ImplicitCAD: Haskell all of the Things

Julia Longtin

ImplicitCAD Project

julia.longtin@gmail.com

February 27, 2020

Overview

Who am I?

- Free Software Developer for 20+ years
  - Contributor to OpenEMR, FAI, TinTin++,...
- Work at Wire.com
  - The presented work has no affiliation with my employer
- ImplicitCAD maintainer

Who am I?

- Have run two hackerspaces

- Spent 8+ Years teaching 10 hours a week
- Tested ImplicitCAD on the members at HacDC

Who am I?

- Transhumanist, 3D printer aficionado
- Scratch building mutant printers for 10+ years
- Created method for converting 3D prints to aluminum with microwaves

What is ImplicitCAD?

- Programatic 3D Modeling System
- Written in Haskell
- Licensed under the AGPLv3
- Originally written by Christopher Olah in 2011
  - I Took over as maintainer in 2014

Source Code

- https://github.com/colah/ImplicitCAD/

Online Editor

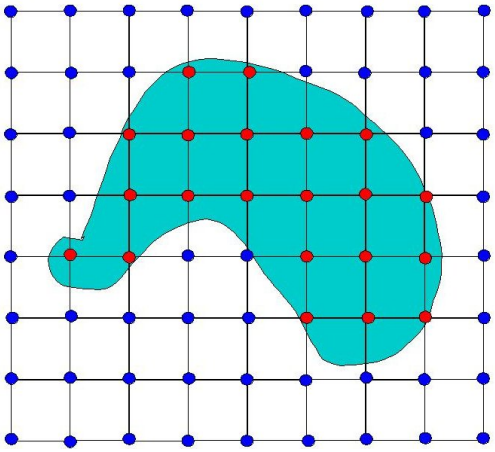
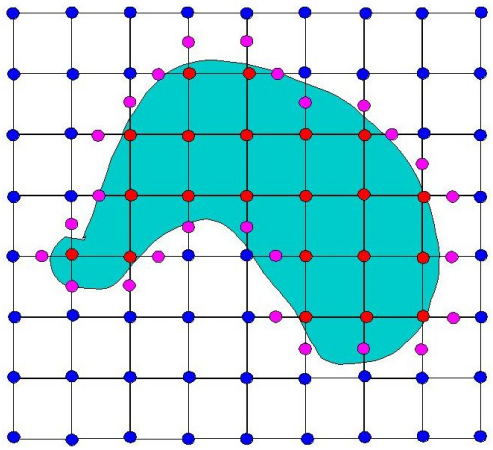
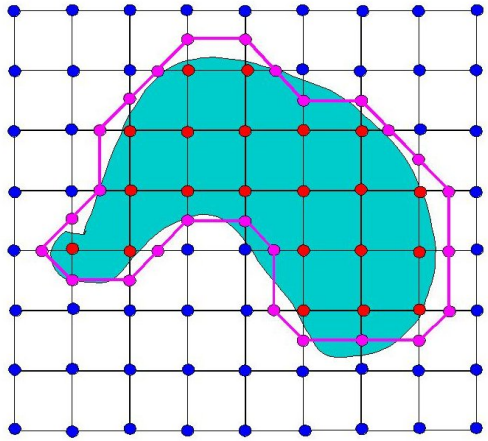
- http://implicitcad.org/editor/

What is ImplicitCAD?

- Three Components:
  - 3D/2D Rendering Engine
  - SCAD Execution Engine
  - Library (Haskell DSL)

ImplicitCAD's Rendering Engine

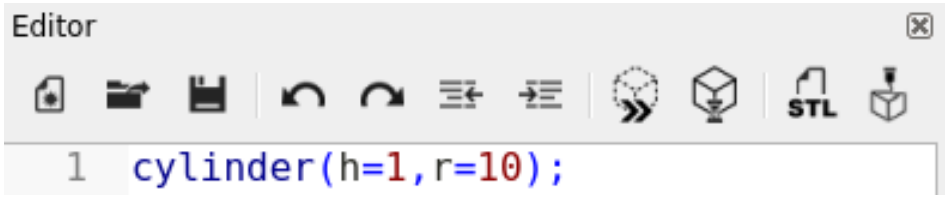
- Renders 3D: STL, OBJ... "Triangles on the outside".
  - Uses marching cubes algorithm for triangulation functions
- Renders 2D: SVG, SCAD, PNG, DXF...
  - Uses marching squares, or raytracing depending on format

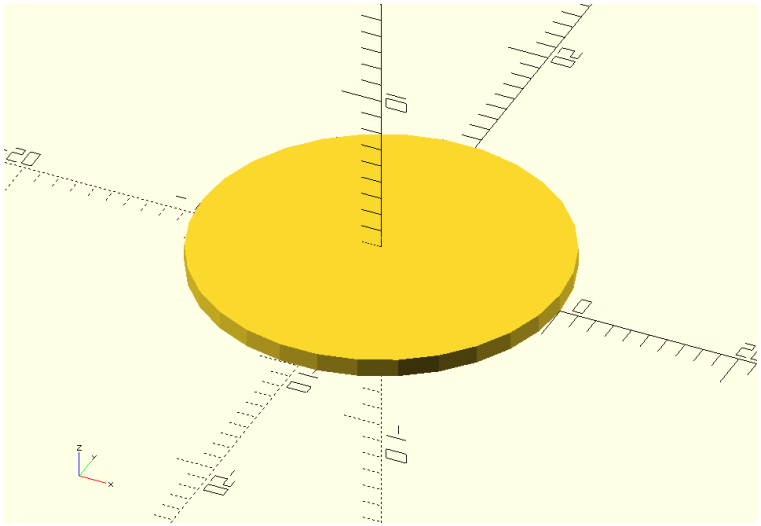




- Highly parallelized, uses all CPUs

ImplicitCAD's SCAD Execution Engine

- OpenSCAD like, not quite OpenSCAD compatible
  - Lack of a real SCAD standard allows us to experiment with the language
- Leans toward more primitive elements, with more power





ImplicitCAD's Haskell Library

- Usable from simple Haskell programs (Haskell DSL!) to generate objects
- Capable of SCAD execution without modeling
- Separable expression executor

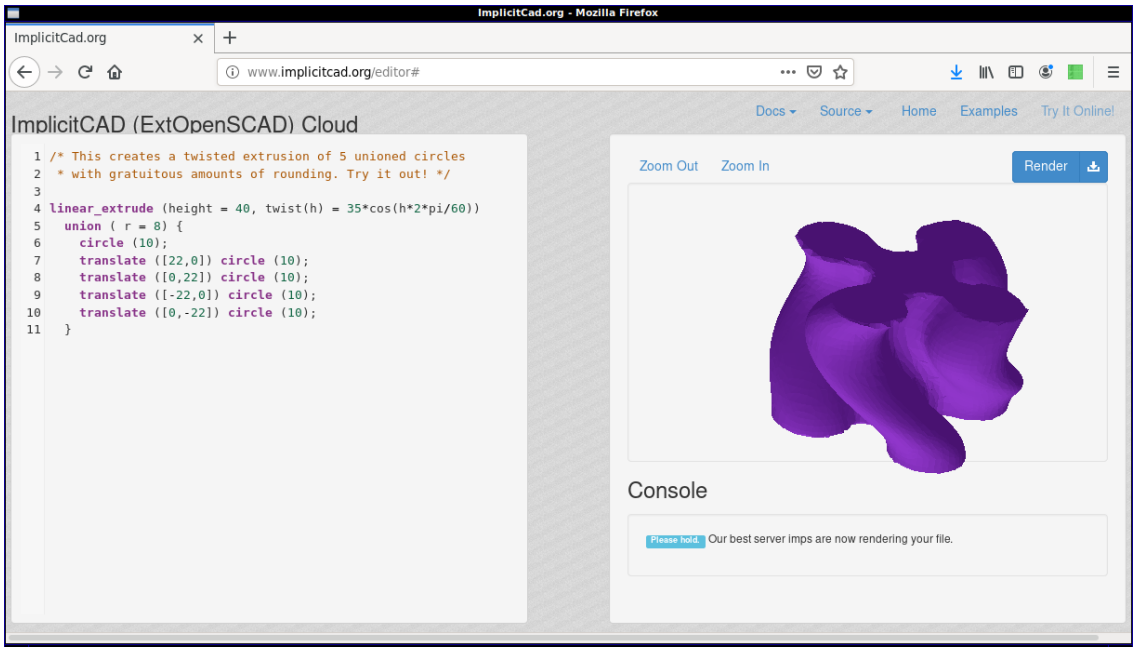
```
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ cat implicitcad-haskell_cylinder.hs
-- A simple cylinder
import Prelude (($))
import Graphics.Implicit(writeSTL, cylinder2)

main = writeSTL 0.2 "cylinder_example.stl" $ cylinder2 10 10 1

juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ ghc implicitcad-haskell_cylinder.hs -o implicitcad-haskell_cylinder
Loaded package environment from /home/juri/ImplicitCAD/ImplicitCAD-newish/.ghc.environment.i386-linux-8.6.5
Linking implicitcad-haskell_cylinder ...
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ time ./implicitcad-haskell_cylinder

real    0m2.697s
user    0m2.649s
sys     0m0.036s
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ ls -la cylinder_example.stl
-rw-r--r--. 1 juri juri 5855239 Feb 23 09:52 cylinder_example.stl
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$
```

ImplicitCAD Executables

- extopenscad: SCAD engine in command line form

```
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ cat implicitcad-scad_cylinder.escad
cylinder(r=10,h=1);
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ extopenscad implicitcad-scad_cylinder.escad
Loading File.
Processing File.
Rendering 3D object from implicitcad-scad_cylinder.escad to implicitcad-scad_cylinder.stl with resolution 0.3349119544218533 in box ((-10.0,-10.0,0.0),(10.0,10.0,1.0))
ExtrudeR 0.0 (Circle 10.0) 1.0
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$
```

- ImplicitSNAP: Engine backing the web site.



Other SCAD Engines

OpenSCAD - https://openscad.org

- Graphical Interface
- GPLv2 License

OpenJSCAD - https://openjscad.org

- Web based UI
- MIT License

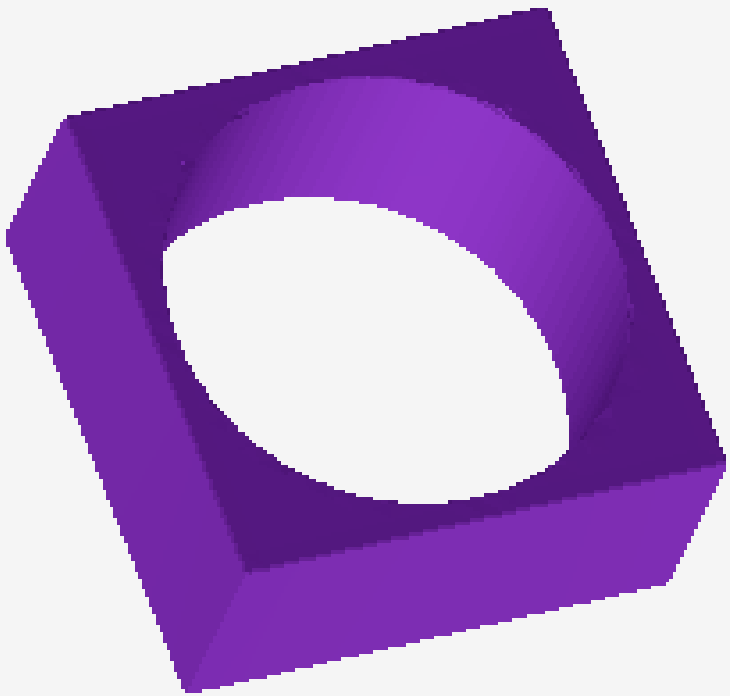
Curv - https://curv3d.org

- Graphical Interface, GPU required
- Apache License
- Uses math based on ImplicitCAD

Why ImplicitCAD?

- Simple SCAD Language

```
1 linear_extrude (height = 10)
2   difference () {
3     translate (-10, -10) square (20);
4     circle (r=9);
5   }
```



Why ImplicitCAD?

- Haskell DSL

```
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ cat example.hs
-- A simple cylinder
import Prelude (($))
import Graphics.Implicit(writeSTL, extrudeR, differenceR, translate, rectR, circle)

main = writeSTL 0.2 "example.stl" $ extrudeR 0.0 (differenceR 0.0 [translate (-10.0,-10.0) (rectR 0.0 (0.0,0.0) (20.0,20.0)),circle 9.0]) 10.0

juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ ghc example.hs -o example
Loaded package environment from /home/juri/ImplicitCAD/ImplicitCAD-newish/.ghc.environment.i386-linux-8.6.5
Linking example ...
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ time ./example

real    0m11.942s
user    0m11.740s
sys     0m0.080s
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ ls -la example.stl
-rw-r--r--. 1 juri juri 13749249 Feb 23 13:36 example.stl
```

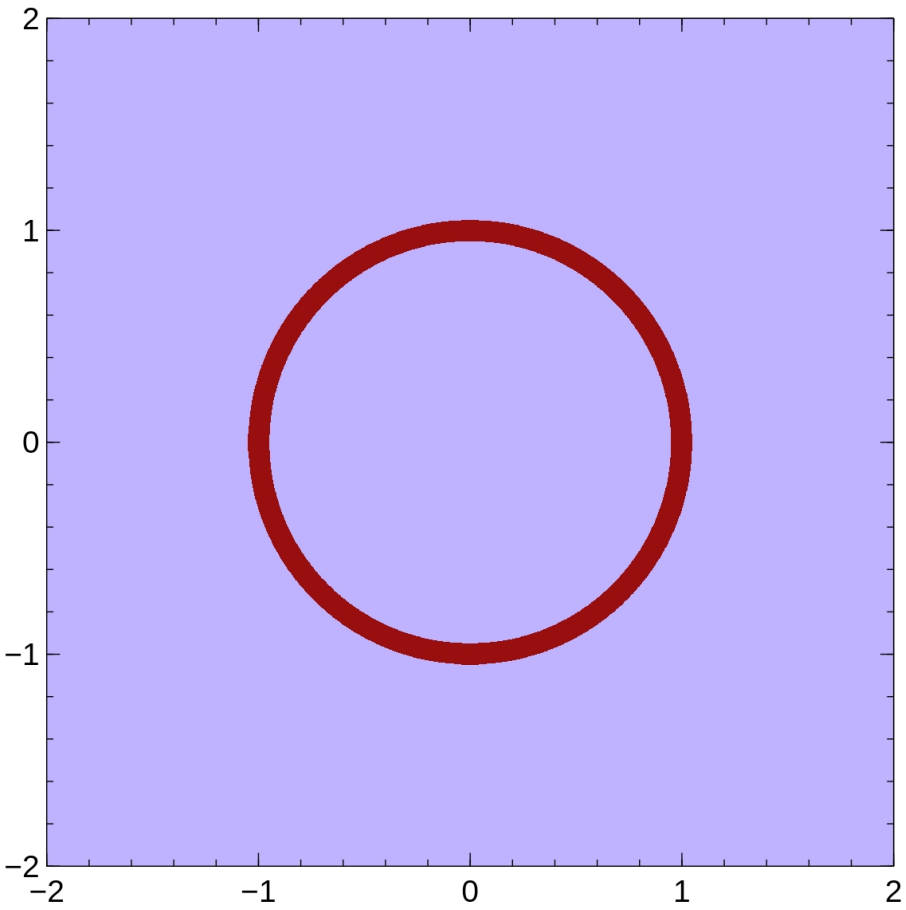
Why ImplicitCAD?

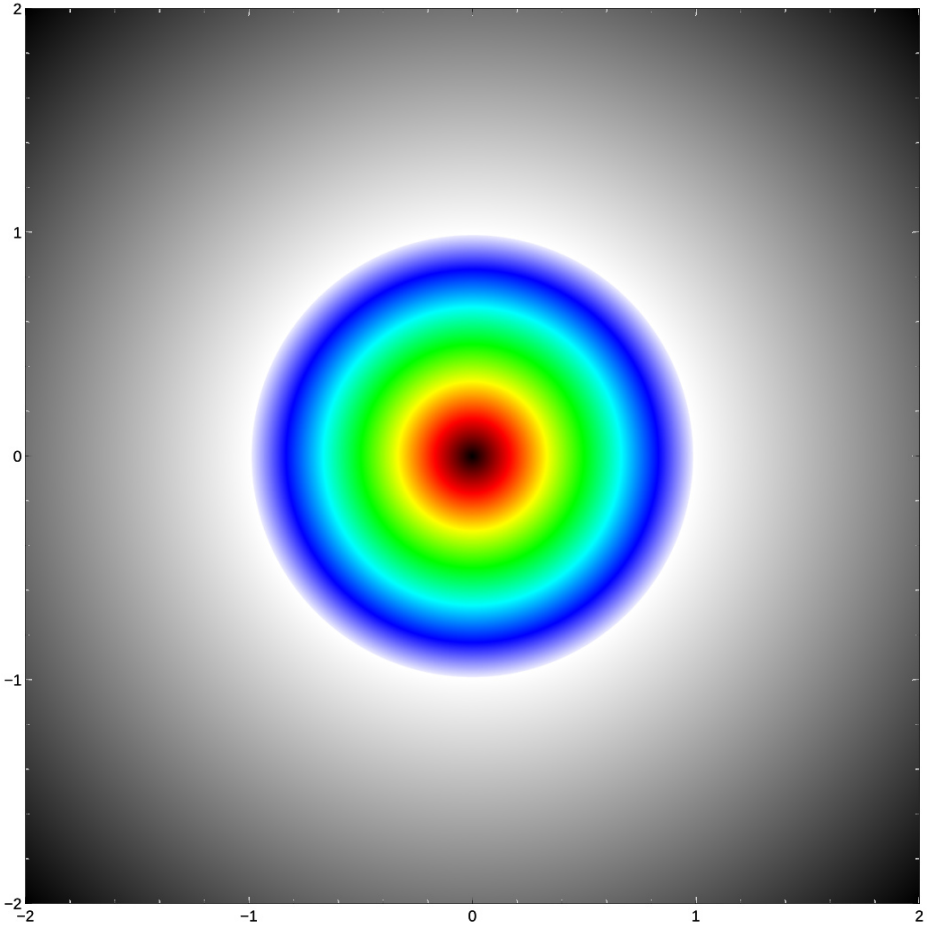
- Nearly Haskell-Native intermediate state

```
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ extopenscad example.escad | grep ExtrudeR
ExtrudeR 0.0 (DifferenceR2 0.0 [Translate2 (-10.0,-10.0) (RectR 0.0 (0.0,0.0) (20.0,20.0)),Circle 9.0]) 10.0
juri@qemuhost:~/ImplicitCAD/ImplicitCAD-newish$ cat example.hs | grep main
main = writeSTL 0.2 "example.stl" $ extrudeR 0.0 (differenceR 0.0 [translate (-10.0,-10.0) (rectR 0.0 (0.0,0.0) (20.0,20.0)),circle 9.0]) 10.0
```

Why ImplicitCAD?

- Implicit Constructive Solid Geometry





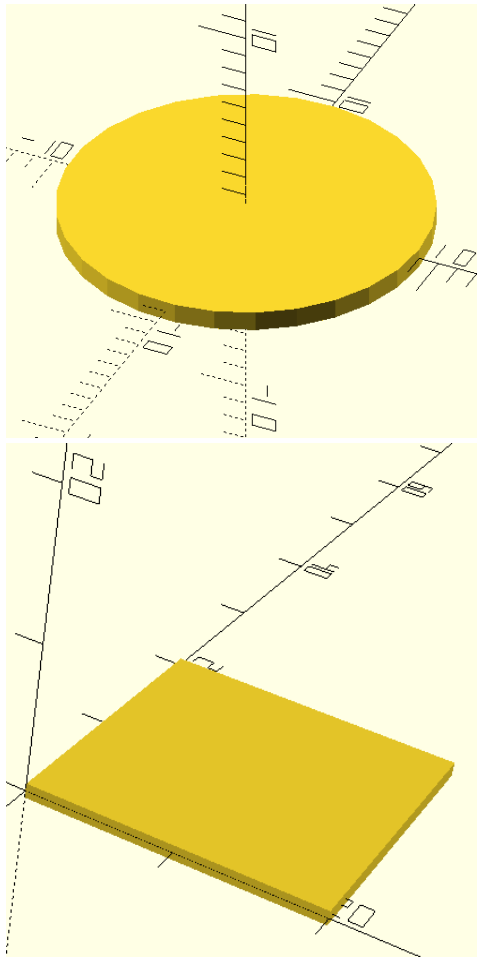
Why ImplicitCAD?

- High Performance through parallel list comprehensions

```
-- | Calculate mid points on X, and Y axis in 2D space.
midsY = [[
        interpolate (y0, objX0Y0) (y1', objX0Y1) (obj $* x0) yres
        | x0 <- pXs |                  objX0Y0 <- objY0   | objX0Y1 <- objY1
       ]| y0 <- pYs | y1' <- tail pYs | objY0   <- objV    | objY1   <- tail objV
       ] `using` parBuffer (max 1 $ div (fromN ny) forcesteps) rdeepseq

midsX = [[
        interpolate (x0, objX0Y0) (x1', objX1Y0) (obj *$ y0) xres
        | x0 <- pXs | x1' <- tail pXs | objX0Y0 <- objY0 | objX1Y0 <- tail objY0
       ]| y0 <- pYs |                  objY0   <- objV
       ] `using` parBuffer (max 1 $ div (fromN nx) forcesteps) rdeepseq

-- | Calculate segments for each side
segs = [[
    getSegs (x0,y0) (x1',y1') obj (objX0Y0, objX1Y0, objX0Y1, objX1Y1) (midA0, midA1, midB0, midB1)
     | x0<-pXs | x1'<-tail pXs |midB0<-mX''  | midB1<-mX'T       | midA0<-mY''  | midA1<-tail mY'' | objX0Y0<-objY0 | objX1Y0<-tail objY0 | objX0Y1<-objY1 | objX1Y1<-tail objY1
    ]| y0<-pYs | y1'<-tail pYs |mX'' <-midsX | mX'T <-tail midsX | mY'' <-midsY                 | objY0 <- objV                       | objY1 <- tail objV
    ] `using` parBuffer (max 1 $ div (fromN $ nx+ny) forcesteps) rdeepseq
```

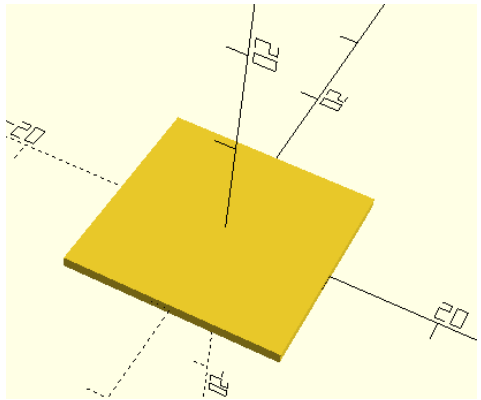
Basic CSG in SCAD

- circle(r=9);
- square(20);

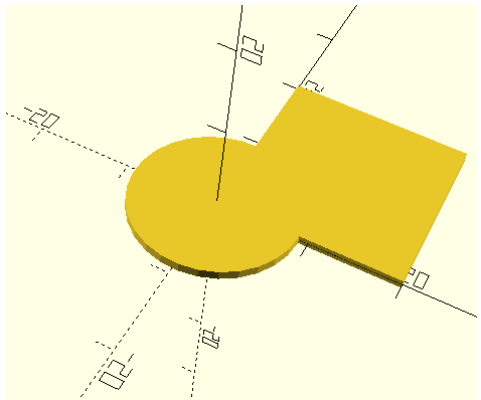


Basic CSG in SCAD

- translate([-10,-10]) square(20);



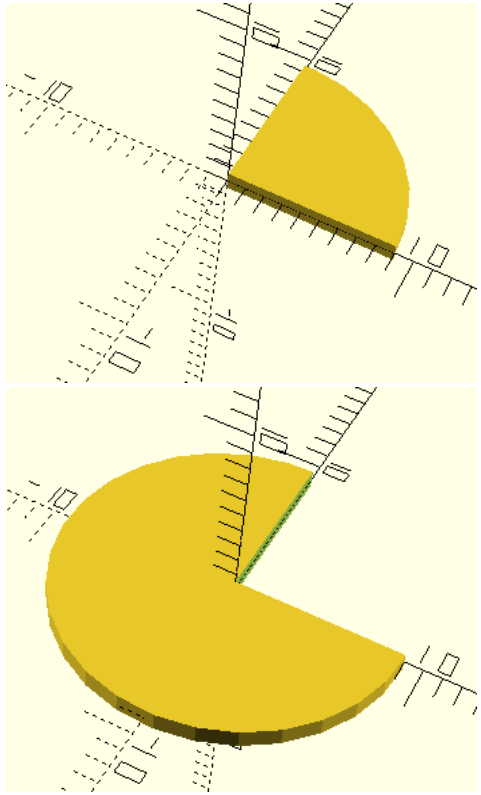
- union() { circle(r=9); square(20); }



Basic CSG in SCAD

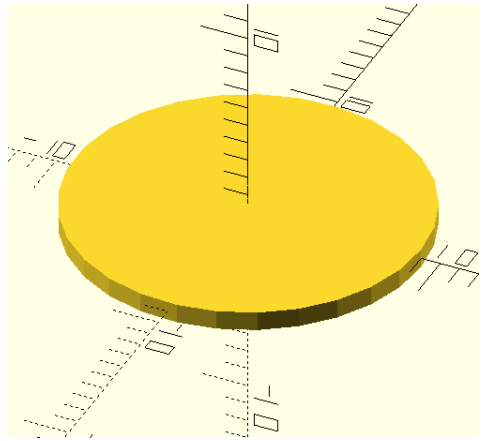
- intersection() { circle(r=9); square(20); }

- difference() { circle(r=9); square(20); }

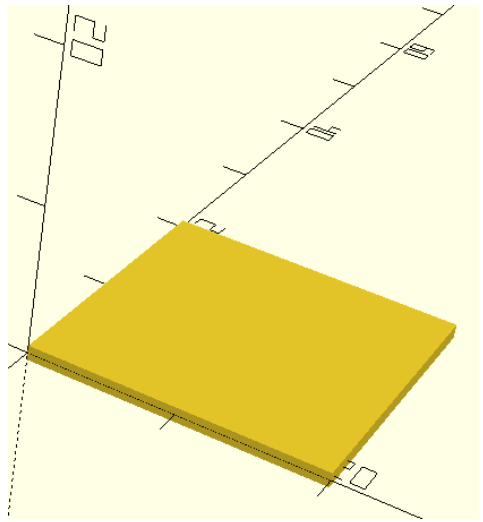


Basic CSG in Haskell

- main = writeSVG 0.2 "example.svg" $ circle 9.0

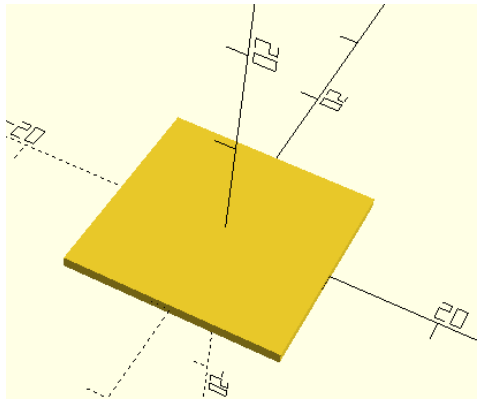


- main = writeSVG 0.2 "example.svg" $ rectR 0 (0,0) (20,20)

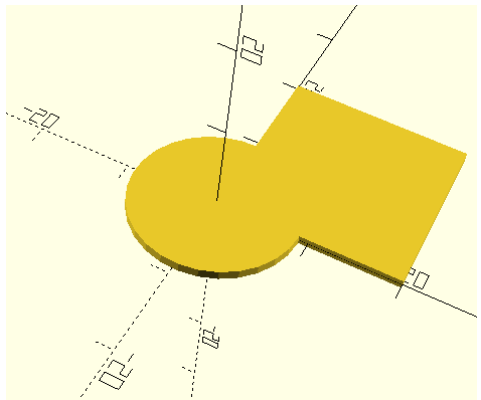


Basic CSG in Haskell

- main = writeSVG 0.2
  "example.svg" $ translate
  (-10,-10) $ rectR 0 (0,0) (20,20)



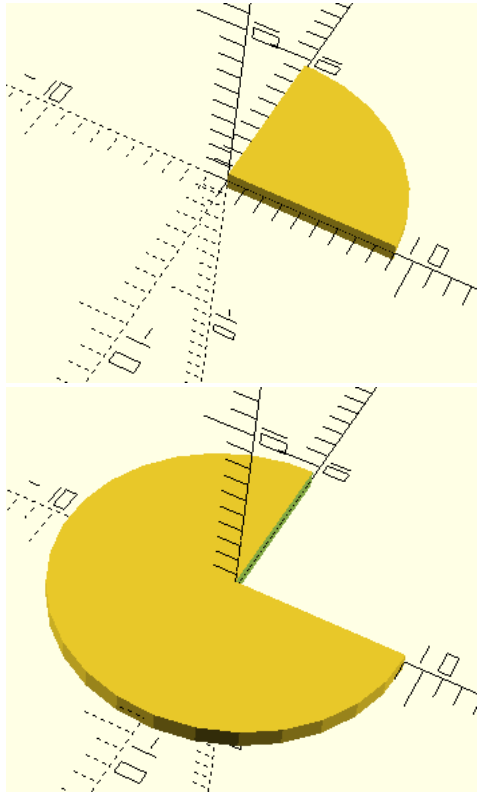
- main = writeSVG 0.2
  "example.svg" $ unionR 0
  [circle 9, rectR 0 (0,0) (20,20)]



Basic CSG in Haskell

- main = writeSVG 0.2
  "example.svg" $ intersectR 0
  [circle 9, rectR 0 (0,0) (20,20)]

- main = writeSVG 0.2
  "example.svg" $ differenceR 0
  [circle 9, rectR 0 (0,0) (20,20)]



Additional Primitives

- polygon();
- cube();
- sphere();
- cylinder();

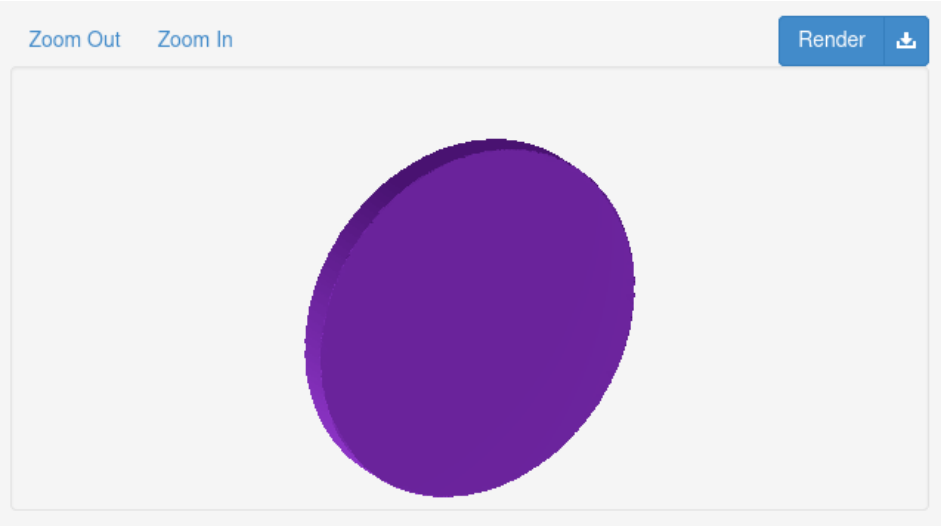
Additional Operations

- scale() { }
- rotate() { }
- linear_extrude( ) { }

Flow Control

- module ( ) { }
- function () { }
- if ... then ... else
- for ( ) { }
- include ...
- use ...
- echo ();

SCAD Example: Disc




```
module disc_2d(diameter){
  radius=diameter/2;
  circle(r=radius);
}
module disc_3d(diameter, thickness){
  linear_extrude(thickness)
  disc_2d(diameter);
}
disc_3d(thickness=10, diameter=120);
```

SCAD Example: Disc


```
module cylinder_3d(height, diameter){
  linear_extrude(height){
    circle(r=diameter/2);
  }
}
cylinder_3d(height=10, diameter=120);
```

SCAD Example: Bead

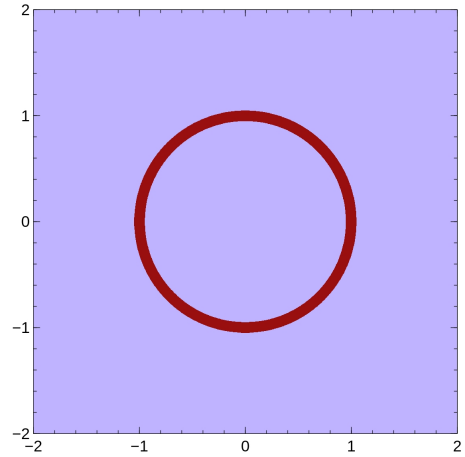
```
module bead_3d(height,diameter,hole_diameter) {
  difference() {
    cylinder(r=diameter/2, h=height);
    cylinder(r=hole_diameter/2, h=height);
  }
}
bead_3d(height=10, diameter=120,hole_diameter=20)
```

Implicit CSG

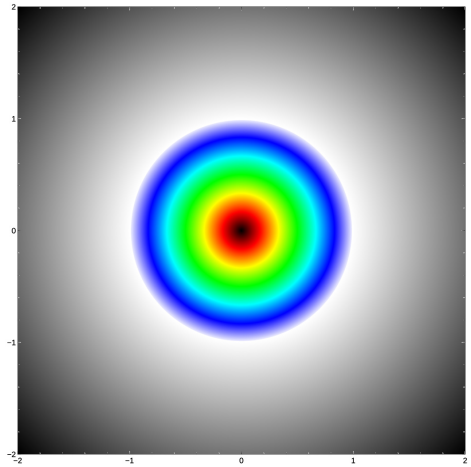
- CSG using Implicit functions
- Implicit functions are functions that define gradient to edge
- Interior of objects = negative value
- Exterior of objects = positive value

Circles

- $x^2 + y^2 = r^2$

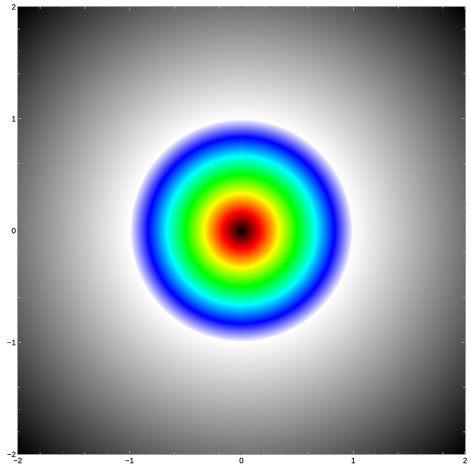


- $f(x, y) = sqrt(x^2 + y^2) - 1$

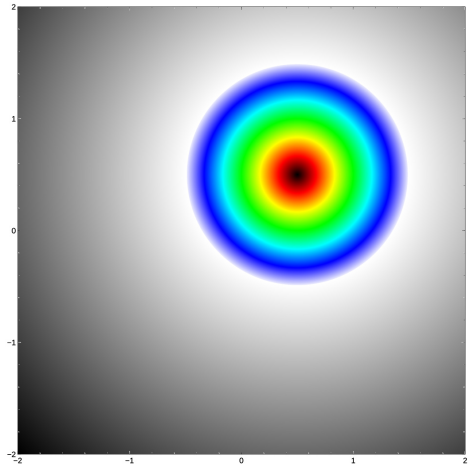


Translation

- $f(x, y) = sqrt(x^2 + y^2) - 1$

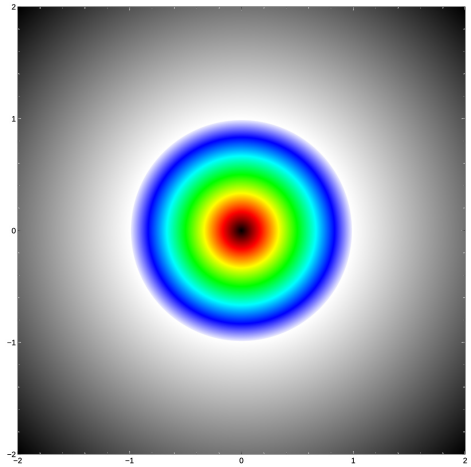


- $tf(x, y, tx, ty) =$
  $sqrt((x - tx)^2 + (y - ty)^2) - 1$

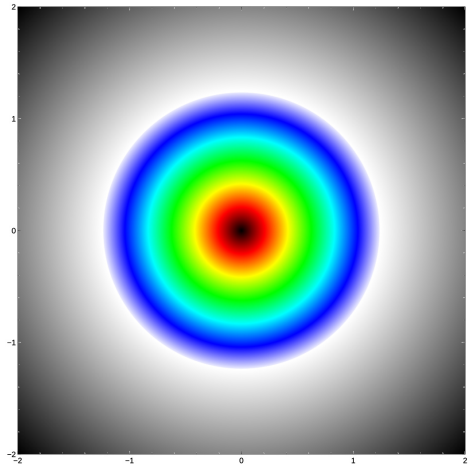


Scaling

- $f(x, y) = sqrt(x^2 + y^2) - 1$

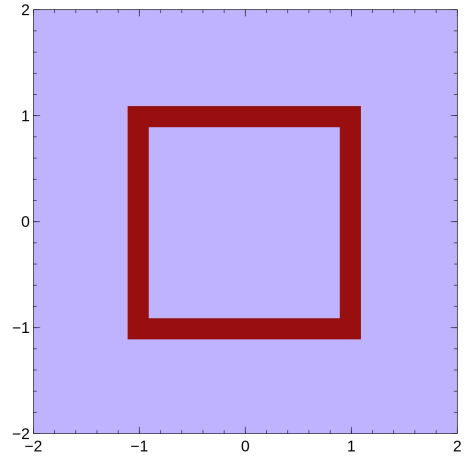


- $sf(x, y, sx, sy) = sqrt((x/sx)^2 + (y/sy)^2) - 1$

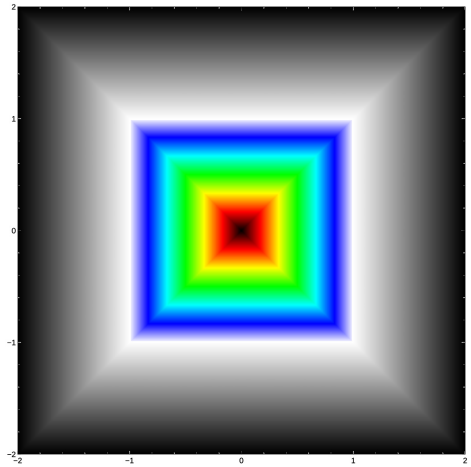


Squares

- $abs(x) = r \wedge abs(y) < r \vee abs(y) = r \wedge abs(x) < r$

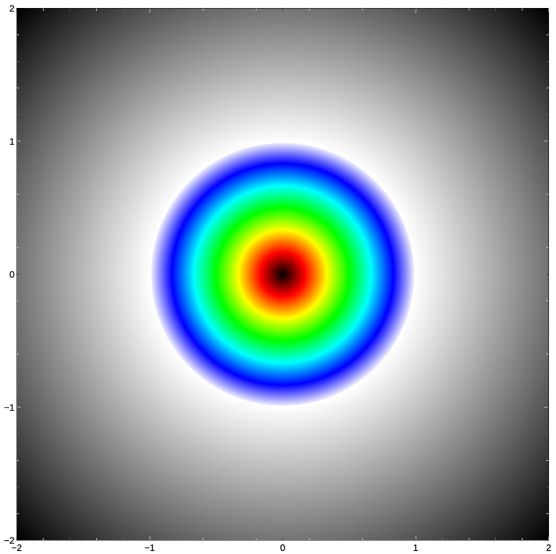


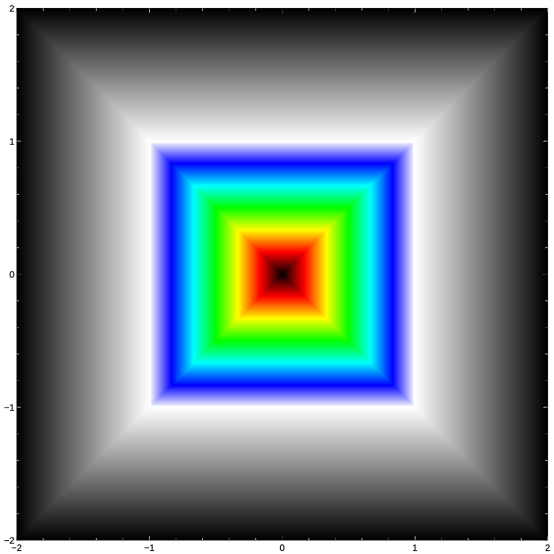
- $f(x, y) = maximum(abs(x), abs(y)) - 1$

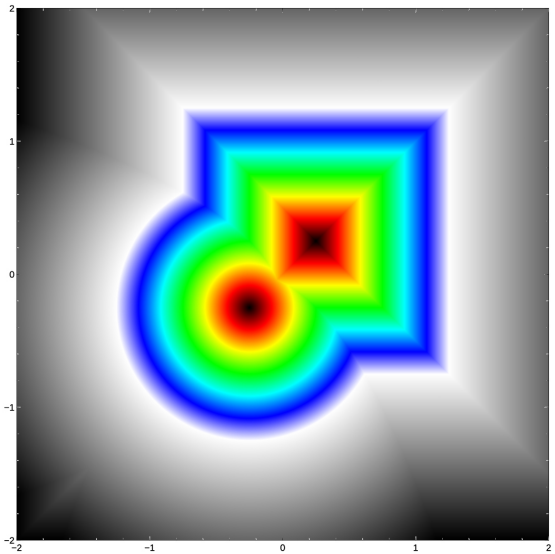


Unioning

- $f(x, y) = sqrt(x^2 + y^2) - 1$
- $f(x, y) = maximum(abs(x), abs(y)) - 1$
- $f(x, y) = minimum(sqrt((x+0.25)^2+(y+0.25)^2) - 1, maximum(abs(x - 0.25), abs(y - 0.25)) - 1)$

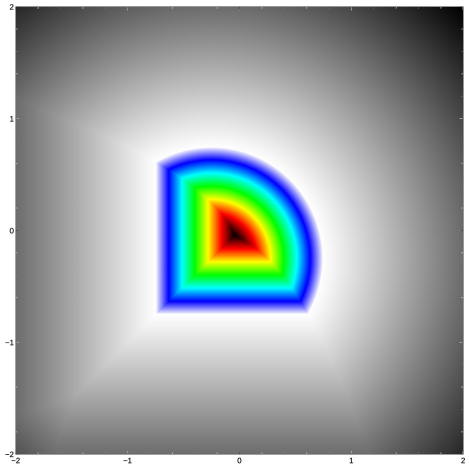






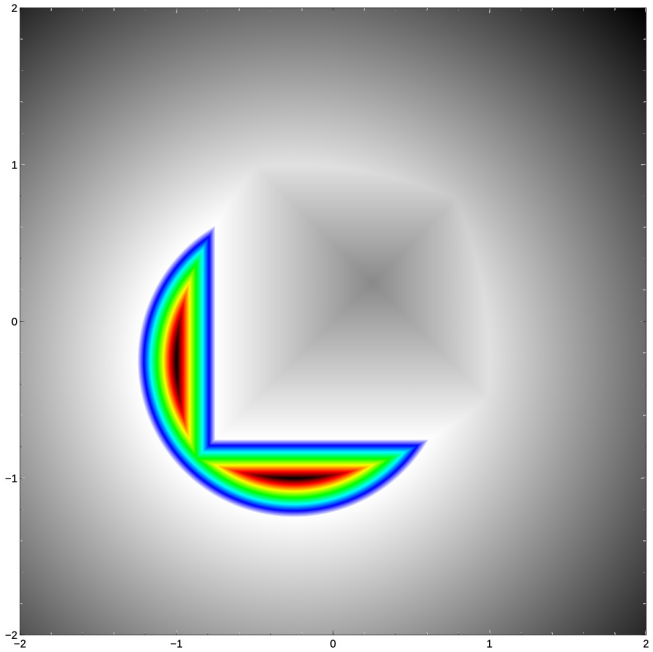
Intersecting

- Taking the maximum of the objects
- $maximum(sqrt((x + 0.25) * (x + 0.25) + (y + 0.25) * (y + 0.25)) - 1, maximum(abs(x - 0.25), abs(y - 0.25)) - 1)$



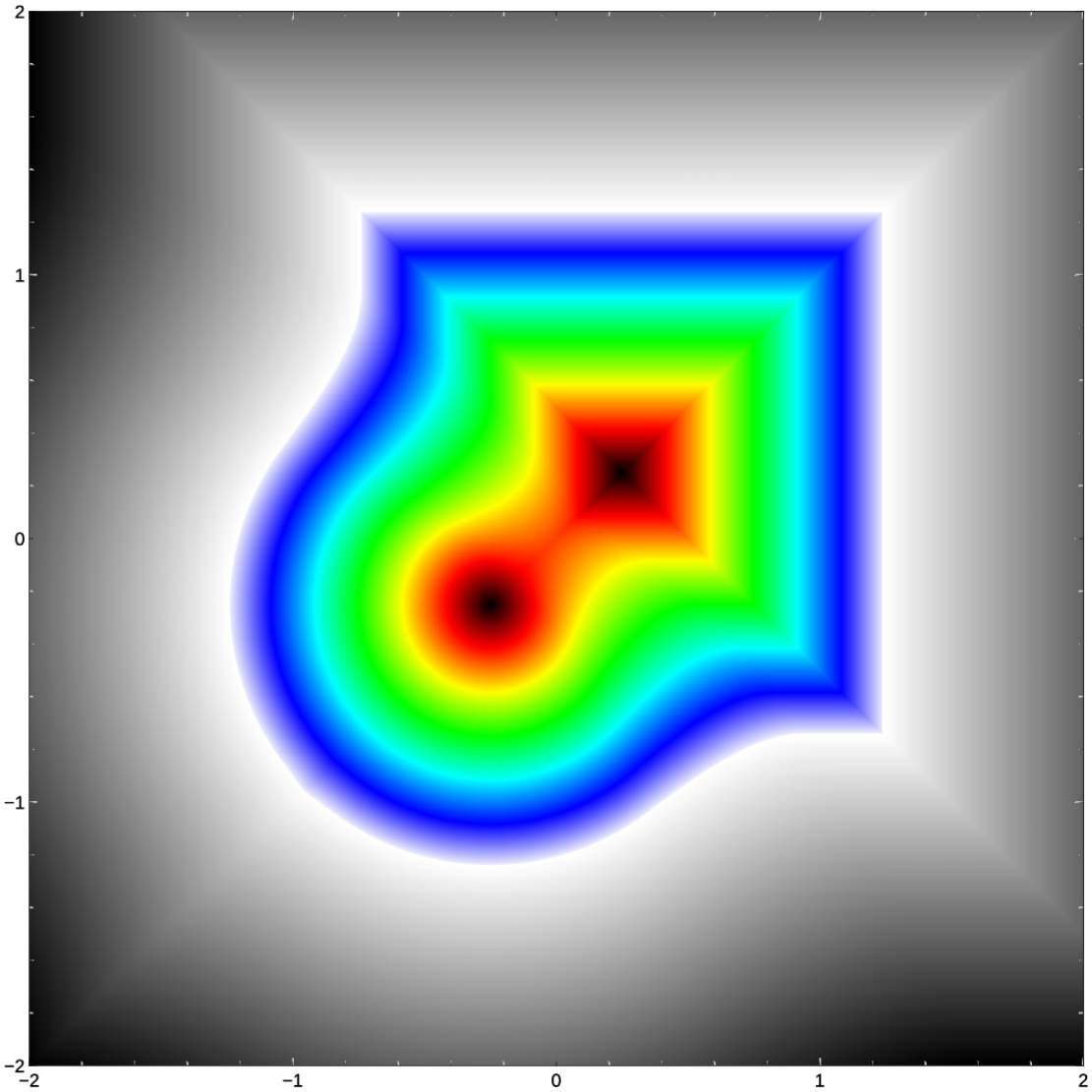
Getting the Difference

- Taking the maximum of the item to be removed from, and the inverse of the items being removed
- $maximum(sqrt((x+0.25)*(x+0.25)+(y+0.25)*(y+0.25))-1, -(maximum(abs(x-0.25), abs(y-0.25)))-1)$



Rounded Unions

$$\mathrm{rmin}_r(a,b) = \begin{cases} \min(a,b) & |a-b| \geq r \\ b + r * \sin(\pi/4 + \mathrm{asin}(\frac{a-b}{r\sqrt{2}})) - r & |a-b| < r \end{cases}$$



Implicit CSG Advantages

- Easy Rounding

- $square(x = 30, y = 30, r = 5)$

- $square(x = 30, y = 30, r = 15)$ == Circle!
- Rendering can be matched to your ability to print

Implicit CSG Disadvantages

- Eats CPU
- Must track render area, as well as function stack
  - Tighter boxes mean less CPU wasted

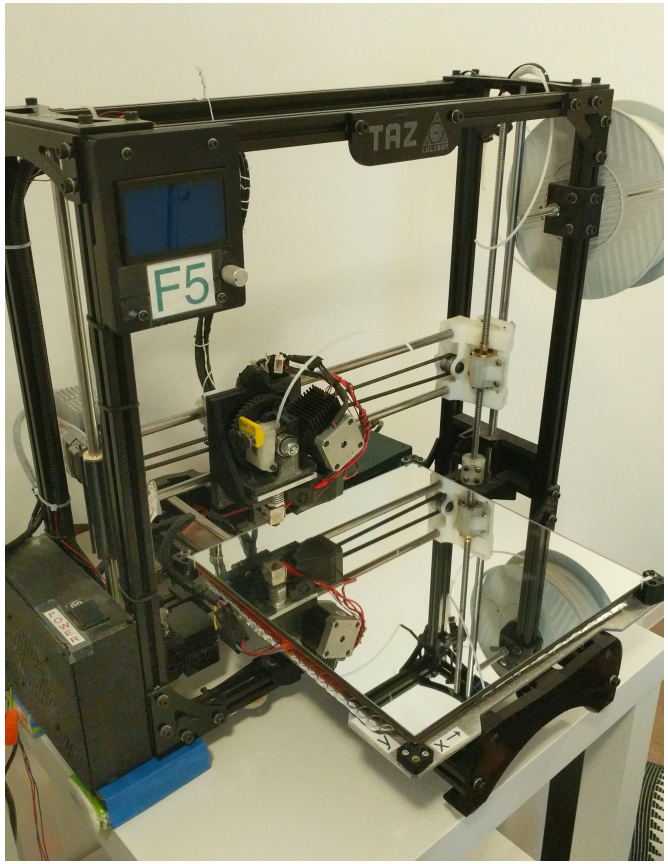
Haskell Implications

- Can generate and use functions
  - Hard to reason about them
  - Cannot print in intermediate form dump

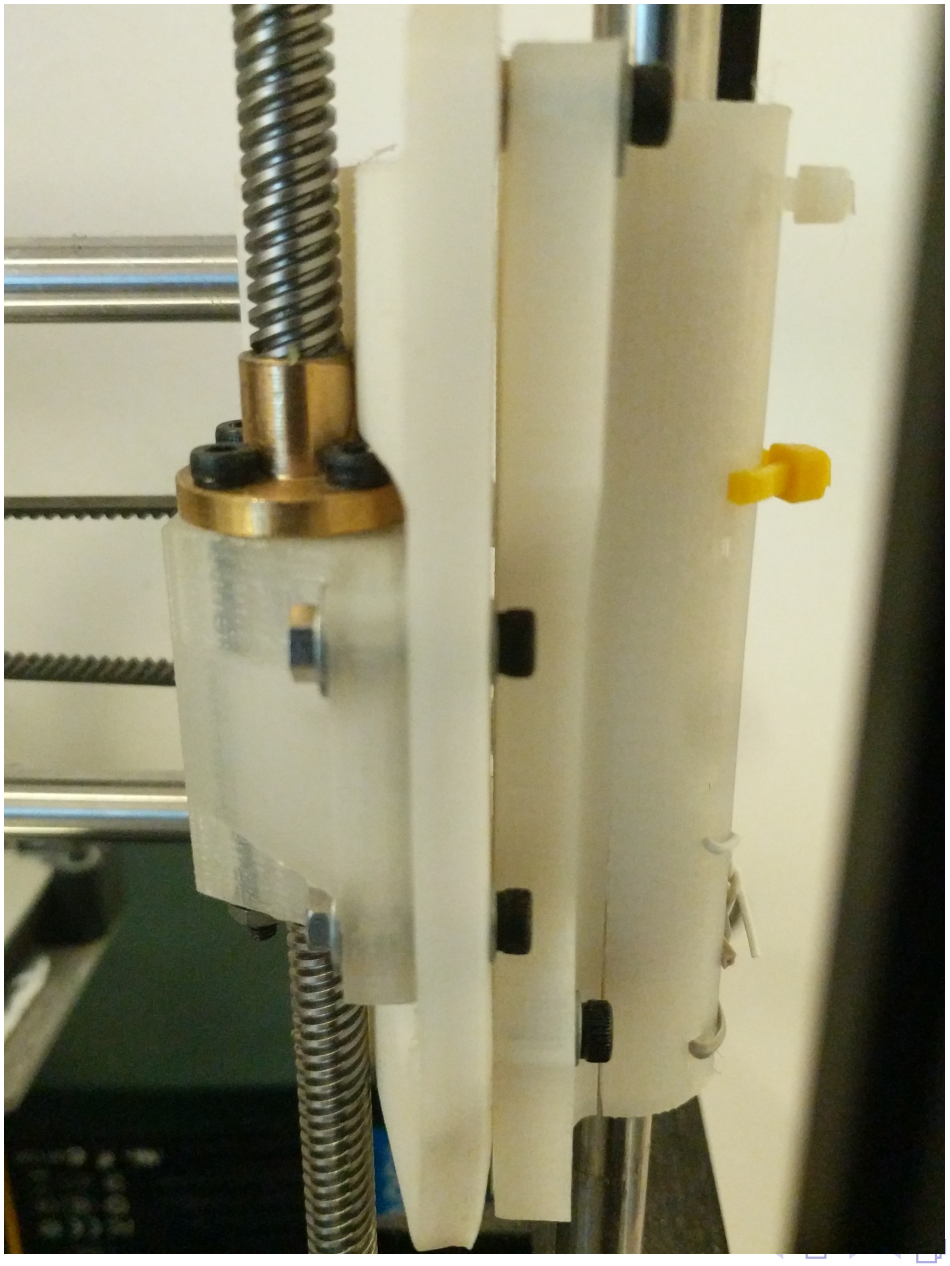
```
linear_extrude (height = 40, twist(h) = 35*cos(h*2*pi/60))
union ( r = 8) {
  circle (10);
  translate ([22,0]) circle (10);
  translate ([0,22]) circle (10);
  translate ([-22,0]) circle (10);
  translate ([0,-22]) circle (10);
}
```

Real World

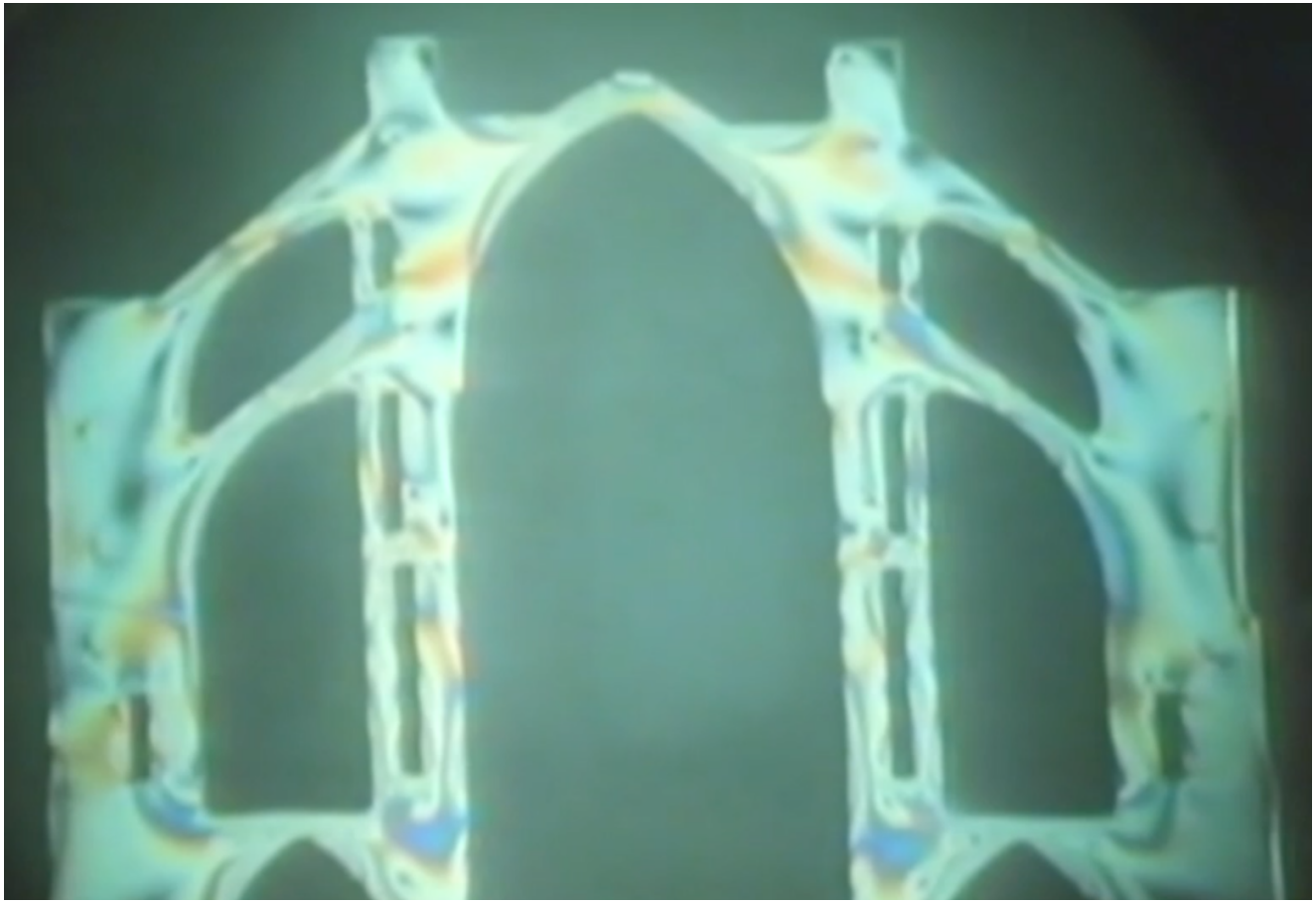
- My 3D Printer
  - Lulzbot Taz 3-5ish
  - Re-printing with ImplicitCAD
  - STLs provided by Lulzbot
    - STLs are not source code!



Cracking Parts



Stress under Pressure



Future ImplicitCAD Functionality

- Functions as a type, so they can be reasoned about by the engine
- User-defined implicit operations

```
raw3D(bbox=function_that_returns_two_points,
      implicit(x,y,z)=function_that_returns_a_value);
```

Fin

Questions?