Computational Design + Fabrication: Geometry

Jonathan Bachrach

EECS UC Berkeley

September 1, 2015

Today

- who are you?
- trigonometry
- geometry
- solid geometry
- tools

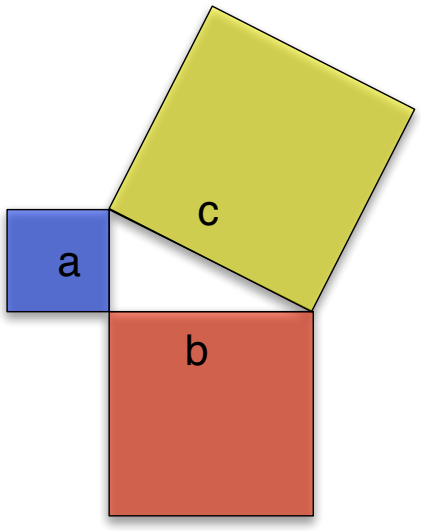
Who Are You?

- name
- undergraduate or graduate
- major
- python programming?
- courses taken (PL, graphics, geometry)
- cad tools?
- proficient in any shop tools?
- cnc tools?
- favorite project most proud of
- why taking class?

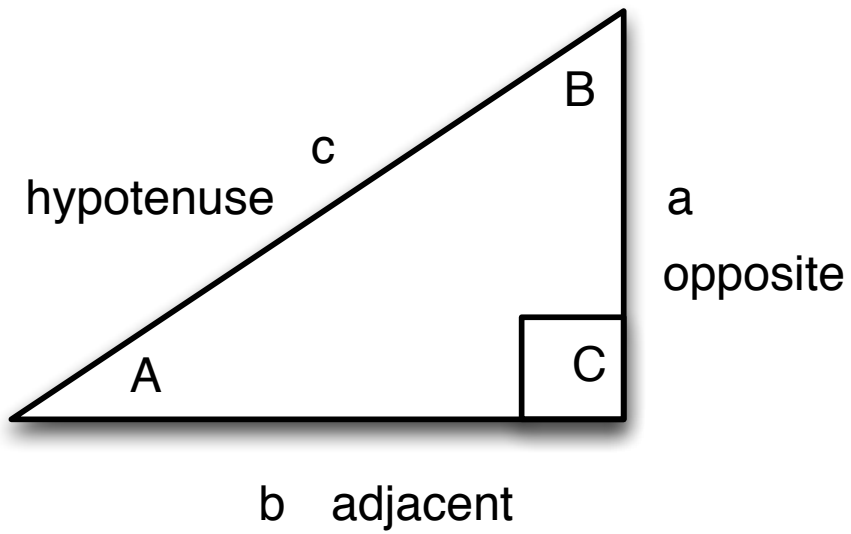
- the future of design by lipson et al
- questions

- right angled triangle

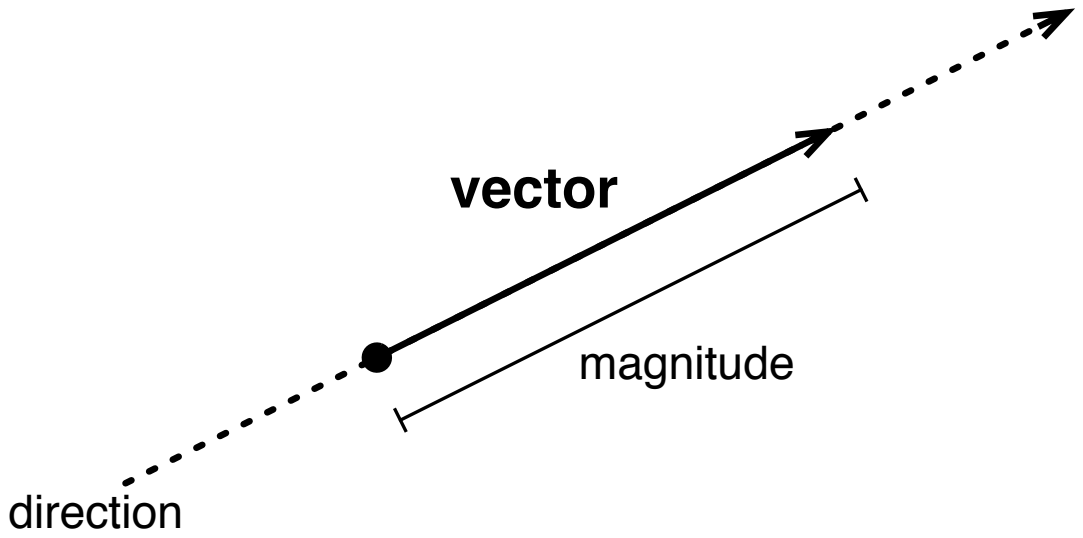
$$a^2 + b^2 = c^2 \tag{1}$$



- $sinA = \frac{opposite}{hypotenuse} = \frac{a}{c}$
- $cosA = \frac{adjacent}{hypotenuse} = \frac{b}{c}$
- $tanA = \frac{opposite}{adjacent} = \frac{a}{b} = \frac{sinA}{cosA}$



- $v = [a, b]$
- magnitude = $|v| = \sqrt{a^2 + b^2}$
- direction

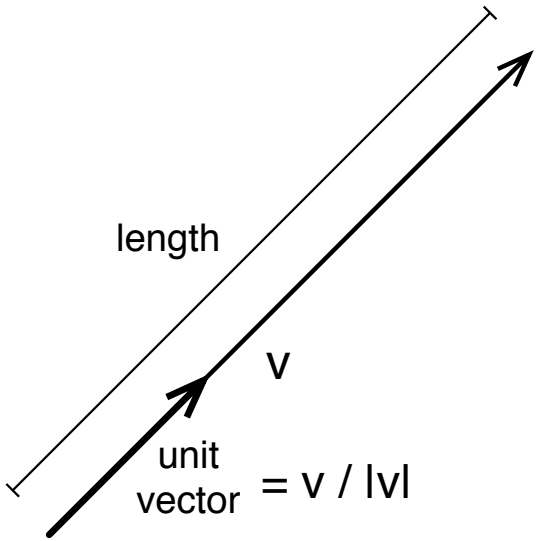


Vector Ops

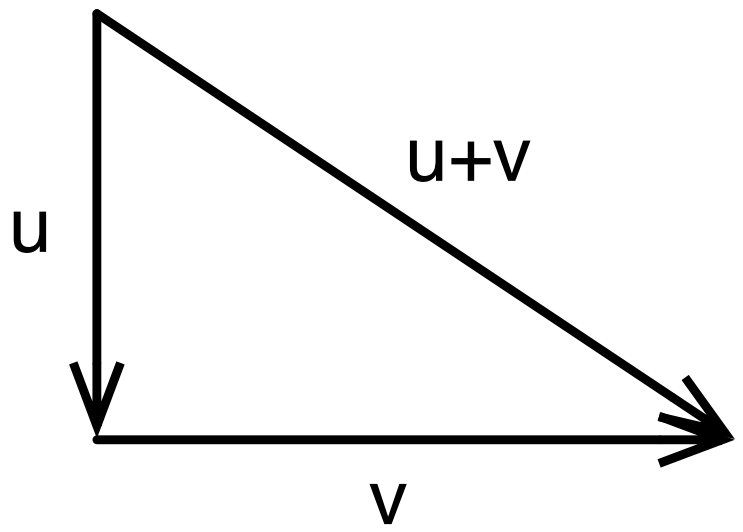
- normalize
- add
- scale

Vector Normalize

- unit vector
- $v = v/|v|$

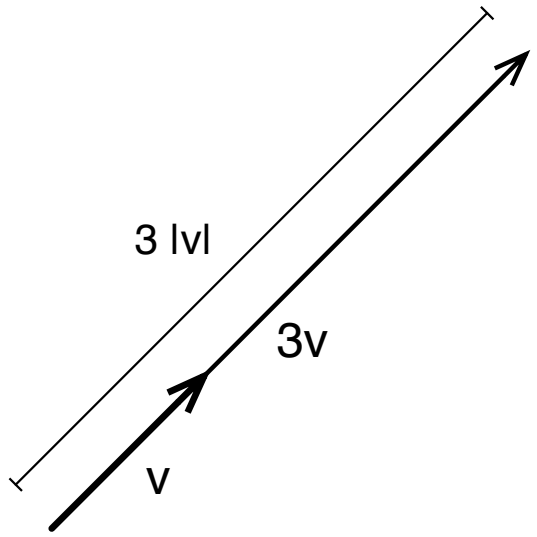


- $w = u + v$

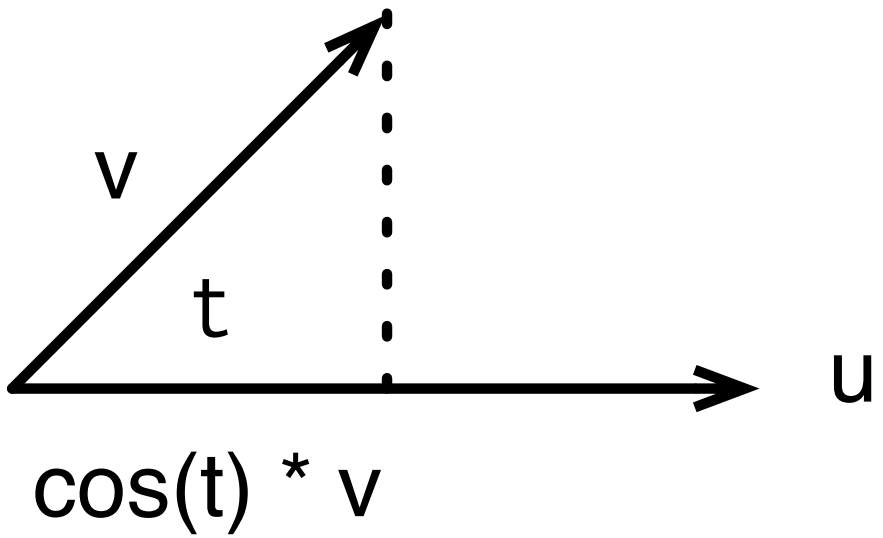


Vector Scale

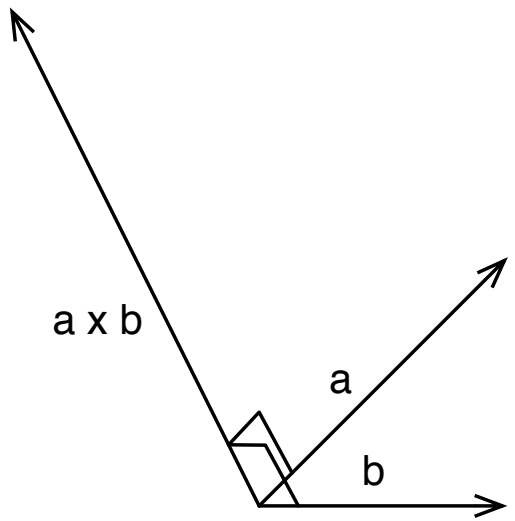
- $w = s * v$



- $w = u \cdot v$
- $u \cdot v = |u| \times |v| \times \cos\theta$
- dot product of vectors at right angle is zero

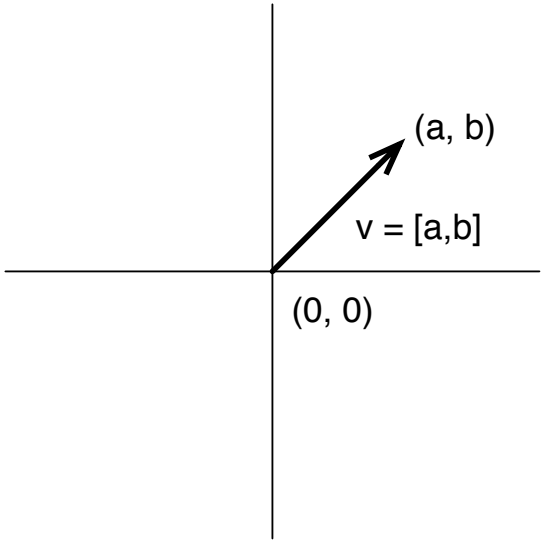


- $w = u \times v$
- $u \times v = |u||v| \sin\theta n$
- right hand rule: $u =$ index finger, $v =$ middle finger, $w =$ thumb

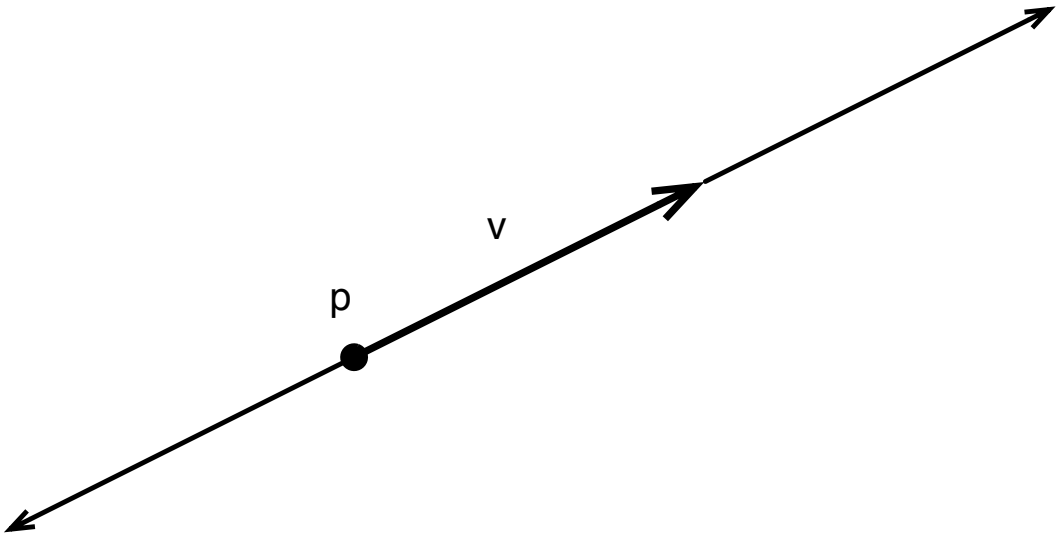


- polar to cartesian: $v = r[\cos\theta, \sin\theta]$
- cartesian to polar: $[r, \theta] = [norm(v), atan2(v)]$

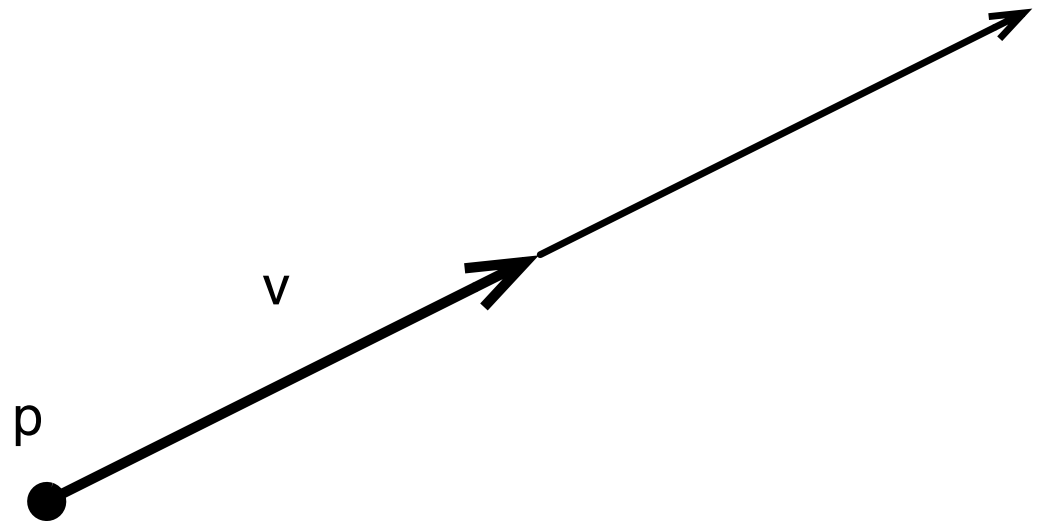
- interpreted as vector from zero
- subtracting points forms vector



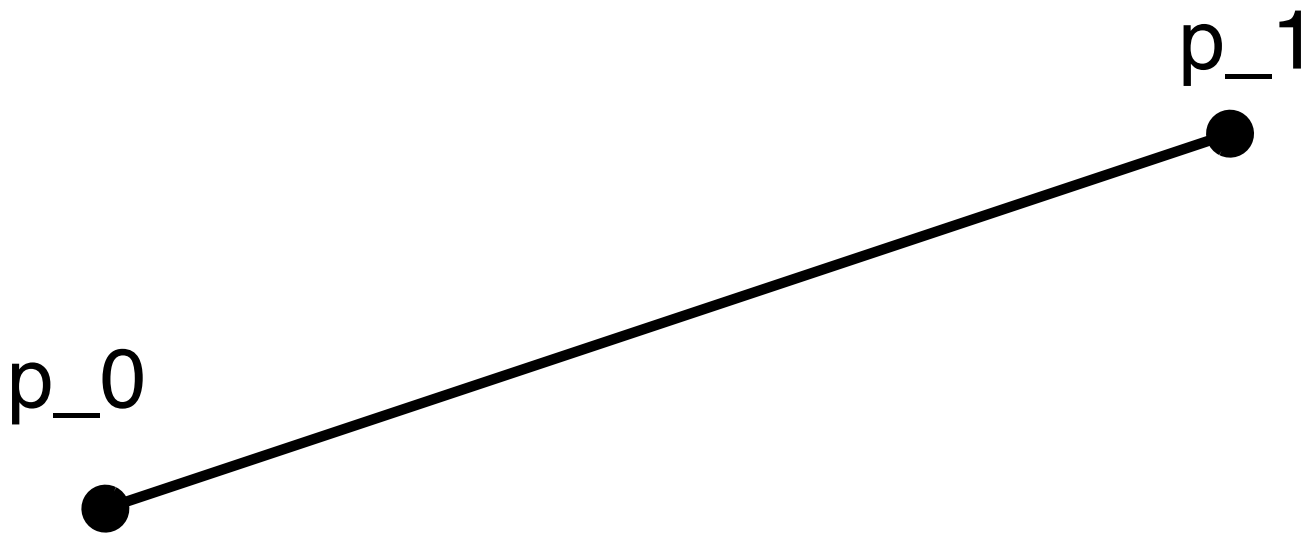
- infinite
- through two points
- parameterized by t



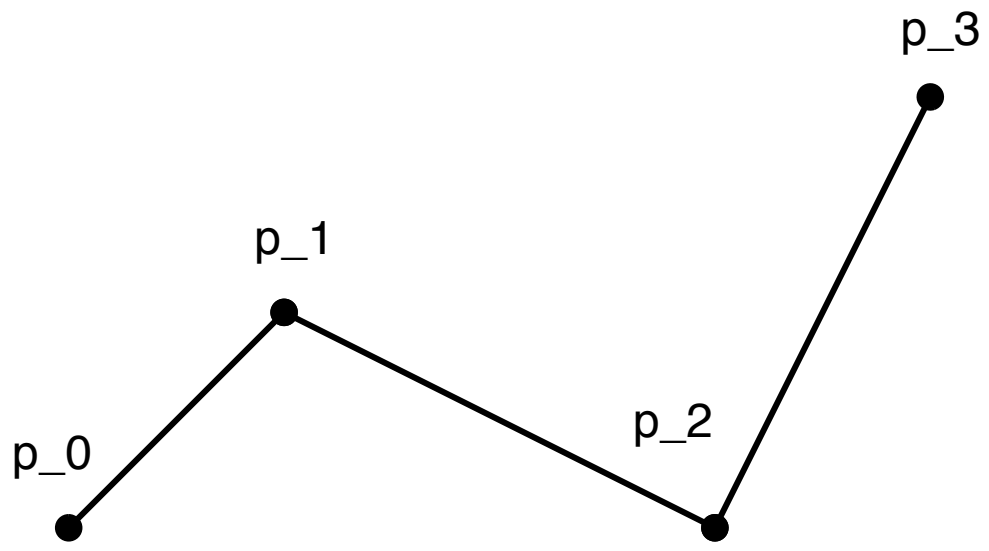
- one half of line bounded by point



- finite line bounded by points

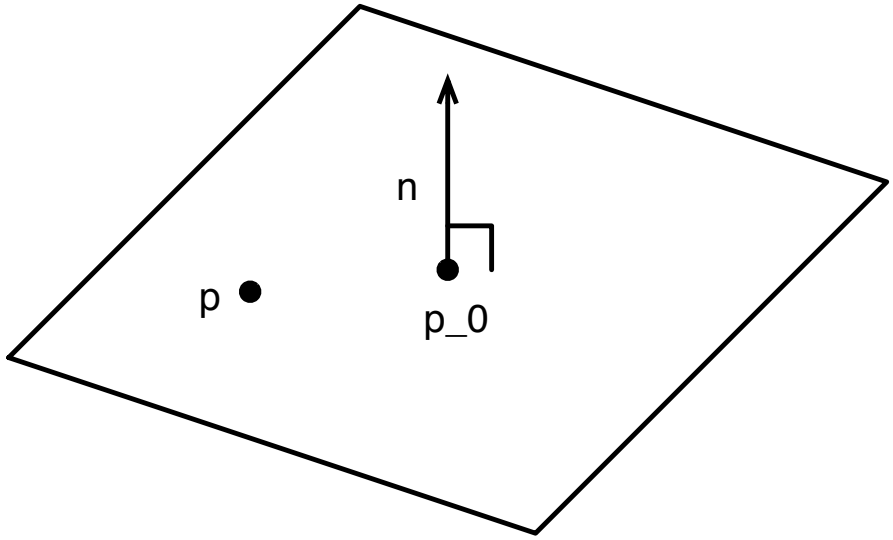


- collection of lines segments connecting points
- open and closed
- collection of polylines
- DXF, SVG

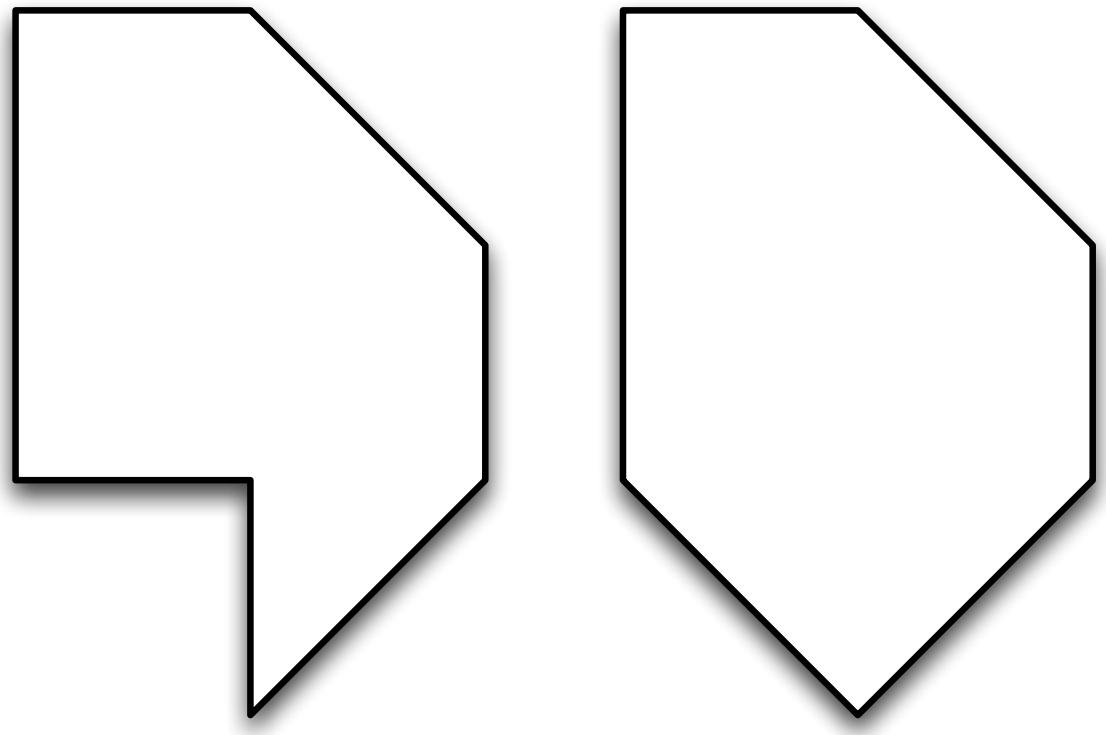


$$n \cdot (p - p_0) = 0 \tag{2}$$

- point p_0
- normal n
- dot product interpretation
- intersect line
- nearest point

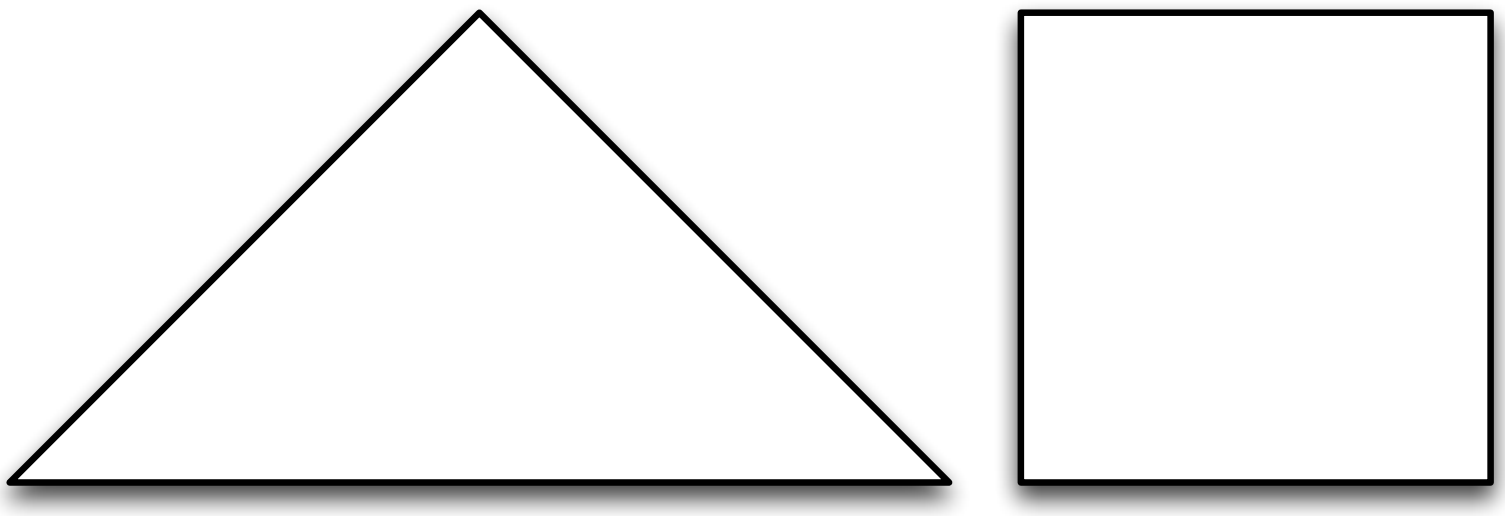


- closed polyline
- perimeter
- area



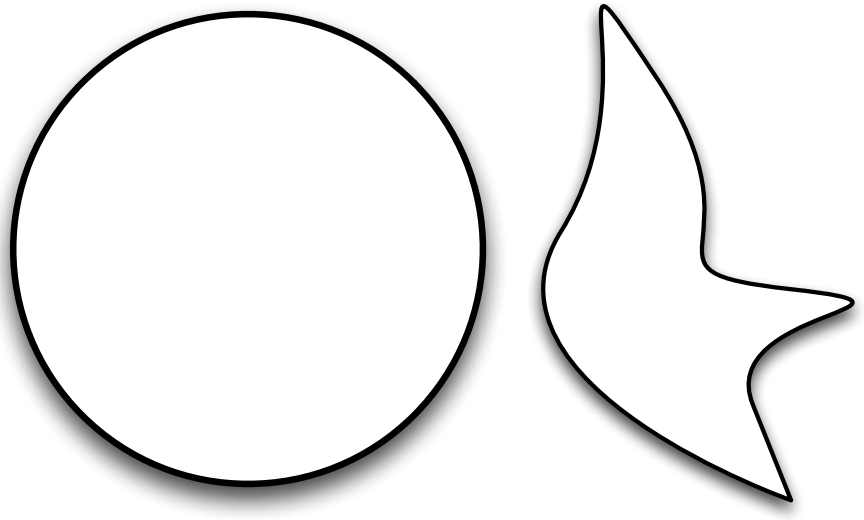
concave convex

- triangle
- square



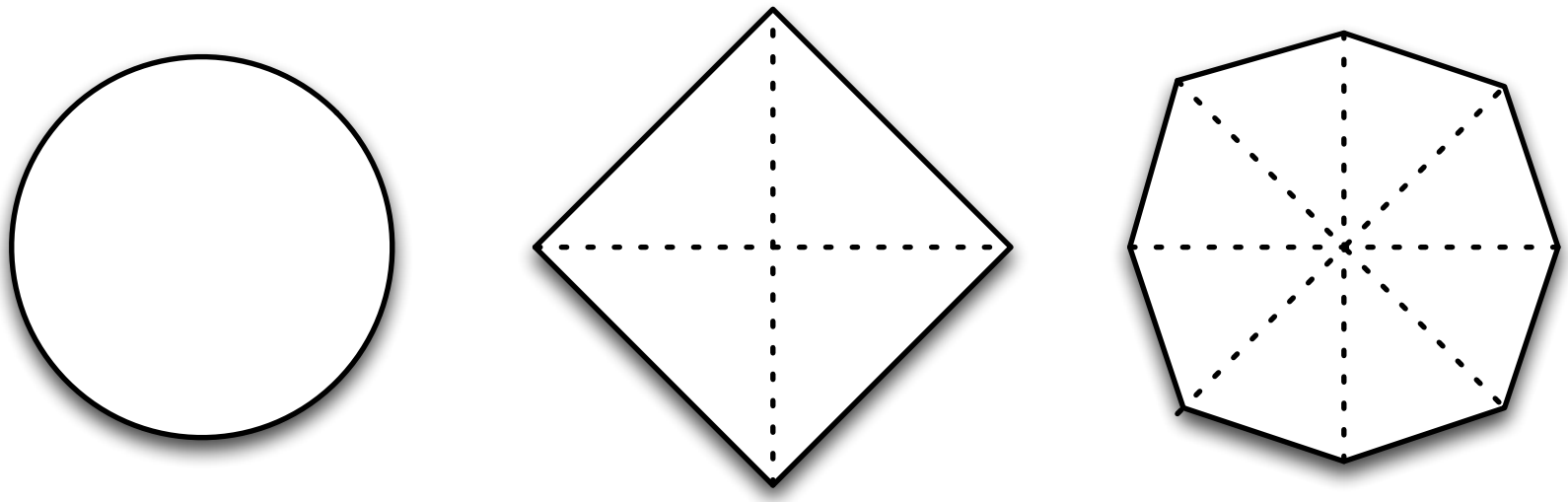
Non Polygons

- circle
- squiggly



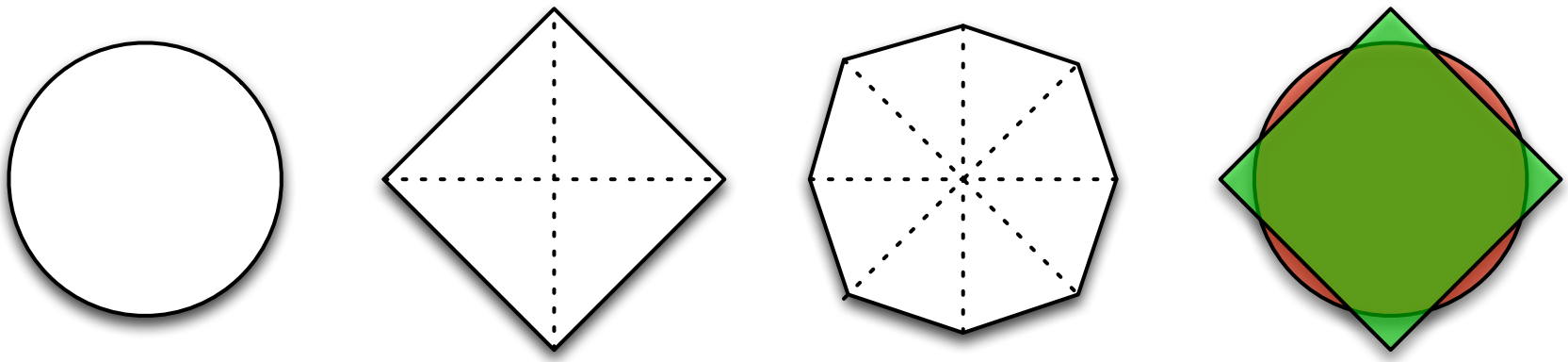
Approximating Non-Polygons by Polygons

- circle
- approximation error



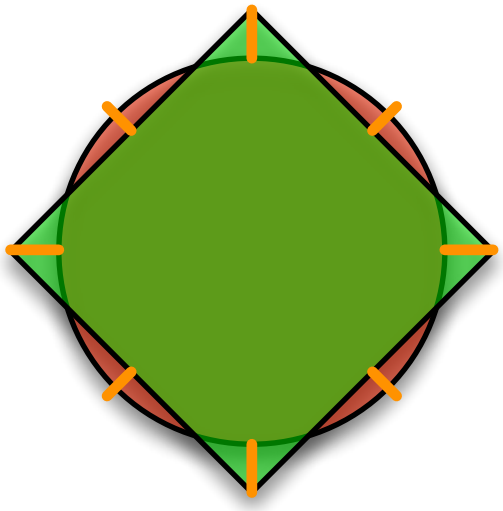
Approximation Error

- number of facets
- angle
- fragment size
- squared area delta



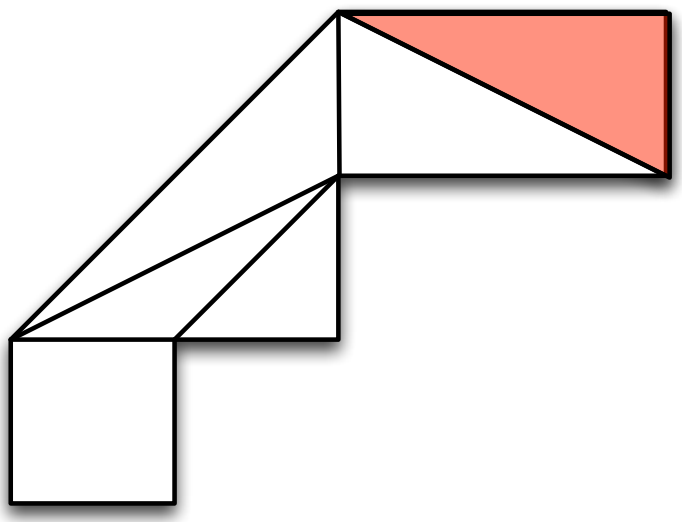
Hausdorff Distance

- two polygons are close if every point in one is close to the other
- max distance from one polygon to closest point in the other
- could be computed using sampling

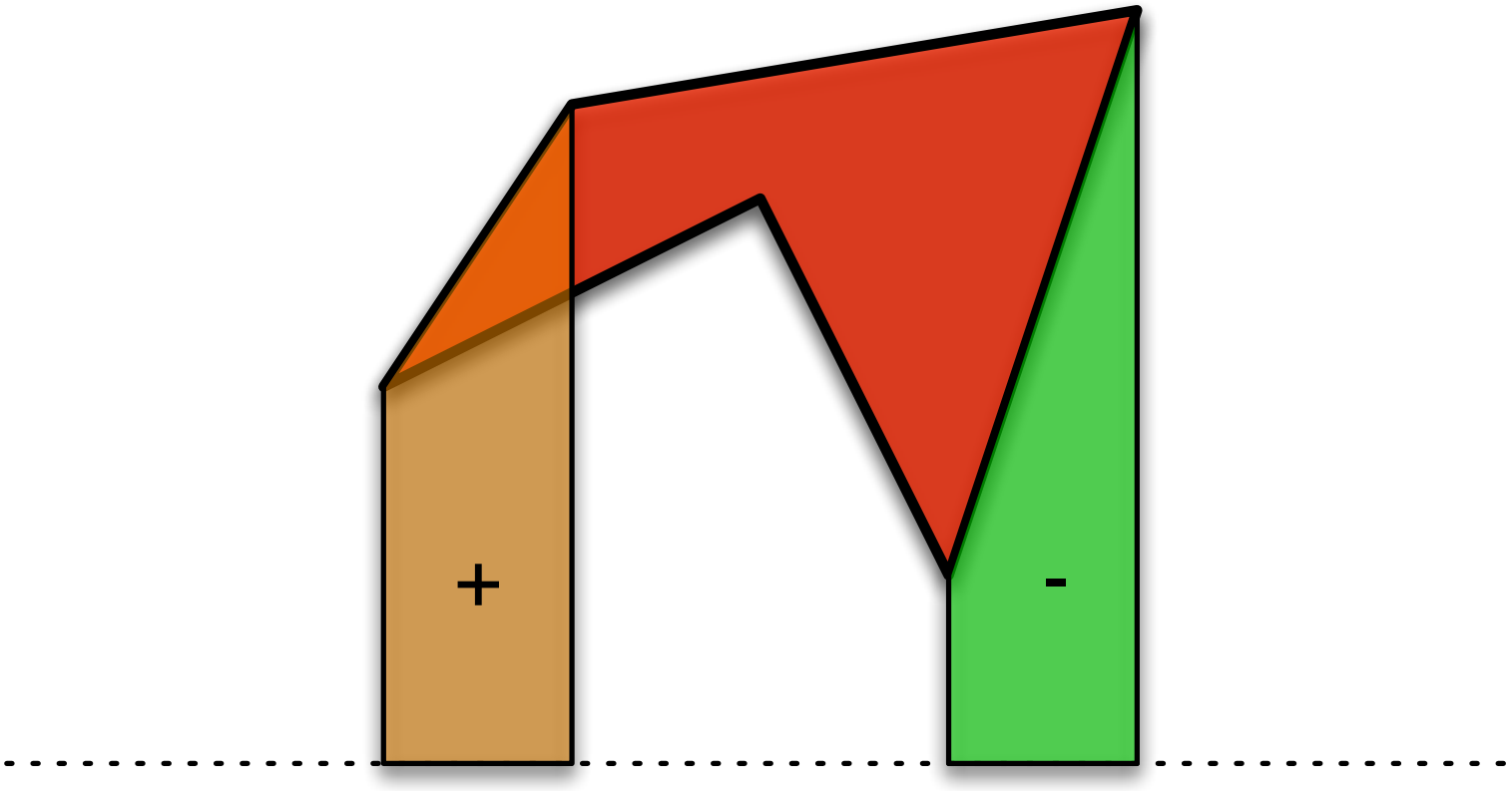


Triangulation of Polygons

- break polygon into neighboring triangles
- how do we do this?
- what do we care about in triangulating?
- ear clipping – triangle with one edge inside
- faster algorithms

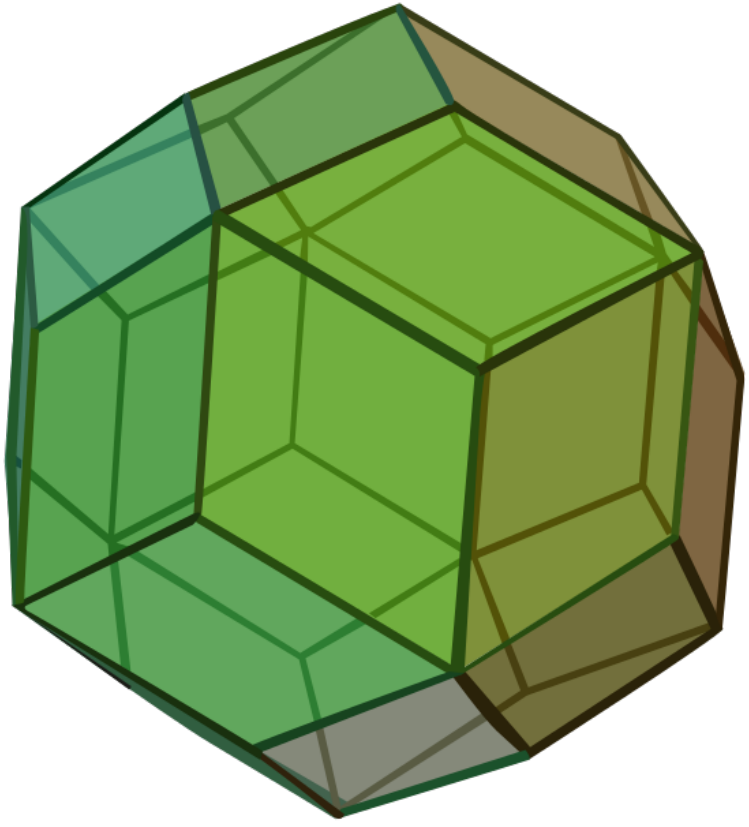


- clockwise
- sum area under successive points
- reverse sign if $x_{i+1} < x_i$
- how else?

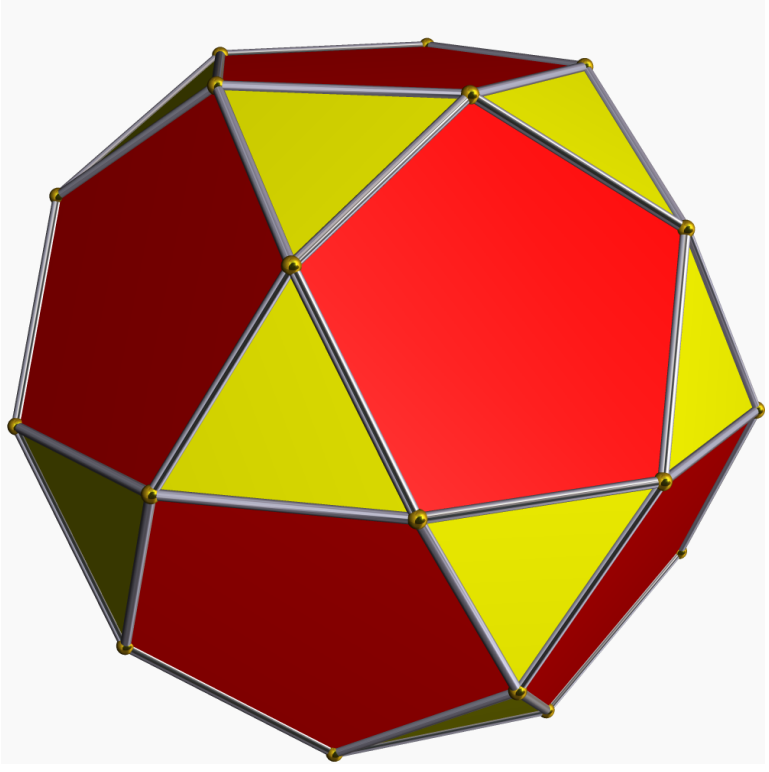


Polyhedra

- collection of intersecting planes
- neighboring polygons called faces forming closed
- how do we know its closed?

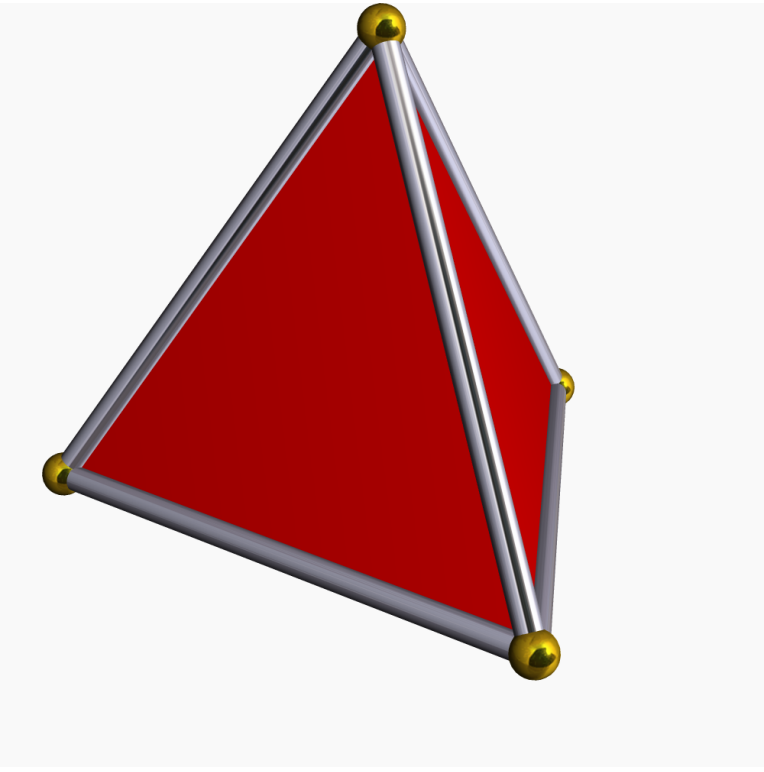


Rhombictriacontahedron by DTR

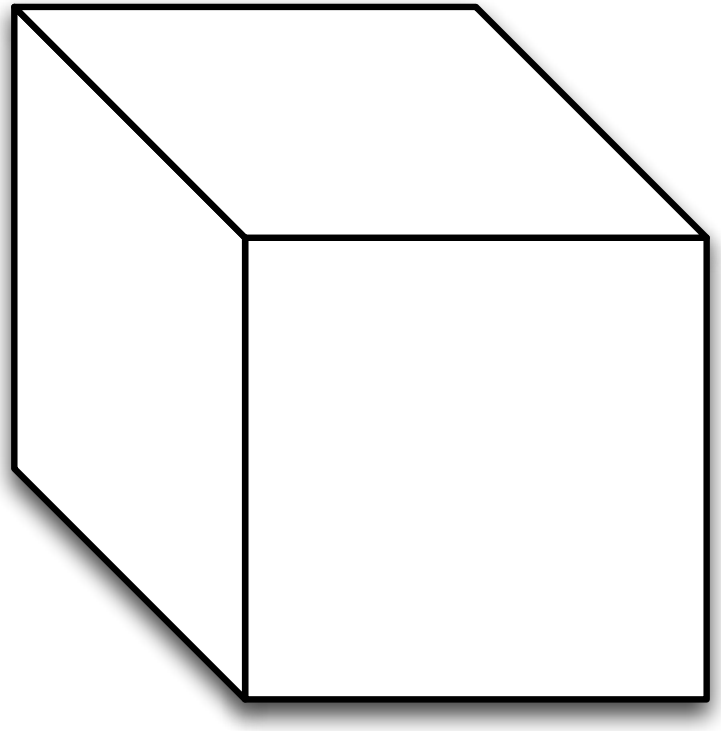


Tetrahedron by Robert Webb's Stella software

- tetrahedron
- cube

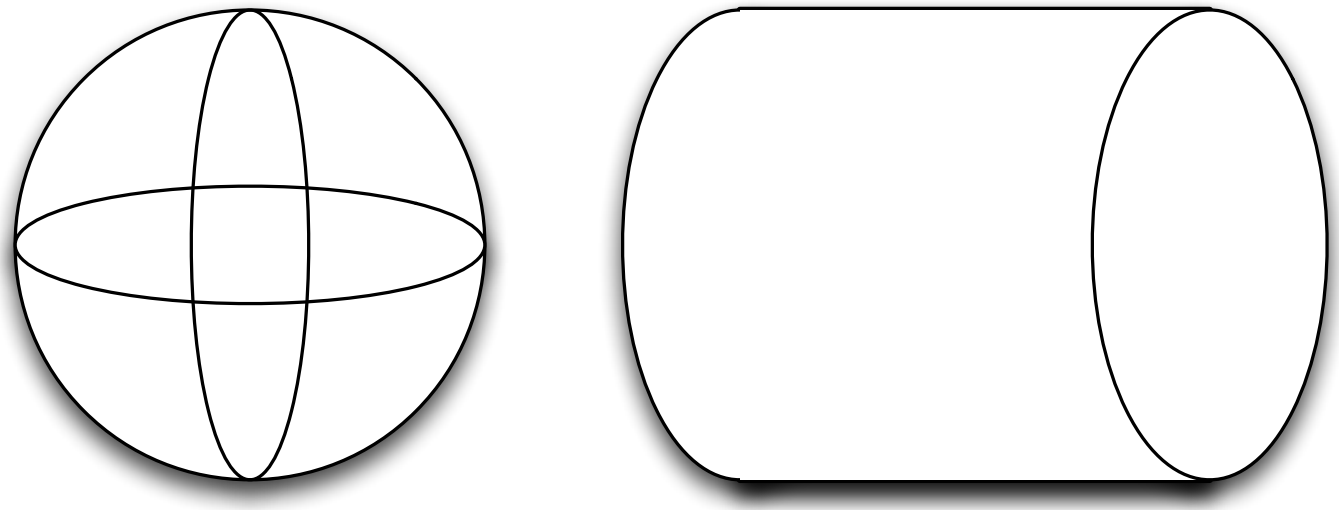


Tetrahedron by Robert Webb's Stella software

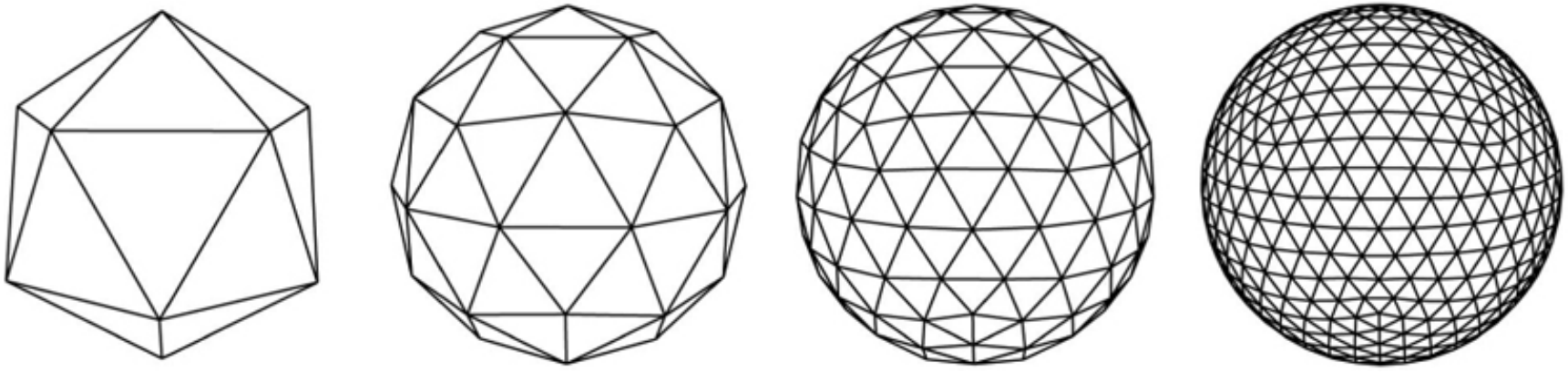


Non Polyhedra

- sphere
- cylinder



- sphere
- cylinder
- approximation error
- advantages?
- problems?

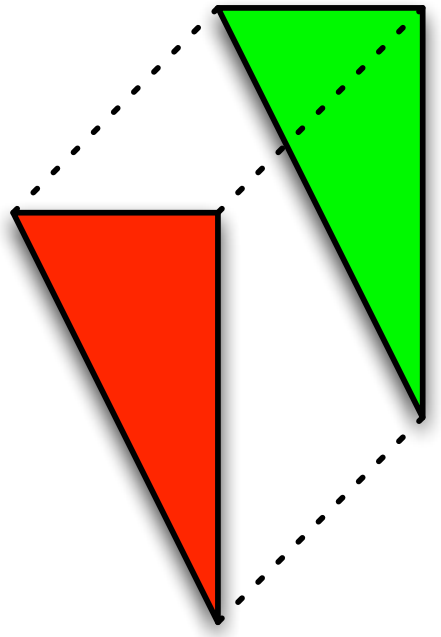


Planetary-Scale Terrain Composition – Kooima + Leigh + Johnson + Roberts + SubbaRao + DeFanti

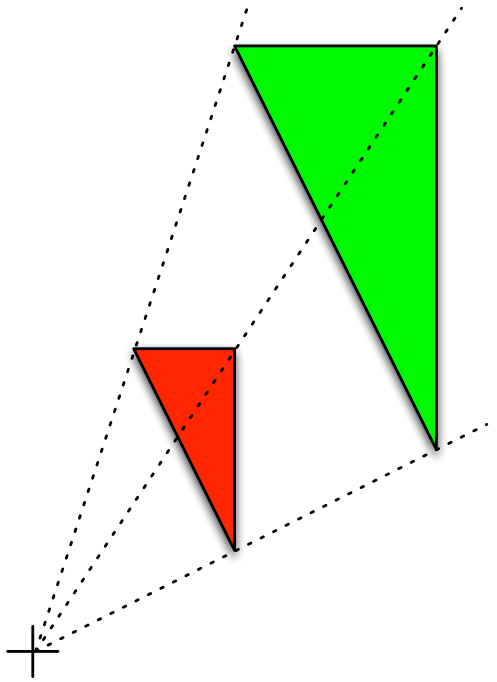
- stl: faces with redundant points
- obj: points faces with point indices
- (winged mesh)

- translate
- scale
- rotate
- (shear)

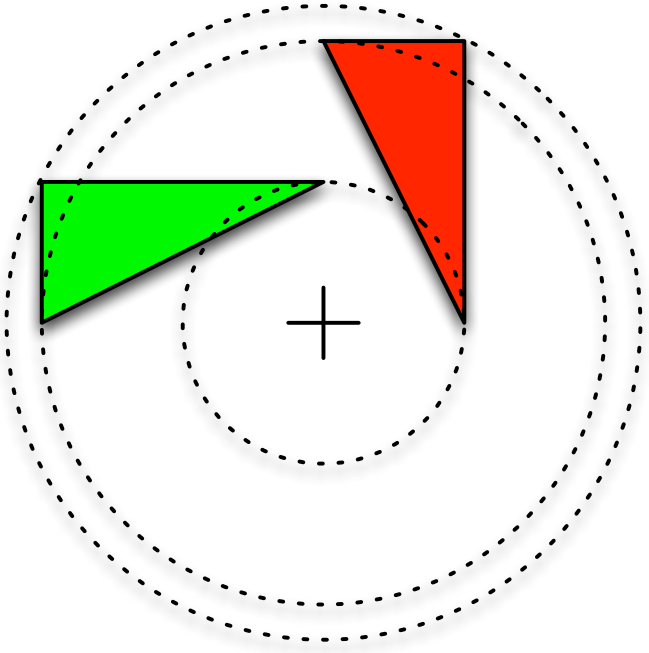
- translation vector



- scaling from origin



- rotation about origin



Affine Transformation Matrices

- 2x2
- 3x3
- basic transformations
- composition
- extends to 3D

$$\begin{vmatrix} x_2 \\ y_2 \end{vmatrix} = A \times \begin{vmatrix} x_1 \\ y_1 \end{vmatrix} + B \tag{3}$$

$$A = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}, B = \begin{vmatrix} 0 \\ 0 \end{vmatrix} \tag{4}$$

$$A = \begin{vmatrix} s_x & 0 \\ 0 & s_y \end{vmatrix}, B = \begin{vmatrix} 0 \\ 0 \end{vmatrix} \tag{5}$$

$$A = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}, B = \begin{vmatrix} 0 \\ 0 \end{vmatrix} \tag{6}$$

$$A = \begin{vmatrix} 0 & 0 \\ 0 & 0 \end{vmatrix}, B = \begin{vmatrix} t_x \\ t_y \end{vmatrix} \tag{7}$$

- can write entire affine transformation in one matrix
- can compute inverse of transformation
- augment input vector with 1

$$\begin{vmatrix} x_2 \\ y_2 \\ 1 \end{vmatrix} = \begin{vmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ y_1 \\ 1 \end{vmatrix} \tag{8}$$

$$v_2 = M \times v_1 \tag{9}$$

- now can compose transformations using matrix multiplication

$$M_A = M_R \times M_T \tag{10}$$

$$\begin{vmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \times \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix} \tag{11}$$

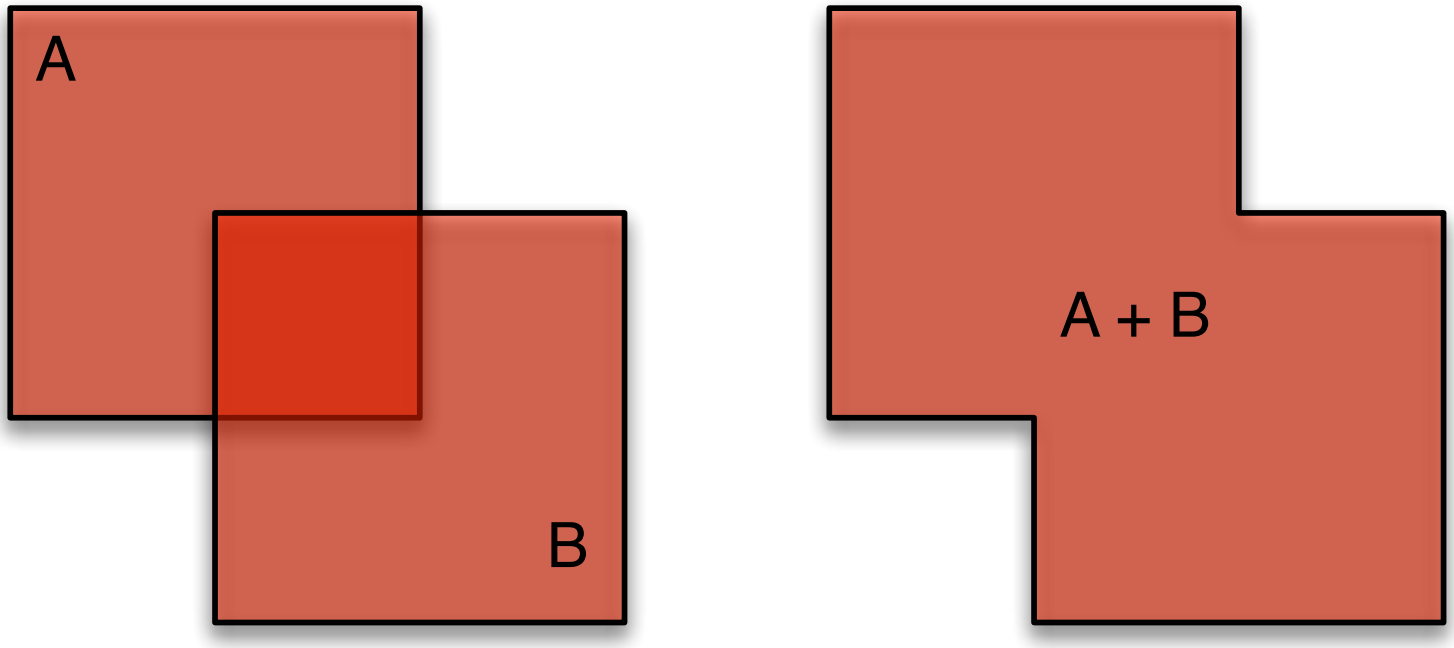
- 3D shapes with volume
- set ops
- offset, convex hull, minkowski sum
- 2D to 3D
- 3D to 2D

Set Ops

- interpretation of solids as sets of points
- ops: union, intersection, difference

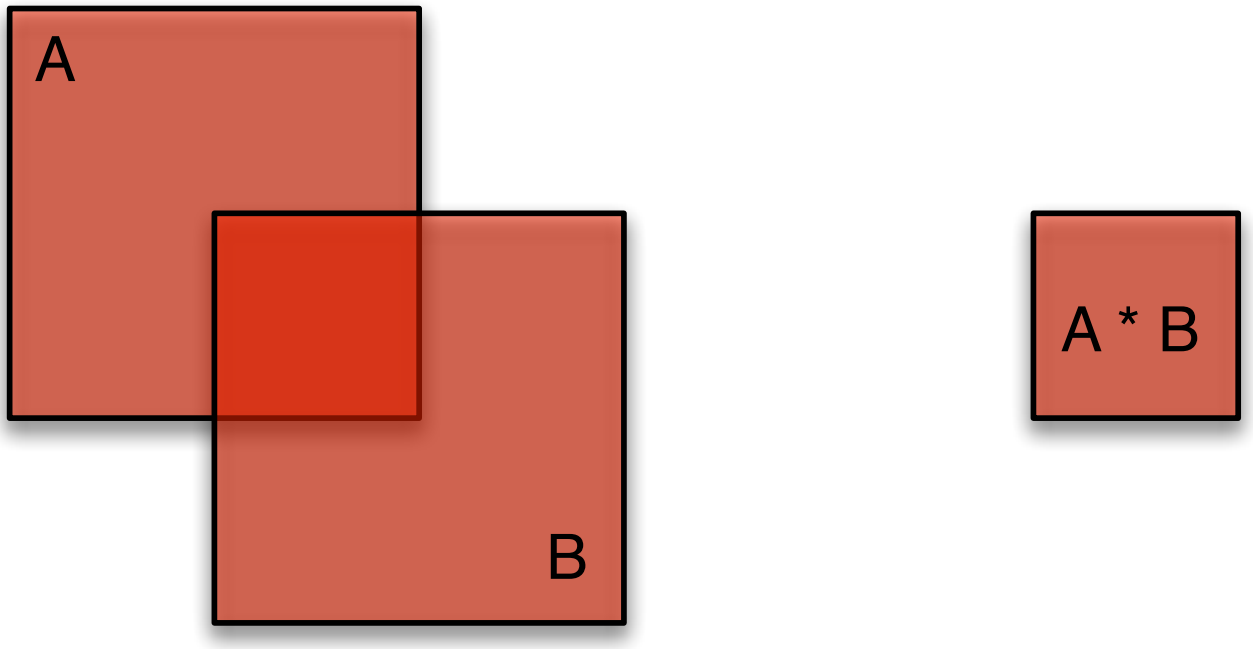
Union

- sometimes written using addition



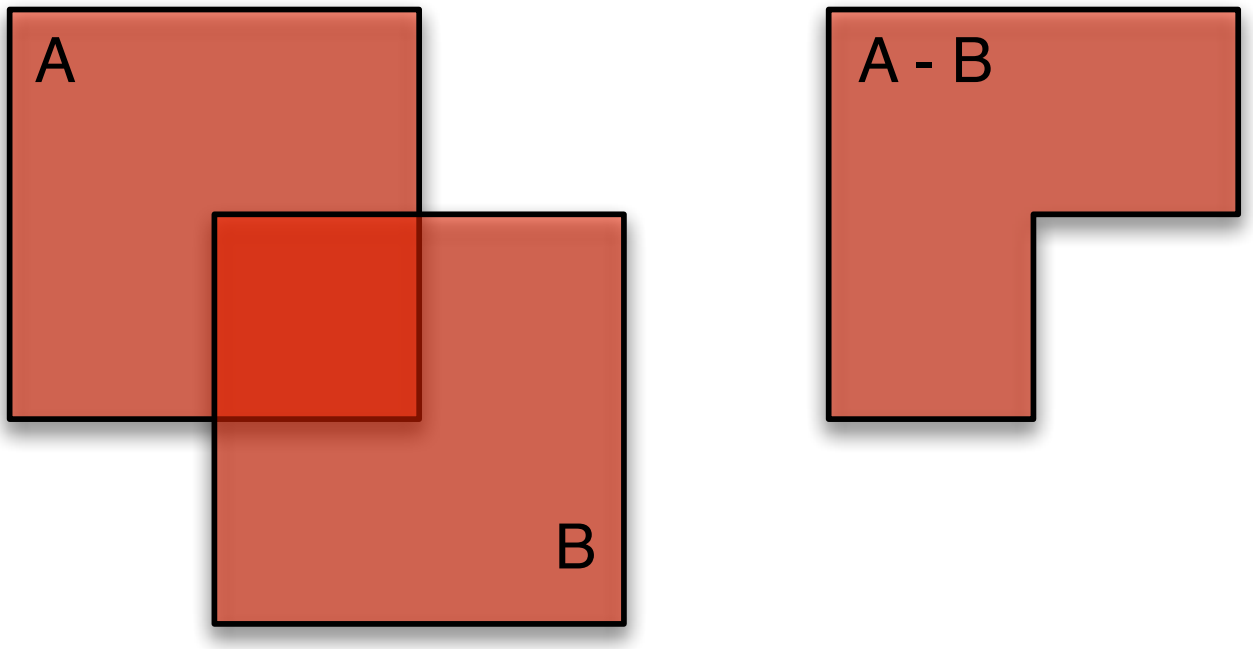
Intersection

- sometimes written using multiplication



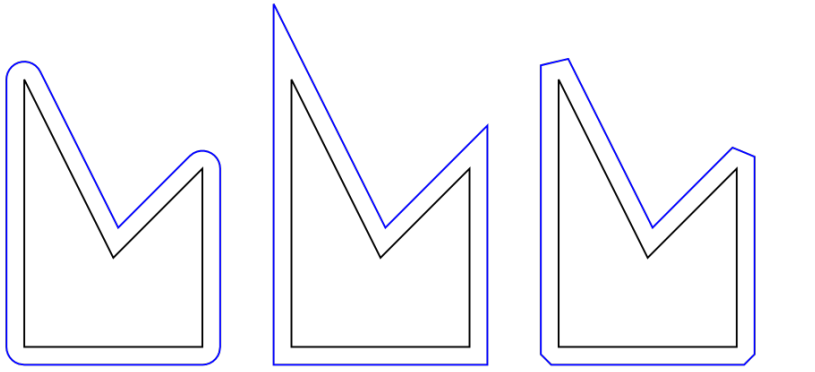
Difference

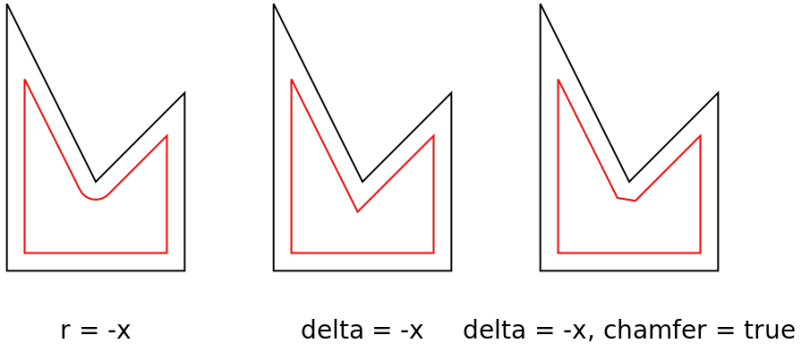
- sometimes written using subtraction



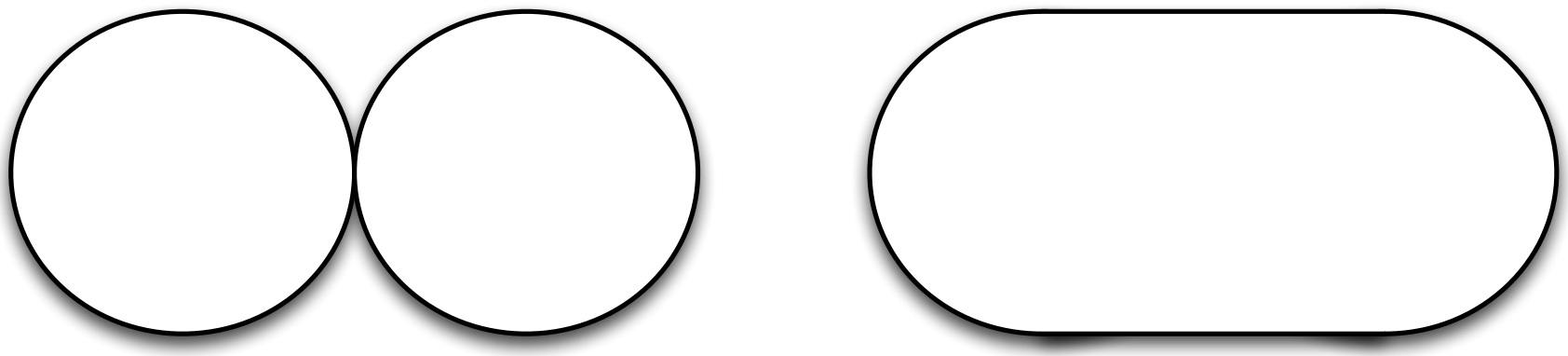
Offset

- amount
- chamfer
- how to implement?

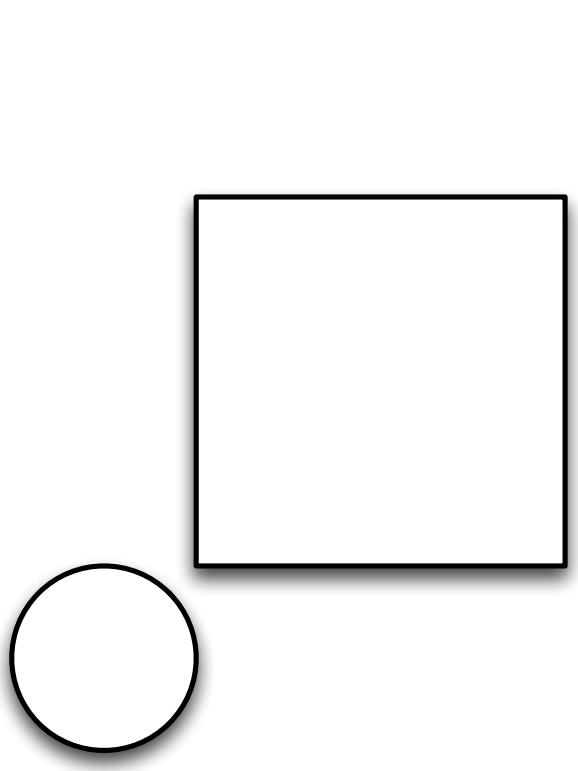


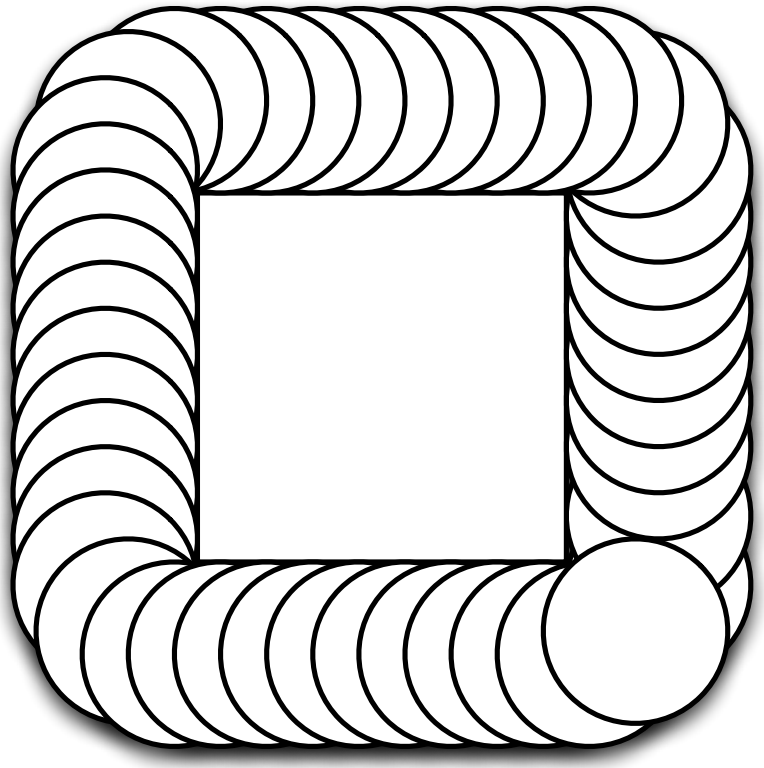


- definition
- arguments
- examples



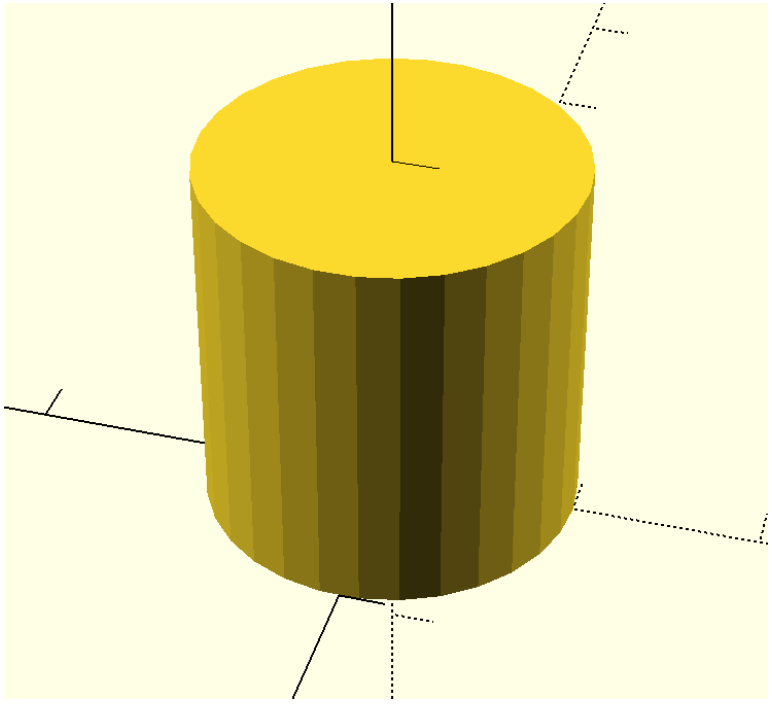
- definition
- arguments
- examples

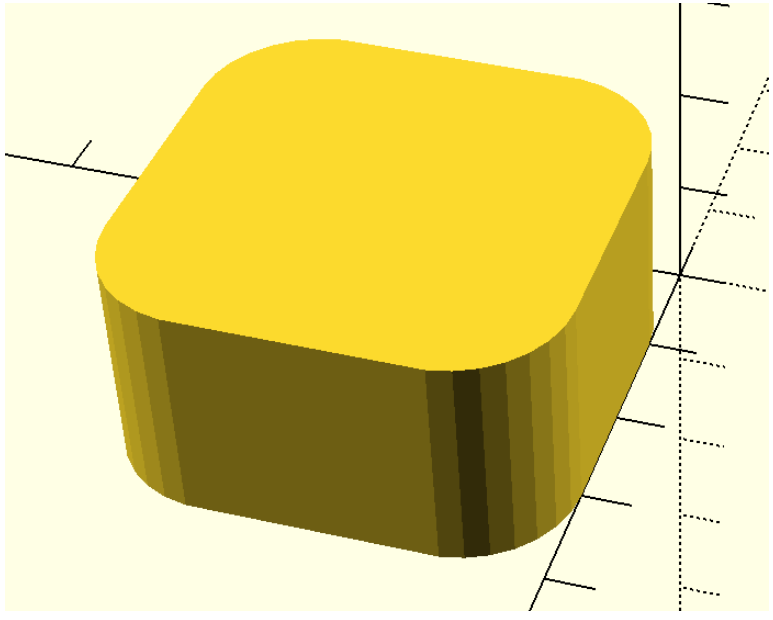




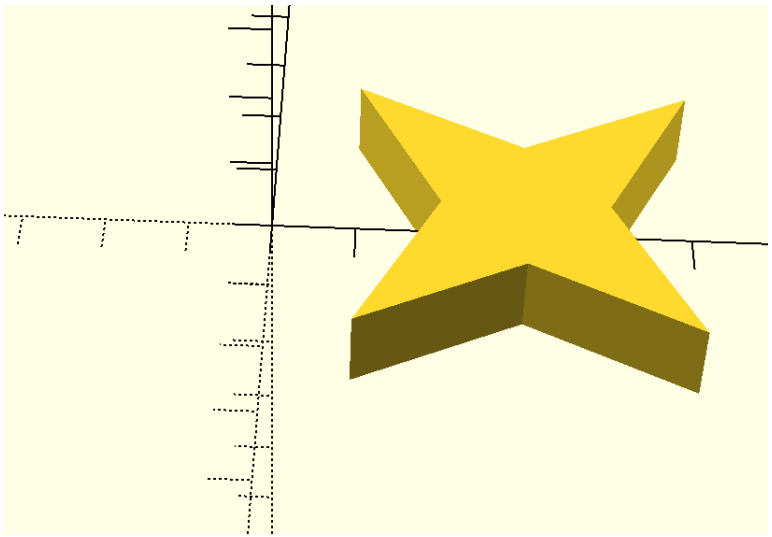
- Extrusion
- Lathe

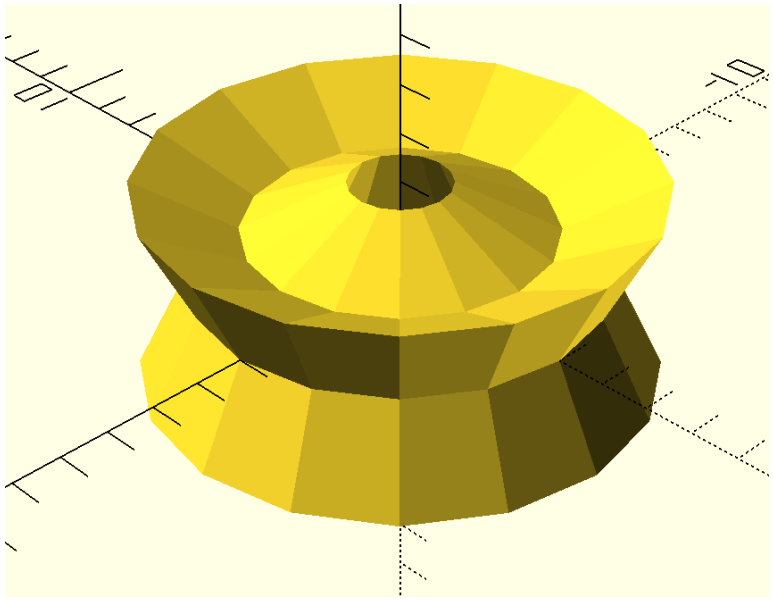
- height
- scala
- rotate



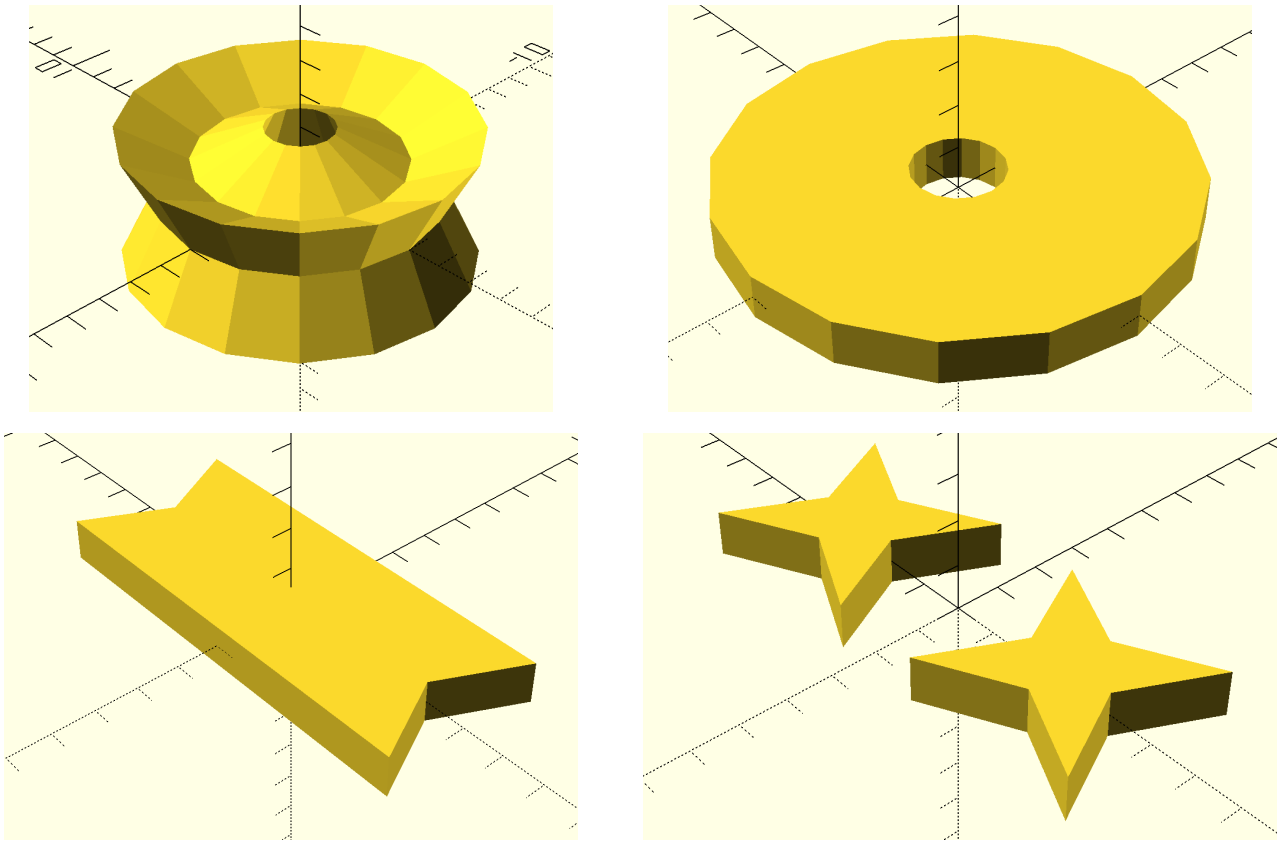


- must translate off center



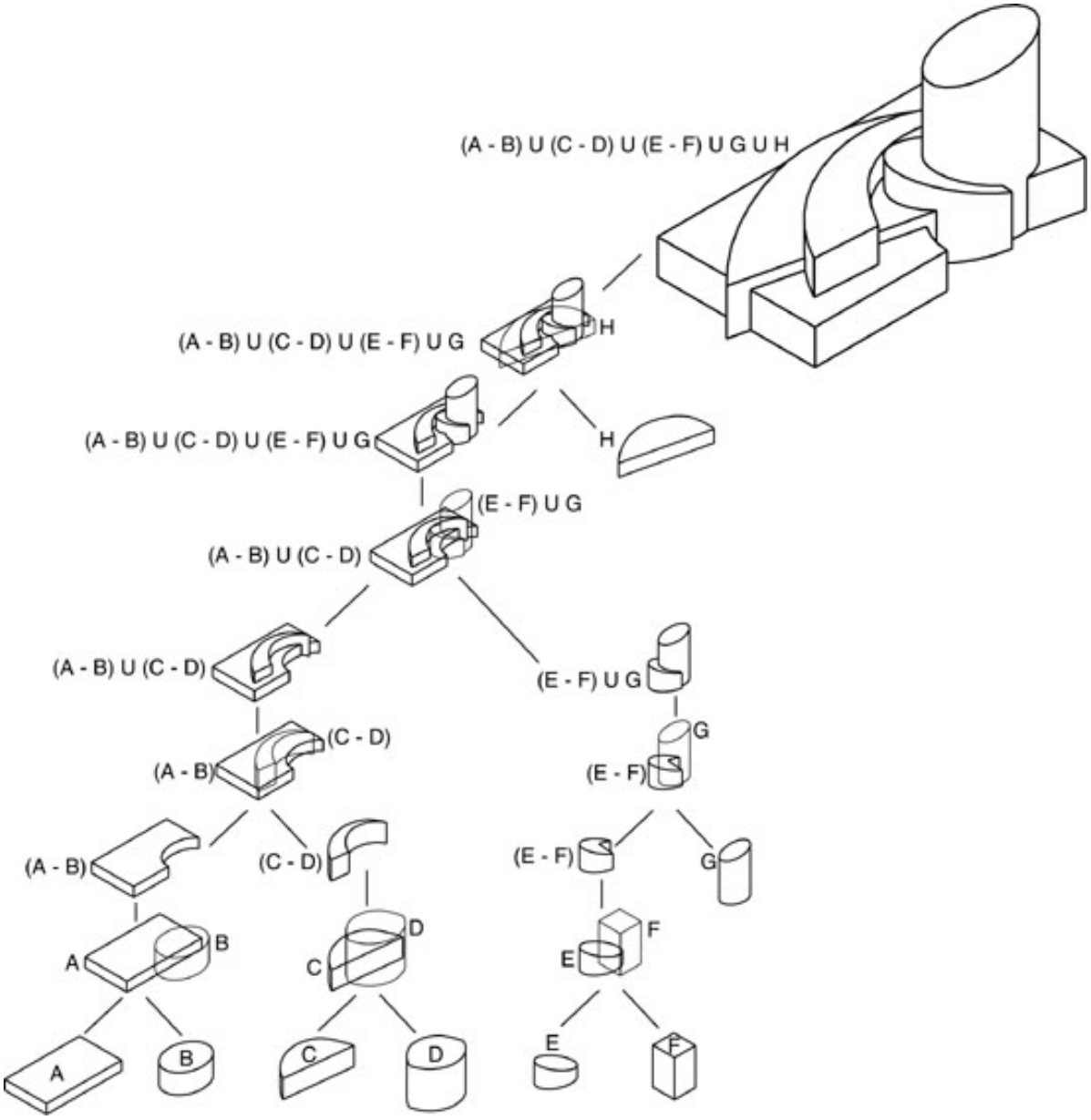


- shadow
- slice

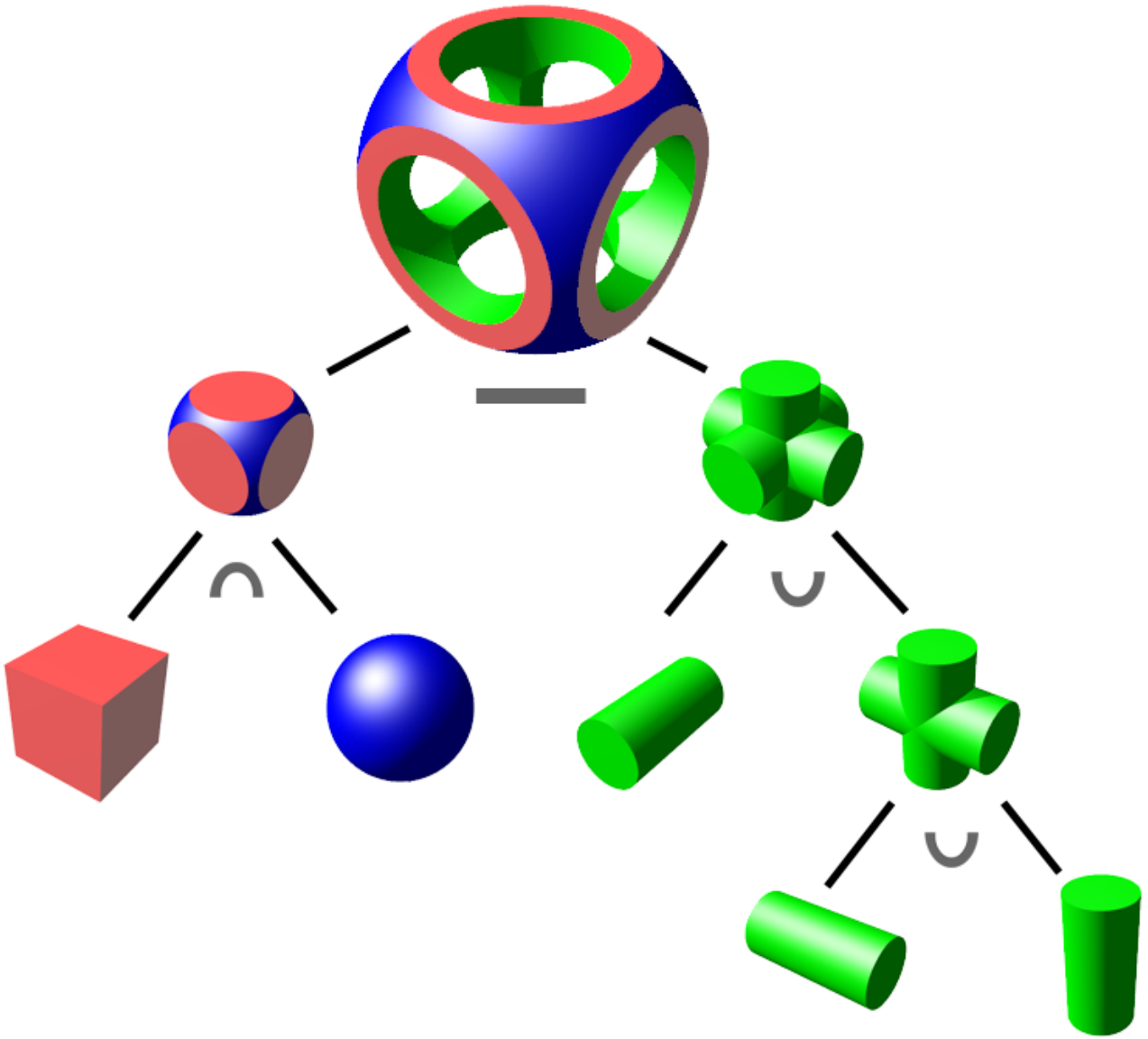


picture by N.Goodger

CSG Hierarchy



chang + woodbury



- outputs stl
- procedural
- based on CGAL nef polyhedra

OpenSCAD CheatSheet v2014.03

Syntax

```
var = value;
module name(…) { … }
name();
function name(…) = …
name();
include <….scad>
use <….scad>
```

2D

```
circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font,
     halign, valign, spacing,
     direction, language, script)
```

3D

```
sphere(radius | d=diameter)
cube(size)
cube([width,depth,height])
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, triangles, convexity)
```

Transformations

```
translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
resize([x,y,z],auto)
mirror([x,y,z])
multmatrix(m)
color("colorname")
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski()
```

Boolean operations

```
union()
difference()
intersection()
```

Modifier Characters

```
*   disable
!   show only
#   highlight
%   transparent
```

Mathematical

```
abs
sign
sin
cos
tan
acos
asin
atan
atan2
floor
round
ceil
ln
len
let
log
pow
sqrt
exp
rands
min
max
```

Functions

```
lookup
str
chr
search
version
version_num
norm
cross
parent_module(idx)
```

Other

```
echo(…)
for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
for (i = […,…,…]) { … }
intersection_for(i = [start:end]) { … }
intersection_for(i = [start:step:end]) { … }
intersection_for(i = […,…,…]) { … }
if (…) { … }
assign (…) { … }
import("….stl")
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(convexity)
surface(file = "….dat",center,convexity)
projection(cut)
render(convexity)
children([idx])
```

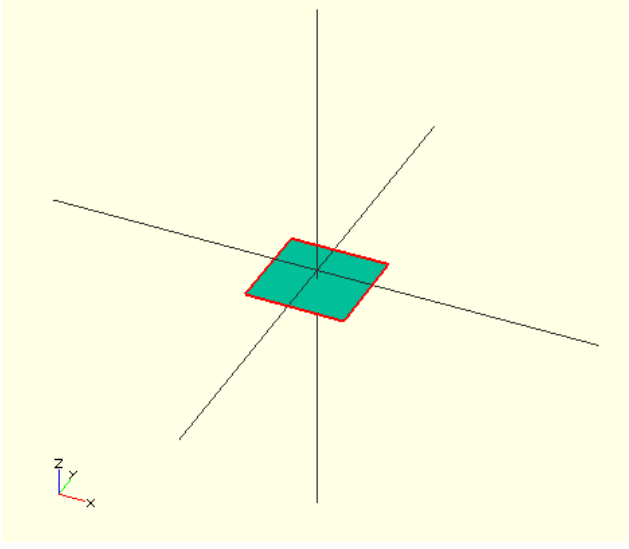
List Comprehensions

```
Generate [ for (i = range|list) i ]
Conditions [ for (i = …) if (condition(i)) i ]
Assignments [ for (i = …) let (assignments) a ]
```

Special variables

```
$fa       minimum angle
$fs       minimum size
$fn       number of fragments
$t        animation step
$vpr      viewport rotation angles in degrees
$vpt      viewport translation
$vpd      viewport camera distance
$children number of module children
```

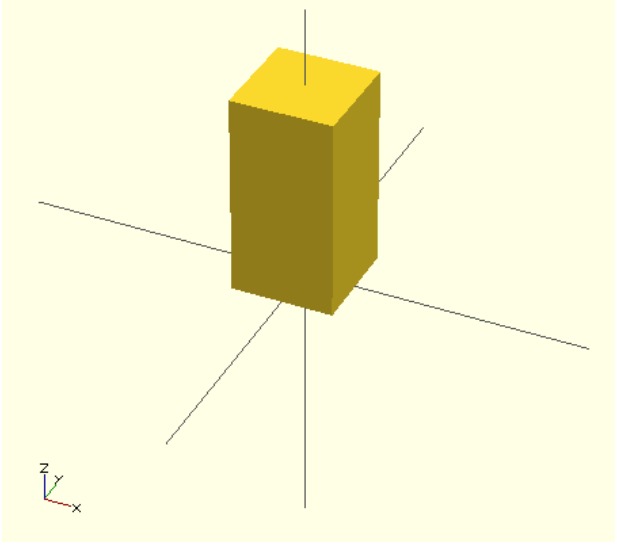
```
circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font, halign, valign, spacing, direction, language, script)
```



```
square(center=true);
```

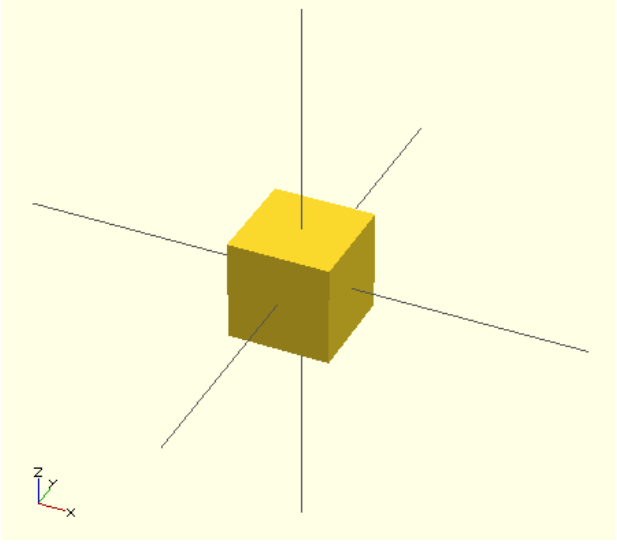
2D to 3D Extrusion


```
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(convexity)
```



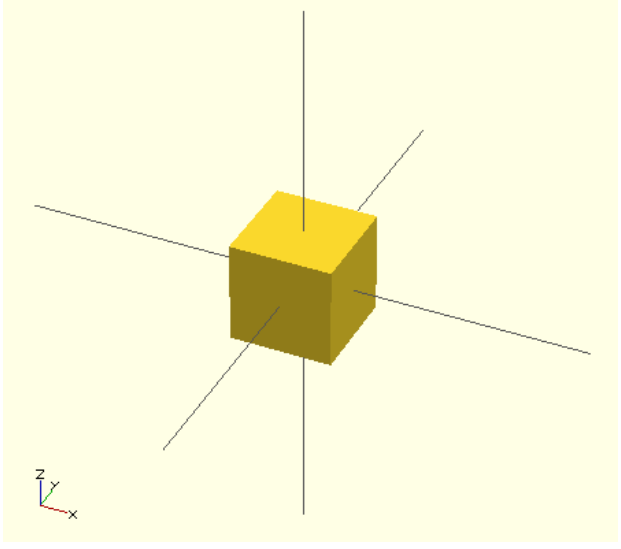
```
linear_extrude(height = 2) square(center=true);
```

```
sphere(radius | d=diameter)
cube(size)
cube([width,depth,height])
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, triangles, convexity)
```



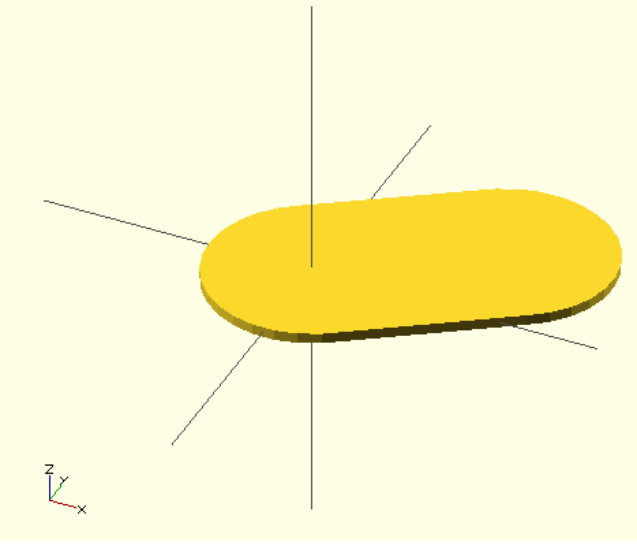

```
cube(r = 1, center = true);
```

```
translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
resize([x,y,z],auto)
color([r,g,b,a])
```



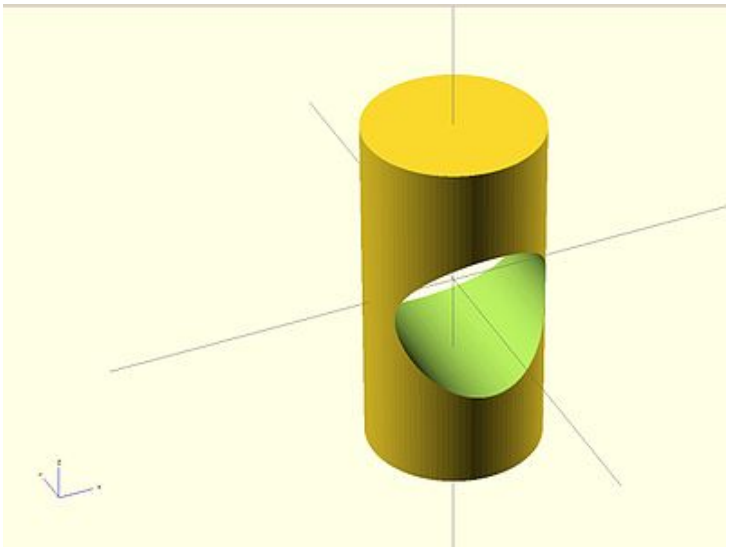

```
translate([0, 0, 1.5]) cube(r = 1, center = true);
```

```
mirror([x,y,z])
offset(r|delta,chamfer)
hull()
minkowski()
```



```
hull() {
   translate([15,10,0]) circle(10);
   circle(10);
}
```

```
union()
difference()
intersection()
```

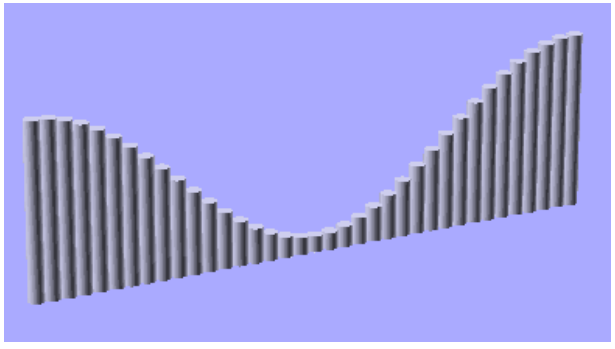


```
difference() {
  cylinder (h = 4, r=1, center = true, $fn=100);
  rotate ([90,0,0]) cylinder (h = 4, r=0.9, center = true, $fn=100);
}
```

```
sin
cos
tan
acos
asin
atan
atan2
floor
round
ceil
```

```
abs
sign
ln
len
let
log
pow
sqrt
exp
rands
min
max
```

```
for (i = [start:end]) { ... }
for (i = [start:step:end]) { ... }
for (i = [...,...,...]) { ... }
intersection_for(i = [start:end]) { ... }
intersection_for(i = [start:step:end]) { ... }
intersection_for(i = [...,...,...]) { ... }
```



```
for(i=[0:36])
  translate([i*10,0,0])
    cylinder(r=5,h=cos(i*10)*50+60);
```

```
var = value;
assign (...) { ... }
```

```
for (i = [10:50]) {
  angle = i*360/20;
  distance = i*10;
  r = i*2;
  rotate(angle, [1, 0, 0])
  translate([0, distance, 0])
  sphere(r = r);
}
```

```
if (...) { ... }
```

```
if (x > y) {
  cube(size = 1, center = false);
} else {
  cube(size = 2, center = true);
}
```

```
module name(...) { ... }
name();
function name(...) = ...
name();
```

```
module hole(distance, rot, size) {
  rotate(a = rot, v = [1, 0, 0]) {
    translate([0, distance, 0]) {
      cylinder(r = size, h = 100, center = true);
    }
  }
}
```

```
hole(0, 90, 10);
```

next time

- Our software stack
- CNC machines and manufacturing

Referencers

- ILM Math Library – `http://www.openexr.com`
- Open SCAD – `http://www.openscad.org`