence tell the printer to place its head at such-and-such coordinates, and turn the plastic extrusion on and off. A common choice for this stage is Slic3r (<http://slic3r.org/>).
- Use a third program to connect to the printer, and actually perform the printing process. Printron / Pronterface (<http://www.pronterface.com>) is a popular choice.



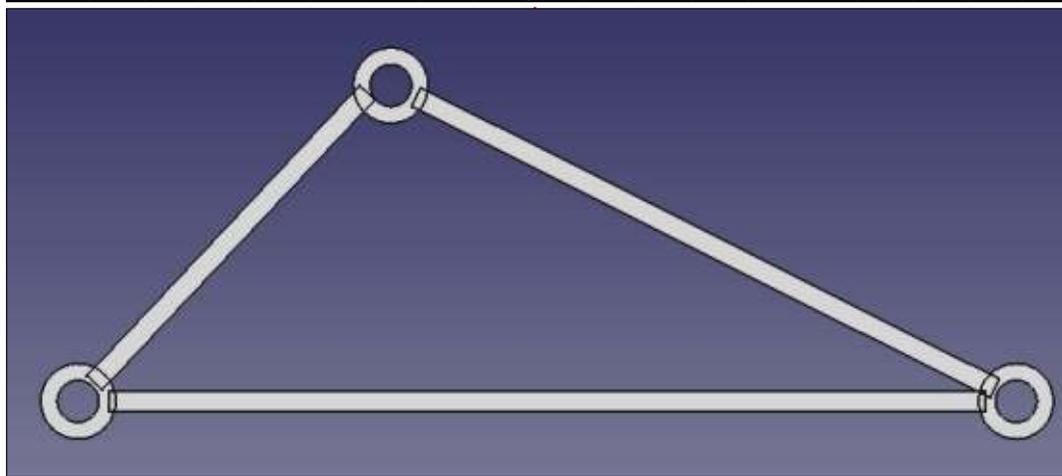
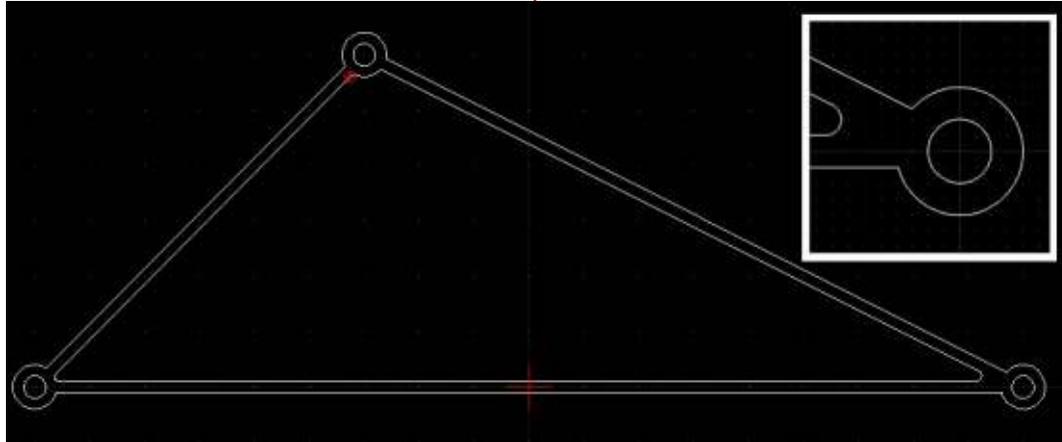
Two file formats form the glue between stages (a) and (b), and between (b) and (c). The STL format previously discussed in parts 7 and 8 of this series is a standard way to transfer our object's form from the design application to the slicer. Other choices do exist, such as OBJ files, but do seem to be slightly less well supported. G-Code files may be used to transfer data from the slicer to the printer controller, though this step is omitted if the slicer can also act as a printer controller. Applications such as Slic3r can control directly a certain number of printer models, mostly open-source hardware. However, many (commercial) models require their own software for slicing and controlling the printer, which is usually found only for Windows. This may be a point to take into account if or when selecting a printer to purchase.

Let us start with a simple truss object, basically a triangular structure of square bars connected with transverse circular bars. The first point we will need to get right is dimensions. Depending on your printer, there will be limits to the overall size of the object to be

printed. In this case, I chose to build a piece 120 mm in length, the size of the longest bar. Bar sections were 3 mm square, to make them easy to print. Finally, the circular joints have an internal radius of 3 mm, and 6 mm external. The overall height of this structure is 4.5 mm.

To set up this piece, a traditional CAD procedure would be to draw a flat representation of

the external shape, make sure all joints fit by trimming lines as required so that there is no intrusion of one bit into another, and then draw in the circles representing the holes in each joint. Using the more advanced features of modern 2D CAD applications such as LibreCAD, one could easily add some filleting to make joints a tad more robust at the unions between bars and cylinders.

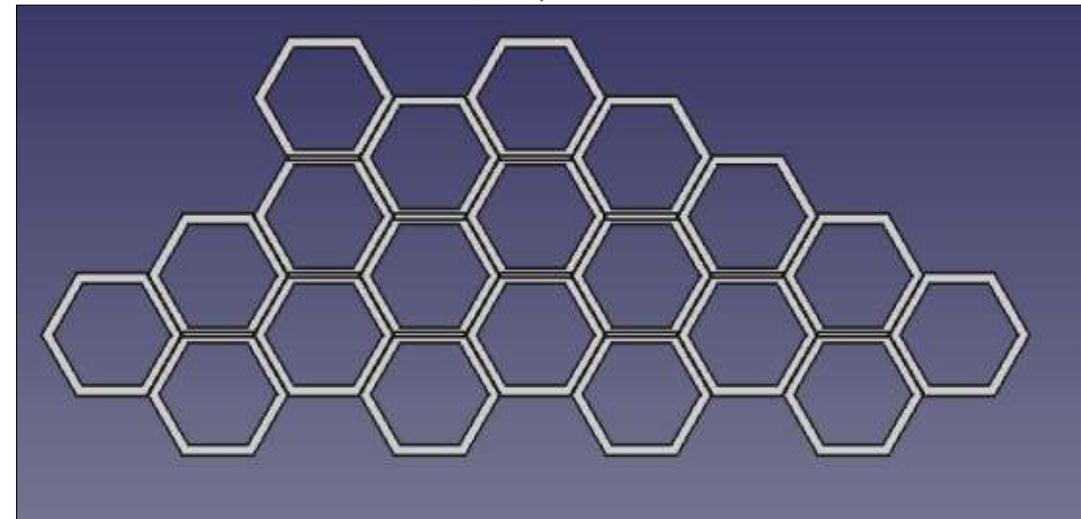


To build a 3D model, however, it is more convenient to think in terms of assemblies. I started out immediately in 3D by drawing a cylinder object in the Part workbench of FreeCAD, to represent one of the joints. I then draw a second, taller, cylinder to represent the cutout for the hole, and subtracted both objects to create a hollow cylinder. I then copied and pasted this complete part into the three final positions for the joints. I then created a flat bar of the appropriate section, and then copied, rotated and scaled it into position three times to form the triangular structure.

Some care needs to be taken in this assembly, since it is clear some overlapping of parts has occurred.

In the real world, the bars would need to abut to the outside surface of the cylinders, and bar extremities would need to be shaped accordingly. As an alternative, vertical slots could be cut into the cylinder walls, and the bar heads left square and slotted into the cylinders.

In the magical world of 3D printing, however, the intrusion of one volume into another may not be a problem. Most printing software can take care of this, so that the plastic material in each volume does not get printed twice over - which would result in a big mess. Instead, software is smart enough to perform a boolean union on all volumes and stitch them together correctly. It must be said, however, that not all printing



software is equal in this aspect, and some experimentation may be necessary to find the limits of a particular printer software and hardware combination.

Once the external truss had been built up, I wanted to fill in the center with a non-structural mesh. There are several ways of going about this. For instance, one could build a flat volume to fit the empty space, and then cut holes in it to suit. I wanted something a bit more fancy, along the lines of the beehive motives seen in some modern car grills. So I started by building a basic hexagonal shape in the same way I had done the cylindrical joints. I began by drawing a vertical six-sided prism

with sides 9 mm long and height 3 mm, then cut out another vertical prism with sides 8 mm long from the center. I then replicated this basic motif to fill the space in a honeycomb structure.

At this point, I had both the external triangle and the inner grille. However, the grille did protrude slightly from the sides of the triangle. So, it was back to the Draft workbench and I drew a rough approximation of the external triangle as a continuous Wire object. This object, extruded upwards, gave me the shape of the internal space, with some overlap with the triangle's bars. I then defined the grille as the intersection of the first grill and

this new volume, which in essence trimmed its shape down to fit within the interior space of the triangle.

The final piece is the combination of the external triangle, plus the grill. This assembly is then made into a single object using a boolean union.

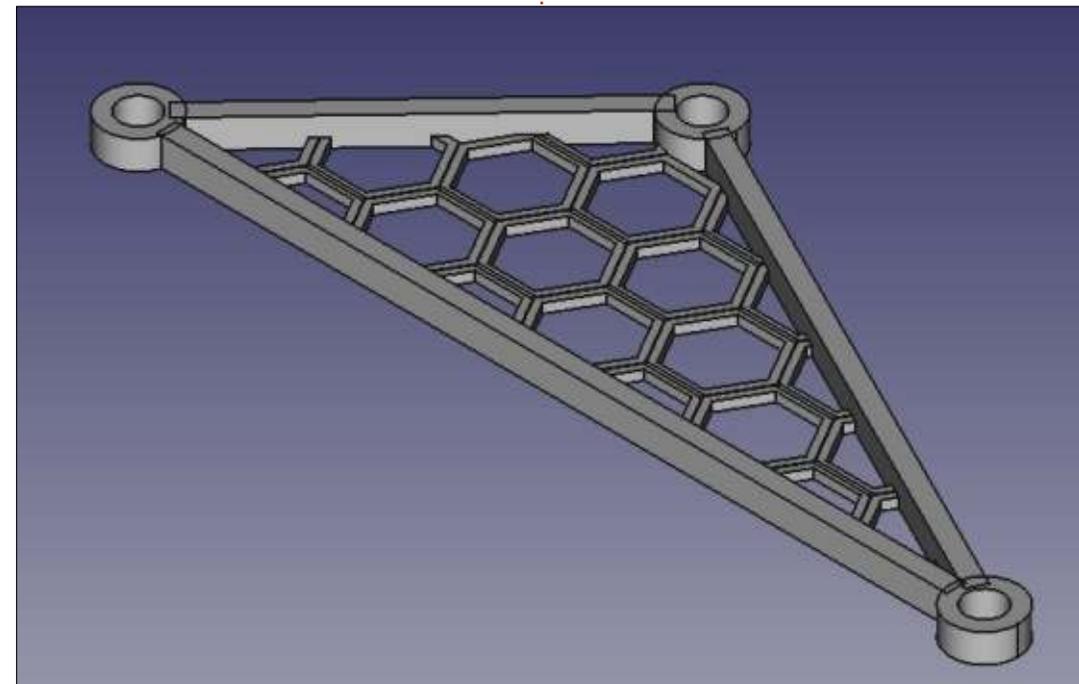
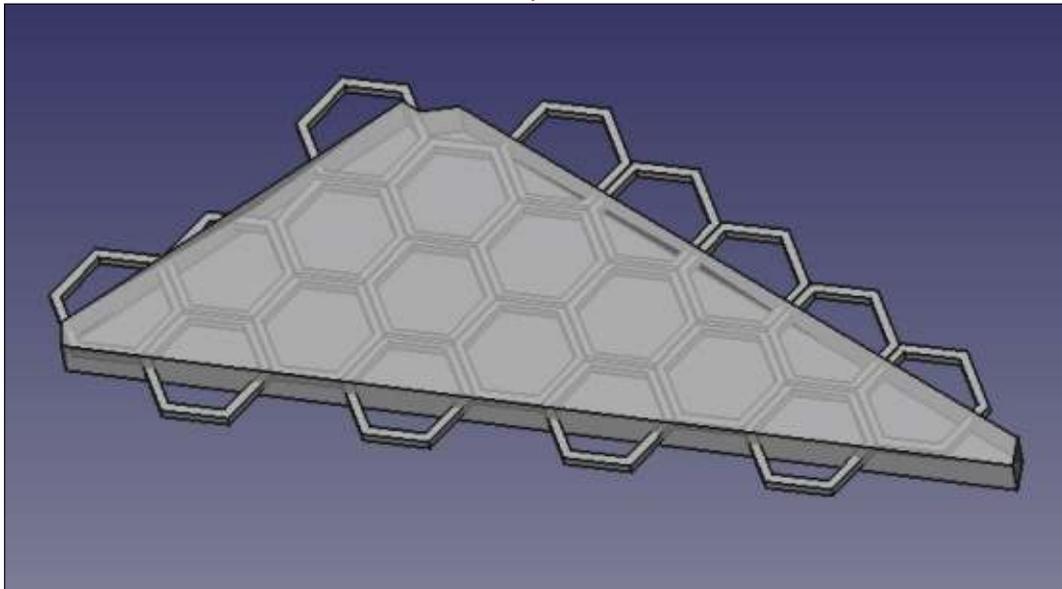
Once we have our object prepared, the printing process should be rather straightforward. Starting in FreeCAD, select the final part and export it into an STL file with menu option File > Export. From there, either use the Slic3r, or any equivalent slicer software,

to slice and print the model.

Once finished, the auxiliary mat can be stripped away. Some surface finishing will probably need to be done, specially on the lower side where it has been in contact with the mat.

WHAT NEXT?

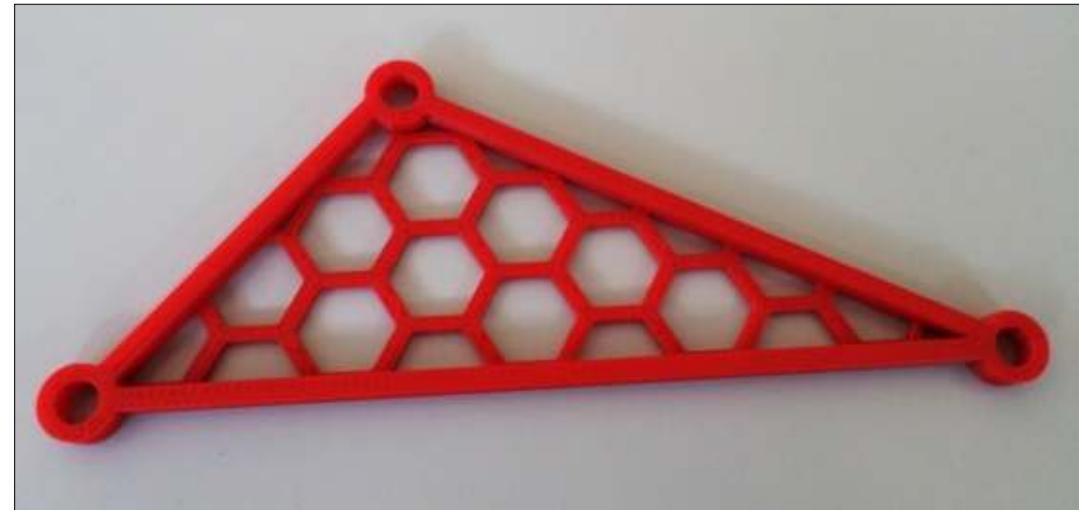
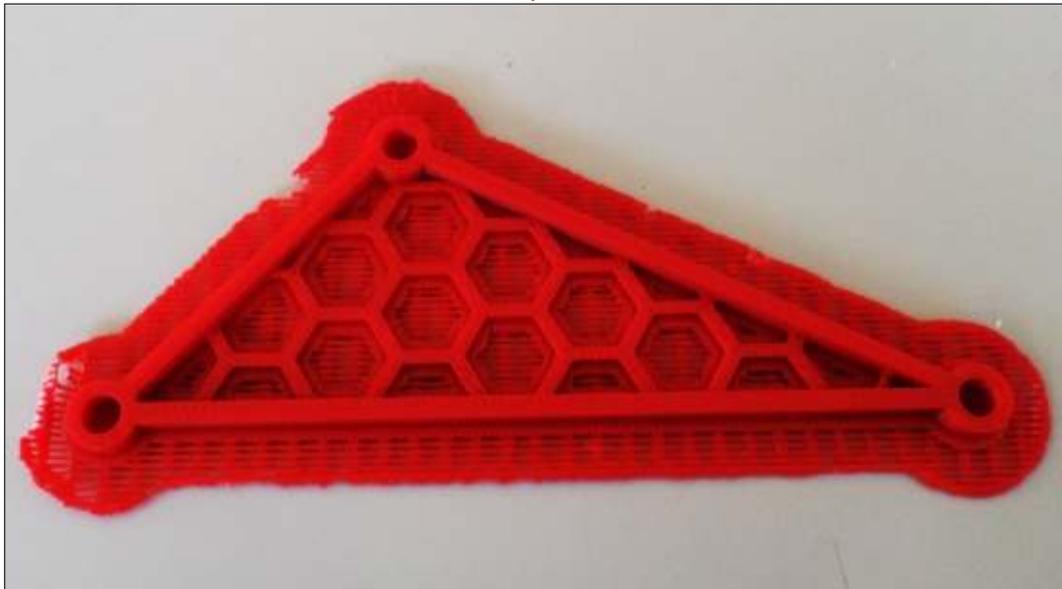
In this article on using FreeCAD, we explored going from a computer model to something physical, using a 3D printing technique. We went through both some of the strong points of 3D printing, and the weak points. We



HOWTO - FREECAD

discussed 3D file formats, we built a model in FreeCAD, and printed it using the Slic3r software. This specific model, built in plastic, would probably have no intrinsic purpose. However, it could be used as a basis for a mold for a metal copy, or simply as teaching material on truss structures and internal stresses within a structural object.

In the next part of this series, we will change direction once more and explore some of the uses of copying objects to create a repetitive pattern such as chainmail.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



HOW-TO

Written by Alan Ward

In this series, we will be examining the world of FreeCAD, an open-source CAD modeling application that is still in Beta, but has been gaining acceptance in recent years. Naturally, it is readily available in the Ubuntu repositories. In the ninth article on using FreeCAD, we used a 3D printer to create a physical representation of a construction. In this episode, we will explore some of the uses of copying

objects to create a repetitive pattern such as chain-mail.

PRINTING INTERLOCKING PIECES

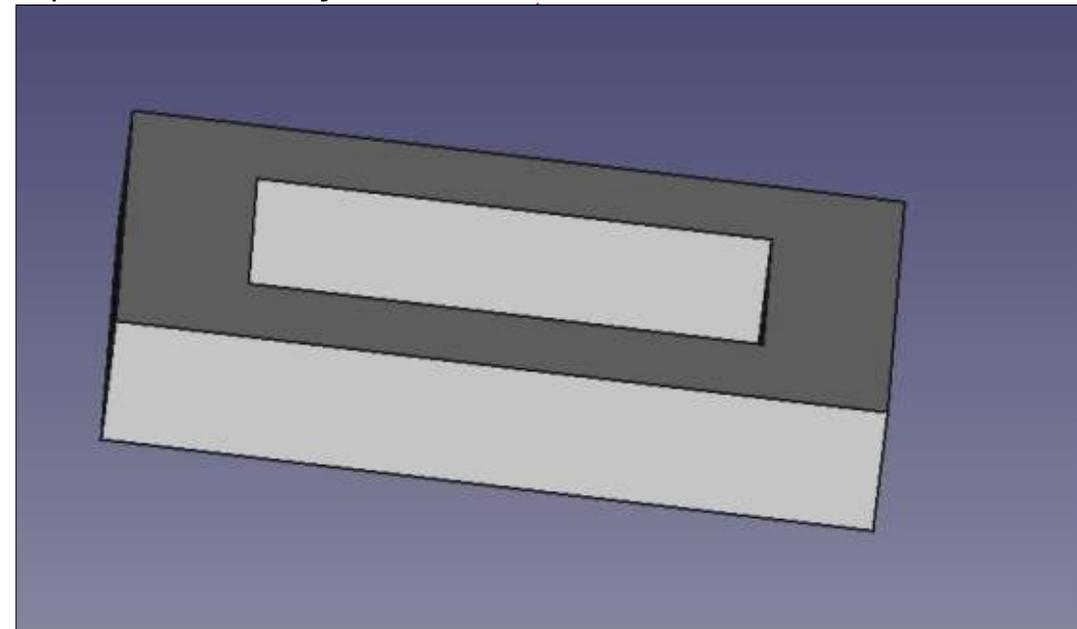
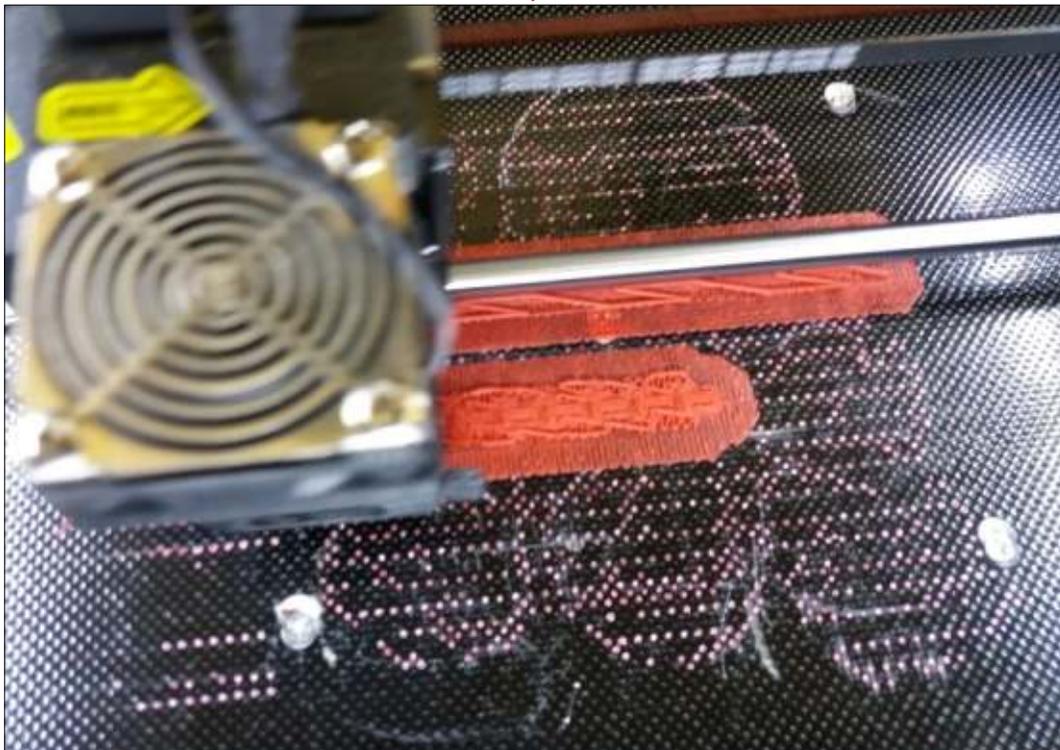
Objects with repeated, but disjointed parts, such as chains or chain-mail, can be printed in 3D in much the same way as a regular, connected, part. The main difference is that the printer's driver software should be smart

enough to add in some extra plastic between bits. This is usually done in much the same way printers begin printing by laying down one or several mats on the bed, so as to ensure the parts being printed stick well to its surface, and do not move around as successive layers of plastic are added. In the adjoined image of a printer, two separate lengths of chain are being printed. In each case, the printer began with the mats, approximately 6 mm (a quarter-inch) out from the object's outline. At this point, several layers of plastic have already been

deposited, and the objects (chain links) are starting to appear. Excess plastic -among them, the diagonal traces and wavy bits- are also there, to ensure links stay upright and in a correct spatial relationship to each other.

CREATING SOME CHAIN

Back to FreeCAD, let us begin by creating a very simple length of chain. We will start by doing a rectangular flat link, with a rectangular hole cut into it using the "cut" operation (Boolean



volume subtraction).

This link shape can be worked upon, and made rather more aesthetic or more functional, as desired. It is worth spending some time at this stage -perhaps rather more than I have done- since what we are producing here is a basic motif that will be repeated many times to create the complete chain object.

Now, let us copy and paste our finished link. The new copy will need to be displaced a sufficient distance (e.g. along the X axis), and rotated by 90 degrees about the axis of displacement. Precise measurements will depend on link dimensions, but, in general, I do

tend to leave, at the least, 1 mm of empty space between links. We now have something similar to this:

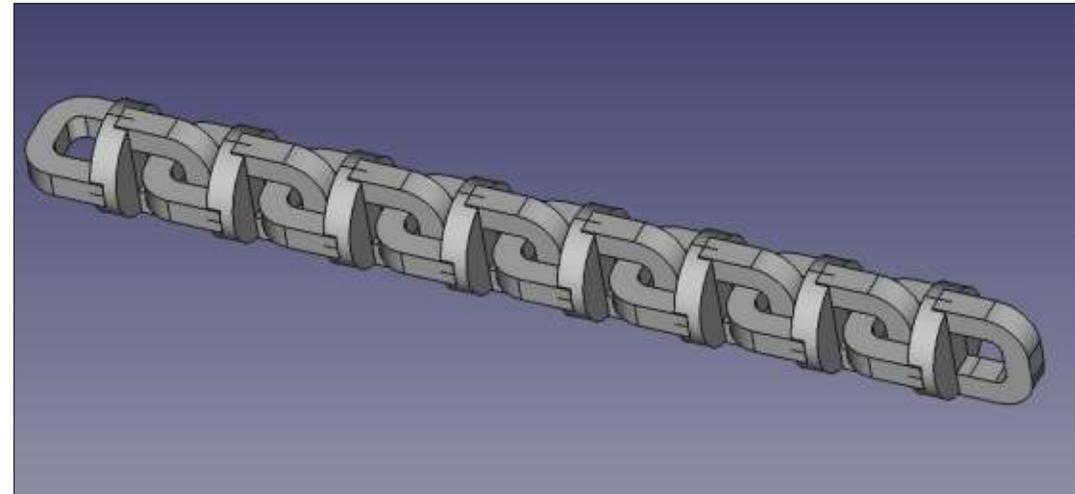
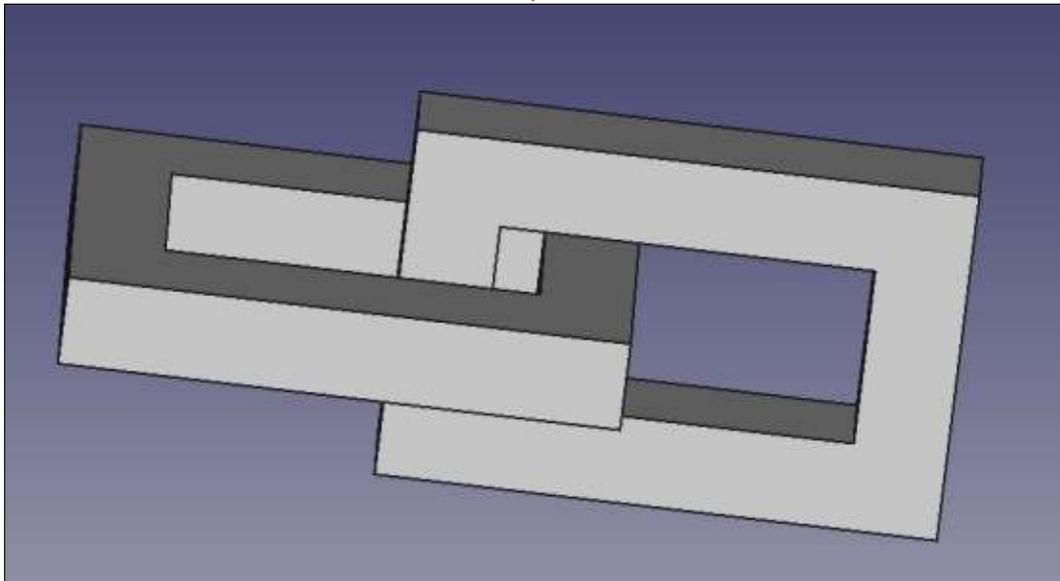
Now, for the weird part. In order to print this in 3D, we will need to move it out to the printer as one single object. But we already have defined two separate volumes, with some free air between them. So, now, we simply need to combine the two objects with a Boolean Union operation. This is really strange for someone with a mathematical background, since we are defining in essence a single volume with two separate and disconnected parts to it. But it does work.

Once we have defined the couple of links as a single union object, we can now copy and paste that, obtaining two strings of two links. One of the strings needs to be displaced along the same axis into a suitable position, and then the two bits need to be fused to each other into a single Union object, as before. We can then continue in this fashion, multiplying the number of links by 2 in each operation: 1, 2, 4, 8, ...

Once we have the workflow set out, we can experiment with more complex basic link forms. For instance, we could create a link with a flat eye and a vertical one. Thus, each successive copy can be displaced only along the axis, without any rotation. As before, the complete chain will need to

end up as a single Union object, which can then be exported as an STL file and sent to the printer.

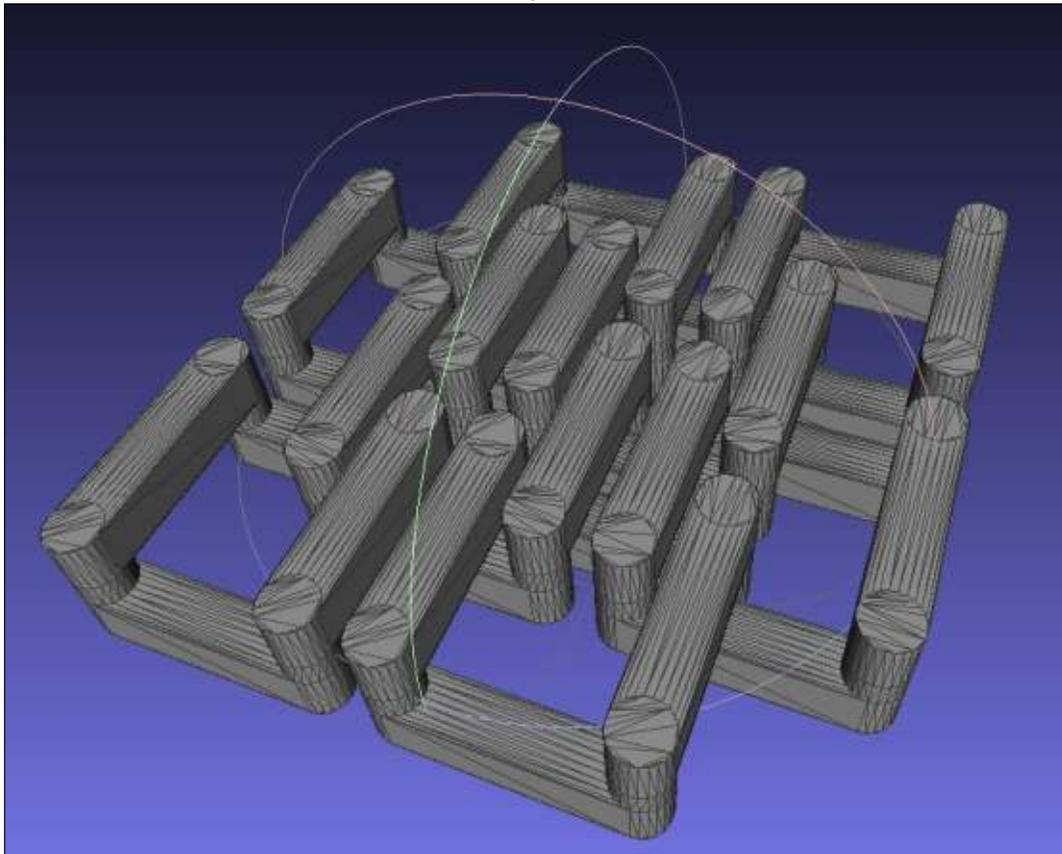
It may be prudent to point out that the excess plastic will need to be pared away from the final object. This will include the mats set out beneath the chain, but also all the various bits and pieces the printer will have added to support the links, and also between each pair of links. A very complex link geometry may hinder getting all these bits out from our assembly. Practical experience shows that an object such as the above chain can take as long to clean up as to do the actual printing – if not more. Thinking ahead, and adjusting link shapes to make the interstitial spaces more easy to access, can be of help.



SPREADING OUT TO CHAIN-MAIL

Going from a linear piece of chain to flat chain-mail is not too complex. The main difficulty is that each link will need to be interlinked with many other links, typically four, so the central space will need to be created large enough to allow neighboring links to pass – while maintaining a separation of about 1 mm

between any two links. As long as this is assured, links can take any form. One typical shape would be flat toroid rings (“donut-shaped”), set at different angles for each alternate row: a close-up of real mail can be seen at this link: [https://en.wikipedia.org/wiki/Mail_\(armour\)#/media/File:European_riveted_mail_hauberk_close_up_vie_w.jpg](https://en.wikipedia.org/wiki/Mail_(armour)#/media/File:European_riveted_mail_hauberk_close_up_vie_w.jpg). Other setups are also possible, for example curving each link into a saddle-shaped ring to aid fitment. Going even further in



this direction, rounded links can be transformed into a collection of simple volumes (cylinders and bars), such as in the adjoining STL file (captured in Meshlab).

Once a basic link motif is created, it can be copied and pasted and separate links displaced into position. Then, several links can be fused into a single Union object, such as the above 8-link assembly, which is then repeated to create a larger piece of mail. Motifs can be added along two axis, to create a flat piece of material, or along one single direction to create a chain-mail band.

Finally, chain-mail is not limited to square links or motifs in which links are connected to four other surrounding links. Three-sided symmetry can be used to create motifs in which triangular or hexagonal links are joined each to three other links. Links with a single ring can be alternated with links made of two parallel rings, joined by vertical parts that interlock with flat rings. This setup actually increases freedom of movement between links, producing a chain-mail fabric that folds and bends much better than

a more traditional square motif.

WHAT NEXT?

In this -final- article on using FreeCAD, we explored some of the uses of copying objects to create a repetitive pattern such as chain-mail. This could then be used as a basic material for different purposes, such as preparing costumes, historical recreations of armored clothing, or even rapid prototyping of jewelry.

Going through the various articles that have come out over the last months, it is clear that a 3D design program such as FreeCAD has many different applications, ranging from mechanical engineering (gears), to architecture (buildings) and arts and crafts (chain-mail). It is always nice to see such software available for the various Ubuntu variants, where casual users benefit from free access to these applications, while more advanced users can use widely tested applications on a very stable platform. This is not always possible with commercial offerings that are often available for a very limited number of operating systems.

After this review of some of the possibilities FreeCAD offers, this series of articles will go dormant for a time. Further along, it may be started up once more, if there is sufficient reader interest. Specific proposals would be very welcome.



Alan holds a PhD in Information and the Knowledge Society. He teaches computer science at Escola Andorrana de Batxillerat (high-school). He has previously given GNU/Linux courses at the University of Andorra and taught GNU/Linux systems administration at the Open University of Catalunya (UOC).



HOW TO CONTRIBUTE

FULL CIRCLE NEEDS YOU!

A magazine isn't a magazine without articles and Full Circle is no exception. We need your opinions, desktops, stories, how-to's, reviews, and anything else you want to tell your fellow *buntu users.

Send your articles to: articles@fullcirclemagazine.org

We are always looking for new articles to include in Full Circle. For help and advice please see the Official Full Circle Style Guide: <http://url.fullcirclemagazine.org/75d471>

Send your comments or Linux experiences to: letters@fullcirclemagazine.org
Hardware/software reviews should be sent to: reviews@fullcirclemagazine.org
Questions for Q&A should go to: questions@fullcirclemagazine.org
Desktop screens should be emailed to: misc@fullcirclemagazine.org
... or you can visit our site via: fullcirclemagazine.org

Please note:

Special editions are compiled from originals and may not work with current software versions.

Full Circle Team

Editor - Ronnie Tucker

ronnie@fullcirclemagazine.org

Webmaster - Lucas Westermann

admin@fullcirclemagazine.org

Special Editions - Jonathan Hoskin

Editing & Proofreading

Mike Kennedy, Gord Campbell, Robert Orsino, Josh Hertel, Bert Jerred, Jim Dyer and Emily Gonyer

Our thanks go to Canonical, the many translation teams around the world and Thorsten Wilms for the FCM logo.

For the Full Circle Weekly News:

You can keep up to date with the Weekly News using the RSS feed: <http://fullcirclemagazine.org/feed/podcast>

Or, if your out and about, you can get the Weekly News via Stitcher Radio (Android/iOS/web):

<http://www.stitcher.com/s?fid=85347&refid=stpr>



and via TuneIn at: <http://tunein.com/radio/Full-Circle-Weekly-News-p855064/>

Getting Full Circle Magazine:

EPUB Format - Most editions have a link to the epub file on that issues download page. If you have any problems with the epub file, email: mobile@fullcirclemagazine.org

Issuu - You can read Full Circle online via Issuu: <http://issuu.com/fullcirclemagazine>. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu.

Magzster - You can also read Full Circle online via Magzster: <http://www.magzster.com/publishers/Full-Circle>. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu Linux.