

3D Printing with Julia

Presenting Euclid, a new high performance multi-material slicer

Jack Minardi
Co-Founder and Software Lead at Voxel8



VOXEL8

The Voxel8 Developer's Kit

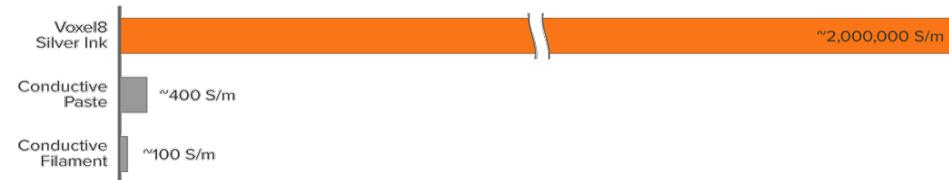


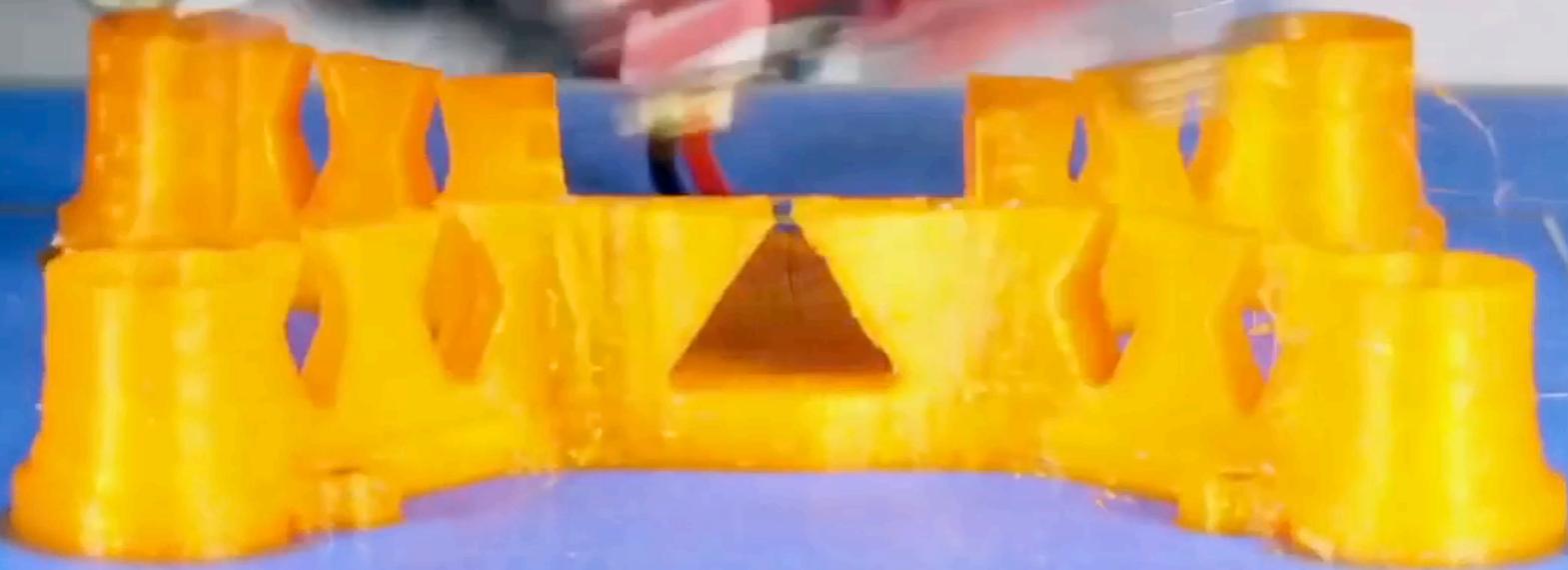
EDISON AWARDS

"ONE OF THE 9 BEST IDEAS AT CES"
- FASTCOMPANY

New Platform for Prototyping 3D Electronics

- Dual modular print heads for multi-material printing
- Room temperature curable conductive ink, 20,000x more conductive than conductive filaments for 3D printing
- Integrated pneumatic dispensing allowing wide materials pallet
- Auto bed leveling and nozzle alignment with laser profilometer







What is Euclid?

Euclid is a high performance path-planner for 3D printing written in Julia.

Path-planning (a.k.a. “Slicing”) is the process of converting a 3D mesh file into a series of X, Y, Z moves that when executed will produce the desired part.



Why Julia?

- Path planning is computationally expensive.
- The expressiveness of a language matters.
- `PyCall` provides any missing “batteries”.
- I am young and naïve.



Why Julia?

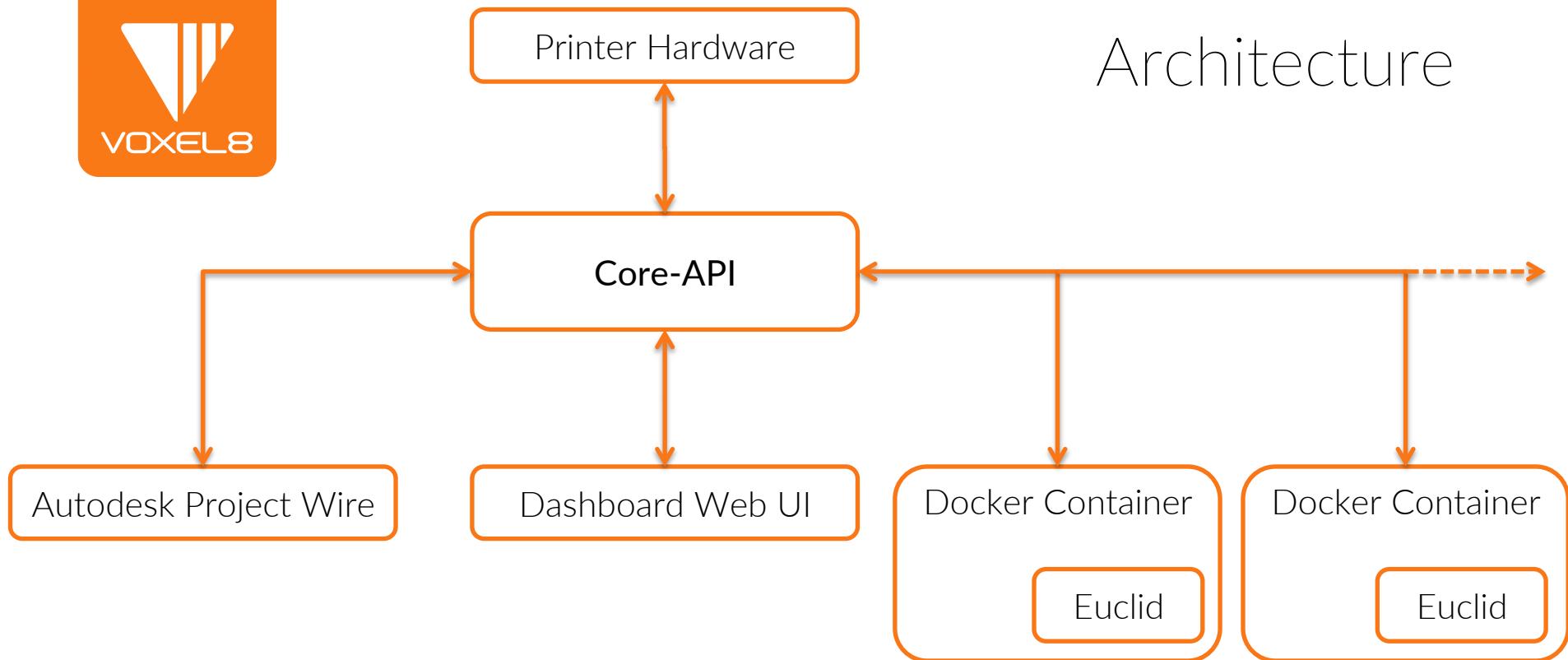


“A company that gets software written faster and better will, all other things being equal, put its competitors out of business.”

– Paul Graham on why Lisp was his secret weapon



Architecture





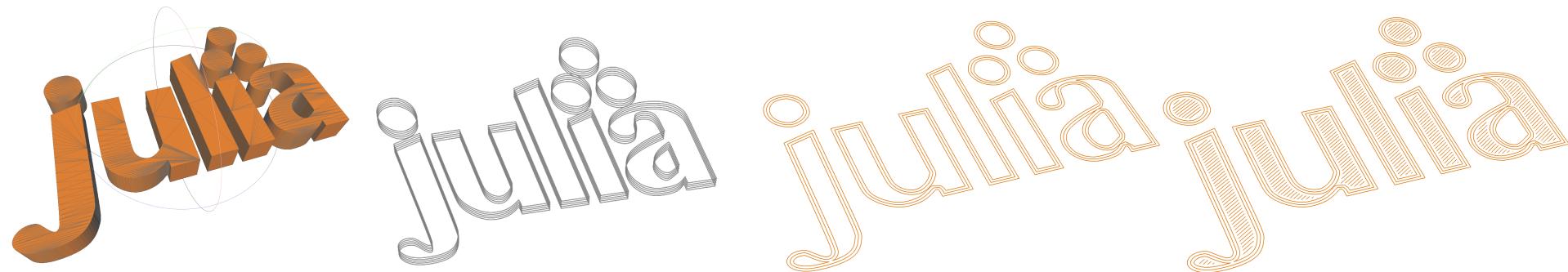
Euclid REQUIRE

ArgParse
BoundingBoxes
Compat
Coverage
Dates
Distances
Graphs

ImmutableArrays
JSON
Lint
Meshes
PyCall
ZMQ

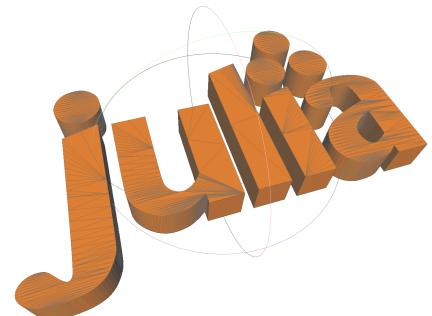


Euclid Pipeline





Design

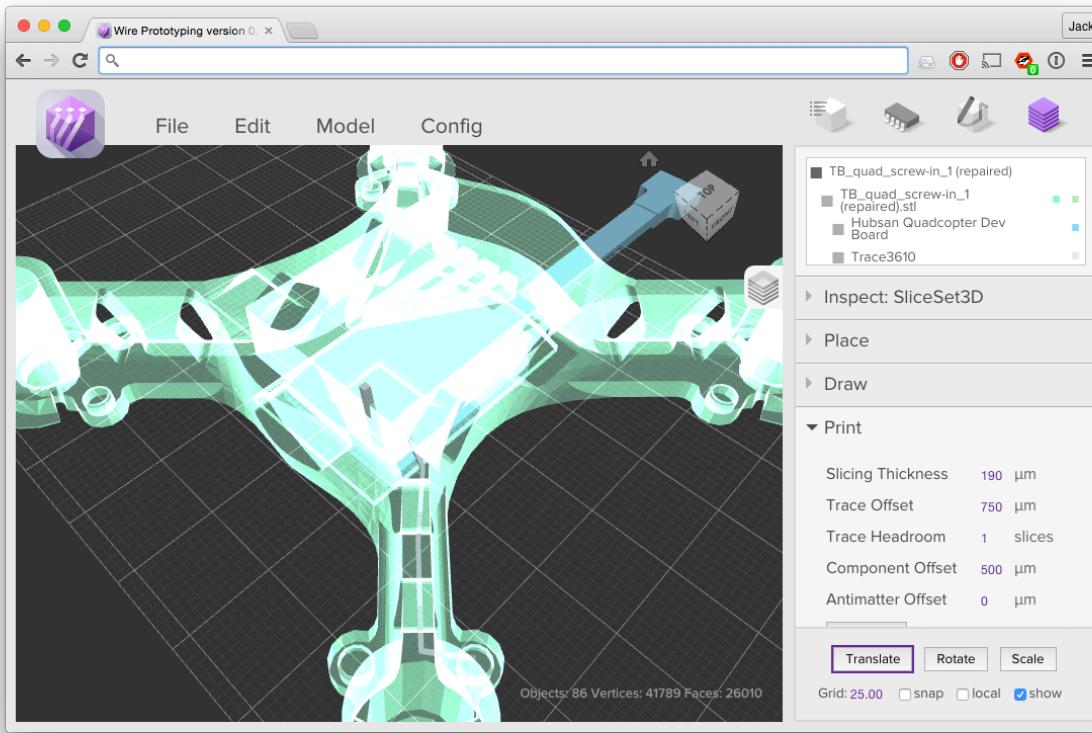




VOXEL8

Autodesk Project Wire

Design





Common Mesh File Formats

STL

```
facet normal ni nj nk
  outer loop
    vertex v1x v1y v1z
    vertex v2x v2y v2z
    vertex v3x v3y v3z
  endloop
endfacet
...
...
```

OBJ

```
v v1x v1y v1z
v v2x v2y v2z
v v3x v3y v3z
...
f 1 2 3
...
...
```



The three main types in the Meshes.jl package are Vertex, Face, and Mesh.

The Mesh type is internally stored similarly to an OBJ file.

Meshes.jl

github.com/JuliaGeometry/Meshes.jl

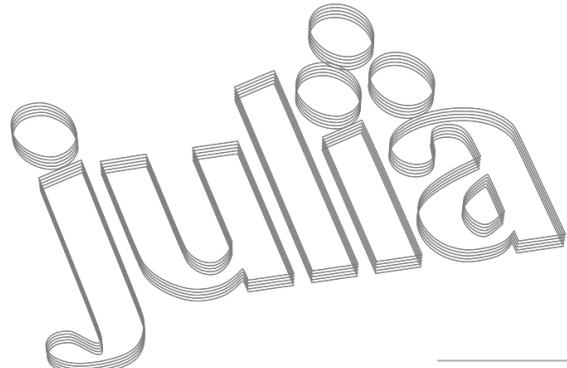
```
typealias Vertex Vector3{Float64}

immutable Face{T}
    v1::T
    v2::T
    v3::T
end

type Mesh{V, F} <: AbstractMesh{V, F}
    vertices::Vector{V}
    faces::Vector{F}
end
```



Slicing





The mesh file is sliced
with Meshes.jl

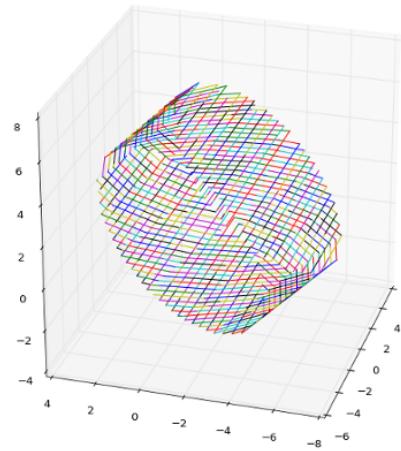
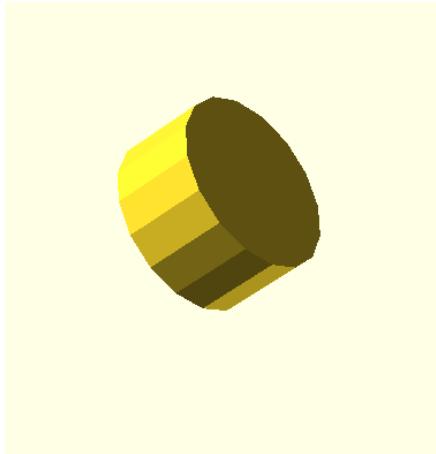
using Meshes

```
m = mesh("/path/to/mesh.stl")
heights = [1, 2, 3]
slices = slice(m, heights)
```



VOXEL8

Slicing





Offsetting Polygons

Design Model

Slice Mesh

Offset

Infill

Optimize Path Order

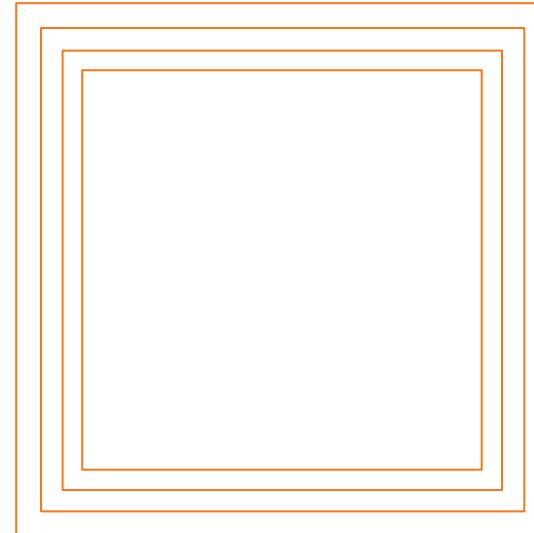
Output GCode

julia



Offsetting Polygons

The inner offset with a distance d of a polygon P is the boundary of the set of points each of which is contained in P and has a distance greater than d from the boundary of P .¹





Offsetting Polygons





Offsetting Attempt 1

Homegrown solution – Polygons.jl

- Greiner-Hormann Algorithm for clipping
- Hormann-Agathos Algorithm for “Point in Polygon” tests

Abandoned because it was too slow and hard to debug.



Offsetting Attempt 2

Wrap Angus Johnson's Clipper Library

- Clipper is the gold standard library used by many other slicers
- Written in C++, our wrapper used Cxx.jl
- See our progress here: github.com/voxel8/Clipper.jl

**Abandoned because it requires unstable Julia,
unstable Cxx, unstable LLVM.**



Offsetting Attempt 3

Use pyclipper with PyCall

- pyclipper is a cython wrapper for Clipper
- PyCall makes this very easy to interop with

This is the method we are currently using.



Offsetting Polygons

Ultimately when Cxx.jl is stable we will use it to interact with the Clipper C++ library.



Infill

Design
Model

Slice Mesh

Offset

Infill

Optimize
Path Order

Output
GCode

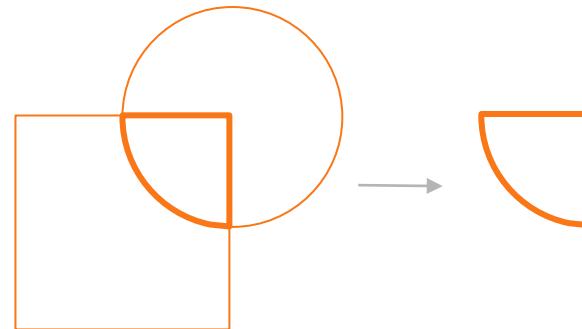
julia



Infill utilizes clipping, which is finding the intersection of two polygons.

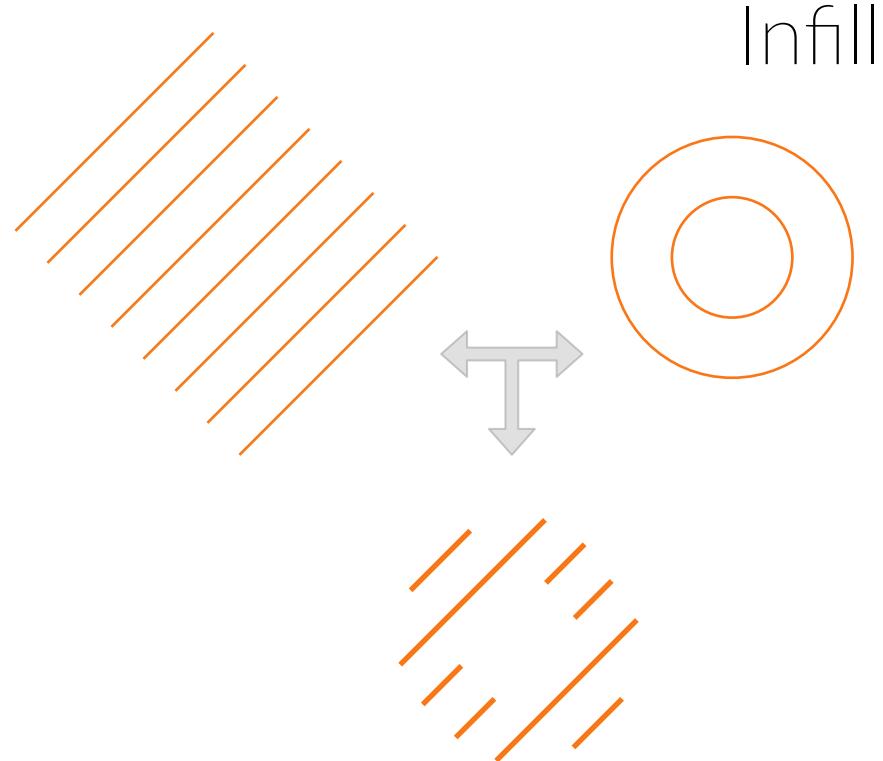
We also use the Clipper library here.

Infill





Infill is generated by clipping a rectilinear pattern against the inner-most offset.





VOXEL8

Optimize Path Order

Design
Model

Slice Mesh

Offset

Infill

Optimize
Path Order

Output
GCode



VOXEL8

At this point we have all the lines we need to execute to build the part, we just need to choose the order in which we get to them.

This is essentially the traveling salesman problem

Optimize Path Order

Goals

- Minimize print time
- Maximize print quality
- Minimize travel (moves w/o extrusion)
- Minimize “retractions”



GCode

Design
Model

Slice Mesh

Offset

Infill

Optimize
Path Order

Output
GCode



GCode

GCode is essentially a series of X, Y, Z moves, how fast to execute them, and whether or not to extrude material while moving.



GCode

What is GCode?

- It was invented in the 1950s right here at MIT
- Conceived as a numerical control language for CNCs, has been adopted by the 3D printer community
- There are various flavors and implementations but they all share a common core.



VOXEL8

GCode

G0 E-0.3 F960

G0 X12.5 Y70.23 F9000

G0 E0.32 F960

G1 X121.69 Y70.44 E0.08 F3000

G0 X121.69 Y70.75 F9000



Euclid Command Line Interface

```
$ euclid \
  --config /path/to/config.json \
  /path/to/plastic.stl \
  /path/to/conductor.stl
```



Dockerizing Euclid

What is Docker?

“Docker allows you to package an application with all of its dependencies into a standardized unit for software development.”

Why?

- No dependency on AWS's tech
 - .ebextensions
 - Amazon Linux
- Easy to test the environment and deploy anywhere Docker runs.



```
# Start from a basic installation of Julia v0.3 on Ubuntu.  
FROM julia-on-ubuntu  
  
# Install package dependencies.  
RUN apt-get update && \  
    apt-get install -y \  
        python-pip \  
        python-scipy && \  
    pip install pyclipper==0.9.3b0  
  
# Julia package installs write to the home directory and shouldn't be  
# run as root.  
USER julia  
  
# Install dependencies and euclid.  
RUN julia -e 'Pkg.clone("git@github.com:Voxel8/Polygons.jl.git"); \  
    Pkg.clone("git@github.com:Voxel8/Euclid.git"); \  
    Pkg.build("Euclid")'  
  
# Expose euclid's listen port.  
EXPOSE 2014
```

Dockerfile



We've built a Ruby on Rails server that handles incoming requests and sends them to the Docker container.

The Rails server provides a RESTful API to Euclid.

Core-API

Endpoint	Methods
/jobs	POST
/jobs/:id	GET
/configs	GET, POST
/configs/:id	GET, PUT, DELETE



Core-API

Client Workflow

- Client POSTs to jobs/ with a config_id and the model data.
- Core-API returns a Job ID
- Client can request details of any job by Job ID



Core-API

To start the slice job we simply run the following command from Rails:

```
$ docker exec '#{container_id}' euclid --config '#{config}' -o '#{outfile}' #{infiles}
```

When Euclid finishes it writes the GCode file to a shared directory that is accessible from Rails. This file is then uploaded to an S3 bucket.



The screenshot shows a web browser window with a dark grey header bar. The header includes a search bar, a refresh button, and several icons. Below the header, the word "dashboard" is displayed in a smaller font. To the right of the dashboard title are navigation links: MODELS (which is highlighted in orange), CONFIGURATIONS, PRINTERS, ACCOUNT, and VOXEL8 HOME.

The main content area is titled "MY MODELS". On the left side of this section is a small orange square containing the Voxel8 logo. On the right side is a "NEW MODEL" button.

Three model cards are listed:

- Model Name.ext**
Created by: Username
• Tue, Jun 09 2015 14:08:52
• Configuration: Leroids 0.35mm Big Fast Prints 1.1
[DOWNLOAD G-CODE](#)
- Model Name.ext**
Created by: Username
• Tue, Jun 09 2015 14:08:52
• Configuration: Leroids 0.35mm Big Fast Prints 1.1
[DOWNLOAD G-CODE](#)
- Model Name.ext**
Created by: Username
• Tue, Jun 09 2015 14:08:52
• Configuration: Leroids 0.35mm Big Fast Prints 1.1
[ERRORS DETECTED - CLICK TO VIEW](#)

Dashboard



The screenshot shows a web browser window with a dark header bar. The header includes a search bar, a refresh button, and several icons. Below the header, there is a navigation menu with tabs: 'dashboard', 'MODELS', 'CONFIGURATIONS' (which is highlighted in orange), 'PRINTERS', 'ACCOUNT', and 'VOXEL8 HOME'. On the left side of the main content area, there is a small orange square containing the Voxel8 logo. The main content area has a title 'CONFIGURATIONS' and a 'NEW CONFIGURATION' button. Below the title, there are four configuration cards, each with a 'Config Info' button and an 'Edit Config' button. The first card is titled 'Configuration name_Rev1' and shows last update and creation details. The other three cards are identical.

Configuration	Last updated	Created by
Configuration name_Rev1	Thu, Jun 11 2015 19:44:11 GMT -0400 (EDT)	Username
Configuration name_Rev1	Thu, Jun 11 2015 19:44:11 GMT -0400 (EDT)	Username
Configuration name_Rev1	Thu, Jun 11 2015 19:44:11 GMT -0400 (EDT)	Username
Configuration name_Rev1	Thu, Jun 11 2015 19:44:11 GMT -0400 (EDT)	Username

Dashboard



The screenshot shows a web browser window with a dark grey header bar. The header includes a search bar, a refresh button, and several icons. Below the header, the word "dashboard" is displayed in a light grey font. To the right of the dashboard name are five navigation links: MODELS, CONFIGURATIONS, PRINTERS (which is underlined in red), ACCOUNT, and VOXEL8 HOME. The main content area has a white background. At the top left of this area is a small orange square containing the Voxel8 logo. To its right, the word "PRINTERS" is centered in a large, black, sans-serif font. On the far right of this row is a dark grey rectangular button with the white text "ADD PRINTER". Below this row, there are two entries, each enclosed in a light grey box. Each entry has a small orange square with the Voxel8 logo at the top left. To the right of the logo is the printer's name. To the right of the name is a "EDIT PRINTER" button. Below the printer names, there are two tabs: "Printer Info" (which is highlighted in orange) and "Printer Control". Under the "Printer Info" tab, the first entry is labeled "Workshop Printer" and the second is labeled "Office 1". Each entry also lists its status: "Status: online" for the Workshop Printer and "Status: offline" for the Office 1 printer.

dashboard

MODELS CONFIGURATIONS PRINTERS ACCOUNT VOXEL8 HOME

PRINTERS

ADD PRINTER

Printer Info Printer Control

Workshop Printer

Status: online
Active: Yes
Configuration: Leroids 0.35mm Big Fast Prints 1.1

EDIT PRINTER

Printer Info Printer Control

Office 1

Status: offline

EDIT PRINTER

Dashboard



Conclusion

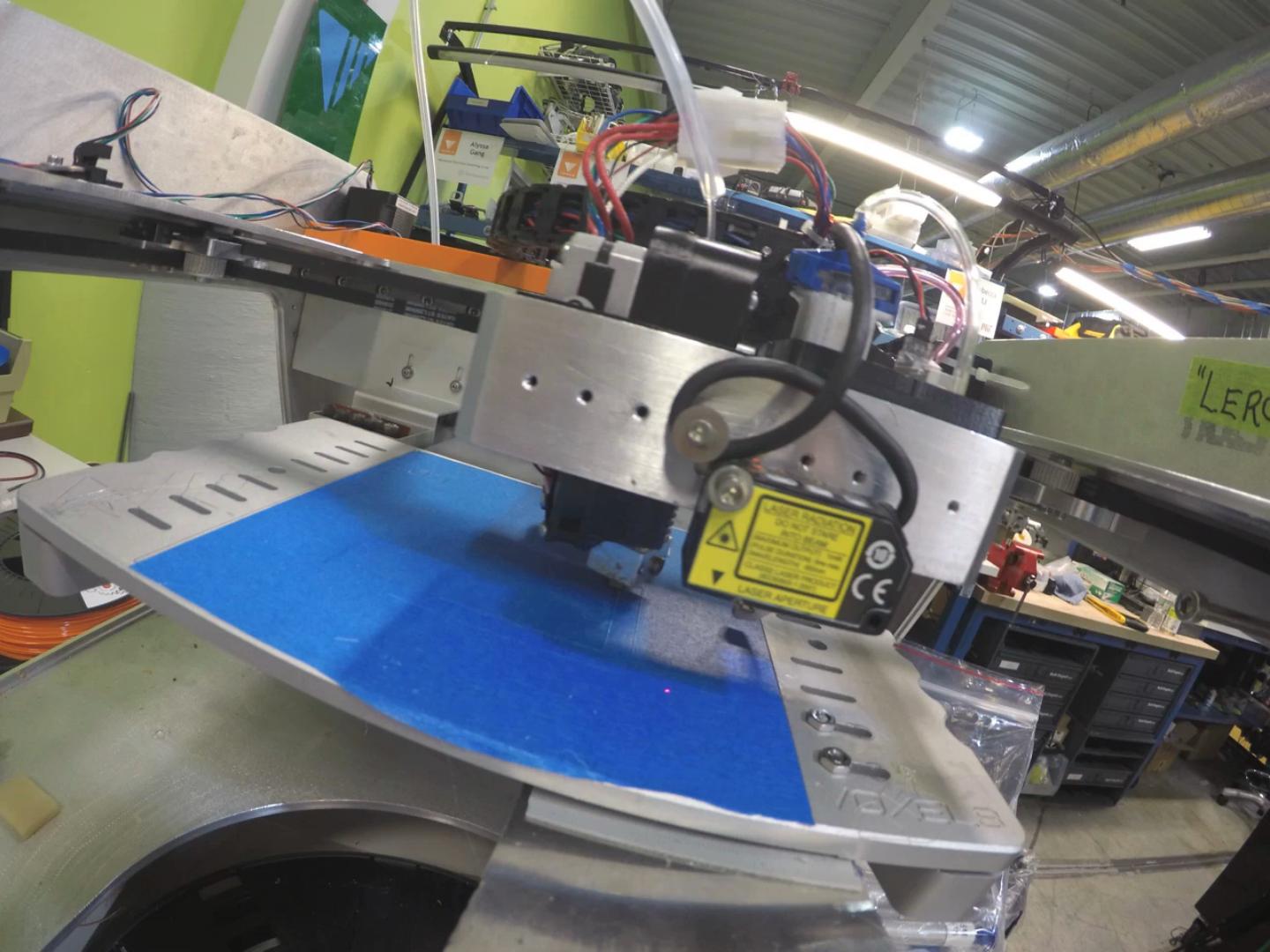
- The speed at which we can add features has been great for productivity.
- Even though Julia is young, it has a growing number of useful packages.
- Core language has been stable and fast.
- Most packages (that claim stability) have been very stable.
- It is easy to install and manage packages (as they are just git repos)
- If there is no package available, PyCall always works and has plenty of "batteries".
- Even though Julia wasn't designed for the web, Docker makes it easy to put in the cloud.



Conclusion

Wishlist

- Julia 0.4 stable release
- Package pre-compilation
- Docstrings
- Stable Cxx.jl
- Debugger



Questions?



VOXEL8



Jack Minardi
Voxel8

@jackminardi
@voxel8co