Open source & HW @ Prusa Research

Vojtěch Bubník

About me

- Software engineer / architect / project lead with more than 20 years of experience designing custom 3D CAD, simulation and 3D printing software.
- Last 7 years spent with full time open source development at Prusa Research.
- Now leading the Prusa Slicer team, worked on firmware.
- Lot of technical hobbies (HAM radio, electronics, flying model planes & gliders, popular science).

Why to love Open Source & HW

- We love to make things.
- Without Open Source & HW, Prusa's brothers would not have had fun with FDM printers.
- We want to give back to community and share the joy of tinkering and self learning.
- We believe in collaboration and exchange of ideas.

TV show "Na plovárně"
Josef Průša, 2012

FDM 3D printing basics

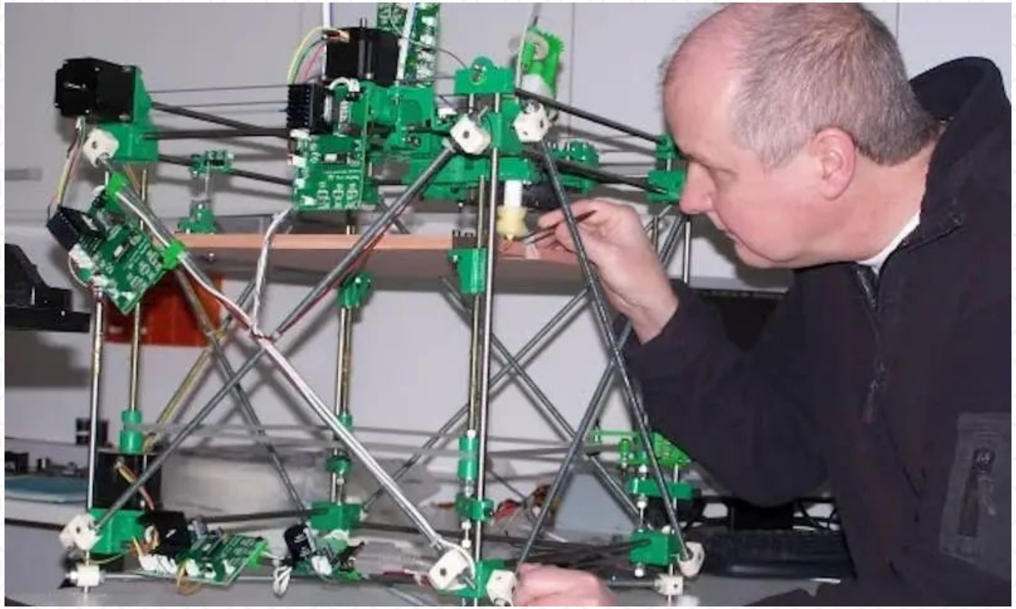
and why it took so long for FDM to become a commodity:

- Hardware: Frame, motion system, extrusion system, heating / cooling, electronics, control system.
- Firmware: Reading and interpreting commands, controlling the above.
- Slicer: Converting a 3D model geometry into commands for the firmware.
- Key patents by Stratasys expired in 2014

History of RepRap (self replicating machines)

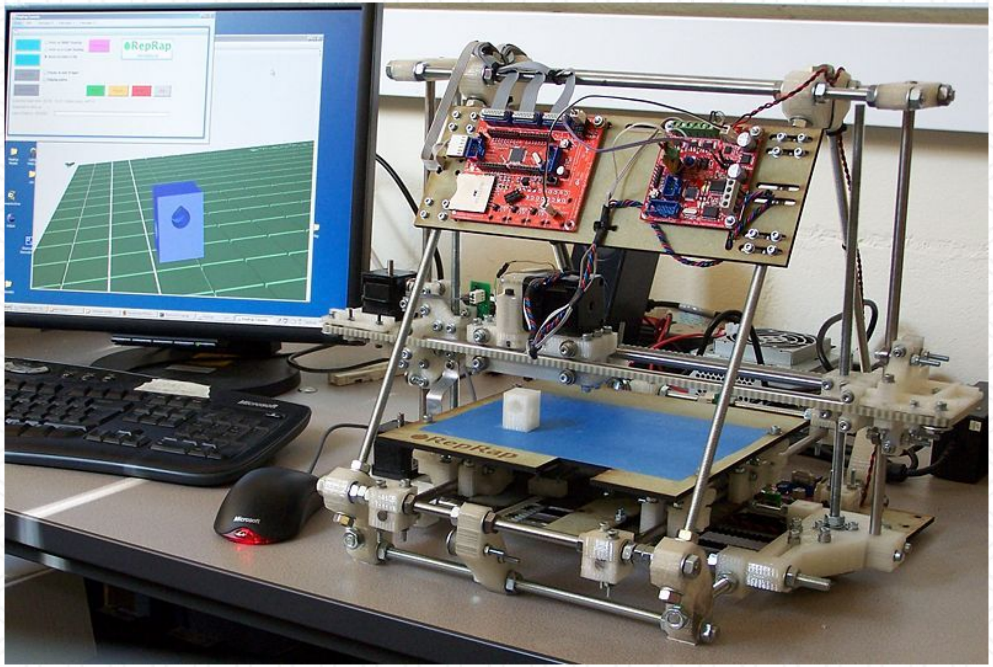
Founded by Adrian Bowyer in 2004 (lecturer at the University of Bath)

1st design shared: RepRap Darwin (spring 2007), printed on Stratasys

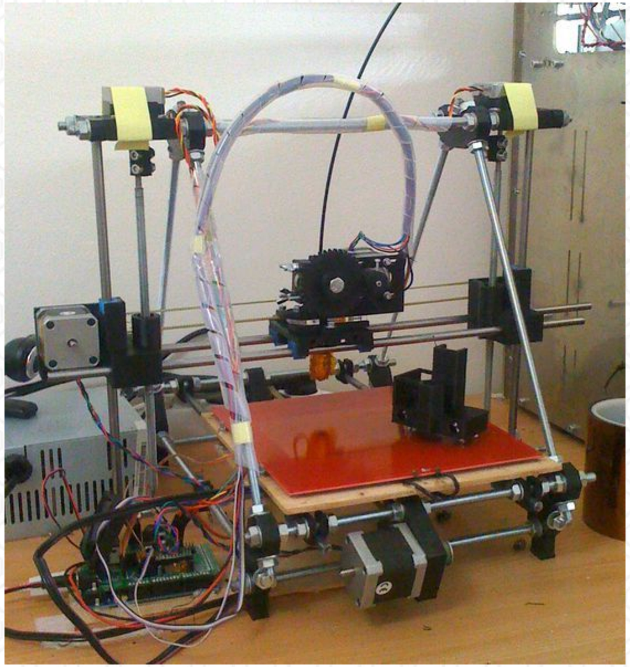


History of RepRap (self replicating machines)

RepRap Mendel
(summer 2009)

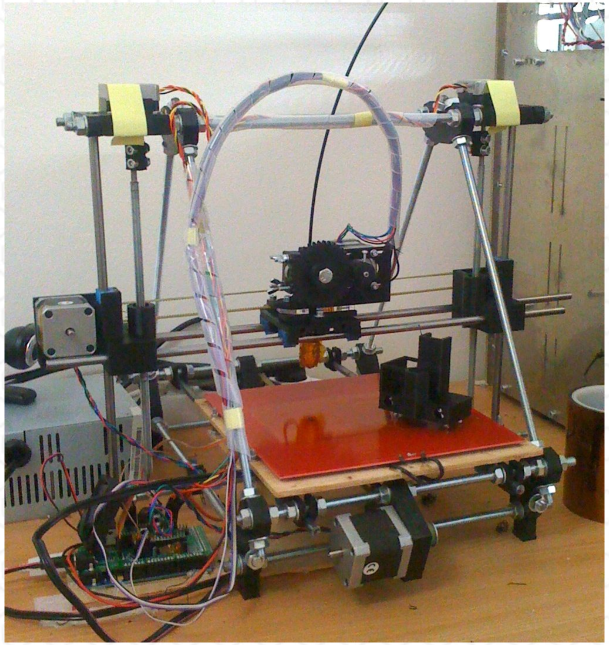


Prusa Mendel
(September 2010)

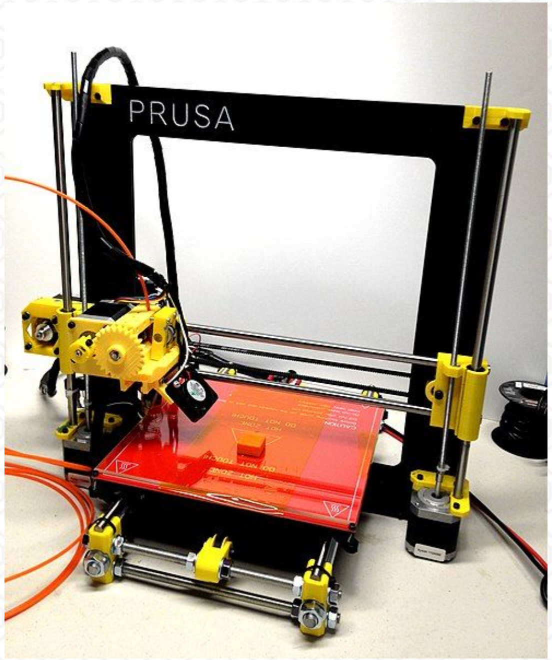


History of RepRap (self replicating machines)

Prusa Mendel iteration 2 (November 2011)



Prusa i3 (May 2012), Prusa Research founded (bros Prusas).



History of RepRap (self replicating machines)

Prusa i3 MK2 (May 2016)

- Automatic bed leveling (inductive sensor)
- Automatic skew compensation
- PEI sheet on print bed
- intuitive menu system

Fully open source / open hardware, on github

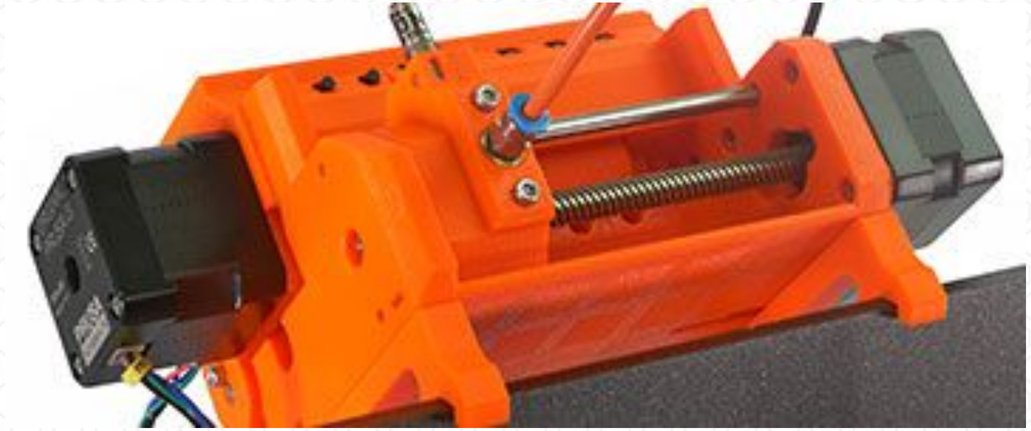
RepRap - open source & HW for the win

- Our FDM 3D printers are based on RepRap. Very most FDM 3D printers have some RepRap heritage.
- RepRap and early Prusa printers were designed solely in OpenSCAD (open source scripted CAD based on CGAL dual licensed geometric library).
- RepRap & derived printers run some open source firmware (Marlin, RepRapFirmware, Klipper), all of them sharing the path planner of grbl.
- KiCad is often used for electronic design (prefered at Prusa Research)

Hardware designs / ideas shared

Prusa Research shares:

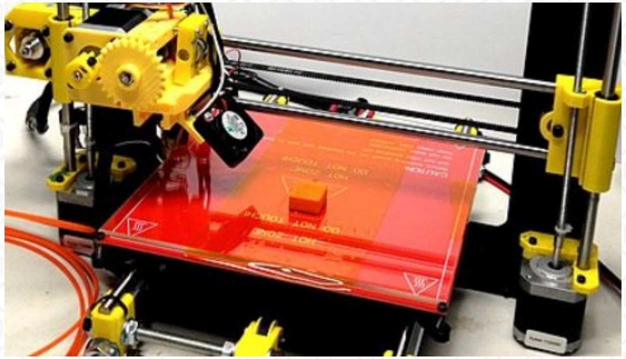
- Complete Mendel / i2 / i3 / i3 MK2, MK3, Mini printer designs
- Multi-Material / Multi Material 2 upgrades
- Electronic boards (Prusa Mini control board, filament sensor boards)

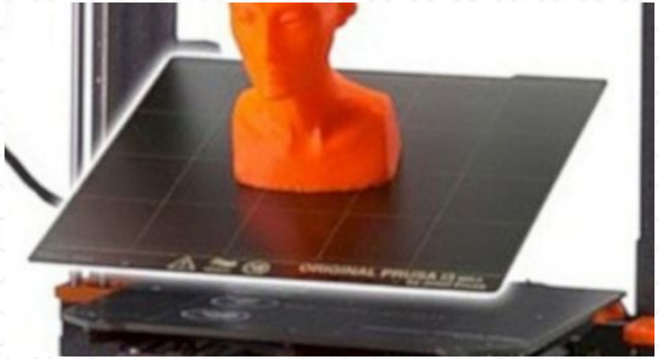


Hardware designs / ideas shared

Prusa Research designed, others cloned, now industry standard:

- Heated print bed as printed circuit board
- PEI (Polyetherimide) sheet on print bed for adhesion
- Removable textured spring steel print sheets held by high temp magnets, PEI powder coated





Hardware designs / ideas shared

However not everything could be open sourced easily:

- Some parts are custom (custom Delta power supply with power fail signal), no design documentation from OEM.
- Open Inventor is used nowadays for complex 3D modeling instead of OpenSCAD.

RepRap 3D printer firmware: Marlin

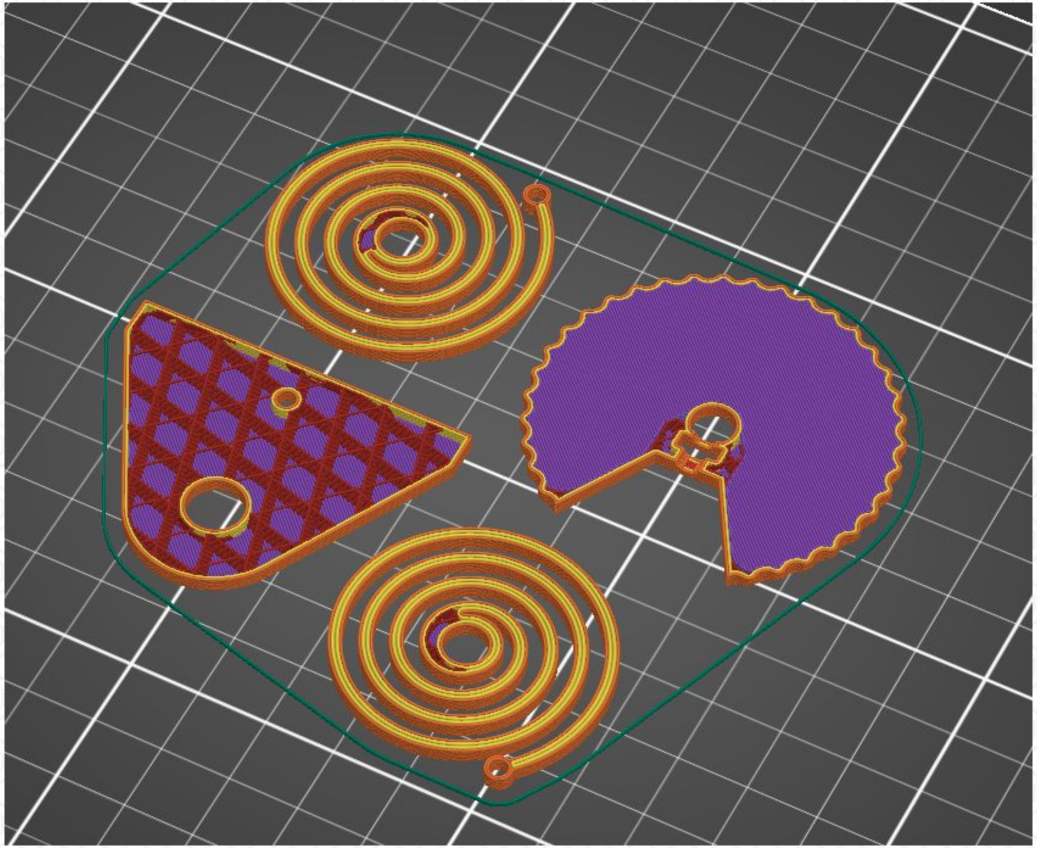
- Initial commits winter 2011, based on grlb (2009)
- Highly optimized, runs on 8 bit Arduino boards.

Prusa Research developed:

- Well designed user interface.
- Automatic mesh bed leveling.
- Automatic skew calibration and correction.
- Power panic (power outage recovery).
- Sensorless homing & crash detection (Trinamic drivers).
- Filament run-out laser optical sensor.

Constant flow of ideas / code between Prusa Research and Marlin code bases.

Slicing: 3D print preparation



Slicing: 3D print preparation

- Slicer loads a 3D model, slices it with parallel planes to polygons, optimizes paths and produces G-code.
- G-code language since 60s, standardized 1980 by NIST.
- Slicer programs XYZ axes, E axis (material flow), temperatures and cooling.
- Optimization problem (traveling salesman, NP-complete), discrete problem (fill region with lines).
- Conflicting requirements (machine resonances vs. steady plastic flow, cooling vs. layer bonding …)
- Sensitivity to geometry, numeric stability.

History of Open Source slicers

- Skeinforge (2009?): Python, slow, first?
- RepRapPro: Java, quickly abandoned.
- Slic3r by Alessandro Ranellucci: Summer 2011, originally Perl, still alive.
- Cura: December 2011, based on Skeinforge, Python / C++.
- Cura 2: January 2013, adopted by Ultimaker. Most widely used 3D printing software.
- PrusaSlicer: Spring 2016. Based on Slic3r, 2nd most widely used 3D printing software.
- Kiri:Moto: 2013, web browser (JavaScript, WASM) based

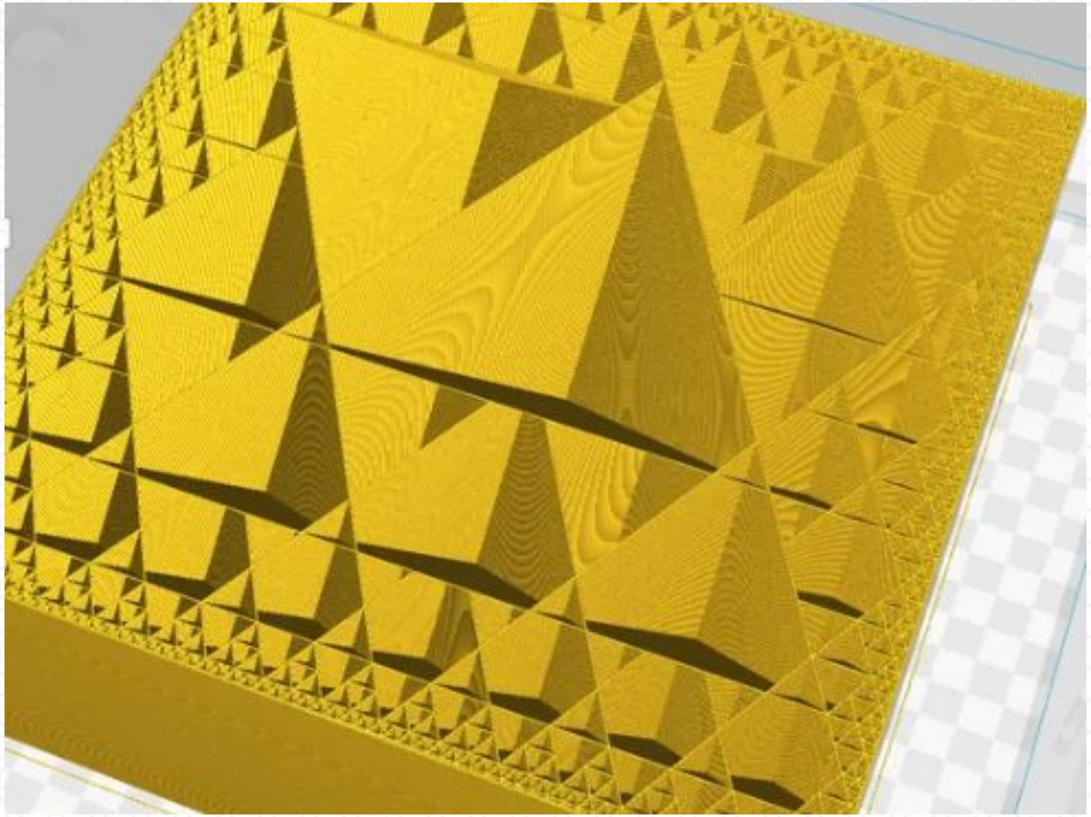
Why building upon Slic3r (AGPL)

- >6 years of development by Alessandro, testing & feedback by a large community. Many ideas and heuristics that seem now obvious were RnD’d by the community and contained in the slicer.
- Keeping slicer open and compatible with 3rd party printers supports spurring new ideas.
- Cost savings on testing, allows for faster release cycles (at least in early startup stage).
- Community provides bug fixes, supports Linux builds & packages.

Community contributions

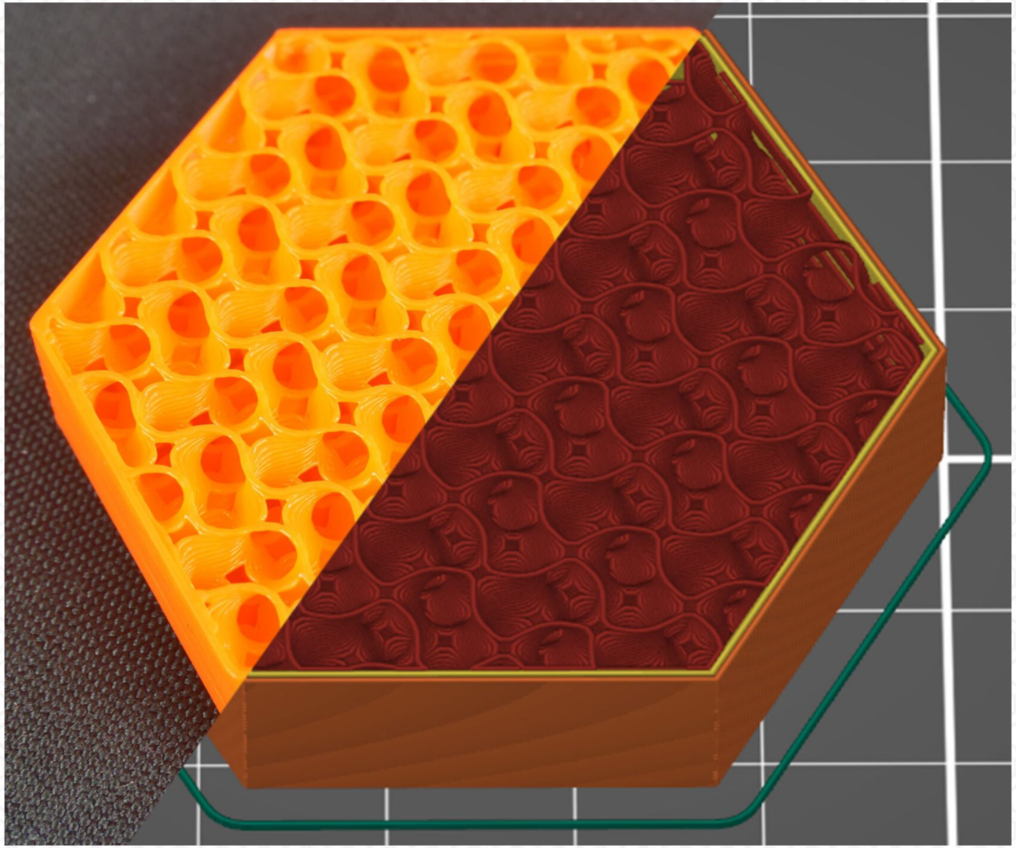
Adaptive Cubic Infill by Martin Boerwinkle

(high school student, won 2017 3rd award at Intel International Science and Engineering Fair, 4th award from ACM)



Community contributions

Gyroid Infill by Remi Durand (@supermerill) in February 2018 for Slic3r Prusa Edition

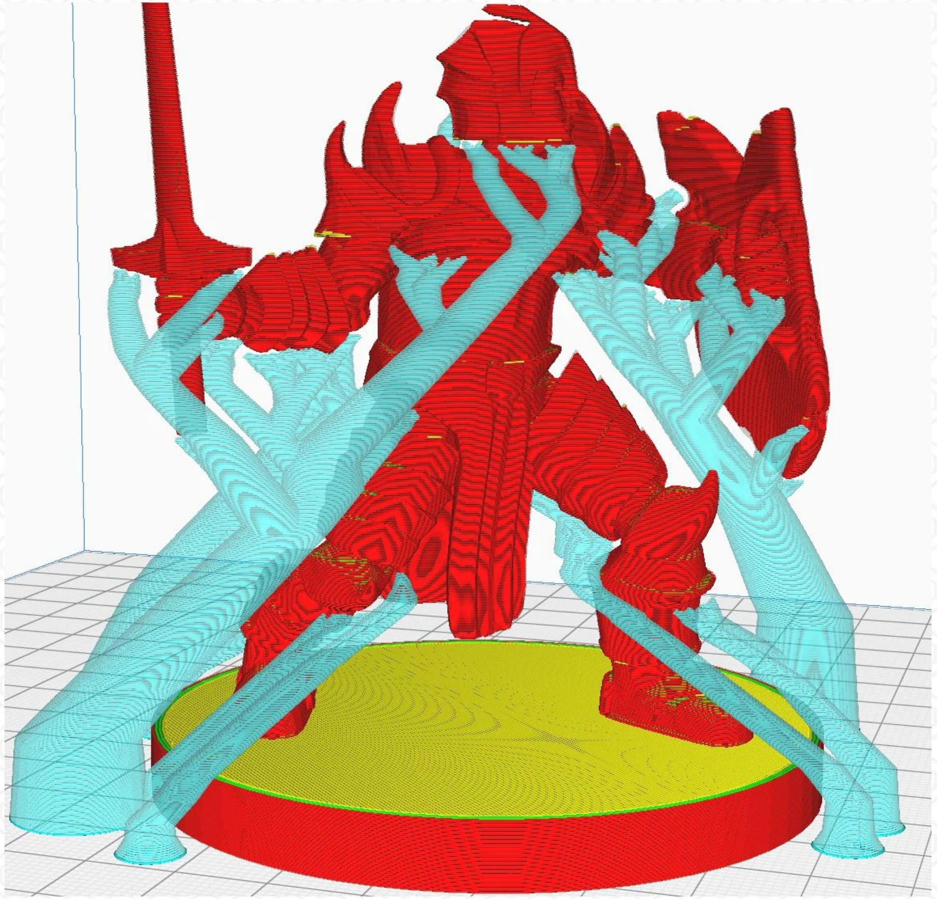


Community contributions

Tree Supports by Thomas Rahm (college student)

Based on Cura tree supports

Improved upon and integrated into PrusaSlicer as Organic Supports



Source code sharing and exchange

- PrusaSlicer uses Cura’s code and ideas:
  - Variable width extrusion Arachne
  - Lightning Infill
  - Adaptive cubic infill by Martin Boerwinkle
  - Some code of Organic supports originates from Cura.
- Cura uses PrusaSlicer’s code and ideas:
  - Build plate arrangement
  - Monotonic infill
- Everybody uses Clipper polygon clipping library by Angus Johnson.

Open source pitfalls

- Successful project generates huge trafic of issues / proposals / contributions. Just processing the traffic consumes a huge amount of mental energy.
- Developers very close to the community, not everybody loves that and not everybody behaves.
- We have to serve our customers first and the day has 24 hours only. Who is our customer?
- Feature creep: We have to keep the software stable, maintainable, usable, support our products, deliver on time.
- Vast majority of our customers prefer simpler discoverable UI with less knobs, we cannot make everybody happy.
- Adding too many features is not compatible with any of the above.

Open source pitfalls

- Open source contributions are often one trick ponies or low quality, don’t fit architecture or project’s goals & direction.
- We are often busy with our own assignments and we don’t have the resources to process pull requests and proposals.
- If contributions are not accepted, contributor may feel unappreciated and forks will spawn.
- Forks with a feature that we don’t offer generate a bad publicity.
- Any successful “corporate” open source will be “librified” anyways.
- Linux is not a desktop “platform”, badly fragmented.

Open Source as a business strategy

Sharing code is good, but not everybody is playing by the rules. Many 3D printer manufacturers just:

- Reskin / rebrand (legal)
- Don’t release source code (illegal)
- Release source code when pressed, but not the whole of it (illegal)
- Release source code when pressed, but hides the origin of the source code (deletes commit logs, copyright headers).
- Release features based on our code before we do.
- Public has no idea who developed what.

Open Source as a business strategy

In a competitive environment, investing into RnD of a core product and then sharing it with competition makes sense if everybody plays by the rules:

- Code is shared as required by the license.
- Credit is given and clearly understood by the customers of all parties borrowing the code.
- It would be nice if code contribution was symmetrical.

Are GPL licenses up to date? Are they clear and understandable? Do they guarantee the above? Is there a need for new open source / hardware licenses?

Thanks for your attention & Happy printing