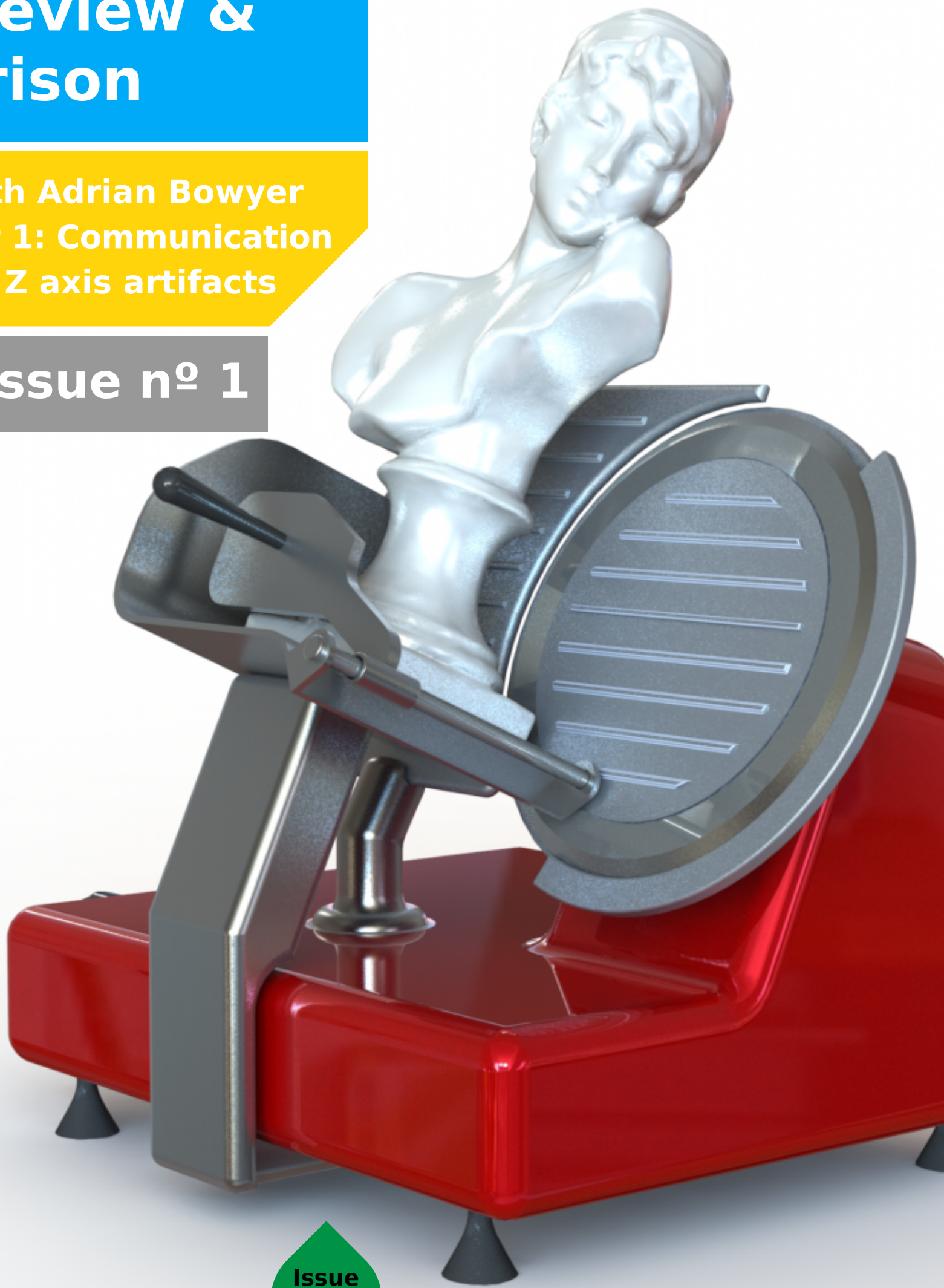
WWW.REPRAPMAGAZINE.COM

REPRAP MAGAZINE

Practical exploration of 3D printing

Slicer review & comparison

Interview with Adrian Bowyer
Repetier part 1: Communication
Taxonomy of Z axis artifacts



Issue nº 1

February 2013

Interviews - Pro tips - Reviews

Issue 1

Diy projects - Tutorials - Galleries

WWW.REPRAPMAGAZINE.COM

REPRAP MAGAZINE

Web:

www.reprapmagazine.com

Magazine team

Developers

Paulo Gonçalves
Gary Hodgson
Richard Horne
Viktor Dirks
Cameron MacLachlan

Contacts

general@reprapmagazine.com

editorial@reprapmagazine.com

RepRap Forums

Reprap Magazine

RepRap Magazine

Welcome to the first edition of RepRap Magazine! This project is brought to you by a small team of enthusiastic and motivated users, experts and developers of RepRap 3D printers project. We aim to develop this project in an open way, working on a close relationship with our readers.

For this first issue, as our cover mentions, we focus the main article on slicers, but also take a close look at the pro-tip of the issue, as well as the software section where we start a series of articles dedicated to Repetier.

While it is hard to choose the highlight of this issue we would have to point to our interview with Adrian Bowyer, the person who started the RepRap project and who was kind enough to give us an interview for our first edition.

With all that said, we hope that you enjoy this edition and stay tuned on this project.

Paulo Gonçalves
Editor

This issues cover credits:

Design **Paulo Gonçalves**
3D slicer model **Nebule**
3D Upset lady model **Onur AYTEKIN**

Our mission

To the readers

We want to be have a close relationship with our readers. For that we encourage you to participate in this project. Send us photos of your best prints and your setup for possible publication to our dedicated email at general@reprapmagazine.com. Also take part at the discussion at the http://forums.reprap.org/list.php?305.

To the contributors

This is an open magazine, and for that we encourage you to submit your articles for possible publications to our email at general@reprapmagazine.com. If you are also a developer of a tool that RepRap users use you can also send an email to be in our database for future contacts.

Independent

We are 100% independent. The manufacters of the products featured do not determine our content nor our opinions.

Contents

This issues team

Roland Littwin

Gary Hodgson

Richard Horne

Rich Cameron

Paulo Gonçalves

Follow us, and get in touch, at
www.reprapmagazine.com

In the works

Many in the Maker community are always looking for new and interesting projects to follow and perhaps take part in, and so this section gives a brief look at up-and-coming projects from in and around the RepRap community.

It should be noted that unless otherwise stated these projects are most definitely works in progress and not ready for general consumption.

If you like to live on the cutting edge then these projects might be for you.

by Gary Hodgson

CoffeeSCAD

Mark Moissette

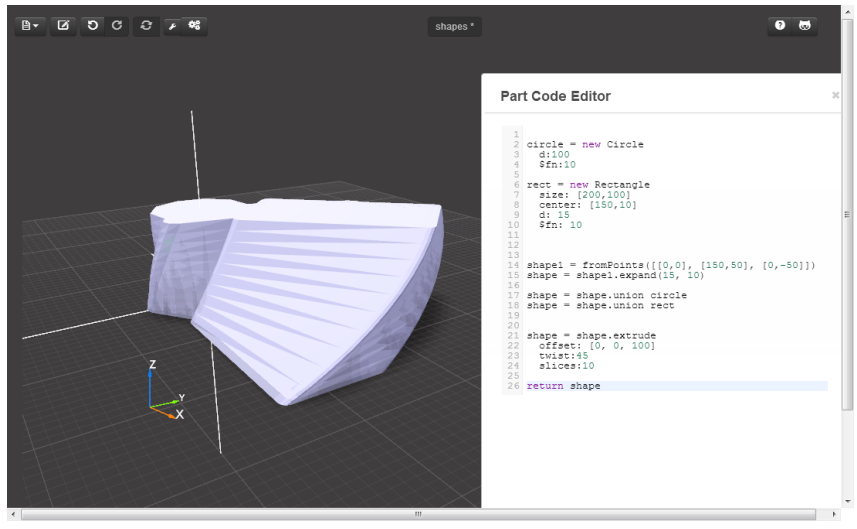
https://plus.google.com/CoffeeSCad

https://github.com /kaosat-dev/CoffeeSCad

Creating 3D models via a scripting language brings several advantages: precise dimensions, modularisation, and being able to take advantage of programming constructs to effortlessly generate complex shapes. Chief amongst the current offerings is OpenSCAD [1], a free, open source, multi-platform application with it's own syntax for creating models. Inspired by this, and others efforts to bring CSG modelling to the browser, namely OpenJsCad [2], Mark Moissette has developed CoffeeSCAD which allows the use of CoffeeScript as the scripting language. CoffeeScript [3] is the current darling of the web development community - a language which compiles to Javascript and brings a clearer syntax, plus some sugar, to the table. This also means that plain old Javascript is valid as well.

So what are the advantages of CoffeeSCAD over OpenSCAD? The first is zero install. CoffeeSCAD runs in the browser and is only a hyperlink away. Want to quickly design a part whilst away from your PC? CSG designing is available wherever there is an internet connection and a browser that supports WebGL. However, this brings with it an interesting challenge, namely where do the files get saved to. Currently they are stored in the browser using the HTML5 storage feature, and Mark is working on alternative solutions to save files in an online repository, or locally, via a small, optional backend, for those not wishing to have everything in the cloud.

The next advantage is the power of Coffee/Javascript. OpenSCAD's syntax allows the user to do a lot, but is limited in several respects: runtime variables and objects are missing for example. Having a mature scripting language available opens up a whole range of options, such as dynamic arrays and object-oriented programming.



CoffeeSCAD has several additional features, including real-time visualisation updates, project organisation, and BOM generation.

There is also a further, geeky, and perhaps slightly less obvious, advantage. OpenSCAD is written in C++ which has a considerable learning curve and a reputation for being a formidable language to master. Producing a similar application in CoffeeScript may attract others to join in development by lowering the barriers to entry. Developing for such a web-based application is decidedly easier than native C++ applications, if only because the build requirements are so much lower.

The project is still under heavy development but a demo system [4] is already available to play with and the source code is available under github [5] for browsing or hacking on. Progress can be followed via the dedicated Google+ page [6].

[1] http://www.openscad.org/
[2] http://joostn.github.com/OpenJsCad/
[3] http://coffeescript.org/
[4] http://kaosat-dev.github.com/CoffeeSCad/
[5] https://github.com/kaosat-dev/CoffeeSCad
[6] https://plus.google.com/u/0/117965920069380418940

OctoPrint

Gina Häußge

https://plus.google.com/OctoPrint

https://github.com/foosel/OctoPrint

What started as a fork of Cura [1], for adding a web interface, has grown and developed into it's own fully-fledged web-based controller for 3D printers. Gina Häußge started work on what was originally called PrinterWebUI at the end of 2012, several iterations later it has a new name, a new home and even a funky new logo.

For those users who have a dedicated laptop or PC attached to their printer, a web interface allows them to check on and control the print from a remote machine. Some of the desktop applications have started introducing web interfaces, for example Kliment's Printrun, but OctoPrint ditches the local interface completely in preference for a web application.

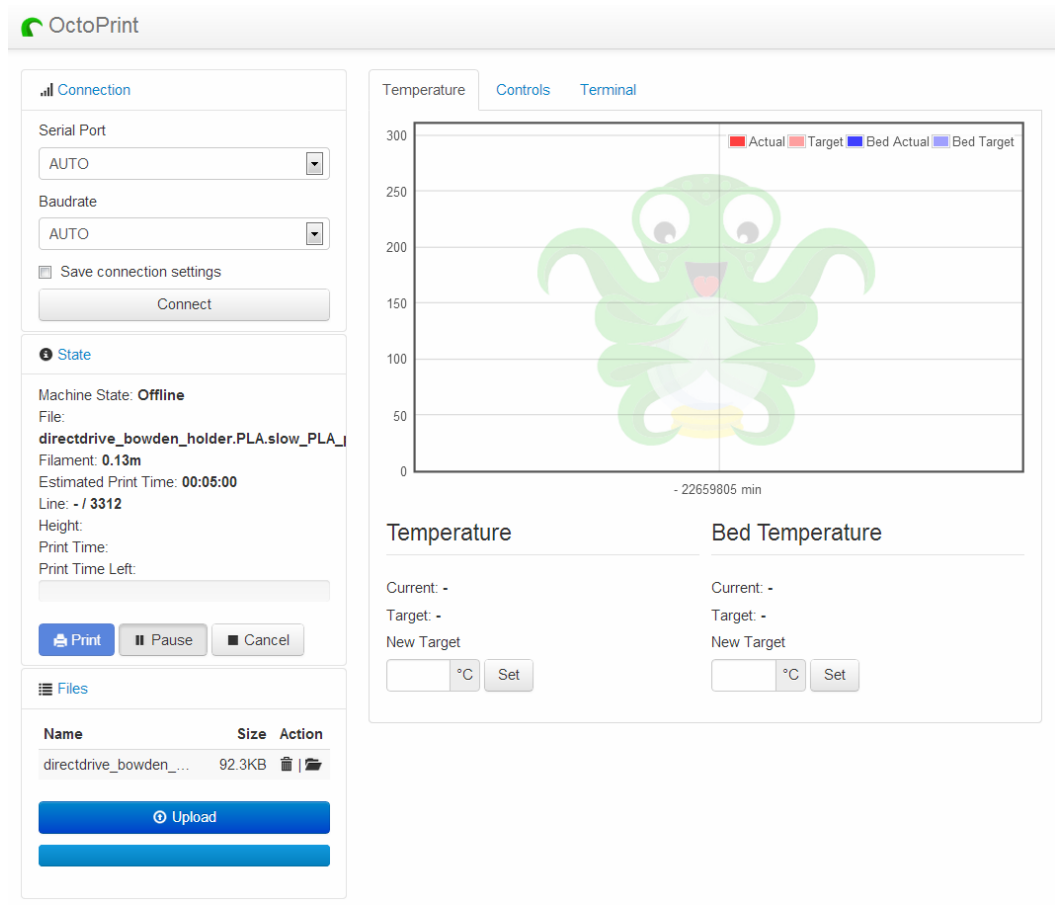
Such an interface is also ideally suited for a setup using the Raspberry Pi board as printer controller, and the instructions on the OctoPrint Github page [2] explain how to set this up.

The interface has many of the features expected by today's controllers: pausing and cancelling of prints; jogging of the axes; status of the current print; sending commands to the printer, etc.

Another feature that is sure to be useful for controlling a printer remotely is the ability to stream from a webcam, showing the progress of your print. No matter how reliable your printer appears to be, accidents and fails can happen at any time, and being able to monitor the progress of the print certainly adds that piece of mind.

The webcam can also be used to take time lapse movies of the print emerging. This is not only useful for showing off your printer in action, but also for potentially analysing the cause of print failures. Jason Gullickson's review of OctoPrint has a great example of how a seemingly good print turns bad just as the end was in sight [3].

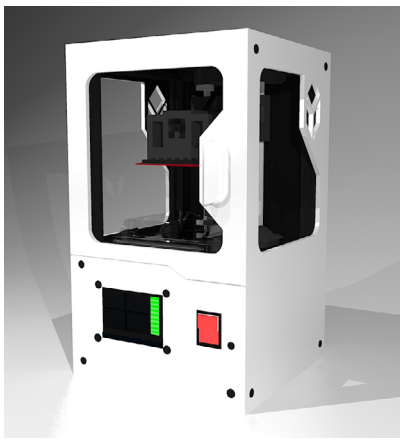
The pace of development is brisk and so Gina recommends to those wishing to test out the new stuff to take a build from the "devel" branch on Github. Once the feature is complete it is merged into the more stable master branch, ready for general consumption. At this stage the project is suitable for testing and Gina is actively looking for feedback, particularly from a wide range of printer types.



Progress of the project can be followed over OctoPrint's dedicated Google+ page [4].

[1] https://github.com/daid/Cura
[2] https://github.com/foosel/OctoPrint
[3] http://www.gullicksonlaboratories.com
https://plus.google.com/u/0/110130855001142142895/posts

sLAMPS



Justin Hawkins

https://plus.google.com/
Justin Hawkins

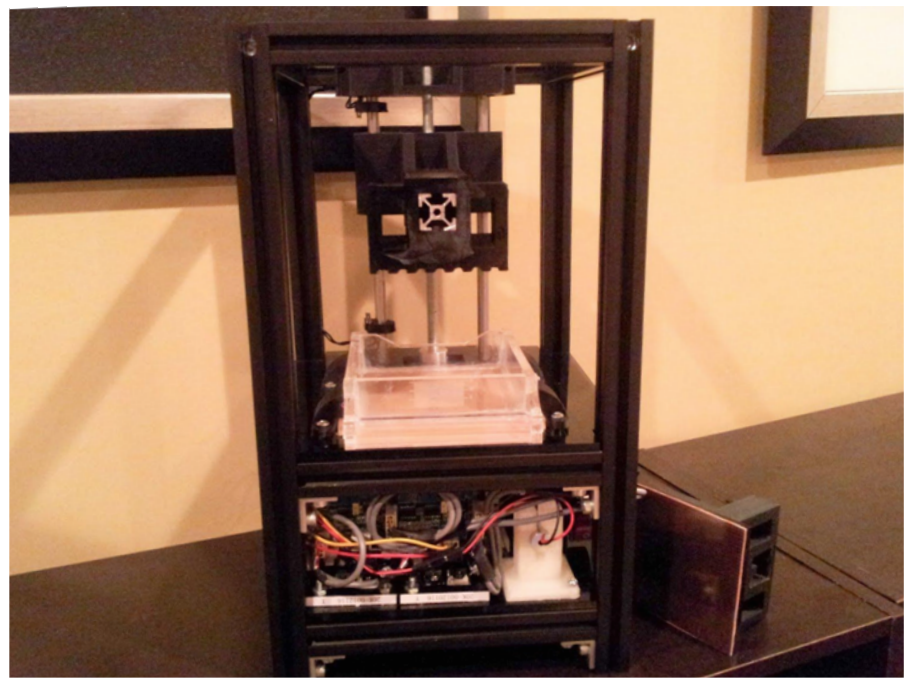
Resin-based 3D printers open up a whole new level of high-resolution prints. However there are several key factors holding back widespread adoption: the cost of resin, and the lack of affordable, open source designs. Justin Hawkins is attempting to tackle the latter by developing a low-cost SLA (Stereolithography) printer which will be made open for the RepRap and 3D printing community.

Stereolithography has been around since the 1980's but patents and high operating costs have meant that it has only recently garnered the attention of the DIY crowd. In 2011, Rob Hopeless posted an Instructable [1] detailing how to build a home-made SLA printer, capturing the imagination of enthusiasts everywhere. Since then there have been several projects emerge, either using lasers e.g. the recent Kickstarter based campaign from FormLabs, or using DLP projectors, such as the successfully funded B9Creator [3].

With the exception of the B9Creator, whose source designs are supposed to become available once all the Kickstarter units have shipped, there is little in terms of open source options for the Maker movement. The lemoncurry project aims to provide information for those wishing to build a DLP printer, but contains no detailed plans, and the RepRap project has focussed almost exclusively on FDM printing.

Justin's hopes are for a sub-$1000, open source SLA printer that could be a part of the RepRap stable. It consists primarily of off the shelf parts so as to make sourcing them as easy as possible, and the custom parts - such as the laser-cut case, custom electronics and printed parts - will have the design files made available for people to source themselves. He is also expecting to sell the required components via a web-shop, and is considering doing a funding campaign to bring the printer to a wider audience at the best price possible.

The current version consists of eletronics based on a Arduino shield, ala RAMPS, whitch control the laser galvanometers. This is being redesigned into a more efficient, and cheaper standalone board with the help of RepRap regular, Kliment. A custom version of Marlin provides the firmware.



The project is currently in intensive development and whilst progress is quick there is still much to be done before the printer sees the light of day. Chief amongst the things left to do is source and test the perfect resin, of which a bulk order should bring down the costs for the end-users.

Justin adds: "Once I release the design, I look forward to seeing where everyone can take the design, and just how far we can take Open Source High Resolution printing." We can only agree, and look forward to seeing the results, and enjoying the benefits, of his hard work.

[1] http://www.robhopeless.com
[2] http://formlabs.com
[3] http://b9creator.com
[4] http://code.google.com/p/lemoncurry

Interview

Adrian Bowyer (English pronunciation: /'boʊjər/) is a British engineer and mathematician, formerly an academic at the University of Bath.

In 1977 he joined the Mathematics Department at the University of Bath. Shortly after that he received a doctorate from Imperial College London for research in friction-induced vibration. Whilst working in the Mathematics Department he invented (at the same time as David Watson) the algorithm for computing Voronoi diagrams that bears their names (the Bowyer–Watson algorithm).

He then spent twenty-two years as a lecturer then senior lecturer in the Mechanical Engineering Department at the University of Bath. He retired from academic life in 2012, though he is still a director of the company RepRap Professional Ltd. He invented the RepRap Project - an open-source self-replicating 3D printer. The Guardian said of this, "[RepRap] has been called the invention that will bring down global capitalism, start a second industrial revolution and save the environment..."[

Adrian Bowyer
From Wikipedia, the free encyclopedia

Interview with

Adrian Bowyer

Gary Hodgson

Author

Alias:
garyhodgson
Country:
UK, living in Germany
Website:
http://garyhodgson.com

The RepRap project started at Bath University, how exactly did it come about? Did the idea originate from yourself or did it come out of discussions with the University?

Despite the dictates of modesty, I have to say that the idea was entirely mine. Like most ideas, it was a convergence of several previous ideas coming together; here's a list, tidied and fictionalised by the unconscious processes of recollection:

1. I have been interested in the idea of artificial replicators since childhood. I can't remember where that came from.

2. Around the turn of the century, Bath University got an equipment grant, and I suggested that they spend it on two 3D printers. This was nothing to do with Item 1. - I just wanted access to them to make things.

3. As soon as the machines arrived I realised that here, at last, was a manufacturing technology that was powerful enough to replicate a significant fraction of itself.

4. I also realised that a self-replicating machine had to be a solid Darwinian success, independently of superficial and ephemeral froth like mere economics.

5. I decided that the way to do this was to copy an evolutionarily stable strategy from nature. The one I chose was the mutualist symbiosis between the flowers and the insects, as I have described elsewhere.

6. Almost as soon as I had the idea I realised that it was very powerful, and that the only way to prevent that power from falling into the wrong hands was to give it to everyone.

7. Literally minutes after I thought that I realised that you have to give any self-replicating device away anyway, otherwise you put yourself in an eternal battle trying to stop people doing with your idea the one thing it was intended to do.

Items 4. through 7. where what made me start RepRap as open-source from its very beginning.

In the interest of others who may wish to attempt something similar, how did you convince the University to support the project? Was the indirect educational effect sufficient?

I told them what I was going to do. Academic freedom means (interalia) that the individual academic has absolute discretion over how they disseminate the results of their research. It is usually academic cupidity (rather than Universities' bureaucracy) that causes academics to, for example, pursue patents before publication. But anyway, my University was happy with my decision, which more-or-less came to them as a fait accompli.

What does your recent retirement from the University entail for the project there? Will it continue under the direction of other staff and the students or will it be closed down?

There are other staff still pursuing aspects of the project. Most of this work is of the form "Application of RepRap to Problem X".

The project started with several key tenets which marked it as unique amongst perhaps similar endeavours (the DIY CNC scene for example), namely: the goal of self-replication; the GPL; and the "Wealth Without Money" essay. These, together with the project's academic roots, defined the project as part of a much wider, more ambitious and perhaps philosophical, scheme. Considering the recent commercialisation, and associated hype, surrounding 3D printing, do you feel that this is a natural phase for the project to be going through, and one that ultimately supports the original goals? Or do you feel this is a distraction which must be endured before the project regains its original focus?

More the former than the latter. I always thought that, if RepRap went anywhere, it would become parasitic on commerce.

An old friend of mine is highly amused whenever he visits and I say, "I just have to go and set the RepRaps printing. Won't be long."

"Your prediction has come true," he says. "You always said that RepRap would entrap people by making them want to assist it to replicate. And it has enslaved its very inventor to that end, printing itself - as he sees it - for the benefit of his company. Whereas in reality the company is for the benefit of the machine."

As Samuel Butler said, "A hen is only an egg's way of making another egg."

Are the original goals of the project (self-replication, disruption of traditional means of production) still relevant, or necessary, now that the project has reached the stage it has?

Self-replication was, and remains, the primary goal, for me at least. It is the goal to which every other possible sub-goal is necessarily subservient. The potential disruptions are not really even goals; more by-products. The spread of the machine is pretty much doing what I thought it would.

RepRap is a major part of the current "maker revolution", either directly (spin-offs such as Makerbot), or indirectly (inspiring and enabling Open Hardware projects). What aspect or challenge needs to now be accomplished for RepRap to maintain this position within the DIY 3D printer and maker community?

Two main developments:

1. A design that is very very quick and easy to assemble.

2. Multiple materials.

I don't really think that maintaining the position is a problem. After all, if every non-replicating 3D printer makes just one RepRap at some point in its life, you can see what that does to the population dynamics.

Did you imagine back in 2004 that the RepRap project would gain such global success? Has the subsequent growth lived up to your expectations?

At the start I thought it was 50/50: it would either sink without trace, or go spectacularly global. It seemed to me that its inherently exponential nature would not admit of any point between those two; a bifurcation was inevitable.

The coin toss came down heads.

The evolutionary process is sometimes used to describe the project. Considering that the RepRap project is a machine that aims for self replication, and has an evolution comparable to living species (survival of the fittest, evolutionary dead ends, etc.), in your opinion what is the place that RepRap holds in the food-chain of the 21st Century manufacturing environment? Who is its main predator and its favourite prey?

I'm aiming for the bottom: I'd much rather it was a bacterium than a leopard - leopards (as a species) are not around for long.

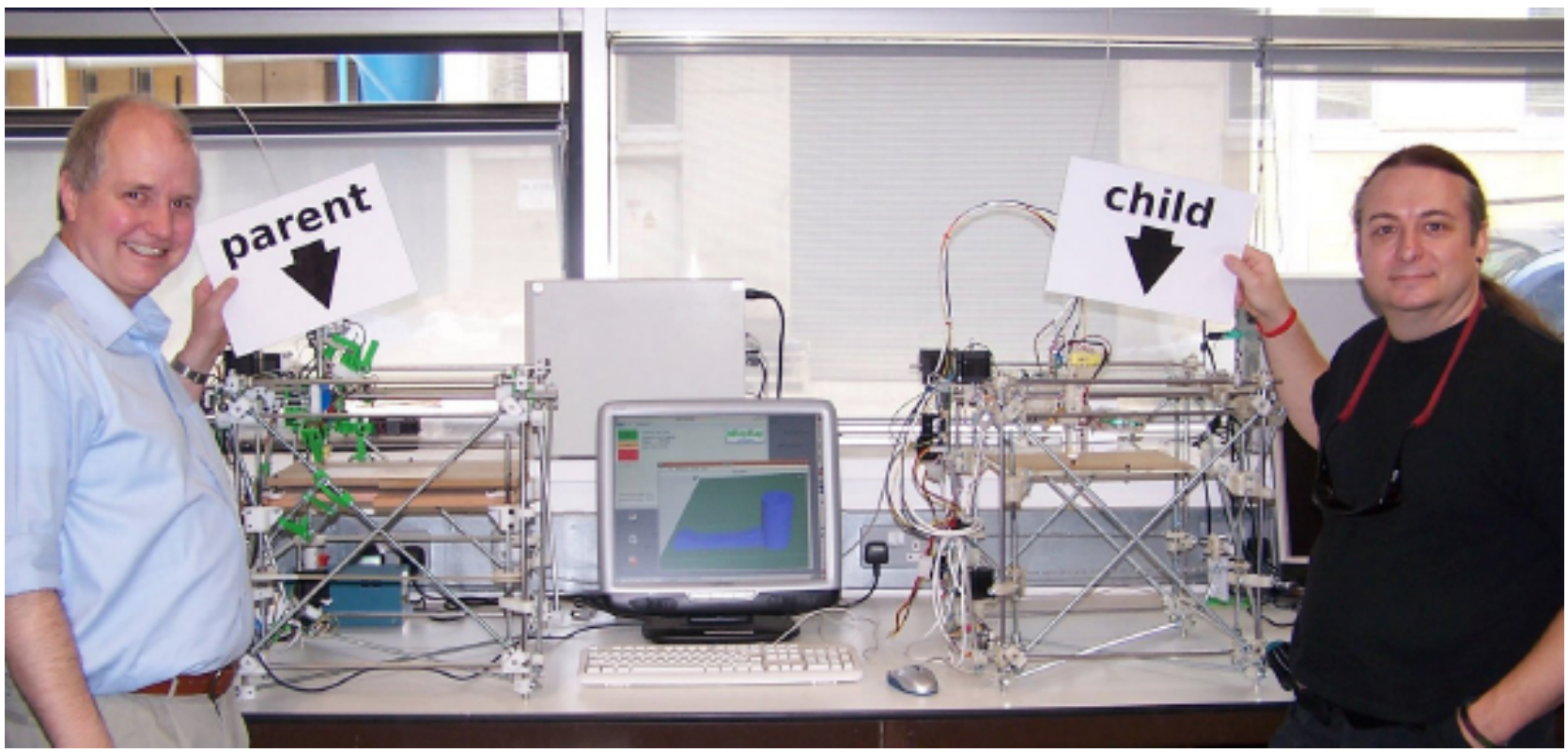
I don't think it has a predator - nothing consumes it, as far as I am aware. And - while it is not autotrophic when looked at in absolute terms - in its environment (which is human society) it is pretty much so. Therefore it has no prey either. It just has its human symbionts.

The project

The organisational aspects of the project, or lack of, has always been controversial, with many wanting more control and direction, and many preferring the slightly chaotic nature. It was recently described as an Adhocracy, which I feel is a wonderfully accurate description. Do you feel that the decentralised nature of the project has contributed to its success?

I have no idea. But I certainly don't think that we would see the wonderful robust speciation that has occurred, with many different RepRap designs out there, if the project had been more controlled. Rather the reverse.

Considering this decentralised nature of the project, do you feel that having such terms as "Core Developers" goes against it? Would the project be better off now disbanding this team? or replacing it with another form of representation?



I really have no strong opinions about this. Indeed, on careful introspection, I find that I have no opinions at all.

Do you feel that the future of the project is now safely in the hands of the wider maker community and that RepRap will live on through it? Or do you feel there is a need for some form of foundation to underpin the core principles of the project, as originally envisioned?

Again, I have no idea.

The psychologists tell us that, if you would know the future of something, it is worthless to consult an expert on that thing. Financiers know nothing about where economies will go; agriculturalists know nothing about where global food production will go, and so on. In every study on this ever conducted, experts in a field are no better predictors of that field than chance. There is one exception: experts whose job it is to predict, and who get regular and immediate feedback on the accuracy of every prediction that they make - people like weather forecasters. They do better than chance.

I am, I think I can say, something of an expert on RepRap in particular and 3D printing in general. So I have no idea what the best way to ensure the success of the project in the future is. My perception is that the way it is organised, and by whom, is vanishingly unimportant compared to somebody's coming up with a really good new design and releasing it, but - as I say - I know that I don't know.

For someone new to the project, and starting to build a RepRap from scratch, what aspect do you think is the most challenging? Have you any tips or advice on how to overcome it?

It entirely depends on their background. I am horribly old, and people of my age are generally quite comfortable with mechanical technicalities - they know how to get two parts square, when to use shims, what sequence to tighten screws in and so on, without having to think about it. But they tend to flounder slightly when it comes to modern electronics with lots of static-sensitive MOSFETS in, and to be quite backward when it comes to compiling software. As you go down the age range the skills reverse, with young people having no trouble with the computing aspects of things, but being quite unsure about the most trivial of mechanical techniques, like how to tighten a nut that is facing away from them.

My best tip for building a RepRap is to form a small team of friends with complimentary skills to cover all those bases, rather than doing it on one's own.

The future

Even ignoring the current hype surrounding 3D printing, there are many exciting possibilities being explored around the world: biological tissue replication, printable architecture, new materials and techniques, amongst many others. Many of these use RepRap as their underlying technology. Is there any particular area which really interests you personally, or you feel has the potential to become exceptionally interesting in the near future?

Yes - I think that the most exciting application of RepRap will be in manufacturing equipment for personal biotechnology.

For example, I suspect (see the caveat about expert predictions above) that we are reasonably close to being able to make human sperm and eggs in vitro from stem cells. Imagine the disruptive potential of an open-source RepRap-made machine into which a couple could put two cheek swabs, one from each of them. The machine would do a complete genetic analysis and offer them a conscious choice over which allele from one to combine

with which from the other all the way down the 23 chromosome pairs, with probabilities and predictions on the alternative phenotypical results.

After they've designed their baby, the couple would then be presented with an implantable fertilised zygote.

Vitally, the open-source RepRap-made aspect of such a machine would put the choices entirely in the hands of the couple concerned and - more importantly - prevent everyone else (relatives, religions, or governments) from interfering.

Just like all the other seven billion of us, I have no idea if this would be a good or a bad idea, though like all seven billion I do have a worthless opinion on it. But it would certainly be interesting.

As versatile and fun as they are, 3D printers are only one tool amidst many. Is there a place for other tools in the RepRap family? Are there other devices you would like to see with a RepRap name badge on it?

A self replicating laser cutter would be possible and useful - several people are working on that. And MIT have a replicating CNC mill. The definition of the word RepRap certainly takes in such developments, though people must name them as they will.

Throughout the project you have explored and experimented with many techniques and tools (Piezo print-head, granule extruder, mould castings). With your day job now being at RepRapPro will you continue such experimenting, or is your time consumed with sales, marketing and support?

About half and half. I'm pleased to say that RepRapPro Ltd makes enough money for me and my colleagues not to have to spend all our time counting springs into plastic bags. I'm also pleased to say that we don't make so much money that some of us never need to count springs into plastic bags.

After all the time you have invested into RepRap, does it still excite you today?

Yes. It has been - and continues to be - more fun that I could possibly have imagined at the start.

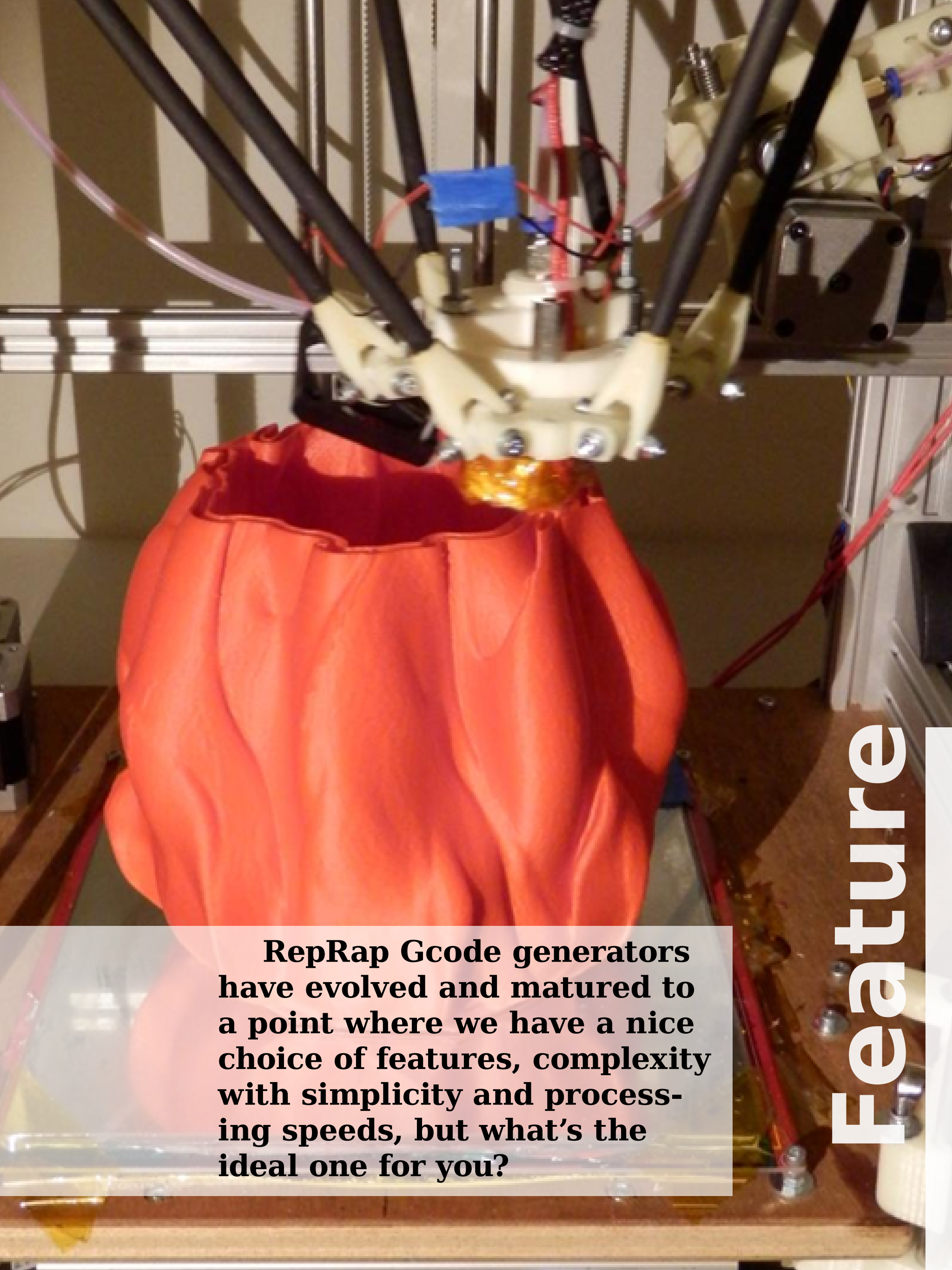
Ignoring your self deprecating "I know as much as the next man" position for a moment, what technological leap forward do you see, or hope to see, happening in the near future which will propel RepRap to the next level?

See above about ease of assembly and multiple materials.

And looking further into the future, which truly advanced, almost science fiction, technologies can you imagine RepRap consisting of? Are nano-factories feasible?

Clearly they are - we have an existence proof: prokaryotic and eukaryotic cells.

I foresee a scale convergence, with the biologists working up from the bottom and the engineers working down from the top. I hope that I live long enough for it to become impossible to distinguish between the creations of the two.



Feature

RepRap Gcode generators have evolved and matured to a point where we have a nice choice of features, complexity with simplicity and processing speeds, but what's the ideal one for you?

Richard Horne

Author

Alias:
RichRap
Country:
UK
Website:
http://richrap.blogspot

Firstly, let's just go over what a slicer is actually doing, why we need to use one and why the settings are so important for good model reproduction.

Its main job is to cut up a model into geometrically and dimensionally correct fine layers and plan paths for the extruded material used on each layer to allow a 3D printer's firmware to process this data and control the movement in a way to print out the finished part.

It's also important to apply some intelligence to slicing a model, so things like overhangs of material and sections of bridging have more outlines or in some cases solid layers going in a specific direction 'to bridge' material in a way so the print is both strong and appealing in final appearance.

It usually means that for most models a slicing program will do a good job of analysing your model and adding extra infill, outlines and layers where needed. This intelligence can cause some problems when you want to force a model to be printed in a specific way. For example a hollow vase, for this you may want to produce a single or multiple walled outlines without any infill. Some slicing programs allow you to turn off the infill 'intelligence' so you get a hollow object, usually with a solid base. Other slicers still attempt to add infill materials for 'support' or miss out solid layers even when you tell them not too, frustrating but not impossible to get around.

Another issue relating to quality and speed is the way your extruders are handled by the slicing program. Gcode can easily be generated that has inappropriate extruder settings with constant retractions for every tiny move; this can reduce part quality and drastically increase print time for no good reason. One critical experiment everyone with a 3D printer should work out is not only how much extrusion retraction is required but when to not retract or how much you can 'skip' without retraction before you get problems with blobs, strings or other defects.

With extruder settings it's usually not one rule for all models, but the speed that most 3D printers can now travel and speed of extruder retraction allows you to make short moves without retracting the extruder filament at all, giving a print speed improvement and no quality reduction.

Another valuable slicer setting for 3D printers is 'lift' or 'hop', this raises the print head up and then back down during a non printing travel move. If your machine can move up and down fast it's often a very good setting to have enabled for almost every type of print. Without lift or hop enabled it's possible for the nozzle to catch of parts of the object being printed either knocking small parts off or worse.

Gcode visualisation is an essential part of processing objects; sadly some slicers do not show a 3D version of the Gcode you have just created, so it can be a mystery if your model has any defects or extra material you were not expecting until the model prints or not as the case can be.

An excellent Gcode viewer is built into Repetier host, this allows you to see your object from the actual Gcode you are about to print with, it's almost essential to view gcode before printing just to make sure it's what you expect and it has been processed correctly.

Another 2D option is built into Pronterface, this allows you to see each layer from the top-down, great if you want to check material is being placed how you expect and the travel moves look ok.

Using pronterface for layer analysis is ideal for things like hollow objects, so you can make sure no extra internal material is being added or spurious moves are going to mess up your print, if you just look at a hollow object in Repetier you can see the 3D model but it's not as easy to see the inside as it is with pronterface.

An ideal addition for any slicing program would be to allow Gcode analysis, even real time highlighting and editing and moving of the paths, but we are a little way from that at the moment. If anyone wants to add this feature to any of the current slicers it would be very welcome.

Slicers all have their pro's and con's for doing different things so here is an overview of the key features to help you decide if it's worth evaluating a different slicing engine.

We are going to look at three Gcode generators in this overview, Cura, Slic3r and Kisslicer, many others exist or are in development.

To give consistency with analysis we are going to use Repetier Host to view the Gcode model and travel moves.

In an attempt to compare general capabilities, we are going to keep the same base settings for all slicing of the basic model tests:

First layer height	0.3mm
Layer height	0.25mm (each slice will be 0.25mm high)
Nozzle size (if setting is used in the slicer)	0.5mm (many printers come with 0.5mm or 0.4mm nozzles as standard)
Infill density	25% (0.25)
Single wall outline	(~0.56mm)
Filament diameter	2.86
Top and bottom solid layers	3(0.75mm)

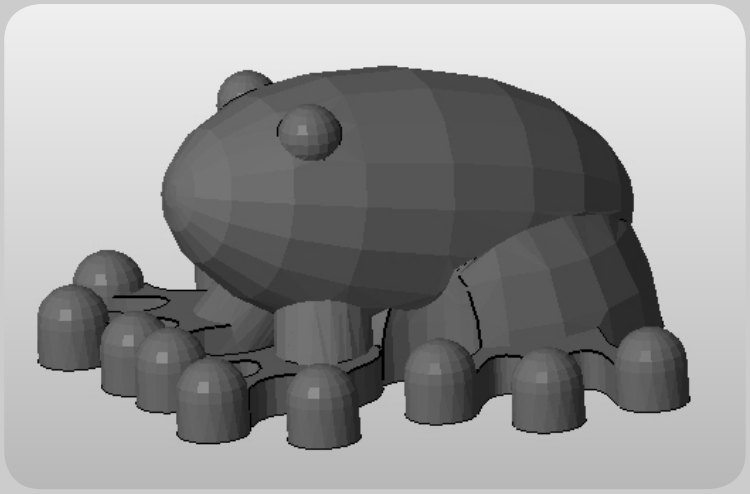
Other settings for the extruder, retraction, travel speed and minimum layer time will all be set to the same nominal speeds in all the slicing programs, being run on the same PC under Windows XP 32bit.

Test 1 - Object with infill test

Frog model by Owen Collins
http://www.thingiverse.com/owenscenic

Dimensions:

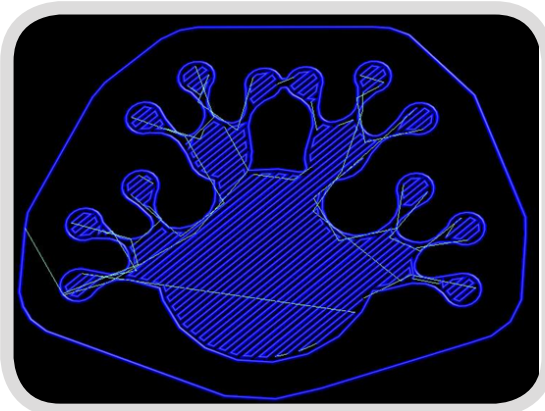
Lenght - 37.29mm
With - 30.83mm
Height - 15.28mm



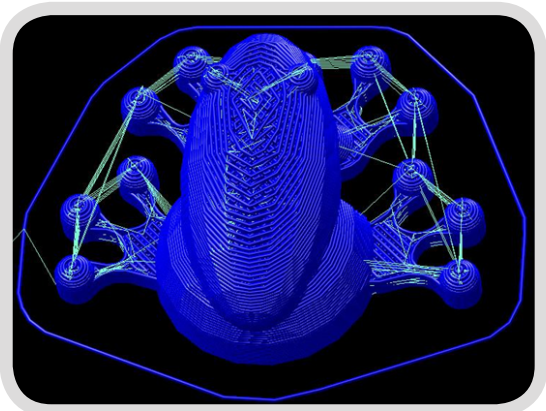
Cura test: Frog

First layer of frog, blue lines are solid infill and Cyan lines are machine moves - note how cura keeps as many of it's machine moves inside the object layer, this helps to stop deftects from the hot nozzle crossing perimeters as it prints.

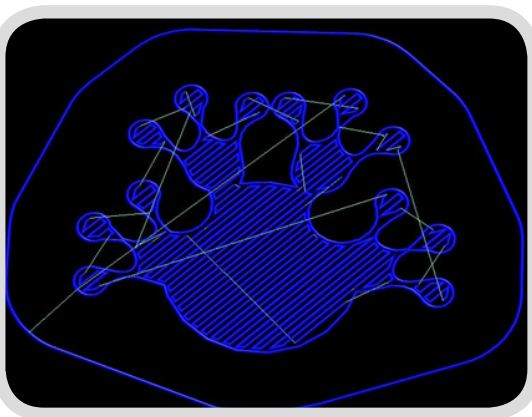
Again notice the perimeter moves are limited to improve surface quality.This frog is quite small so you can see the 0.25mm layer perimeters cause a loose finish on the final top layers.



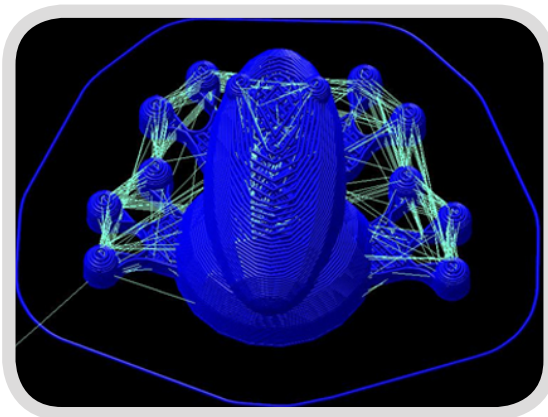
First layer preview



Final preview



First layer preview



Final preview

Slic3r test: Frog

Compare the Slic3r first layer, you can see travel moves break out of the perimeters and do not follow a totally logical path.

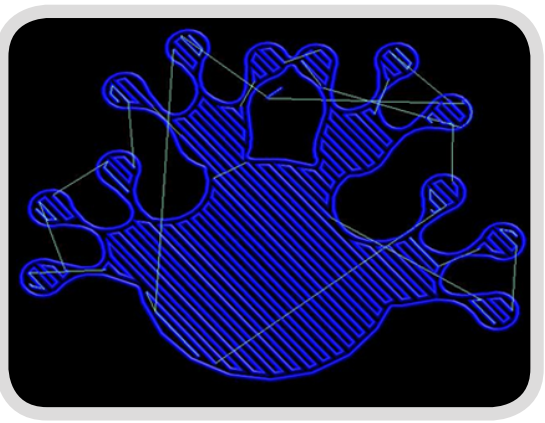
These travel moves continue for all other layers. On a machine without perfect extruder retraction, correct material temperature and fast travel moves, this can produce more artifacts on the outside of the print. But with a well calibrated machine, this should be able to be minimised.

Another good way to help minimise this problem is to use 'hop' or 'lift' this moves the extruder up (Z+) before a travel move and back down at the new location before the next extrusion.

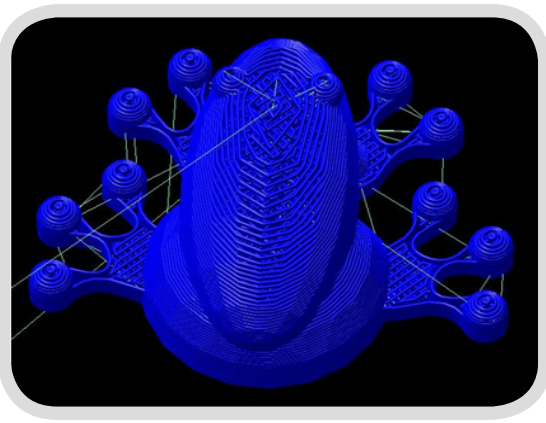
KISSlicer test: Frog

KISSlicer, has mainly internal travel moves, and it back tracks on the last move before a travel move to ‘wipe’ the nozzle before ‘jumping’ to the next location. This will usually have a positive effect on the print and result in less defects.

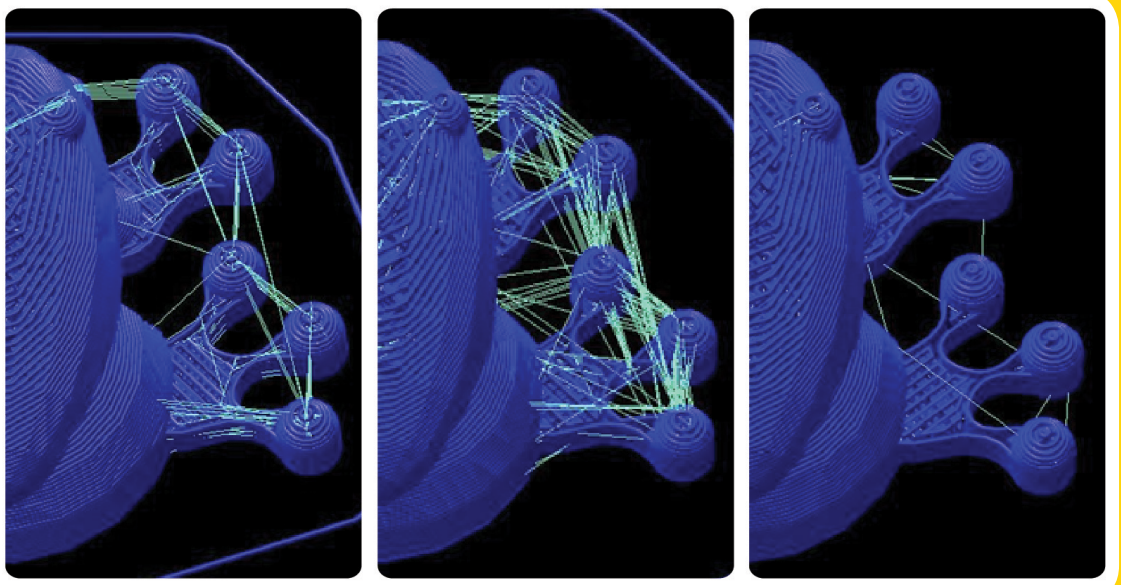
This really highlights how little perimeter breaks KISSlicer has, almost all of them are internal to the model, resulting in a high quality output with a well tuned machine.



First layer preview



Final preview



From left to right:

Cura, Slic3r, KISSlicer

Test 1 results

	Cura	Slic3r	KISSlicer
Slice time	**57 seconds**	**41 seconds**	**12 seconds**
Estimate print time	**12 minutes**	**-**	**11.8 minutes**
Used material	**3.41 grams**	**373.1 mm**	**373.1 mm**
Gcode size	**343 kb**	**325 kb**	**508 kb**

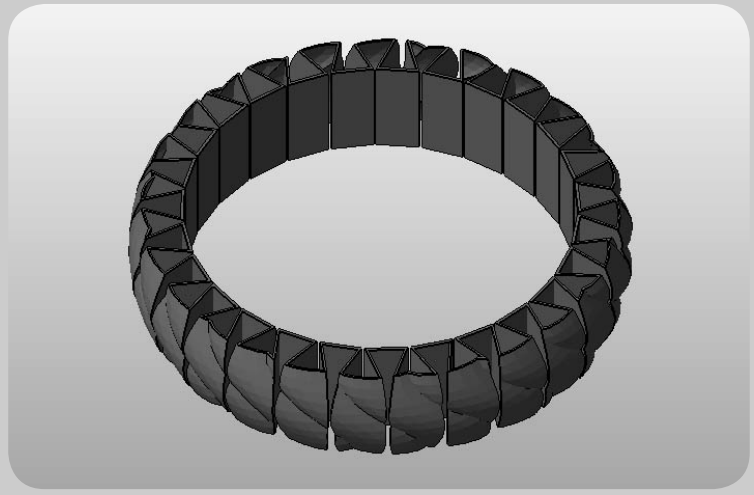
Note: Some Slicers measure filament by length, others by weight.

Test 1 - Model test with no infill, just a single outline

The Stretchy bracelet by Emmet Lalish
http://www.thingiverse.com/thing:13505

Dimensions:

Lenght - 71.77mm
With - 71.78mm
Height - 15mm



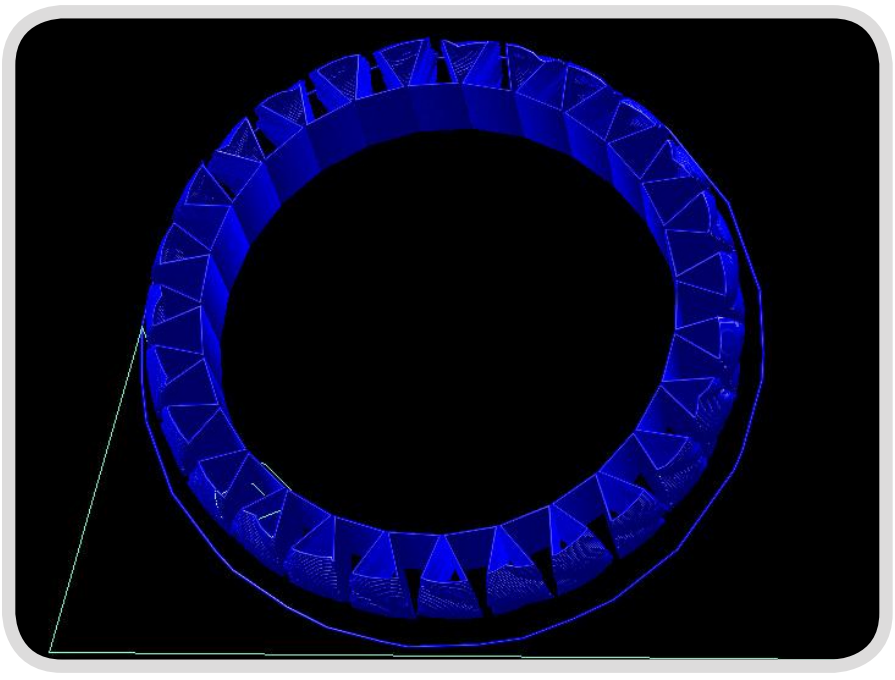
Slic3r

Slicer Version 0.9.7 does not like this model, it hangs while processing the triangulated mesh. It will slice just fine with Versions 0.8.3, you need to change to zero infill and no filled layers, and you should then get a single walled outline to print. You should not need to use Brim.

Cura Test

Same settings as other models, but with the 'joris' setting enabled – this constantly raises the Z height during a single walled print so you get no seam where layer change usually happens, a really great setting for vases and objects like the stretchlet bracelet.

Cura produces some very nice prints with the joris function enabled.

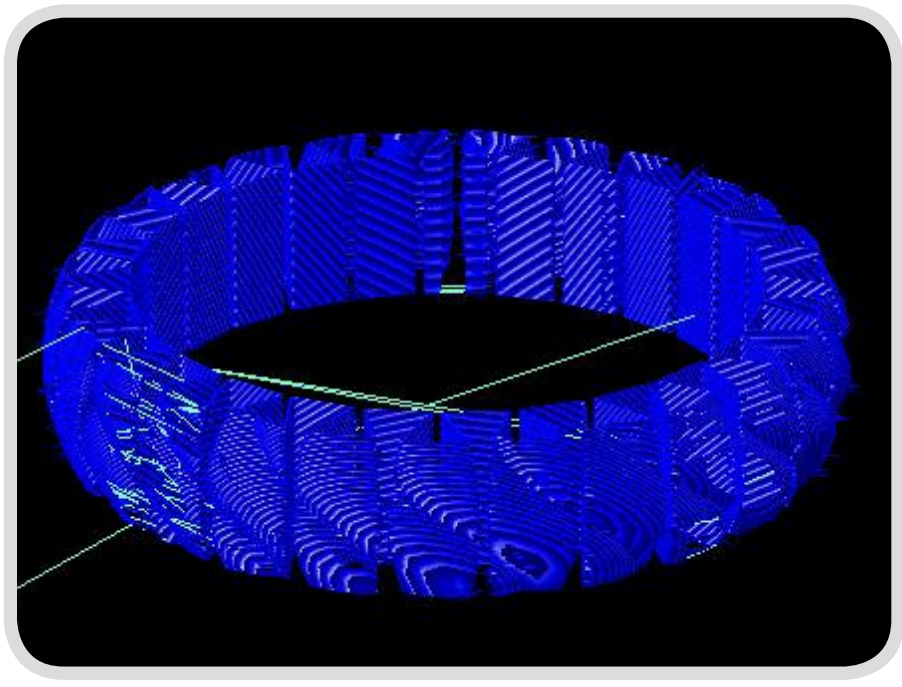


Cura final preview

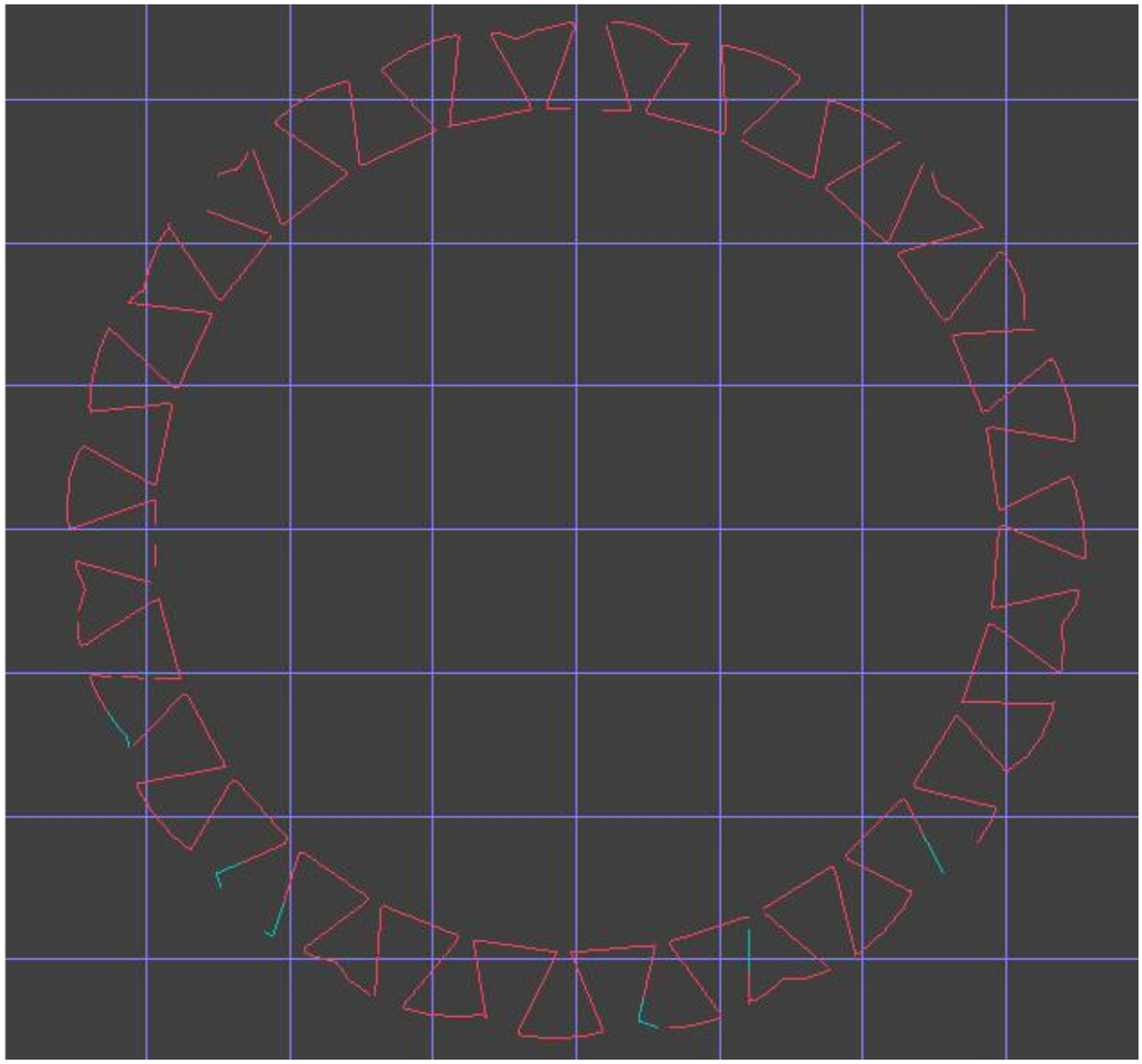
KISSlicer

In the Preferences – Advanced, under the style tab, you can wind down the infill to the right (0) and it will select 'Vase' – this is the ideal setting for these types of single walled objects.

KISSlicer - didn't manage a good job of this model and the Gcode produced takes a long time to load into Repieter Host, it just hangs the program for a few minutes. Eventually the model appears and shows some major issues, along with making Repieter run at 1/10th the normal speed. There are a lot of random travel moves and missing extrusion sections, this output Gcode is not good.



KISSlicer final preview



KISSlicer

Viewing the gcode path in KISSlicer shows some more missing sections, not something you would want to try and print.

Test 2 results

	Cura	Slic3r	KISSlicer
Slice time	3min 32 seconds	---	2min 15 seconds
Estimate print time	12 minutes	---	14min 16 seconds
Used material	6.57 grams	---	5.960 cm3
Gcode size	695kb	---	693kb

Note: Some Slicers measure filament by length, others by weight.

Test 1 - Really big and complex model

The Lava Vase by Dizingof scaled by 150% to 270mm tall
www.thingiverse.com

Dimensions:

Lenght - 71.77mm
With - 71.78mm
Height - 15mm

It going to be sliced with the same settings but with 10% infill.
This model is a complex organic shape with thick walls and minor overhangs.

Cura Test – 7mins 15 seconds to export the file to Gcode – 2.91MB file - very fast indeed!
!EDIT! - Cura did not produce a valid Gcode file, it did have a start and end, but not valid Gcode for the Lava Vase - it just crashed Rep

Slic3r test – Slic3r crashed after 15+ minutes during 'Generating perimeters'
Second try after reducing the number of threads from 3 to 2 = Crashed again at a similar point, it's probably running out of memory on this computer.

I had to give up with this under Windows Xp, but it did slice fine with the 64bit version of Slic3r under Windows7.

KISSlicer test – After 23 min 44 second during the Gcode export KISSlicer crashed with "Assertion failed! Ext_count >0"
2.989MB of Gcode was written out

Second try – After 38mins all the Gcode was written out 156MB – unfortunately I had moved the lower slider for support to be -0 thinking this would disable support for the model, but it did the exact opposite and added support everywhere!

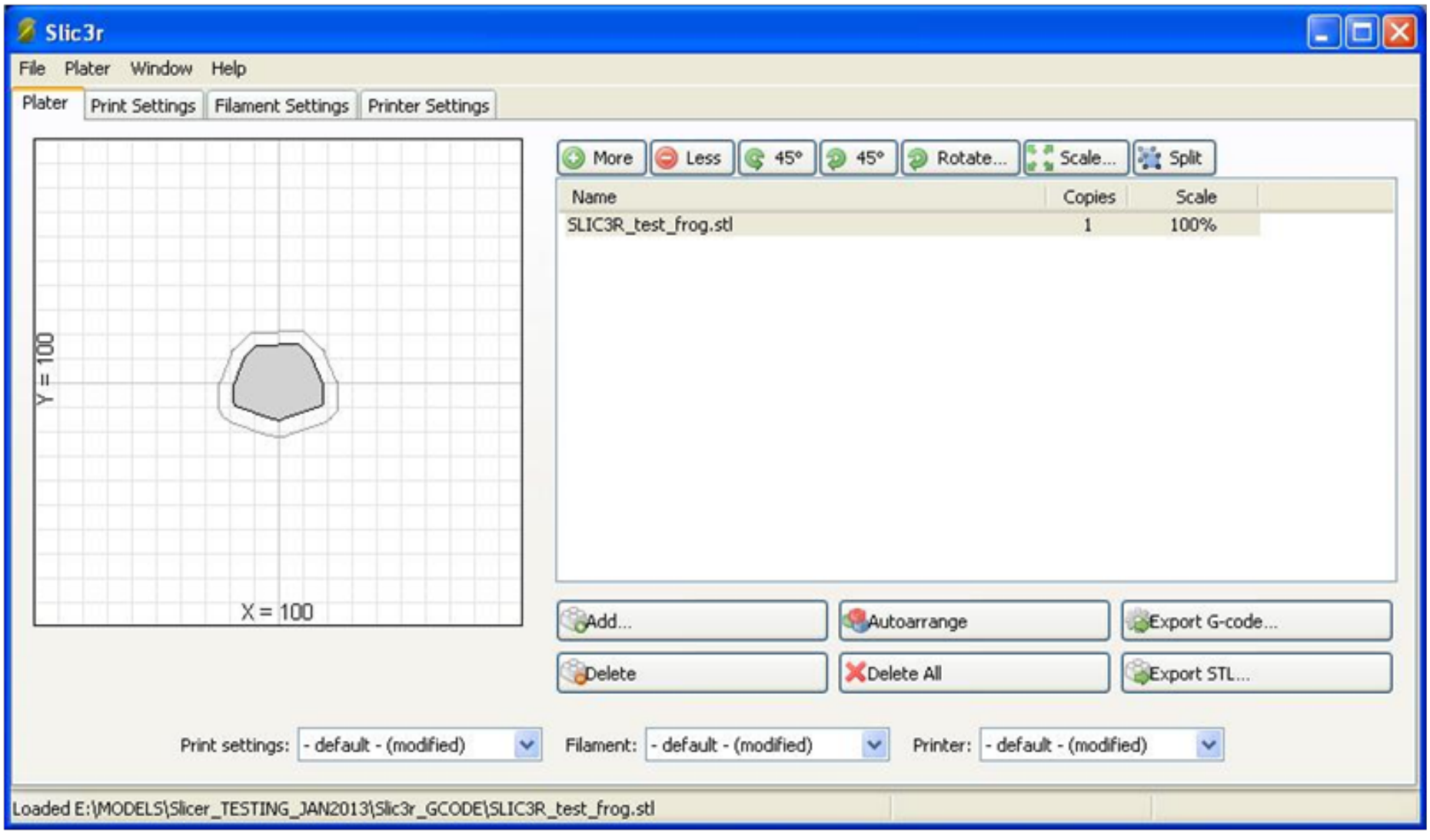
Third try – I changed the support slider to be 90 degrees, and this time the output Gcode was 102MB and generated in 26mins 34 seconds, no support structure used.
Output Gcode looks good.

Note on support – Next time, just use the top slider, this allows you to switch off support completely.

I printed the Lava vase from the Slic3r Gcode on a Rostock printer - just over 5 hours to print.

Closeup of Lava vase printed in Natural PLA.
Printed on Rostock,with0.25mm layer height.

Slic3r - Final review



Slic3r has had a swift development over the last few years, with many and frequent releases. It's currently at Release 0.9.7 so is officially still in beta, its open source and very community driven from feedback, feature requests and improvements and is one of the fastest Gcode generators available.

Slic3r is easy to use and has built in advice for most of the fields you need to enter data about your machine and material. It handles most models well and will attempt to repair and produce Gcode and give you a warning if any problems were detected. Usually running any bad models through Netfabb cloud fixes the issue and you can slice again.

Slic3r has support for multiple extruders and also the additive manufacturing format (AMF), this is the proposed future format for 3D printing to replace the standard .STL format we mainly use today. AMF has support for multi-part objects with mixed material properties, along with other advanced features it will allow full colour models to print as long as you have firmware and a machine to handle it.

Slic3r is fast at processing objects; it can have a few memory issues and occasional crashing with highly complex models. Most of these issues can usually be resolved by using Meshlab or similar to reduce the number of faces down.

Many models, even complex objects usually have somewhere under 50,000 faces, but it's quite possible for high quality model scans or very large organic objects to have 100,000 to over 500,000 faces, that's usually a struggle for most slicers to handle and even with a reduction in faces you will not usually notice any quality reduction in the actual printed model so do check the model resolution and reduce (decimate) if necessary.

One of the main features for Slic3r is that it is very easy to use, fast at processing so you can make changes to settings, view the output code and see exactly what that does to the model very quickly. It's an ideal starting point for learning how your machine works and what it can and can't do.

Areas for improvement

Support structure.

This is a tricky area as the slicer needs to first work out if a part of the model needs support to be printed, it then needs to print a fine structure under the area of the object to allow a gentle support of the actual model but allow removal of the support material from the model without leaving it stuck to the part needing more aggressive removal. Much more work needs to be done on support material usage, temperature settings to allow weaker bonds and easier removal, and also hopefully at some point soon dissoluble support materials that could be extruder and possibly reused again.

Slic3r has limited settings for support and it can puts support material in both the wrong place and or mixed in with the part being printed so it's very hard to remove, experiment with care.

Different versions give very different output Gcode.

This can actually be a good thing, for example using the same settings Version 0.9.1 seems to generate smaller Gcode and will usually do a better job of producing hollow objects (vases or cylinders) or fine objects with only outlines (like snowflakes)

While Version 0.9.7 generally does not seem to do hollow printing as well, usually adding infill or unwanted extrusions. It does produces a file with more Gcode, but tends to have finer lines and so usually more infill, that can make parts look better and be stronger but it will take a little longer to print.

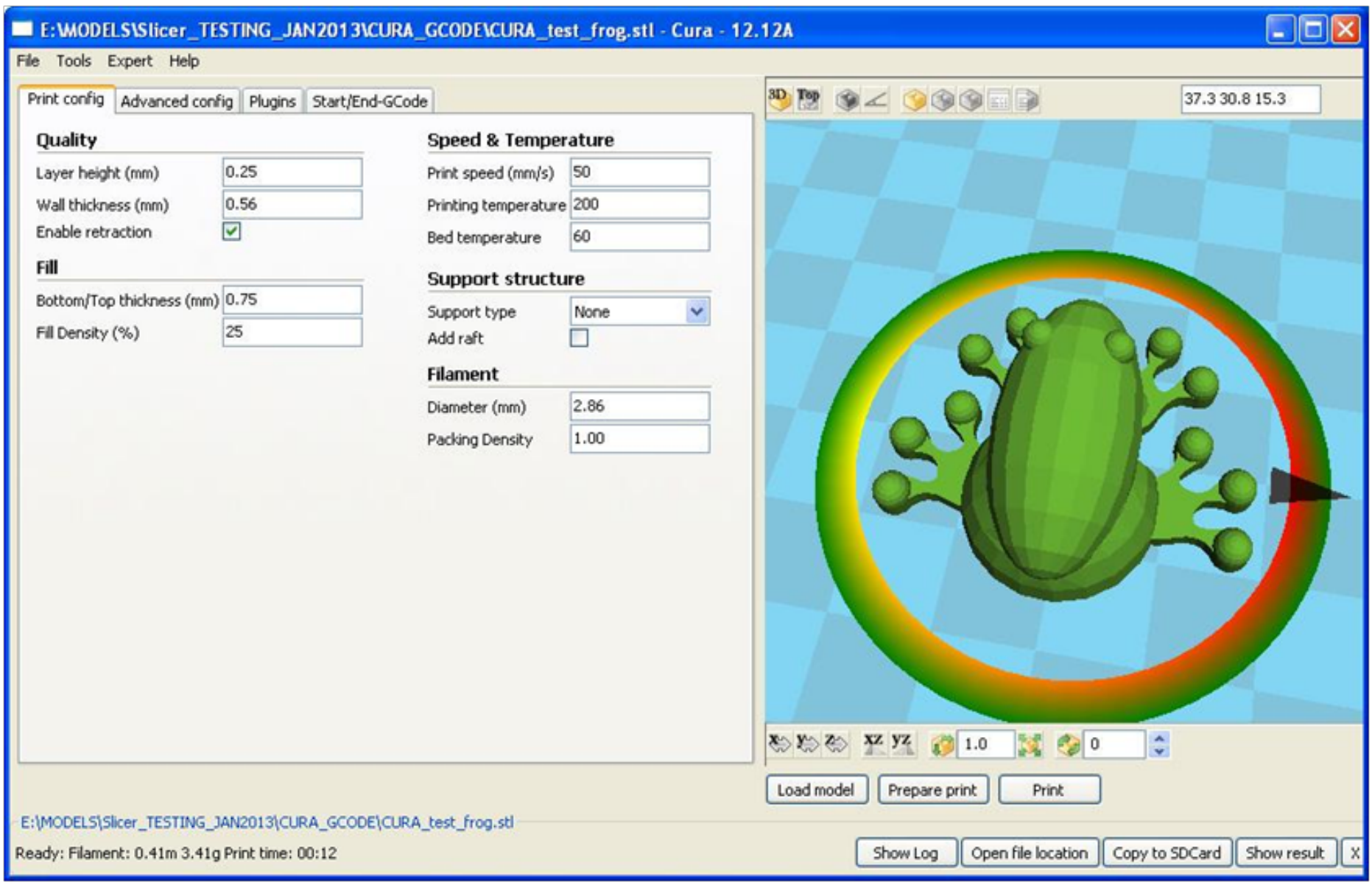
Version 0.7.2b is very popular with machines like Tantilius and Version 0.8.3 was also a good all round release but does not have some of the newer functions, settings and multiple extruder support of 0.9.X versions.

Slic3r is being well supported by individual community members and 3D printing companies like LulzBot, who have sponsored recent releases as well as funding a full time second programmer to further develop the software along side Alessandro.

Slic3r is a fantastic program to setup your machine and experiment with the capabilities, it slices so fast you can try out things quickly and see how settings change the output code and printed parts. You may still need to use different versions for specific things and its support material is tricky to use but you get multiple extruder support and both pre and post Gcode processing so it's a very flexible base for any experiments and most general 3D printing.

http://slic3r.org/

Cura- Final review



Cura is a community developed package that was originally based on the great but highly complex ‘Skeinforge’ and a faster implementation of Python ‘PyPy’

Cura has been refined for use by Ultimaker but it also fully supports other RepRap machines.

Skeinforge has a lot of options, did I say a lot, I mean hundreds, it’s amazing but very tricky to use and experiment with, and it’s quite slow compared to more recent slicers,

Cura however is fast and produces some of the best and cleanest Gcode possible, it’s hyper rugged and usually manages to slice any mode you throw at it, even models with errors.

Cura has some very nice features, and a simple tool-chain that anyone can use. It can print from Collada files – these can be directly exported by Google Sketchup, so you no longer need to do a translation to .STL format which can be very handy.

After loading your model into Cura you can use the model inspection view to see if your model has any issues before slicing. It has a nifty transparent view so you can check internal areas like nut-traps in parts or spot if any unwanted bits have accidentally been left inside your model after designing. The X-ray view shows problem areas in the model, so you can fix them in the design or at least be aware it may cause the slicer a few issues at those points.

The 3D view and Gcode visualisation is really good; it's very easy to see different layers and perimeters, infill and travel paths.

Normal mode allows you to get going straight away, most of the settings are easy to understand, and you only have three limited tabs to setup, after that you can produce Gcode and print.

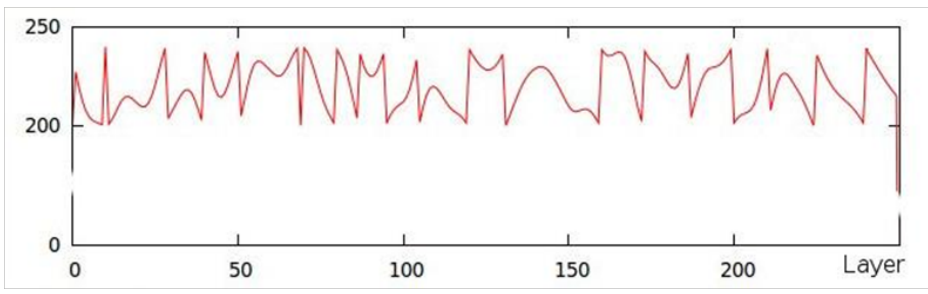
Expert config and experimental settings allow much more control over the print and generated gcode, so after a while you will want to play with these settings for certain types of objects. For example Cura has a checkbox called 'Joris' this wonderful option is perfectly designed for vases, cylinders and many artistic and sculptural pieces you may want to print, it prints the part as normal but the outer edge 'shell' is printed by a constant raising of the Z axis around the part, this means there is no visible seam where you would normally get a layer change and Z axis move up to the next point.
This makes the difference in a nice printed vase with a tiny seam to a perfect print without any signs of a seam or layer change at all.

Cura has a batch run mode where you can ask it to slice many 3D models from a list. Another great feature is project planner where you can place multiple objects on the bed to be printed. You can arrange, copy and sequence the parts to be printed. You need to tell Cura the dimensions of your printing head as it uses this to place each model so they can be printed in sequence without knocking off an already printed part.

The support structure generation is good in Cura, many of the options from Skeinforge are tidied up so all you can select now is the amount of material being used for support, the less material the easier it is to remove but it's more fragile to print as support for your object.

Areas for improvement

As with Skeinforge, plugins are available to do interesting things – Jeremie Francois produced a recent plugin for Cura using a new wood fibre filament that can simulate a wood grain by random temperature fluctuations every few layers, it's quite effective.

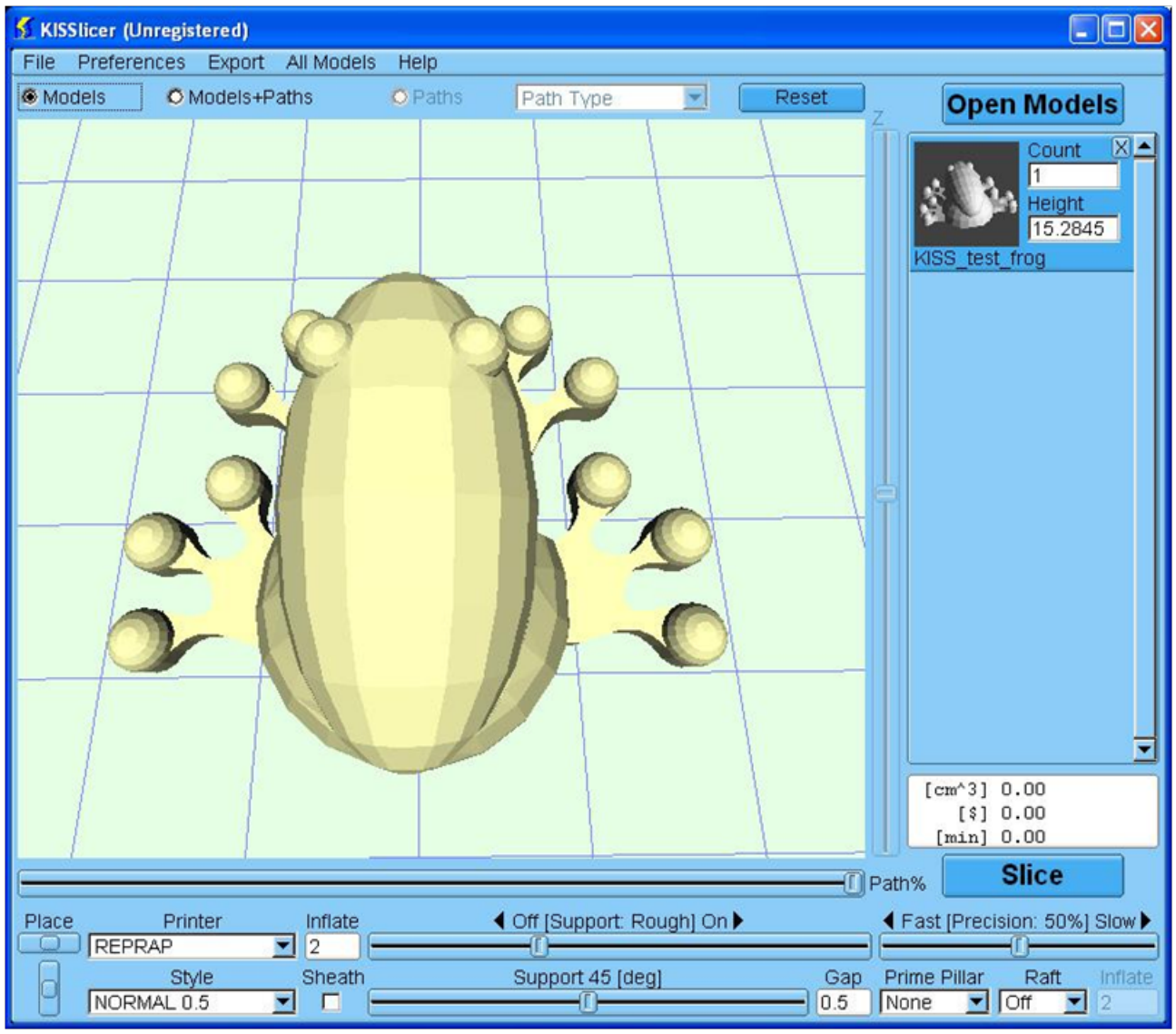


More info on http://betterprinter.blogspot.fr

Cura has no obvious support for multiple extruders yet so at the moment it's adequate for most machines, but it will need to add support for 2, 3 or more extruders in the near future.

http://daid.github.com/Cura/

KISSlicer - Final review



KISSlicer is quite easy to setup and once the printer details and a number of printing modes are configured it has one of the easiest user interfaces, especially when you just want to select a quality, load file and print, it's extremely fast as slicing models and although it usually produces bigger Gcode output than other slicers, that code is not just extra wasted moves, they are designed to give a strong part with hexagon infill and can include a reverse travel move called wipe that backs up along the last path before moving to a different region giving a very nice finish to the part being printed.

The user interface is clear and after changing the colour scheme to something less intense it's nice to look at too, similar to Cura it has a 3D view of both the model and the processed gcode layers and the gcode in 3D.

KISSlicer seems to have a few issues saving settings under Windows, the files need to have both a carriage return and line feed after each field. I found it simple to edit the setting files with a text editor rather than the KISSlicer settings menu.

Sample setting files for various machines are appearing up on the RepRap forums.

Areas for improvement

The Free version of KISSlicer can only load one model at a time, so if you wish to print multiple parts, you will need to combine them onto a plate and export a new .STL file so KISSlicer can load it as one file. – As a note Slic3r can easily combine multiple .STL files and export a plate of the parts as a single .STL meaning you can then load that into KISSlicer.

Another limitation of KISSlicer is that the free version can only handle one extruder, where the PRO license can do multiple extruders and load multiple parts together.

KISSlicer's license allows you to use it for both hobby and commercial use as long as you don't reverse engineer it as its not open source software.

http://kisslicer.com/index.html

Many other 3D printing slicers are also available, some open source and some commercial, many have fallen out of development or are now refining their way to be the next big thing. Keep an eye out on the RepRap forum and Wiki for more news and updates on Slicers and Gcode generators.

Next time we will be looking at different RepRap Firmwares and the electronics to run them on.

Taxonomy of Z axis artifacts

Rich Cameron

Author

Alias:
Whosawhatis
Country:
US
Website:
http://whosawhatsis.com

Rich has been a part of the RepRap project for three years, and is the designer of the RepRap Wallace. He is currently working at Deezmaker (http://deezmaker.com/).

Taxonomy of Z axis artifacts in extrusion-based 3d printing

Recently, printer designs have been multiplying like Stanford Bunnies. With the "Kickstarter generation" of 3d printers, I've noticed a lot of old mistakes being made by people who are new to the community and haven't done as much research as one might hope. Some of these mistakes have mostly been documented in comments and mailing list replies, so I thought it was time they were more formally documented. In this post, I want to talk about issues that cause distortion on vertical surfaces of prints.

Regular Z artifacts:

The most recognizable cause of this problem is Z-wobble. It is caused by the misalignment of layers in a repeating pattern with a period equal to the Z thread pitch (technically the lead, but this is the same as the pitch unless you are using a multi-start thread), and was a famous problem of the original Makebot, the CupcakeCNC. The CupcakeCNC's Z axis was constrained by four M8-threaded rods that turned to move the axis up and down. The problem is that while some of these rods are straighter than others, none are ever perfectly straight. The top and bottom of the rods were held in place with bearings, but when the rods aren't straight, the Z axis will be offset. Rods clamped in bearings will also be off-center due to the hole being larger than the thread's maximum diameter and some quirks of nut manufacturing (Cupcake users long ago realized that it was better to rest a locked pair of nuts on the bearing rather than clamping the bearing with one nut on each side). When the rods turn, the Z platform moves up or down along the Z axis, but it also moves in small circles around it, causing the layers to be misaligned and resulting in sinusoidal ridges along the vertical surfaces, with inverted ridges on the opposite side.

Photo credit: John Abella

Z-wobble was never a problem for Mendel-like designs until recently. The original Mendel did not have this problem because its Z axis was constrained by smooth rods, with threaded rods held by bearings on one end that only move it up and down. The Prusa Mendel created a similar system by coupling the threaded rods directly to motor shafts at the top (which in turn were held in place by the twin stepper motors' integral bearings), and the Prusa i3 (which, thankfully, moves those motors to the bottom) still has those

threaded rods constrained at only one end. I mention these models specifically because there have been several derivatives and so-called “upgrades” for these designs that place another bearing at the opposite ends of the threaded rods.

In these designs, when the rod is only constrained at one end, if the rod is not perfectly straight (which none are), the free end will move in small circles the way the CupcakeCNC’s platform did, but the Z axis will not be affected because the smooth rods constrain it to a linear path, and the shaft couplers allow a bit of flex when the Z nuts push the rod in one direction or another. Some less experienced users were confused or concerned by the wobble exhibited by the free ends of these rods and decided to put bearings on them to prevent this wobble. In fairness, on the Prusa Mendels with the motors at the top, they also served to take the weight of the X axis and extruder off of the shaft coupler, which was prone to being pulled apart in early versions, but the downside was the same. With both ends of the threaded rod now constrained as on the CupcakeCNC, the curvature of the rod again started pushing the Z platform in small circles around it. The smooth rods resisted this force, and tried to bend the threaded rods straight to counteract it, but the threaded rods were just as thick, so the result was that both rods deflected a bit, and Z wobble was allowed to occur.

As people have been pushing for higher-quality, error due to Z axis quantization and precision limits have started to appear, creating banding in an interference pattern that I’ll call Z ribbing. I’ve seen this issue over and over recently, and it is the primary reason for this post. Instead of entire layers being offset in one direction, this error appears as layers sticking out too far or not far enough in every direction. Like Z wobble, this error occurs with a regular period, but it is not the same as the Z thread pitch (though it may be very close). It occurs due to a rounding error in the number of steps per layer causing some layers to be shorter or taller than others, and threads being laid down with the same cross-sectional area will come out wider if they are compressed to a shorter height, and thinner for error in the other direction. This error is often mistaken for Z wobble, and is easily obscured by actual Z wobble as well as several other issues.

Photo credit: Jason Gullickson

Z ribbing is caused by choosing the wrong layer height for your Z screws. It is more apparent with thinner layers because the percent-error in the layer height is greater, and because the vertical surfaces of prints with thinner layers should otherwise be smoother. The error is much harder to avoid if you use an inch measurement thread pitch (more on this in later) or if you have some microstepping inaccuracy. Because microstepping inaccuracy can be hard to get rid of entirely, you should assume that you have some, and should always choose layer heights that are a multiple of your full-step length. If you are in half-step mode, your half-steps will all be accurate, but the 3/16 step position will not reliably be half a step away from the 11/16 step position, so you should only rely on accurate half-step intervals if you are actually in half-step mode.

To avoid Z ribbing, you should always choose a layer height that is a multiple of your full-step length. To calculate the full-step length for the screws you’re using, take the pitch of your screws (I recommend M6, with a pitch of 1mm) and divide by the number of

full-steps per rotation on your motors (usually 200). Microsteps are not reliably accurate enough, so ignore them for this calculation (though using microstepping will still make them smoother and quieter). For my recommended M6 screws, this comes out to 5 microns. It's 4 microns for the M5 screws used by the i3, and 6.25 microns for the M8 screws used by most other repraps. A layer height of 200 microns (.2mm), for example, will work with any of these because 200 = 6.25 * 32 = 5 * 40 = 4 * 50.

This also illustrates why you should never use screws with an inch measurement thread pitch. They are fine for the structural construction, but should not be used for the Z screws. The 5/16-18 threaded rods commonly used as a substitute for M8, for example, have a pitch of 1.41111111mm (the repeating decimal tells you you're screwed, so to speak) for a single-step length of 7.0555555 microns. Here are the only layer heights you can use without a rounding error causing artifacts (in microns): 63.5, 127, 190.5, 254, 317.5, 381, 444.5. Any multiples of the full-step length between these will have rounding errors every 9 layers due to the repeating decimals. Layer heights that are not multiples will exhibit Z ribbing with a different frequency depending on how often they sync up with the full-step length. The errors will be smaller if you have accurate microstepping, but there will still be error.

Similar banding has been seen with a period equal to the thread pitch. This banding is caused by a mechanical issue causing the Z screws to move up and down slightly and/or vary in degrees per step over the course of their rotation, but these are rare now that most printers have their Z screws coupled directly to motors rather than through a belt system that can introduce this type of error. This banding is more closely related to Z wobble, but it results in layers being offset vertically rather than horizontally.

Irregular Z artifacts:

There are several issues that can cause irregular banding. These cause layers to spread differently like Z ribbing, but they are more stochastic rather than occurring at regular intervals. These can be lumped into two categories: those that, like Z ribbing, cause variations in layer height, and those caused by differences in the volume of filament extruded into layers of equal height.

Irregular variation in layer height is caused by the platform or the extruder not maintaining the proper height while printing. On a cantilevered Z axis, this can be caused by binding or by wires pulling. In these cases, this resistance will cause the axis to not move enough in some layers, making those too short and wide, and then catch up in others that will be too tall and narrow.

A similar and probably more common error is resistance from the spool pulling against the extruder, which will tend to lift it. This is most prevalent on machines that move the extruder on the Z axis, especially if it is on a cantilevered platform like on the Thing-o-matic. In these cases, the filament will pull until there is enough tension to pull it loose, and the extruder will fall back to its normal position. How long this takes will depend on how much plastic is fed through the extruder rather than how far the Z axis moves, so the artifact will look different on prints with a lot of area per layer vs. ones with very little. You can prevent this problem by running the filament through a low-friction tube from the extruder to the spool (or to some structural fixed point between the extruder and the spool. The material doesn't have to be quite as low-friction as a bowden tube, but it should be looser around the filament because a tight tube increases friction and preventing backlash inside it is not a concern. Dirt-cheap and widely available (Home Depot stocks it) .17" ID HDPE tube will do the job nicely.

Low-frequency temperature swings can also cause some layers to spread out more than others because they are extruded too soft, and I've also seen temperature swings on a heated platform cause this uneven heights. Some platforms will expand or bow as they heat enough to affect the layer height and create inconsistent layers. This should be particularly true of platforms with a low enough resistance to heat quickly with the supplied voltage because the fast heating will cause more differential expansion and bowing as the heater turns on and off. In the case I saw this, the board was a prototype and the next version didn't have the issue. Short of replacing the heater, I'm not entirely sure what to do about this, but you can identify this issue if PLA on blue tape prints fine, but you get inconsistent layer heights when you heat it to print ABS. Tune your PIDs and make sure your temperatures are as steady as possible. The popular Marlin firmware can automatically tune its PID with the M303 code.

The other class of irregular Z artifact is caused by layers that are the same height, but do not have the same amount of plastic. These artifacts are caused entirely by the extruder and filament. For layers with a low volume (thin layers that also don't have much area), this can be caused by an eccentric extruder gear, but most gears will make several full rotations and spread the error out within a layer rather than causing an inconsistency between layers. A more likely cause is inconsistency in the filament.

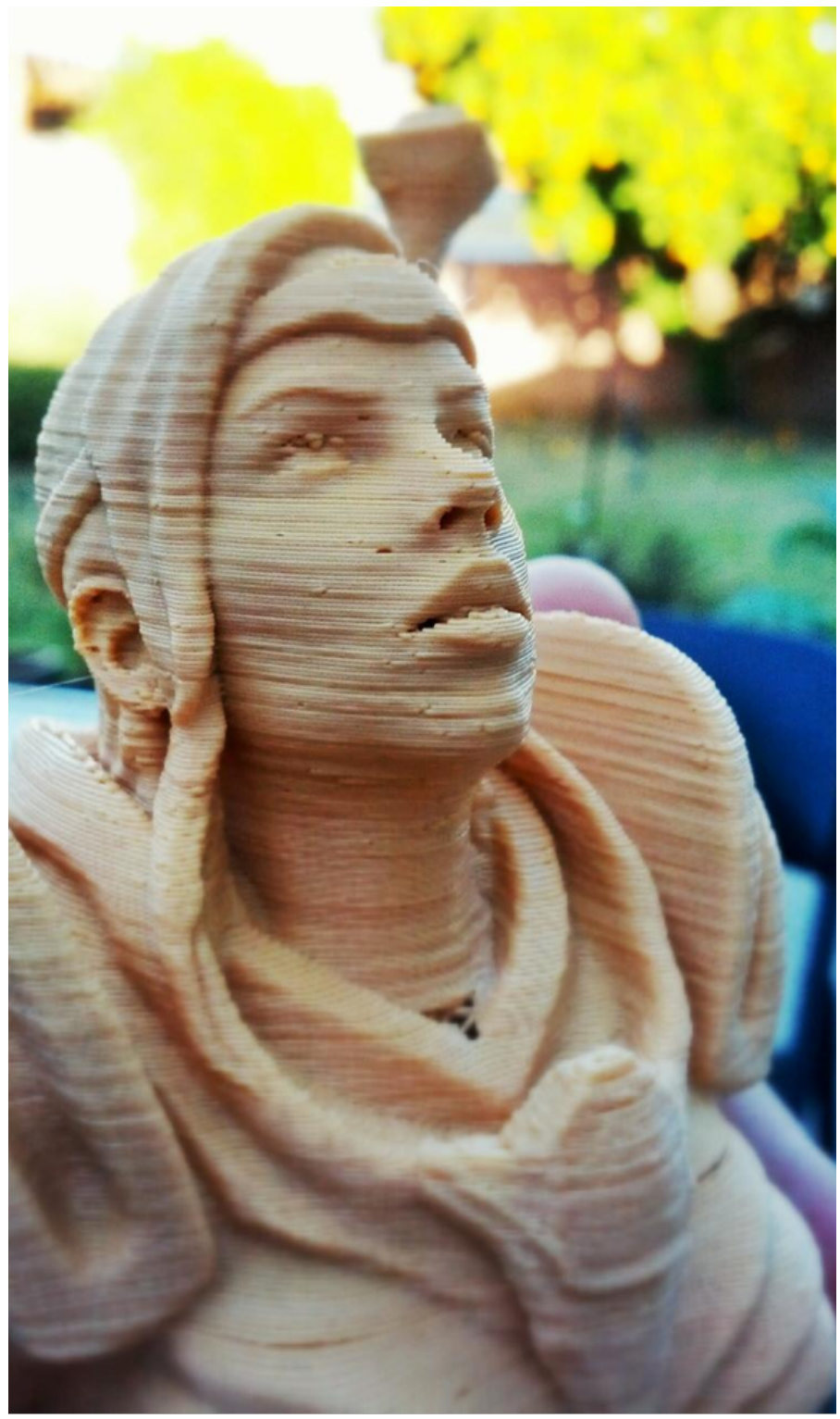


Photo credit: Ben Van Den Broeck

Filament must be uniformly round (flattened filament will have different minor and major diameters, which will affect how it presses into the drive gear and thus how fast it feeds) and must have as little variation in diameter as possible. Some low-quality filament also has bubbles down the center that change the cross-sectional area. These are not issues for plastic welding, which is what a lot of low-quality filament is actually manufactured for, and real 3d printing filament is manufactured to more exacting specifications, so you won't get the best quality prints from El Cheapo filament. This is not to say you need to break the bank, just check tolerance ratings and look for reviews or test samples when looking for a supplier. There are some good value suppliers popping up these days. This is also a major reason that none of the desktop filament extruder projects have yet been successful. There has been talk of developing sensors to detect changes in filament dimensions, but this is complicated by the delay in reaction between plastic feeding through the drive mechanism and coming out of the nozzle.

Repetier part 1: Communication

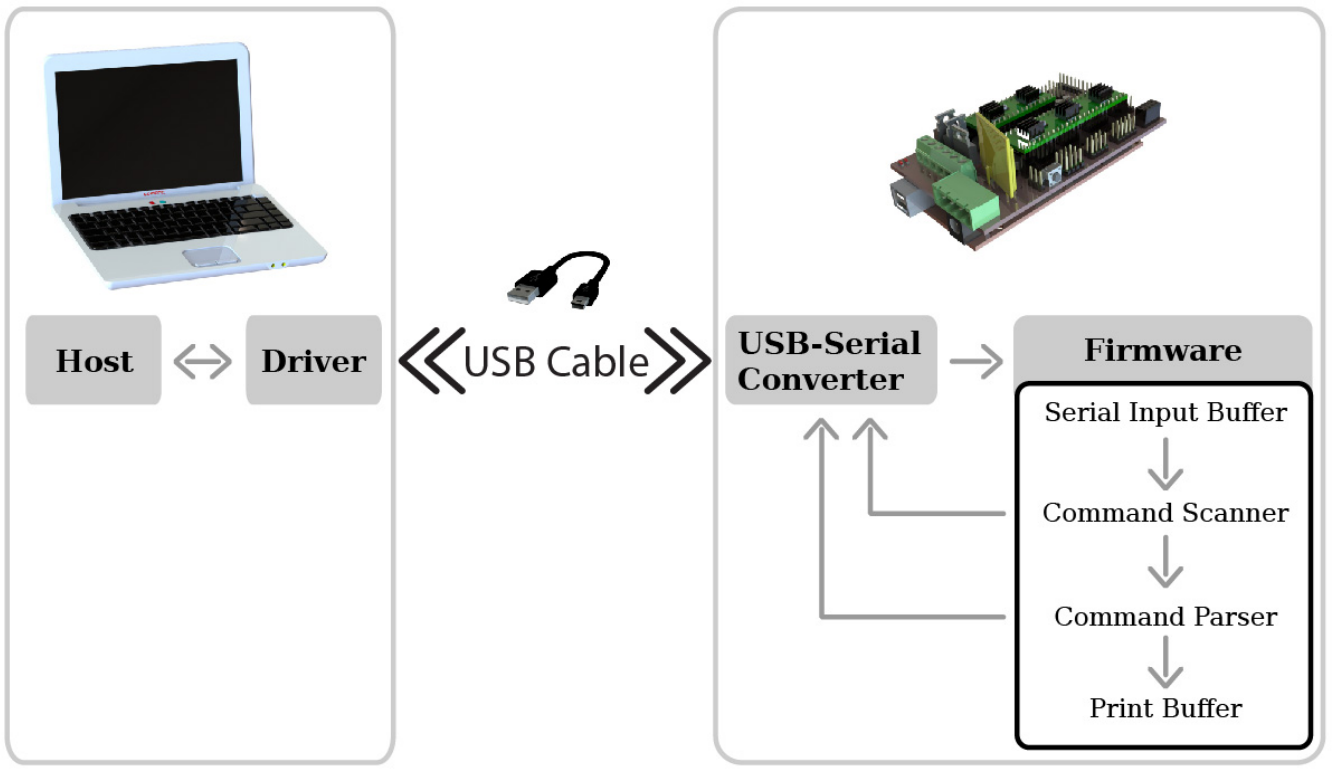
Dipl.-Ing. Roland Littwin

Author

Alias:
Repetier
Country:
Germany
Website:
www.repetier.com

What is the problem?

With some luck you have no problems at all. You connect your printer, set the right baud rate and there you go. If it is that simple then why write a complete article on it? Well, working is not the same as working as best as possible. In combination with Repetier-Host you can change the communication behavior and change the speed considerably. But at the beginning take a look at fig. 1, which shows where communication occurs, until the steppers are moved.



How it works?

As you see, the host uses the serial driver of the operating system. The first thing you should note is that this is not a real serial connection. In the past, computers had a serial connection, these were the D-Sub connections with 9 pins, if you remember them. When you sent a byte there, it appeared immediately at the port.

Nowadays, modern computers have lost this useful port. As a solution to continue using the simple serial connection for microcontroller, the boards now have a USB-serial converter. The bad news is, that it is not the same as a native serial connection, however, the program uses it the same way. USB is a packet device. It sends a data packet, whenever the driver thinks it should. To make it clear, think of the host wanting to send a new command, e.g. M105 to request current temperatures. The string is sent to the serial driver, and the data is 5 bytes, which is less than a USB data packet can hold. The driver waits a while before it sends the line to the printer hoping to get some more bytes. This is a latency delaying the communication. The exact time depends on the driver used. For example, the widely used FTDI driver has a

default value of 16ms. You can change this in the windows hardware settings. So after the set latency, the USB-serial converter gets a data packet with our command and converts it into signals for a serial port.

These are then sent with the selected baud rate to the Atmel microcontroller on the board. The microcontroller reads a complete byte and then issues an interrupt, so the firmware can store the byte. This has to be done fast enough to be finished before the converter sends the next byte.

To allow fast response times, the stepper interrupts are made interruptible in non critical sections. As a consequence, the communication can shift the stepper timing a few micro seconds. Not enough to cause trouble, so don't worry.

When the end of the line is detected and a command buffer is free, the command is parsed and the checksum, if sent, is checked. If the sum is correct an 'ok' is sent back to indicate that the host can send the next command. If the line number of the checksum is wrong, the firmware requests the line in question again from the host. In this way no commands are lost. On the way back we have the same latency problem. The converter may also wait a while before sending the packet. Adding up all times we have:

Time for a line :

time to driver +computer latency + usb send time + serial in time + serial out time + converter latency + usb receive time

Considering the speed of usb, we can ignore the packet send times and concentrate on latency and serial communication time.

Improving throughput.

Let's make some tests and see how we can improve the results.

I will start with 250000 baud. In the host I set ASCII protocol and enable ping-pong. That way we get exactly what I described and how communication was planned for the first controller. If you are using Repetier-Firmware you can send M111 S20 to test the communication only. All commands get acknowledged, but will never be executed. Sending 41496 lines took me 332 seconds, which gives 125 lines/second. So the time per line is 8ms for an Arduino Mega R3. The R3 has no FTDI chip, so I can't see the latency used here. Later I will use a different board with known latency for comparison.

Are 125 lines/second fast enough for you? If you are uploading to an SD card, you will never get enough speed. Oh, you only want to print and you don't think your printer can print that fast! Ok, let's test this theory. If we are printing with 125 mm/s you can send segments with a length of 1mm. If your segments get smaller your print buffer will run empty, which can cause blobs and stuttering. And I can assure you, there are always parts where you have smaller segments. Fine circles, infill between narrow walls or the perimeter of scanned objects tend to have parts with many short segments.

Convinced? You want more lines/second?

Ok, here comes trick #1, which works with all firmware. The firmware has an input buffer, where it stores the characters coming from the converter. It is large enough for a complete command and, in fact, it can even store 2 or more commands, depending on command size. All firmware has at least 63 bytes of input buffer. Depending on the Arduino IDE ver-

sion and firmware version it may even be 127 bytes. From version 0.80 onwards Repetier-Firmware has 127 bytes, older version 127 if compiled with Arduino 023 or older and 63 bytes for Arduino 1.0 and newer.

Baud Rate	Drive	Protocol	Buffer size	Lines/Second
250000	Mega R3	ASCII	Ping-Pong	125
250000	Mega R3	ASCII	127	358
250000	Mega R3	Binary	127	680
500000	Mega R3	Binary	127	680
500000	Mega R3	ASCII	127	466
250000	FTDI 16ms	Binary	127	355
250000	FTDI 4ms	Binary	127	669

Table 1: Max. Communication speeds for different settings

For the next test I will disable the ping-pong mode. Now the host will count how many bytes it has sent and which commands have been acknowledged to be processed. That way the host can send more commands in one USB data packet and increase the throughput. Sending the same data now took only 116 seconds, which equals 358 lines per second. That is 2.8 times faster than the last run!

Do you want more? Ok, you asked for it.

We have seen that packing more commands in a data packet increases throughput. So let's put some more data into it. Uh uh, buffer is already full to its limit from trick #1! It looks like we need some compression then. If you use Repetier-Firmware you can use trick #2 - send commands as binary data in form of the Repetier-Protocol. This protocol does not send ASCII data, instead it sends all values as binary data. That way the length of an average command is reduced by 50-60%. As a further benefit, the firmware has much less work parsing the commands, reducing processing time. Running the same job again now gets finished in 61 seconds, which gives 680 lines per second. That is 5.4 times faster than the original communication method. This is an average combination speed of 12875 bytes per second or 115875 baud.

We are only at 250000 baud, I'm sure the board can handle more!

Yes, you are right. I tested the same with 500000 baud. Unfortunately the time remained around 60 seconds. So there is no benefit in a higher baud rate with the binary protocol. For the ASCII protocol I got 89 seconds for the test, which corresponds to 466 lines per second instead of 358 for 250000 baud. For firmware without binary protocol a small increase can be achieved.

For the last test sequence I used a board with an FTDI chip. Here you can select the latency in the windows hardware manager. Parameters are 250000 baud, binary protocol, 127 byte buffer. First run with 16ms latency takes 117 seconds or 355 lines per second. For the second run I reduced latency to 4ms. This run took only 62 seconds or 669 lines per second. That is nearly the value we got for the Arduino Mega R3.

From this we can follow, that the Mega R3 serial driver has quite a short latency compared to the FTDI drivers default value. Reducing the FTDI latency to 4ms gives

comparable results and a more consistent command stream.

Communication problems:

While it should be very simple to create a working communication with your printer, I often get questions about communication not working. To help all those in finding the reason quickly I made a list of typical causes:

•Firmware and host need the same communication settings. All firmware tested have no parity and 1 stop bit. The only parameter you can set is the baud rate.

•If you have activated the EEPROM settings for your firmware, the baud rate is taken from the EEPROM. If you change it in your configuration and upload the firmware again, the new baud rate gets ignored.

•If you are not using the ping-pong mode, you need the right receive cache size. If you set it too large, you will get many communication errors. Typical values are 63 and 127 byte.

•With Linux, use only ANSI baud rates like 115200 or 57600. Often 250000 and other non ANSI rates do not work.

•If you sometimes have a sudden disconnect without error messages, try a shielded USB cable. You may even want to put a ferrite around the USB cable at the printer side.

Firmware internal communication:

The next step of communication is internal to the firmware. This part is not the most important for the user, although understanding it could help understanding why some things happen. For this reason I want to give a short overview of the internal data flow.

Step 1: In the internal input buffer.

Data sent from the host is stored here until the firmware is ready to process the input. The buffer has a size of 127 bytes, except if compiled with Arduino 1.0 for Repetier-Firmware < 0.80. This is the buffer meant in the host communication setting.

The firmware runs in an endless loop checking for new commands, parsing new commands, executing parsed commands, watching temperatures and updating the user interface, if one is connected. Beside this loop, 3 timer functions handle the extruder, PWM output and reads temperatures.

Step 2: The next step for the send data is to get parsed.

The firmware has a command buffer, where 2 commands get stored in a preparsed format. If the firmware detects new input and the command buffer is not full, it scans a new command from the input, puts the scanned values into the buffer and frees the input buffer to make place for the next data sent. After clearing the input buffer it sends an ‘ok’ in response, telling the host there is now room for the next command. If the parsing reveals a checksum error or a missing line number, it will instead ask the host to resend the last line.

If the command buffer is not empty then the next command is sent to the command parser. The parser executes the com-

mand. These commands can belong in one of three groups, depending on the time of execution:

Group 1- Move commands:

These commands are not executed immediately. Instead, the firmware does some pre-computations and adds the result into the move buffer. One of the timer functions will check this buffer and turn the stepper according to this data. That way there is no delay between connected moves, at least as long as the buffer doesn't run empty. Such a move buffer also allows for making nice movement optimizations. Knowing the subsequent moves, the firmware does not need to decelerate to the minimum speed at the end of a move.

This path planning is the best way to improve your print quality. The more moves are stored in the buffer, the better the planning can be.

There are only two problems you should consider. First, each entry requires memory and memory is something rare. 16 entries for a 4kb RAM system is no problem. 8kb RAM systems can go higher, if no other large memory users are enabled (e.g. delta printer need much memory for sub segments). Second, longer buffers can lead to longer optimization times. So you have to find a good balance between size and speed.

This setup is critical, if the printer gets many small moves in a row, like on highly detailed curves. Then even the maximum speed may get reduced, if it is not possible to accelerate and decelerate to full speed in the move length stored. In this occasion it becomes critical to refill the buffer as fast as possible as already described earlier.

Group 2 - Immediately executable commands:

Most commands belong into this group. They depend on nothing and get executed as soon as they reach the command parser. Most are executed fast enough, that the move buffer doesn't run empty if the print is interrupted by some other commands like temperature requests.

Group 3 - Move dependent commands:

Some commands require an empty move buffer. Good examples are extruder switches or G4 dwell. One parameter where you might want it is M104. If you define EXACT_TEMPERATURE_TIMING in your configuration, the firmware will empty the buffer before it changes the temperature.

The drawback is that the buffer is then empty and the extruder will pause a short time, allowing blobs to occur. For that reason the default is not to wait. Even if you turn the extruder off, the latency should be long enough to finish the print.

Now you know everything about the communication, you should know. With these informations, you should now be able to understand why things happen as they happen.

Check your communication flow and find the bottleneck causing your problem.

http://www.repetier.com/

What is a RepRap 3D printer?

Paulo Gonçalves

Author

Alias:
pdesigns
Country:
Portugal
Website:
www.reprapmagazine.com

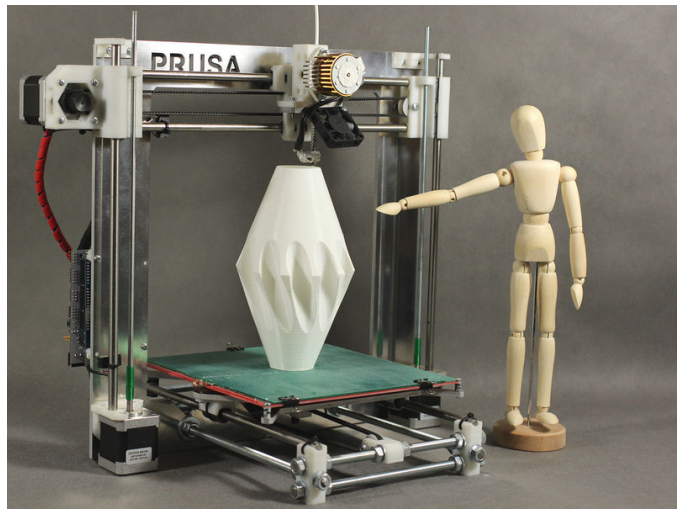
RepRap 3D printers have been around for a couple of years but if you have been missing the party then this section of the magazine is here to assist you getting into this world in an easy way.

For this first edition we are going to cover the basic anatomy of a 3D printer, the workflow of printing, and some advice on choosing the right model to fit your needs. But first lets cover the basics.

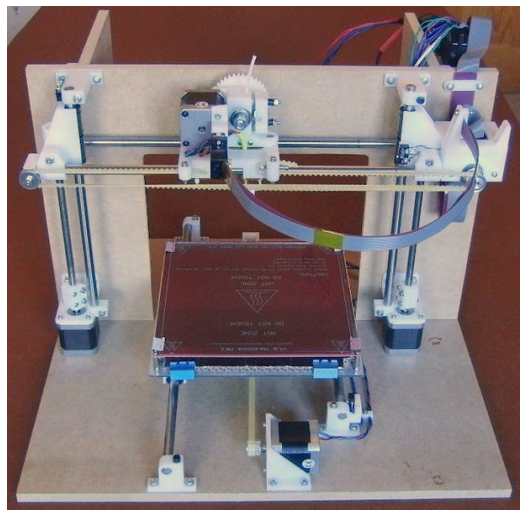
There are several popular models of printers currently available, and the fact that there is not only one popular model, but several, shows you the diversity of the project. Now, one of the aspects that helps this diversity is that for each component of the printer you have more than one type to choose from. Just to give one example, lets focus on the extruder: you can choose from a Bowden, a geared, or a direct drive extruder.

Let's assume we choose the geared extruder we then can choose from Wade's or Greg's extruder for example. This can look a little bit overwhelming and confusing for someone new to RepRap who wants to source the parts and build the printer, instead of just buying a kit, but it can easily be done without much trouble.

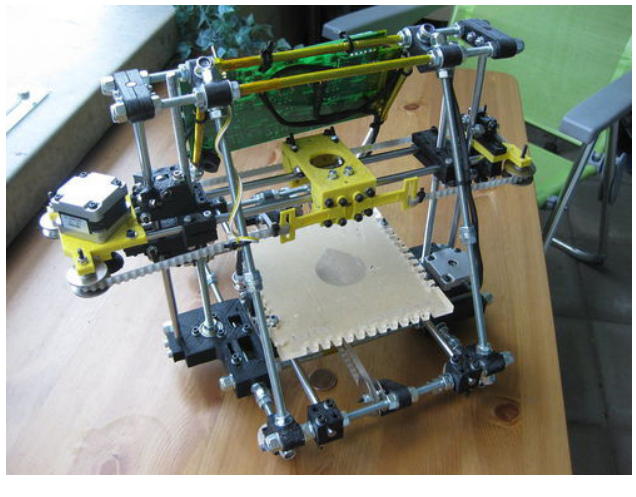
Let's start by understanding your printer anatomy.



Prusa I3
http://reprap.org/wiki/Prusa_i3



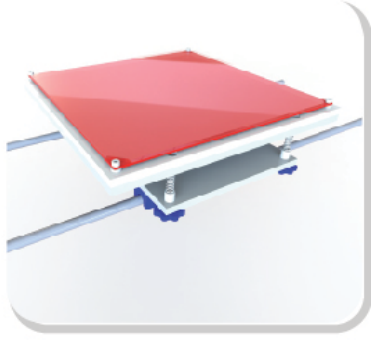
Mendel90
http://reprap.org/wiki/Mendel90



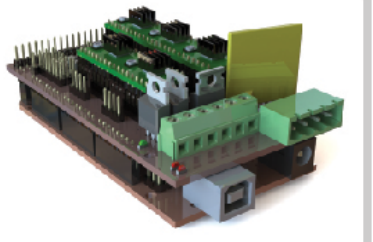
Huxley
http://reprap.org/wiki/Huxley

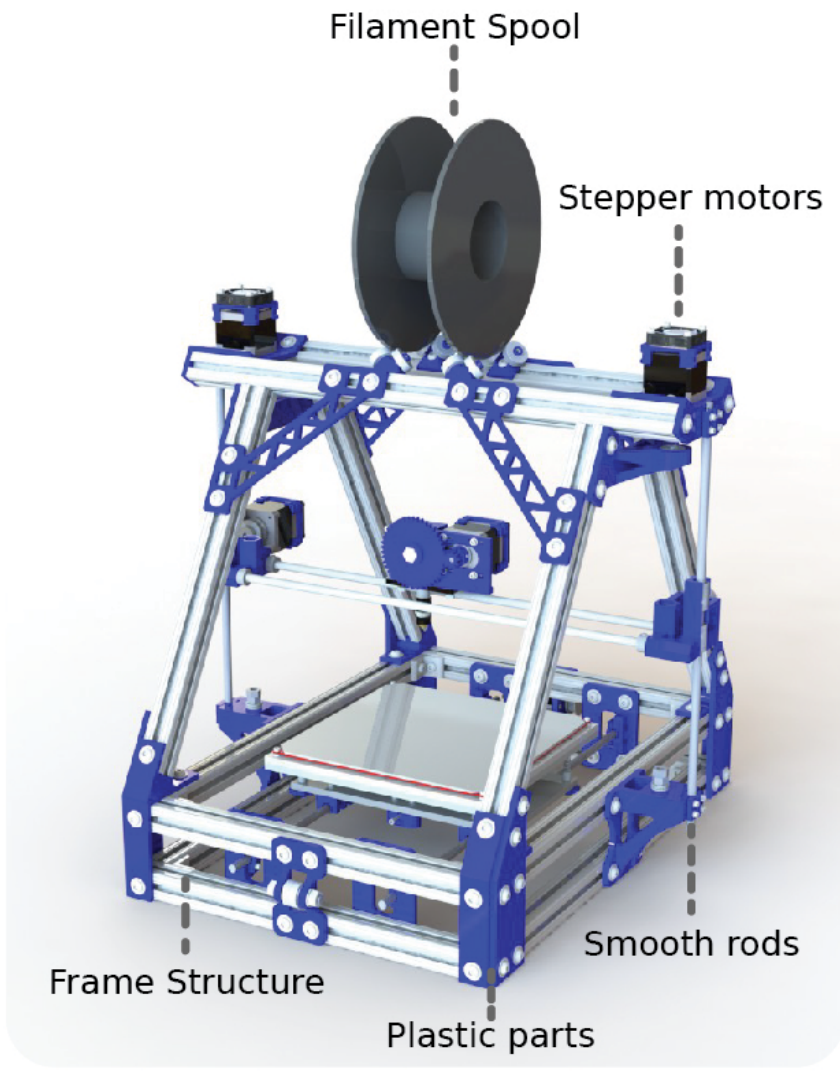
The filament is pushed by the stepper motor to a hot-end, which melts the plastic. It is then squeezed through the nozzle and deposited onto the print bed.

The print bed can be of two types, heated or not heated. A heated bed minimises curling and parts falling off the bed during printing.

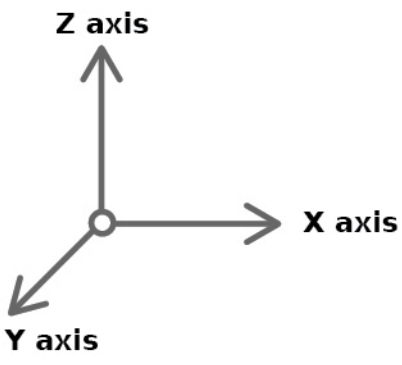


The electronics are the brains of your printer. They control everything, from the motors to the temperatures.





3D model credit: Matt Hodder



For a 3D printer to be able to lay down filament in order to build a 3D object it needs for movement in 3 axis (X, Y and Z). The nature of RepRap 3D Printers being open-source sets the ground for an exponential development of designs.

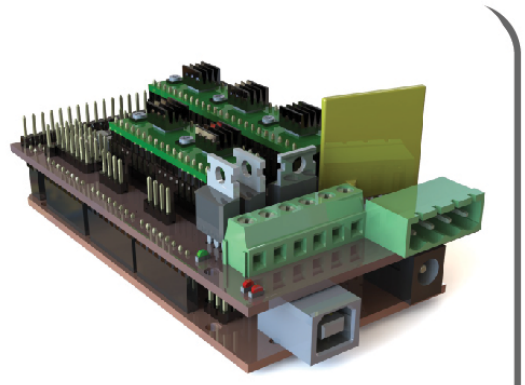
Currently the most popular models are built on two types on motion of the axis, Cartesian and polar.

One thing that makes RepRap interesting and different from some commercial printers is the level of customisation and the easiness of hacking your printer.

You will find that for every part of your printer there is more than one choice available.

Cartesian				Polar
X Head YZ Bed	XZ Head Y Bed	XY Head Z Bed	Z Head XY Bed	XYZ Head
Eventorbot	Prusa I3 Huxley Mendel	Cartesio Tantillus	Pocket printer IRapid	Rostock Kossel Rostock

This are just some of the printer models available. For a complete family tree of the RepRap project visit this page: http://reprap.org/wiki/RepRap_Family_Tree



MAIN CONNECTIONS

- Extruders (1 or more)
- Usb
- Heaters (2 or more)
- Endstops (3 or more)
- Temperature sensors (2 or more)

OPTIONAL

- Fan (up to 3)
- Sd card reader
- Lcd + encoder

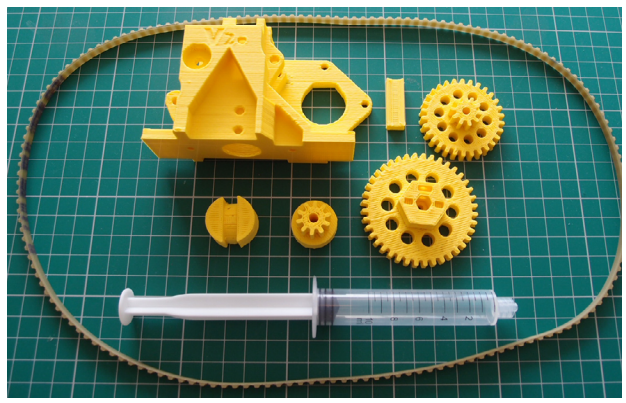
As you can see in this graphic the control board as what we can consider the main connections and some optional ones, but they also need to take care of the stepper motors that move the axis of the printer. To do that they have stepper drives, Pololu being one of the most popular, and depending of the printer model we might need a board with four or five stepper drivers.

Adding to this minimum number of stepper drivers required , and depending on the budget one has available, one might consider a board that alows more motors then the number required so that dual extrusion might be a possible route to explore.

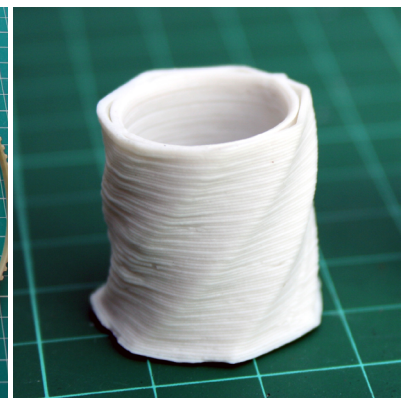
On the optional aspect Sd card readers allow for the use of the printer without it having to be connected to the computer during a print. The Lcd and encoder are a good choice as they allow you to still monitor the printer.

Several fans can be added to a printer, but the most common uses are for cooling the filament near the hotend (especially on a bowden setup), for cooling the controller board and to cool the filament during a print (more common with PLA filament).

Colour Blending



Paste extruder



Ceramic print

Photos credit: RichRap

The RepRap 3D printer can be equipped with a variety of extruders, and some have really interesting uses. There are extruders that have more than one stepper motor and they use more than one filament colour during one print providing us with a multicoloured object as you can see above.

There is also the dual extrusion, where you can combine more than one material at one print. For example it can be used to get two colours, or two types of plastic at the same print.

Another available type is the paste extruder. This type of extruders do not use plastic filament, but as the name points out they print pastes.

This pastes can be ceramic for example, but it can also be food, as chocolate. Yes, we can put a paste extruder on a RepRap and print chocolate cookies!

1 Get the 3D model

To get the a 3d model you can download one from the web, do a 3d scan or design and model it yourself.

2 Prepare the file to print

First check and fix your model using software like Nettfab Studio Basic, and export the file as .STL.

After that you need to adjust the Slicer to the characteristic of your printer and the level of speed and quality you want on your printer.

To finish this phase export the file into a gcode file to upload to the printer board.

3 Print!

Upload the file to the printer (via USB or a memory card), heat up your machine and hit print!

Check if the first layers are coming just fine, and then wait for the print to finish and cood down before removing it.

4 Troubleshooting

If the print is just fine, then you can skip this step.

If the print has problems them you need to check where the problem is:

-Is it a machine problem?

-Is it a slicer configuration problem?

-Is it a design related problem?

5 Iterate

When you are finished with the print don't just save the file to a folder in a dark corner of your computer.

Share it with the community and see if other people use it, or even make improvements.

The process of getting a RepRap can be as simple as buying a full kit or sourcing yourself the parts and building it yourself from scratch. There are advantages with both options, but one of the main advantages of sourcing yourself the parts is that it will probably get to spend less money.

But, first things first. Before you choose between these two options it's probably a good idea to choose which model is the best for you.

For this we created this little check list, trying to make your life that little bit easier, and give you the chance for some more conscious choices.

- Printer dimensions
- Printing envelope
- Cost of building
- Level of customization
- Easiness of sourcing the parts

As a conclusion we can say that RepRap is a very versatile project, that you can explore more and more as your knowledge grows. Probably you will run into some problems on your path, but you will also run into a community of users willing to help you solving them.

Printer model	Printing envelope
Rostock	200x200x400mm
Mendel Max 2.0	245x315x225mm
Tantillus	100x100x100mm
Huxley	140x140x110mm
Prusa I3	200x200x250mm
Mendel 90	200x200x200mm


Rostock


Mendel max 2


Tantillus


Huxley


Prusa I3


Mendel 90

Print volume %
Printer volume %

Get in touch

The nature of this project is to have a close relation with our readers and with RepRap users and developers. Within this spirit you are free and invited to get in touch with us by email or posting on the forums.

RepRap Magazine

Reprap Magazine

RepRap Forums

For any questions and general contacts:

general@reprapmagazine.com

For image donations, content and articles submission:

general@reprapmagazine.com

For developers who want to be available for future contacts:

editorial@reprapmagazine.com

Follow us, and get in touch, at
www.reprapmagazine.com