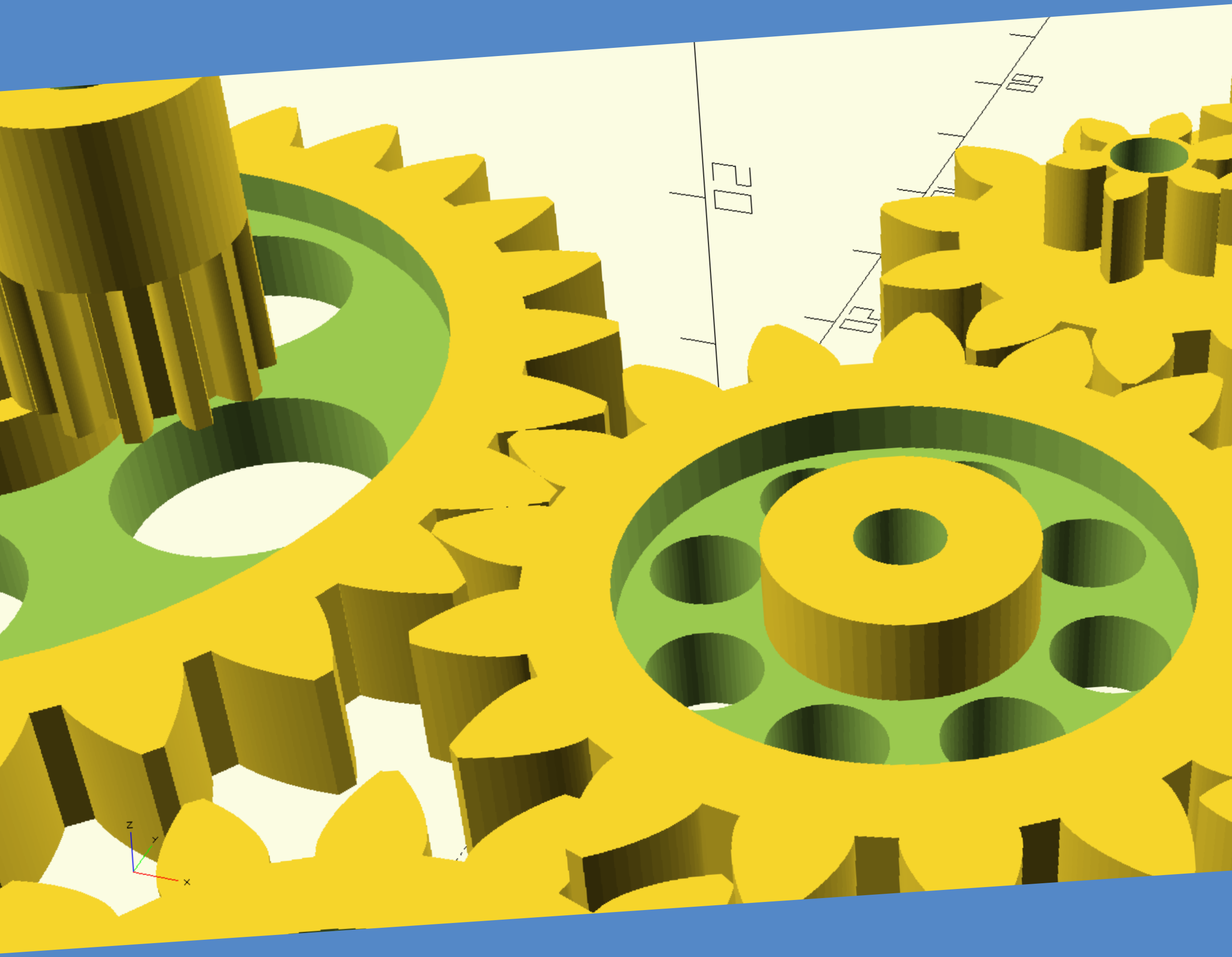
elektor books

Technical Modeling with OpenSCAD

Create Models for 3D Printing, CNC Milling, Process Communication and Documentation



Tam Hanna

elektor
design > share > sell

Technical Modeling with OpenSCAD

Create Models for 3D Printing, CNC Milling, Process Communication and Documentation

Tam HANNA

elektor

● This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
PO Box 11
NL-6114-ZG Susteren, The Netherlands
Phone: +31 46 4389444

● British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

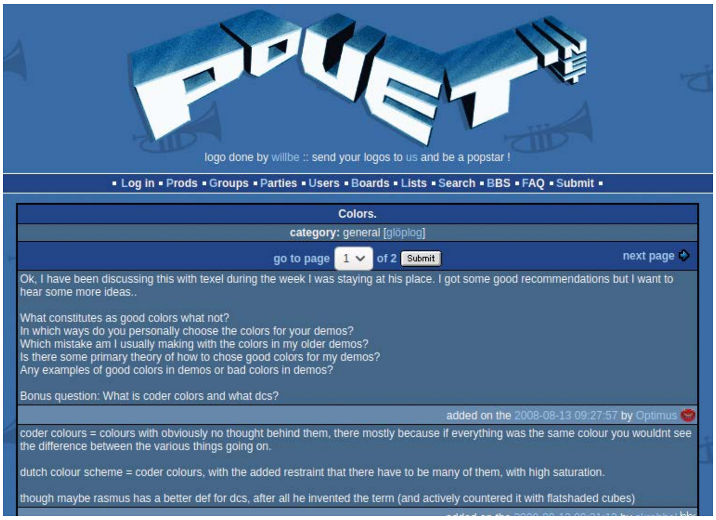
ISBN 978-1-907920-99-8
EBOOK 978-3-89576-394-6
EPUB 978-3-89576-395-3

© Copyright 2020: Elektor International Media B.V.
Prepress Production: D-Vision, Julian van den Berg

Chapter 1 • Why OpenSCAD?

When you think of a 3D design program, you instinctively think of AutoCAD, Rhino, Google SketchUp, or - for the oldies - 3D Studio Max. There's no question that great designs have been and will be created with all these products.

But there is no question that designers and coders have radically different thinking processes - the best proof for this is the definition of the Coder Colors shown in figure one, which stands for such an ugly color scheme that it can only come from the brain of a software developer.


logo done by willbe :: send your logos to us and be a popstar !
▪ Log in ▪ Prods ▪ Groups ▪ Parties ▪ Users ▪ Boards ▪ Lists ▪ Search ▪ BBS ▪ FAQ ▪ Submit ▪
go to page 1 of 2 Submit
next page

Colors.

category: general [glöplog]

Ok, I have been discussing this with texel during the week I was staying at his place. I got some good recommendations but I want to hear some more ideas..

What constitutes as good colors what not?
In which ways do you personally choose the colors for your demos?
Which mistake am I usually making with the colors in my older demos?
Is there some primary theory of how to chose good colors for my demos?
Any examples of good colors in demos or bad colors in demos?

Bonus question: What is coder colors and what dcs?

added on the 2008-08-13 09:27:57 by Optimus

coder colours = colours with obviously no thought behind them, there mostly because if everything was the same colour you wouldnt see the difference between the various things going on.

dutch colour scheme = coder colours, with the added restraint that there have to be many of them, with high saturation.

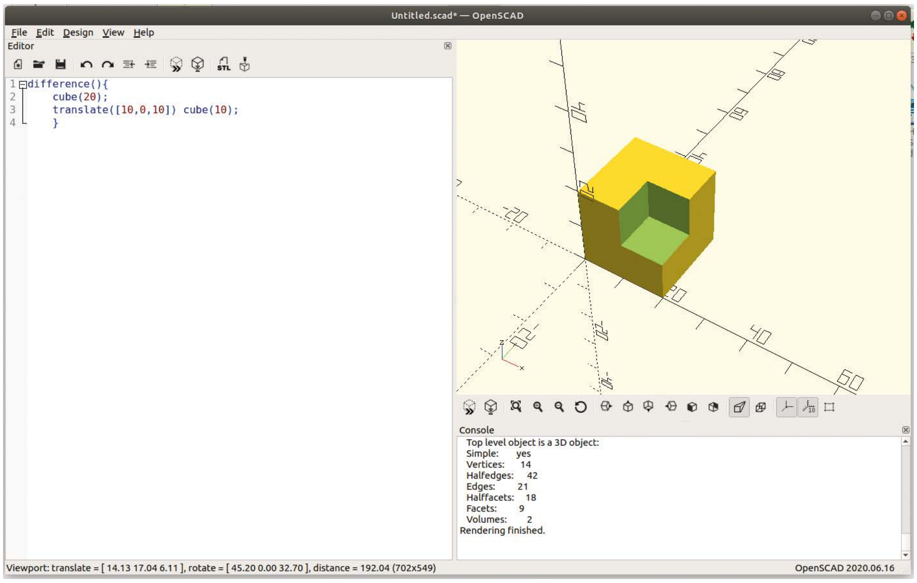
though maybe rasmus has a better def for dcs, after all he invented the term (and actively countered it with flatshaded cubes)

Figure 1-1. The definition of the (now somewhat forgotten) term Coder Colors is not very charming. (Picture source: http://www.pouet.net/topic.php?which=5540&page=1)

Funnily enough, the engineering way of thinking is not exclusively found among computer scientists. The author works as a technical consultant for the American fashion company Icy Beats LLC (see http://www.bopsync.com/), and had a similar experience with the owner. The lady is an excellent and experienced fashion designer but doesn't get along with 3-D modeling software any more than the author does.

The reason for this is that both the fashion designer and the engineer "work with their hands" and therefore naturally think in hand movements and hand actions.

OpenSCAD differs from the systems mentioned in the introduction in that it uses a "step-by-step" approach. Since a picture often says more than 1000 words, figure two shows a "cut out" cuboid next to the source code intended for its generation. Please do not analyze the subtleties of the code at this point - the only important thing to note is that the cuboid is created by "subtracting" two cuboids.



```
difference(){
    cube(20);
    translate([10,0,10]) cube(10);
    }
```

```
Top level object is a 3D object:
Simple:     yes
Vertices:     14
Halfedges:    42
Edges:      21
Halffacets:   18
Facets:      9
Volumes:      2
Rendering finished.
```

Viewport: translate = [14.13 17.04 6.11], rotate = [45.20 0.00 32.70], distance = 192.04 (702x549)   OpenSCAD 2020.06.16

Figure 1-2. A picture is worth a thousand words.

In addition to the engineer-friendly approach, OpenSCAD benefits from the fact that the language implements a kind of object orientation. The best examples of this are shown in figures three, four, and five – they show real parts of the underground habitat of yours truly.

The bottom part of the "brackets" holding the dispenser in place was taken from identical code. The only difference between the two parts was that they were parameterized differently – OpenSCAD took care of generating the rest.

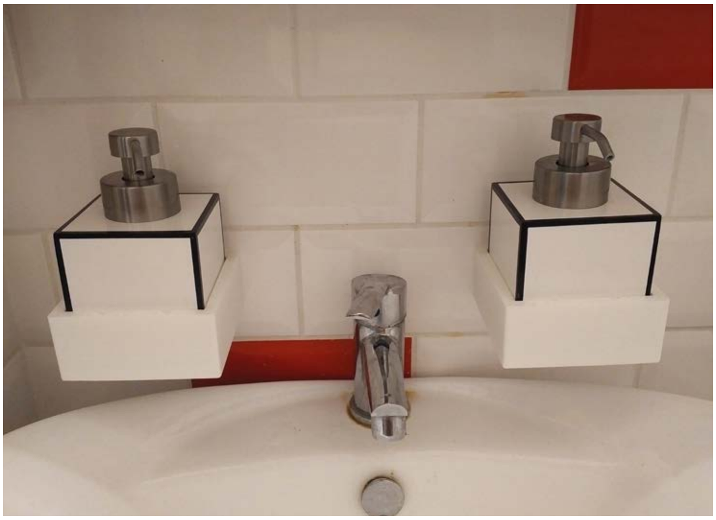


Figure 1-3. No bunker is complete if its users lack soap and lotion dispensers.

Figure 1-4. Naturally, this also applies to the bathroom.

Figure 1-5. ...sadly, black and white dispensers were only available in different sizes.

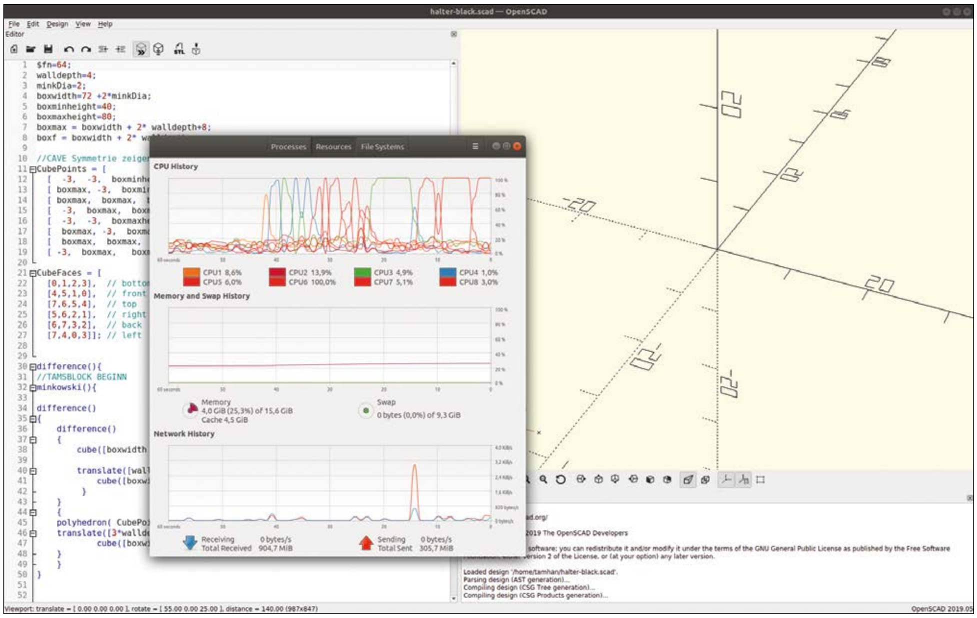
If you always wanted to design "engineering" parts, you will find OpenSCAD a great modeling system. This textbook will introduce you to the basics and advanced application scenarios. I will focus on the actual work with OpenSCAD, and peripherally on interacting with 3D printers. I explicitly do not want to give an introduction to "mechanical engineering - if you are looking for that, find a list of books on different competence levels in the appendix.

1.1 What do we need?

The author demands little from the esteemed reader - spatial awareness is helpful. Safe handling of a caliper gauge is recommended; those who are familiar with a cordless screw-driver and/or a drilling station also have an easier life. Knowledge of electronics is explicitly not required!

From a technical point of view, we do not need much either. OpenSCAD is not demanding, the program works on slow and fast computers. The author illustrates how to set up the program under Windows and Linux; under MacOS, analogous work can be done.
If you don't want to go through the effort of compiling the program yourself, older versions are available as ready-made packages.

If you buy a workstation explicitly for OpenSCAD, pay attention to extremely high single-thread performance. Figure 1-6 shows that the majority of the author's eight-core workstation is bored when working with OpenSCAD, while one core is under full steam.



```
$fn=64;
walldepth=4;
minkDia=2;
boxwidth=72 +2*minkDia;
boxminheight=40;
boxmaxheight=80;
boxmax = boxwidth + 2* walldepth+8;
boxf = boxwidth + 2* wa

//CAVE Symmetrie zeiger
CubePoints = [
  [  -3,  -3,  boxminhe
  [ boxmax, -3,  boxmi
  [ boxmax,  boxmax,
  [  -3,  boxmax,  box
  [  -3,  -3,  boxmaxhe
  [  boxmax, -3,  boxm
  [  boxmax,  boxmax,
  [ -3,  boxmax,   box

CubeFaces = [
  [0,1,2,3],  // botto
  [4,5,1,0],  // front
  [7,6,5,4],  // top
  [5,6,2,1],  // right
  [6,7,3,2],  // back
  [7,4,0,3]]; // left


difference(){
//TAMSBLOCK BEGINN
minkowski(){

difference()
{
    difference()
    {
        cube([boxwidth

        translate([wal
            cube([boxwi
        }
    }
    {
    polyhedron( CubePo
    translate([3*walld
            cube([boxw
    }
    }
}
```

CPU1 8,6% CPU5 6,0% CPU2 13,9% CPU6 100,0% CPU3 4,9% CPU7 5,1% CPU4 1,0% CPU8 3,0%

Memory 4,0 GiB (25,3%) of 15,6 GiB Cache 4,5 GiB
Swap 0 bytes (0,0%) of 9,3 GiB

Receiving 0 bytes/s Total Received 904,7 MiB
Sending 0 bytes/s Total Sent 305,7 MiB

```
ad.org/
2019 The OpenSCAD Developers
software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software
version 2 of the License, or (at your option) any later version.

Loaded design '/home/tamhan/halter-black.scad'.
Parsing design (AST generation)...
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
```

Viewport: translate = [0.00 0.00 0.00], rotate = [55.00 0.00 25.00], distance = 140.00 (987x847)
OpenSCAD 2019.05

Figure 1-6. OpenSCAD no speak parallelisation.

So back to the human factors: OpenSCAD is a programming language. The author of these lines assumes that you have worked with C, Java, Javascript, or Pascal - other languages, whether object-oriented or not, are also suitable.

Otherwise, all you need is a desire to design - the author asks you to at least "skim" the work as a whole to get a grip on the functions contained in OpenSCAD. Learning the syntax by heart is not very useful outside the academic world - there is no reason why you should not use templates, this book, and the cheat sheet (https://www.openscad.org/cheatsheet/) when working with OpenSCAD.

If you can do one, it’s easier to learn another
Experience has shown that a person who is familiar with a programming language can familiarize himself with (most) others relatively quickly. The author can also confirm this from his life experience - he started his career with Pascal, then learned C on the side and has had no major problems with more exotic languages like Python.

If you find yourself confronted with an older version of OpenSCAD, the situation is not too bad. The OpenSCAD developer team was close to a political party that will certainly not be given significant governmental responsibility in Austria for the next ten years and therefore does not have a great deal of money to distribute. As a result - logically, but also somehow, fortunately - a very slow development of the language means that the ecosystem is stable. Harder wars are fought over the question of which 3D printer is suited to the needs of a company or individual. The author of these lines uses two Renkforce RF100 V2 - a small, compact, ready-made device that can be purchased from Conrad for about 175 euros - which (especially with retrofitted component cooling, see the author’s YouTube channel) works well and produces parts up to 12 × 12 × 12 cm in size.

If you have another 3D printer and can handle it with confidence, this is no obstacle. We conduct our experiments with CURA because the author always uses CURA. If you use a different slicer and can already handle it, this is not a problem.

If you don’t have a 3D printer and you can’t find the space (and maintenance time) for such a hangar queen, use the services of model joineries instead. Almost every shop which sells 3D printers accepts print orders and - assuming a friendly approach - is very happy to perform "mercenary printing services" for paying customers.

1.2 Who am I?

Many moons ago, yours truly took to electrical engineering as a way to improve his life. Since then, technical progress gave rich gifts – digital oscilloscopes and 32-bit processors were once unaffordable, but now can be bought even on a limited budget. Colour displays are now so inexpensive that yours truly‘s firm installs them in humidors (!!!) to the delight of all (see picture).

Figure 1-7. A color display in the cigar box: accessible for everyone, thanks to HygroSage.

While advances in all areas of technology solved the electronics problem, the question of realizing custom "in-house" mechanical parts remained open - a wonderful control system for a drone is of no use, even in the presence of civil war and buyers, if you cannot build the actual drone based on it.

When the author was still on active duty, each part required a long and bumpy journey in a ZAZ-965 Zaporizhzhets of the motor pool to talk to the model joinery. A small spar evolved into a task that demanded highly qualified technical personnel.

3D printers have meanwhile answered the "technical" part of this question - you can get admittedly small, but problem-free devices as mentioned in the introduction for less than 200 Euro while in Germany. Imports from China can severely undercut this price.

Sadly, the question of the theoretical design part remained open even as 3D printers became better and cheaper. If you are looking for a way to design mechanical parts, be it housings or brackets, you will find OpenSCAD convenient. Yours truly wouldn't want to do without it anymore - whether furnishing his property, preparing proof of concepts for court hearings, or designing a replacement button for the beloved LeCroy oscilloscope of a French electronics engineer: OpenSCAD is a product that gives pleasure time and time again.

Enough talk: let's now get going in this spirit! And may your designs - whether civil or military - always work to your (and their end user's) utmost satisfaction.

Chapter 2 • Installing OpenSCAD

Given that you are still reading, yours truly is free to assume that you are sufficiently convinced of the benefits of OpenSCAD. To start our experiments, we need a version of the software.

OpenSCAD is open source like many other "novel" engineering programs. This means that you can view the product’s source code and correct any errors found yourself. On the other hand, however, this also means that there are several ways to get our hands onto a runnable binary.

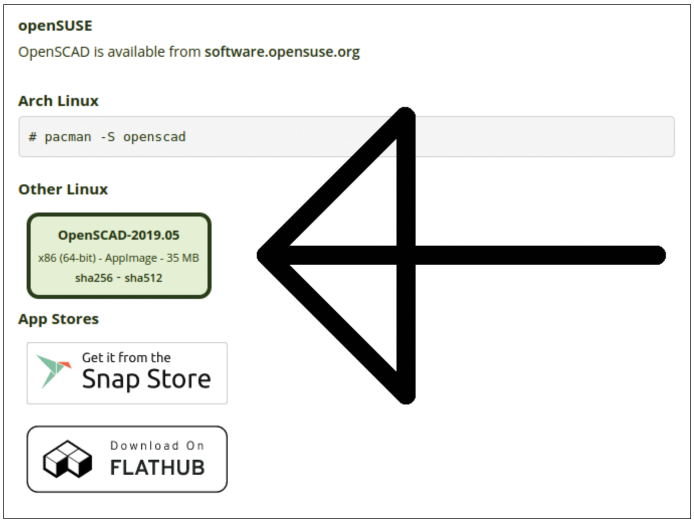
This chapter looks at the deployment under Windows and Linux. For those who want to work under macOS, there are various instructions on the Internet about both compilation and use of provided packages.

2.1 Installing OpenSCAD: Linux, compiled package

Anyone who has spent some time with Linux will certainly know commands like apt-get. They query "package source servers" for a package, and proceed to download and install it afterwards.

Since most distributors perform extensive checks on the packages listed in their package sources, the packages contained there are often not particularly up-to-date - in the case of Ubuntu, for example, the server will provide you with an OpenSCAD version dated from 2015. Quarrels between the OpenSCAD and Ubuntu teams ensure that some releases of Ubuntu have to make do without precompiled binaries.

If you want to work with a reasonably up-to-date OpenSCAD, the first step is to visit the URL https://www.openscad.org/downloads.html. There, scroll down to the Other Linux section. Then click on the link shown in figure one to download a file of about 35 MB.



openSUSE
OpenSCAD is available from **software.opensuse.org**

Arch Linux

```
# pacman -S openscad
```

Other Linux

Figure 2-1. This link leads to happiness.

Please note that "finished" OpenSCAD packages for Linux are only available for 64-bit versions of the operating system. If you absolutely must or want to work with a 32-bit system, you have to do a manual compilation as discussed below.

AppImage files are a "new type of" packaging program which packages Linux applications and the libraries belonging to them. OpenSCAD is started by entering the following two commands:

```
tamhan@TAMHAN18:~/Downloads$ chmod +x OpenSCAD-2019.05-x86_64.AppImage
tamhan@TAMHAN18:~/Downloads$ sudo ./OpenSCAD-2019.05-x86_64.AppImage
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
. . .
```

The first call adds the executable attribute to the file, while the subsequent command instructs to start OpenSCAD. The program reacts by displaying the OpenSCAD start screen. Depending on the distribution, there is the possibility of integrating the pre-built OpenSCAD binary into the program starter - as yours truly prefers to work on the command line level and as distributions change permanently, please refer to the documentation of the respective system. Alternatively, you can also work through the compilation instructions discussed in the following step, which integrate OpenSCAD "completely" into your system.

2.2 Installing OpenSCAD: Linux, compilation.

OpenSCAD version 2019.05, which we downloaded as a finished package, was functional, but is not particularly up-to-date. If you compile OpenSCAD from source, you firstly get the latest version, and secondly can instruct the compiler to provide an experimental version of the OpenSCAD language. This lets you benefit from additional features that cannot be accessed "as is". Annoyingly, OpenSCAD consists of a whole group of components, which is why compilation requires a little work.

The component zoo
If you always wanted to know which libraries work in the background of OpenSCAD, https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Building_OpenSCAD_from_Sources is a highly recommended source.

Like almost all other open-source projects of "larger" dimensions, OpenSCAD development is managed using the git version control system. Our first task is to make sure that git is installed on your workstation:

```
tamhan@TAMHAN18:~/Downloads$ sudo apt-get install git
```

The next step makes the git utility download the latest version of the source code from the repository. Don't be surprised that the author enters this command in the home directory - virtually all git repositories create a new directory during the download process:

```
tamhan@TAMHAN18:~$ cd ~/
tamhan@TAMHAN18:~$ git clone https://github.com/openscad/openscad.git
Cloning into 'openscad'...
. . .
Receiving objects: 100% (64355/64355), 62.52 MiB | 5.11 MiB/s, done.
Resolving deltas: 100% (45989/45989), done.
```

The command downloads the most recent version - this is "disadvantageous" in that during development, non-working or non-compilable versions are sometimes uploaded. In this case, it is recommendable to delete the created directory and then use the branch function to download one of the releases available at https://github.com/openscad/openscad/tree/master/releases.

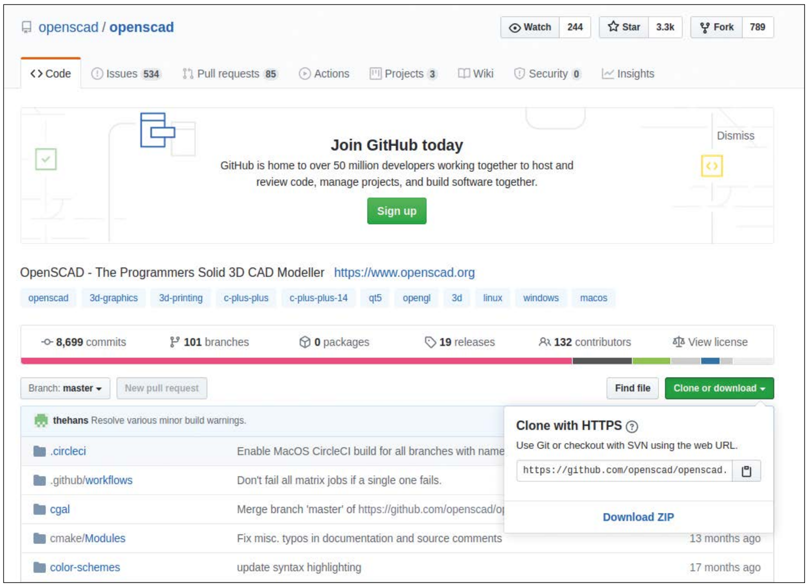
Be that as it may, the next step is to return to the OpenSCAD working directory:

```
tamhan@TAMHAN18:~$ cd openscad/
```

Git repositories have long been capable of mapping relationships between repositories – if a component is based on a library, the component developer can set up a symbolic link to the library's Git repository. This is disadvantageous for us in that the following two commands must be entered within the OpenSCAD folder to provide a fully-fledged source code environment:

```
tamhan@TAMHAN18:~/openscad$ git submodule init
Submodule 'libraries/MCAD' (https://github.com/openscad/MCAD.git) registered
for path 'libraries/MCAD'
tamhan@TAMHAN18:~/openscad$ git submodule update
. . .
Cloning into '/home/tamhan/openscad/libraries/MCAD'...
Submodule path 'libraries/MCAD': checked out
'a7be3d623669d635b7249a327cfce5796ea200b3'
```

The explicit reference to the nested relationships in git repositories is important because git refuseniks like to use the download button shown in Figure 2-2. Unfortunately, its use is not appropriate because the archive it provides does not include the linked repositories.



Join GitHub today

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

OpenSCAD - The Programmers Solid 3D CAD Modeller https://www.openscad.org

openscad 3d-graphics 3d-printing c-plus-plus c-plus-plus-14 qt5 opengl 3d linux windows macos

8,699 commits 101 branches 0 packages 19 releases 132 contributors View license

thehans Resolve various minor build warnings.

.circleci	Enable MacOS CircleCI build for all branches with name	
.github/workflows	Don't fail all matrix jobs if a single one fails.	
cgal	Merge branch 'master' of https://github.com/openscad/op	
cmake/Modules	Fix misc. typos in documentation and source comments	13 months ago
color-schemes	update syntax highlighting	17 months ago

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

https://github.com/openscad/openscad.

Figure 2-2. Git Refugees, beware: this button promises disaster.

The next step provides additional dependencies:

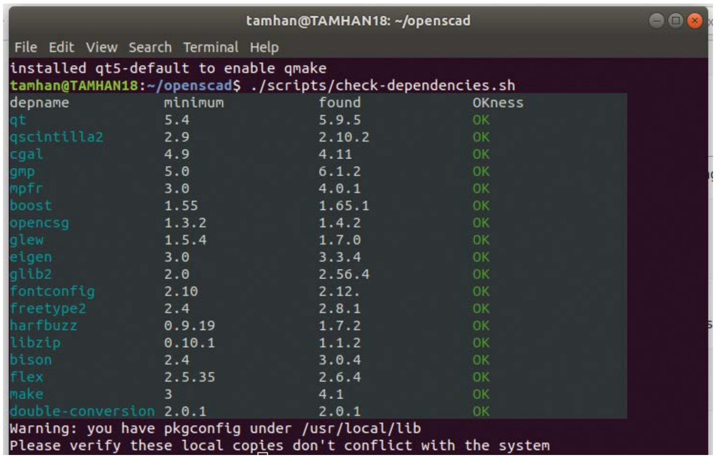
```
tamhan@TAMHAN18:~/openscad$ sudo ./scripts/uni-get-dependencies.sh
. . .
Use ‘sudo apt autoremove’ to remove them.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
installed qt5-default to enable qmake
```

The OpenSCAD development team supports us with a script which automatically downloads the necessary elements using package managers and other tools on somewhat current releases of Ubuntu. If problems occur during this process, you can find help at https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Building_on_Linux/UNIX#Installing_dependencies.

At this point, a reboot of the system is recommended. Afterwards, the existence of the necessary prerequisites can be checked by the following command:

```
tamhan@TAMHAN18:~/openscad$ ./scripts/check-dependencies.sh
```

On the author’s workstation, its processing leads to the result shown in Figure 2-3.



```
installed qt5-default to enable qmake
tamhan@TAMHAN18:~/openscad$ ./scripts/check-dependencies.sh
depname           minimum           found             OKness
qt                5.4               5.9.5             OK
qscintilla2       2.9               2.10.2            OK
cgal              4.9               4.11              OK
gmp               5.0               6.1.2             OK
mpfr              3.0               4.0.1             OK
boost             1.55              1.65.1            OK
opencsg           1.3.2             1.4.2             OK
glew              1.5.4             1.7.0             OK
eigen             3.0               3.3.4             OK
glib2             2.0               2.56.4            OK
fontconfig        2.10              2.12.             OK
freetype2         2.4               2.8.1             OK
harfbuzz          0.9.19            1.7.2             OK
libzip            0.10.1            1.1.2             OK
bison             2.4               3.0.4             OK
flex              2.5.35            2.6.4             OK
make              3                 4.1               OK
double-conversion 2.0.1             2.0.1             OK
Warning: you have pkgconfig under /usr/local/lib
Please verify these local copies don't conflict with the system
```

Figure 2-3. The necessary dependencies are ready for compilation.

Be that as it may, the next step is to run the qmake command-line tool. Its task is to bring the source code into a structure that is "processable" by the compiler and provide various script files and other elements that the compiler will use during its work. If you wish to compile a "normal" version of OpenSCAD, the following input is sufficient:

```
tamhan@TAMHAN18:~/openscad$ qmake
Info: creating stash file /home/tamhan/openscad/.qmake.stash
Project MESSAGE: If you‘re building a development binary, consider adding
CONFIG+=experimental
. . .
to the PKG_CONFIG_PATH environment variable
No package ‚lib3MF‘ found
Project MESSAGE: 3MF Import/Export disabled
```

Don‘t be surprised if the tool throws a group of status outputs during the processing of the qmake command, referring to the absence of various import/export libraries. Unless qmake throws a really serious error, everything will work fine.

If you compile OpenSCAD to take advantage of modern or experimental features, the call to qmake will look slightly different. In this case, you have to provide an additional attribute, which is added via the config command line parameter:

```
tamhan@TAMHAN18:~/openscad$ qmake CONFIG+=experimental
. . .
```

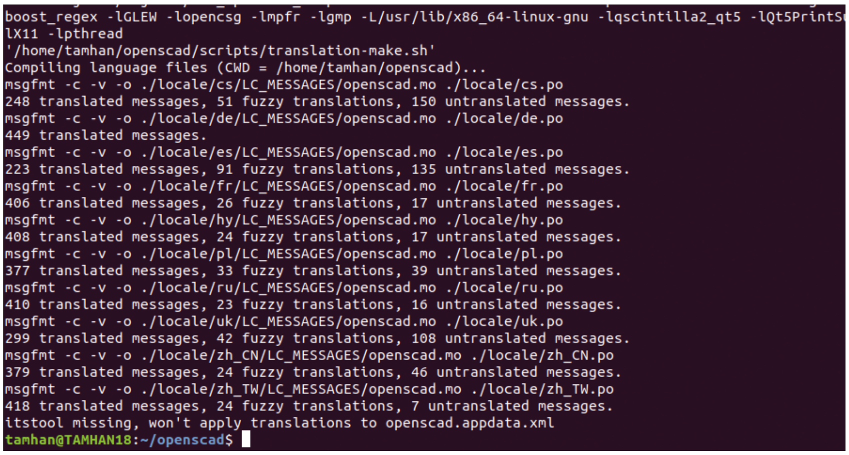
Be that as it may, we can instruct a compilation of our application at this point. Passing the -B parameter instructs make to eliminate all existing Makefiles. This is important if you switch between compiling a normal and an experimental version of OpenSCAD:

```
tamhan@TAMHAN18:~/openscad$ make -B
. . .
```

Due to the considerable codebase, it is recommended to parallelise the actual compilation triggered by make as much as possible. To do this, you have to pass the parameter -j – when sitting on an eight-core workstation, enter the following command:

```
tamhan@TAMHAN18:~/openscad$ make -B -j 8
. . .
418 translated messages, 24 fuzzy translations, 7 untranslated messages.
itstool missing, won't apply translations to openscad.appdata.xml
tamhan@TAMHAN18:~/openscad$
```

As shown in Figure 2-4, make sometimes throws errors during execution, indicating problems with the translation files. However, these are irrelevant to the functioning of OpenSCAD.



```
boost_regex -lGLEW -lopencsg -lmpfr -lgmp -L/usr/lib/x86_64-linux-gnu -lqscintilla2_qt5 -lQt5PrintSu
lX11 -lpthread
'/home/tamhan/openscad/scripts/translation-make.sh'
Compiling language files (CWD = /home/tamhan/openscad)...
msgfmt -c -v -o ./locale/cs/LC_MESSAGES/openscad.mo ./locale/cs.po
248 translated messages, 51 fuzzy translations, 150 untranslated messages.
msgfmt -c -v -o ./locale/de/LC_MESSAGES/openscad.mo ./locale/de.po
449 translated messages.
msgfmt -c -v -o ./locale/es/LC_MESSAGES/openscad.mo ./locale/es.po
223 translated messages, 91 fuzzy translations, 135 untranslated messages.
msgfmt -c -v -o ./locale/fr/LC_MESSAGES/openscad.mo ./locale/fr.po
406 translated messages, 26 fuzzy translations, 17 untranslated messages.
msgfmt -c -v -o ./locale/hy/LC_MESSAGES/openscad.mo ./locale/hy.po
408 translated messages, 24 fuzzy translations, 17 untranslated messages.
msgfmt -c -v -o ./locale/pl/LC_MESSAGES/openscad.mo ./locale/pl.po
377 translated messages, 33 fuzzy translations, 39 untranslated messages.
msgfmt -c -v -o ./locale/ru/LC_MESSAGES/openscad.mo ./locale/ru.po
410 translated messages, 23 fuzzy translations, 16 untranslated messages.
msgfmt -c -v -o ./locale/uk/LC_MESSAGES/openscad.mo ./locale/uk.po
299 translated messages, 42 fuzzy translations, 108 untranslated messages.
msgfmt -c -v -o ./locale/zh_CN/LC_MESSAGES/openscad.mo ./locale/zh_CN.po
379 translated messages, 24 fuzzy translations, 46 untranslated messages.
msgfmt -c -v -o ./locale/zh_TW/LC_MESSAGES/openscad.mo ./locale/zh_TW.po
418 translated messages, 24 fuzzy translations, 7 untranslated messages.
itstool missing, won't apply translations to openscad.appdata.xml
tamhan@TAMHAN18:~/openscad$
```

BILD 2-4. The make run is quite communicative.

Why qmake and make?
OpenSCAD is based on the C++ cross-platform framework QT. It not only offers developers libraries and a GUI stack, but also extends the C++ language with functions such as a signal slot system.

To be able to process applications equipped with these attributes with the usual compiler of the operating system, an additional step is required. qmake takes care of building the Makefiles in a way that instructs Qt's helper infrastructure to be activated. After the qmake command has been processed, a Makefile is found in the project directory, which is responsible for the actual compilation using the operating system's compiler.