Parametric Design with OpenSCAD

- Jay Summet
- jay@summet.com

OpenSCAD

- Open Source parametric design tool
- Executables for Windows, Mac, Linux
- Free to distribute, free to use, cross-platform
- Converts textual instructions into 3D shapes
- Supports Constructive Solid Geometry (CSG) modeling

Resource Links

- Downloads:
  http://www.openscad.org

- User Manual:
  http://en.wikibooks.org/wiki/OpenSCAD_User_Manual

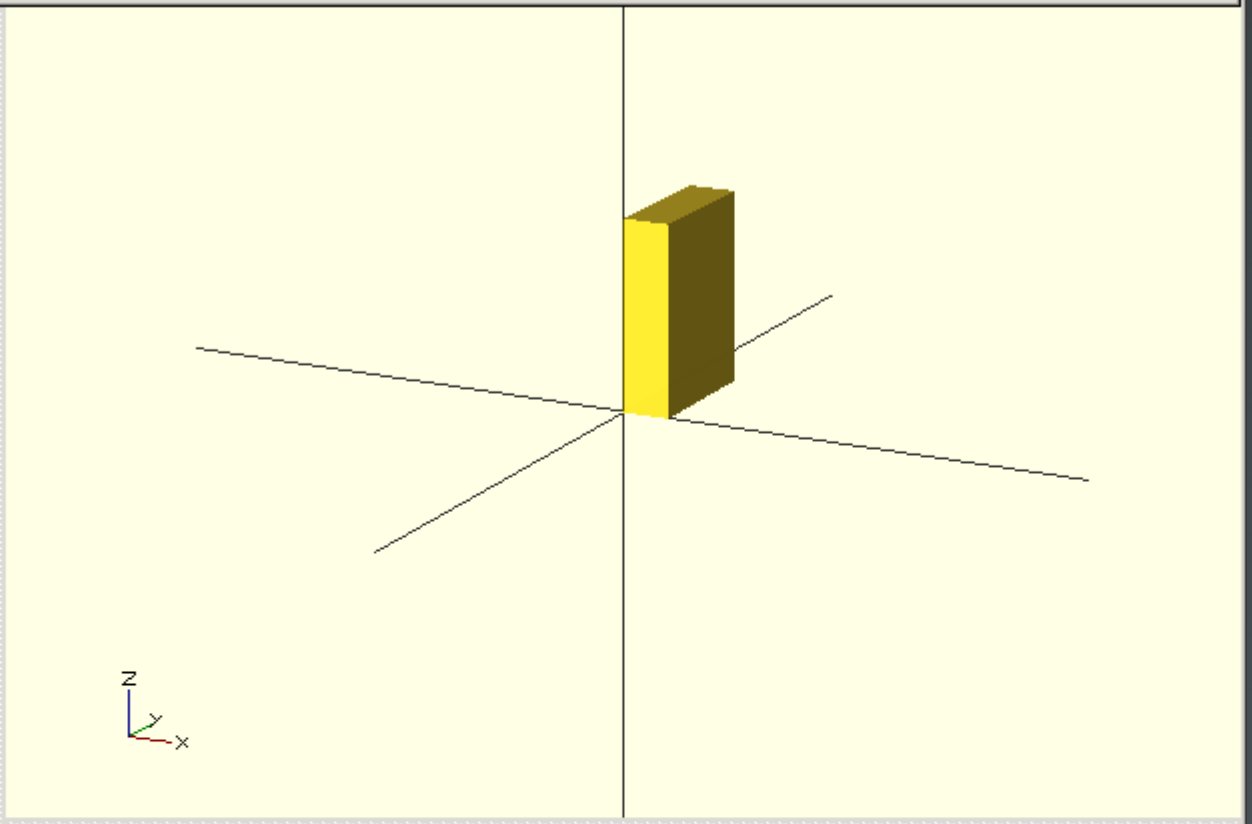
OpenSCAD - New Document*

File Edit Design View Help

```
// Named values
w= 5;
h= 20;
d = 15;

// Vector
dims = [w,d,h];

// call to built-in module
cube(dims);
```




```
Parsing design (AST generation)...
Compiling design (CSG Tree generation)...
Compilation finished.
Compiling design (CSG Products generation)...
PolySets in cache: 3
Polygons in cache: 18
CGAL Polyhedrons in cache: 0
Vertices in cache: 0
Compiling design (CSG Products normalization)...
Normalized CSG tree has 1 elements
CSG generation finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

Viewport: translate = [0.00 0.00 0.00], rotate = [73.20 0.00 27.10], distance = 500.00

Comments

- C/Java/C++ commenting conventions
- Any text after a double forward-slash (//) on a line is ignored by the parser.
- Multi-line comments are started with a slash-star (/*) and ended with a star-slash (*/)

Named Values

- a.k.a. Variables
- A name is assigned a value with the assignment operator (=). Can use expressions on the right hand side of the assignment operator to calculate values.
- CAUTION! : Named Values are set at compile time, NOT RUN TIME! Last assignment takes precidence! (But see the ”assign” statement...)

Example:

```
a = 0;
echo( "A is:", a);

b = a+10;
a = 5;

echo("A is:", a);
echo("B is:", b);
```

```
Parsing design (AST generation)...
Compiling design (CSG Tree generation)...
ECHO: "A is:", 5
ECHO: "A is:", 5
ECHO: "B is:", 15
Compilation finished.
Compiling design (CSG Products generation)...
ERROR: CSG generation failed! (no top level object found)
PolySets in cache: 3
Polygons in cache: 18
CGAL Polyhedrons in cache: 0
Vertices in cache: 0
```

Variable Advice

- To keep yourself sane:
  - Always make new variables, never re-assign new values to old variables.
  - Think of variables as ”constants” or ”descriptive names” instead of ”variables”.

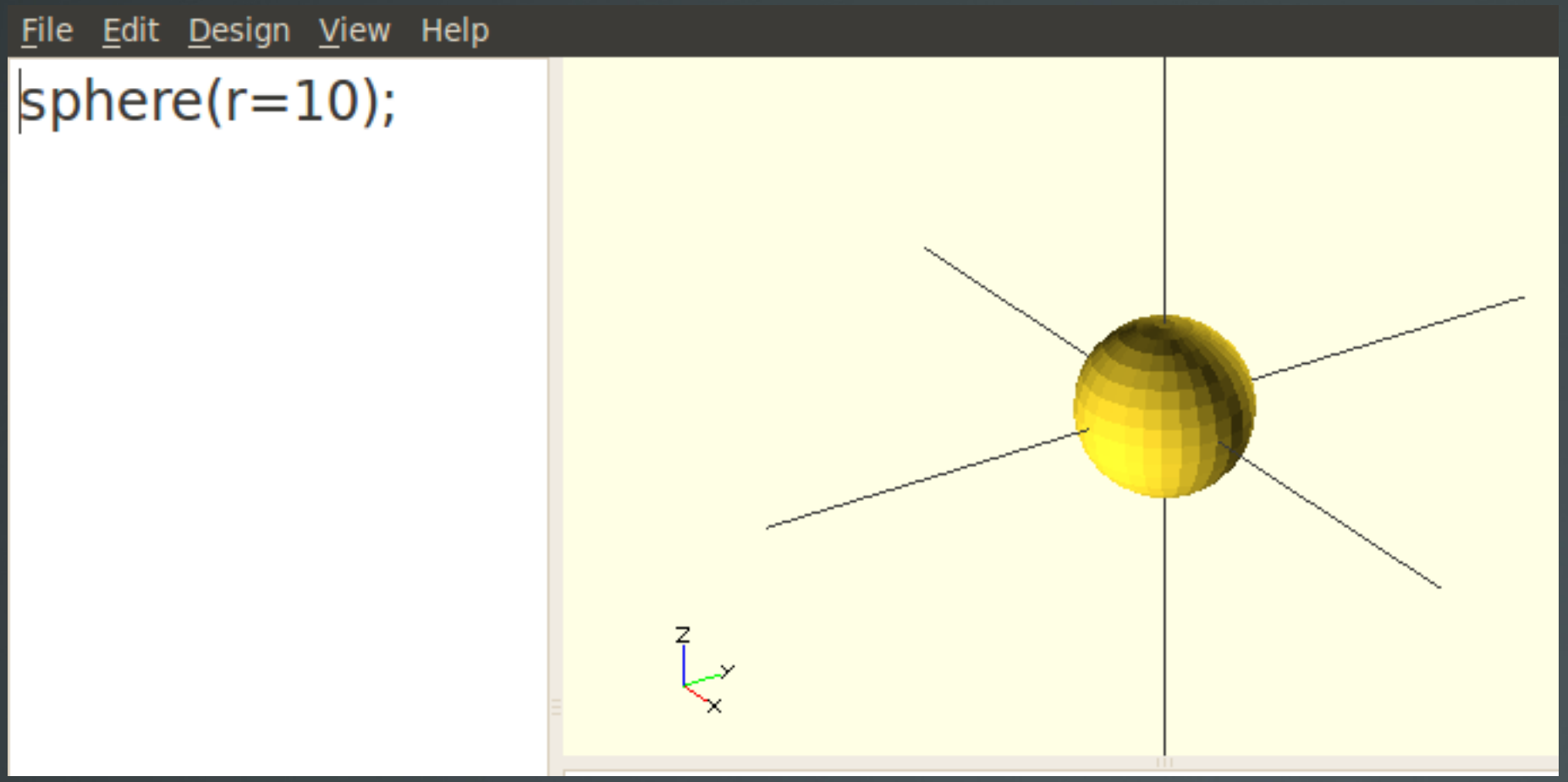
Vectors

- Example: dims = [w,d,h];
- Using square brackets, declare a vector of values, either from constants, expressions, or named values/variables.
- Example: dims = [w*2, d+3, 17];

Statements

- REQUIRE semicolons to end!
- If you forget the semicolon the parser will typically display the error as occuring at the beginning of the next line.
- So check the line before the error!

Sphere



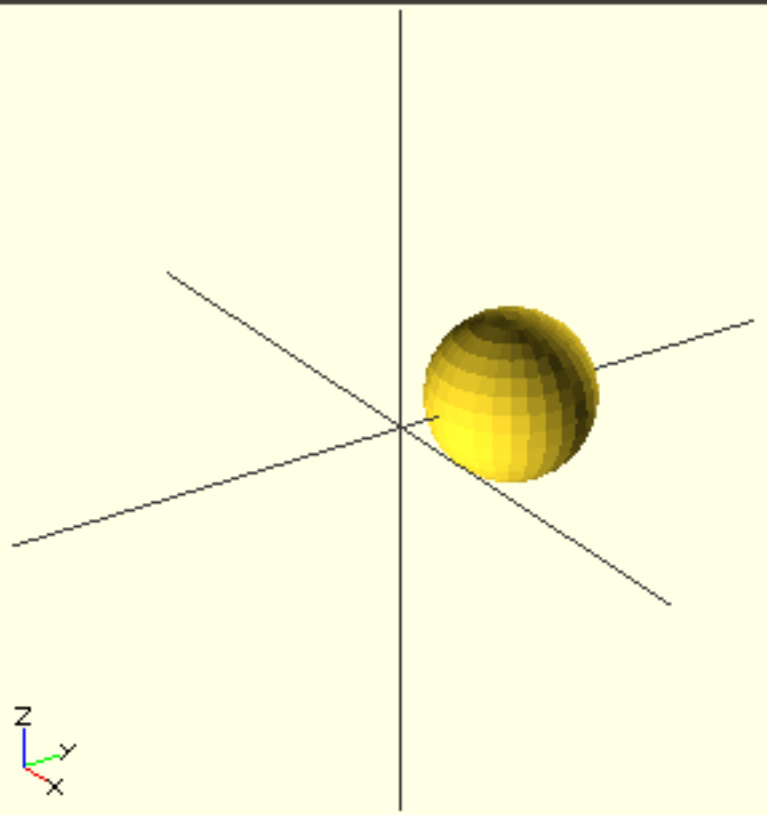
Transformations

- Objects and entire sub-trees can have transformations applied to them that affect their size, placement, color and shape.
- Most commonly used: rotate, translate
- Also useful: mirror, scale, color, hull
- Advanced: multmatrix, minkowski

Sphere - Translated

File Edit Design View Help

```
translate([0,15,0])
  sphere(r=10);
```



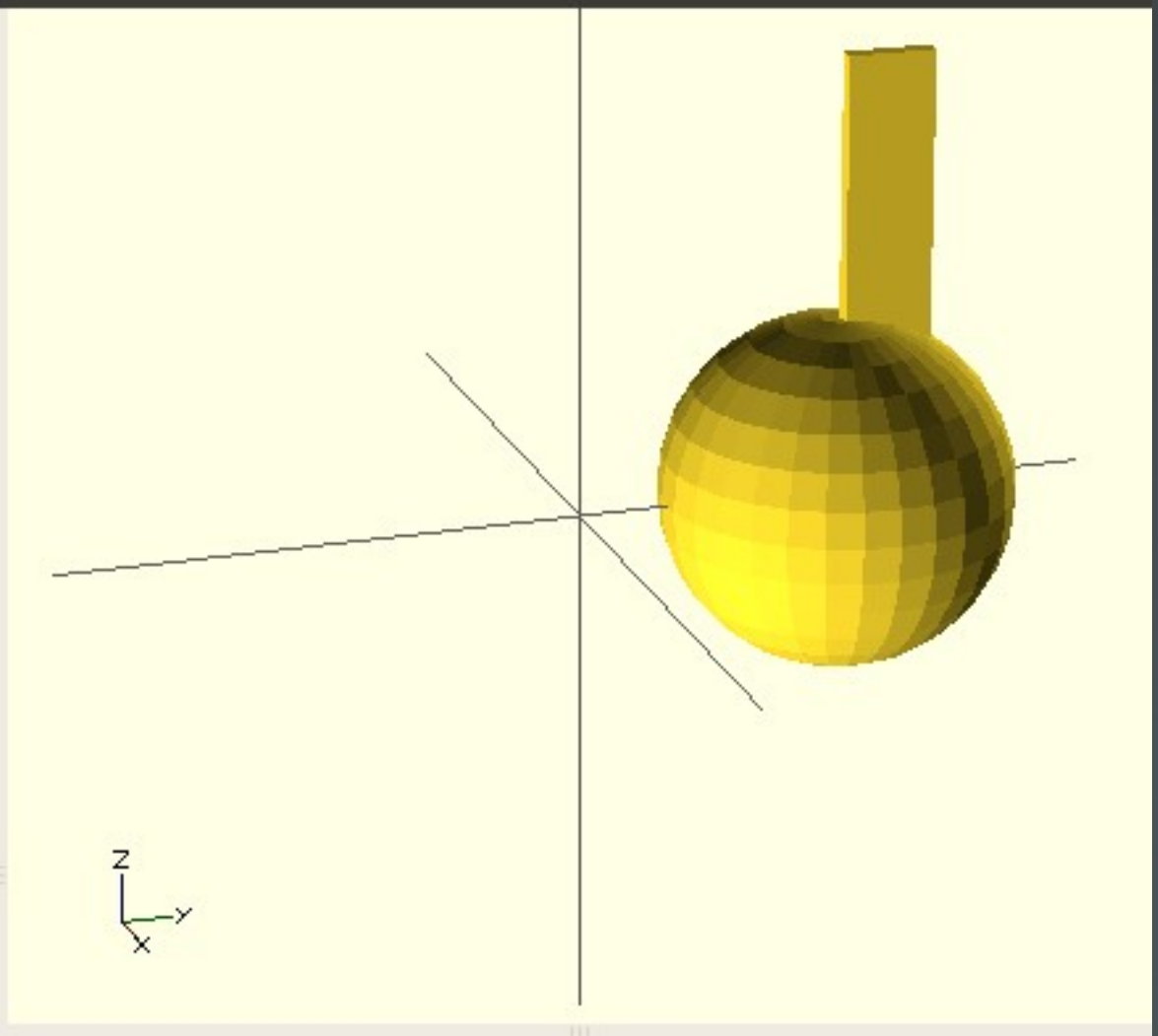
sub-tree

- The translate command works on a sub-tree that contains child nodes. By default, the sub-tree includes the immediately following object, ended by a semicolon.
- If you use {curly brackets} to deliniate sub-trees, you can include more than one module or child node.

Sphere & Cube Translated

File Edit Design View Help

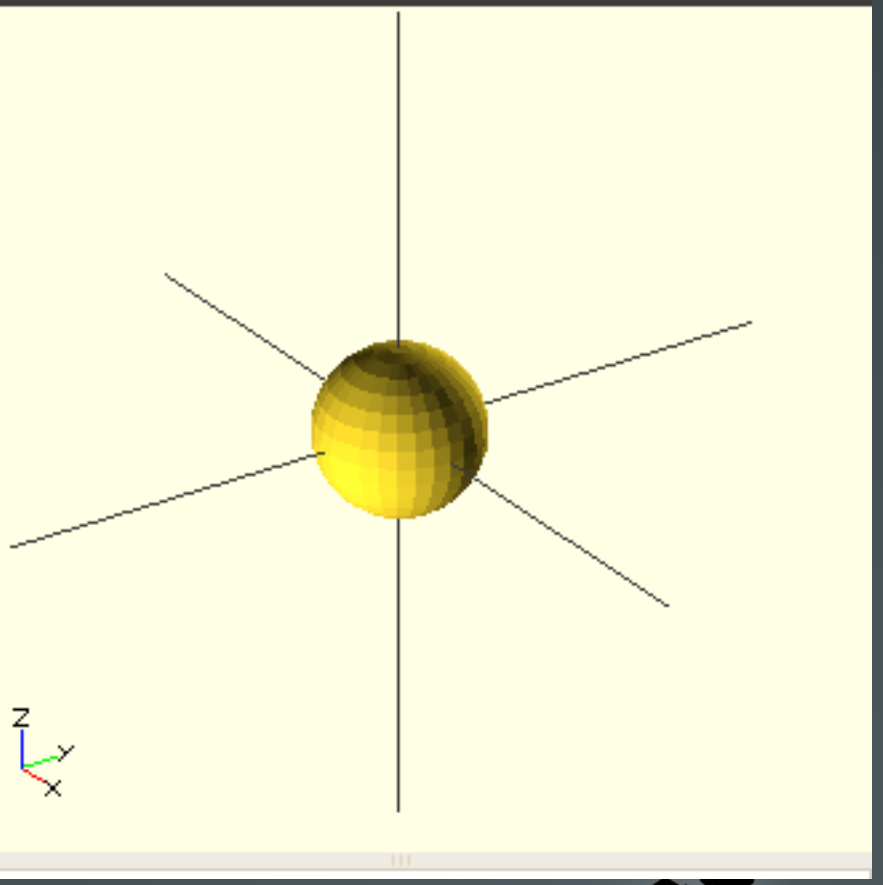
```
translate([0,15,0]) {
   sphere(r=10);
   cube([1,5,25]);
}
```



Sphere - Not Translated

File Edit Design View Help

```
translate([0,15,0]);
  sphere(r=10);
```



Note the extra semicolon!

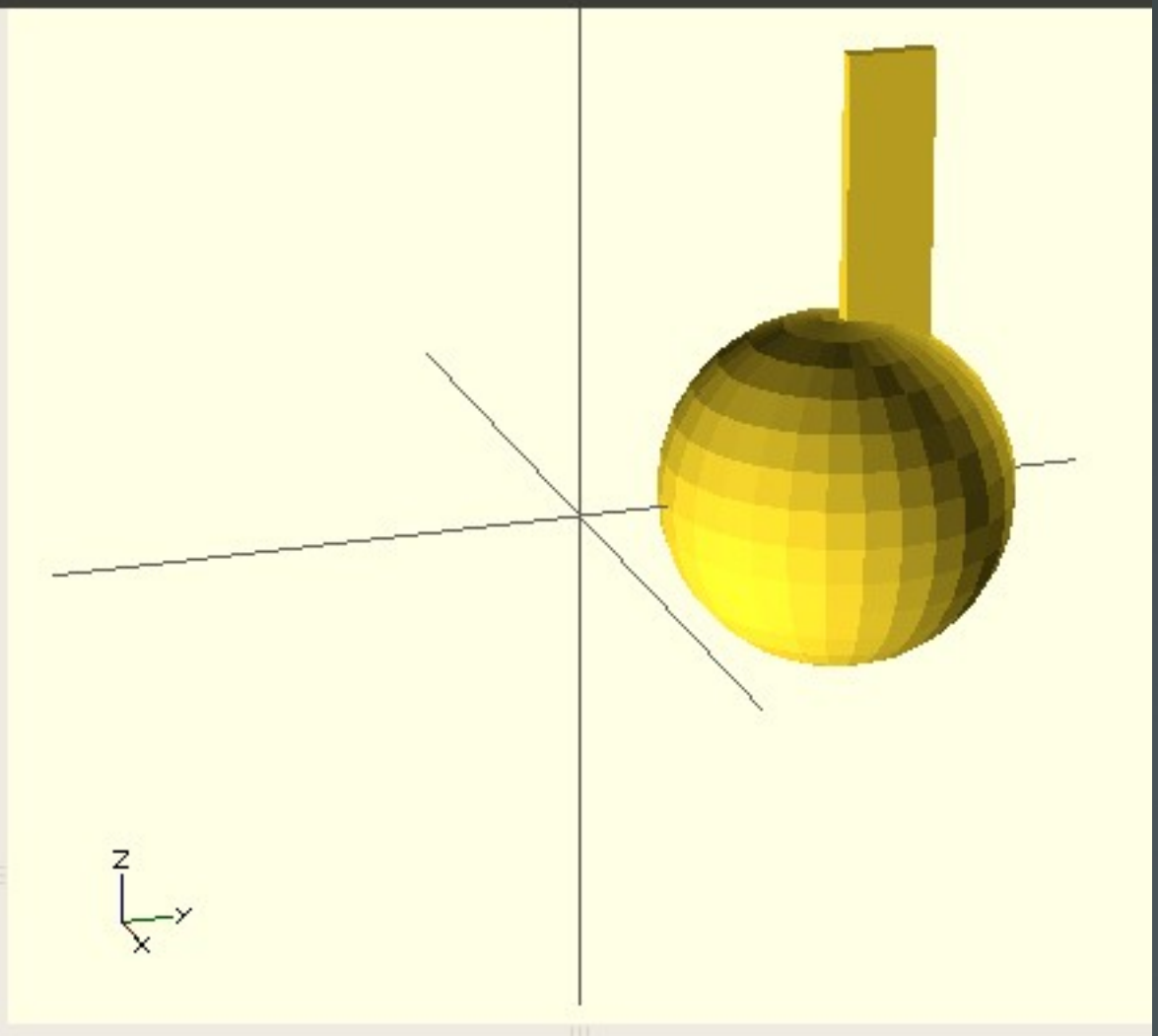
Centering on Origin

- By default, when a sphere is created it is centered on the origin.
- However, a cube is created with its corner at the origin by default.
- This is why the cube in the previous slide is not ”centered” within the sphere after they are both translated the same amount.

Sphere & Cube Translated

File Edit Design View Help

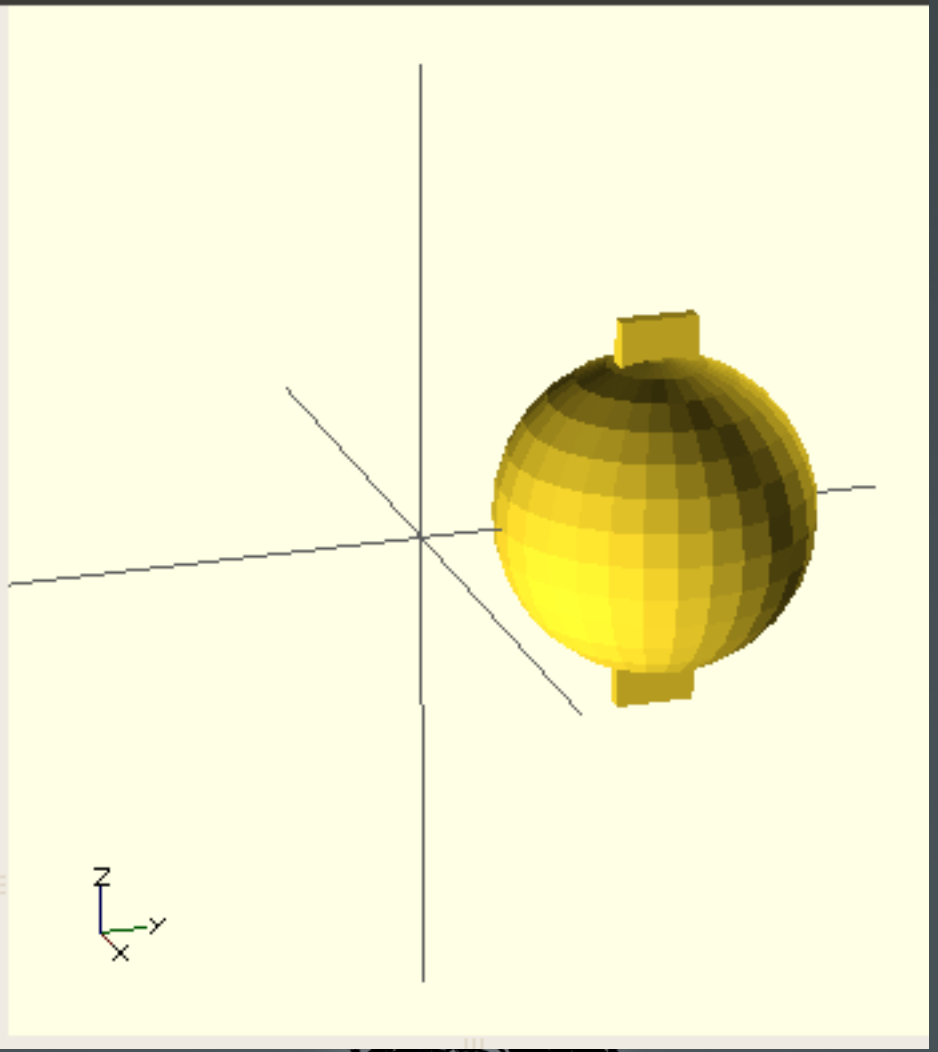
```
translate([0,15,0]) {
   sphere(r=10);
   cube([1,5,25]);
}
```



Creating a centered cube

File Edit Design View Help

```
translate([0,15,0]) {
   sphere(r=10);
   cube([1,5,25], center=true);
}
```



Special Arc Control variables

- $fn – Normally set to zero (0) to allow $fa and $fs to take effect. If set to a number, all circles are made with exactly $fn straight line fragments.
- $fa – Minimum angle for a fragment. Number of fragments = 360 / $fa. Defaults to 12 (i.e. 30 fragments for a full circle)
- $fs – Minimum fragment size. Defaults to 2. Very small circles will have a smaller number of fragments than $fa specifies.

Just use $fn

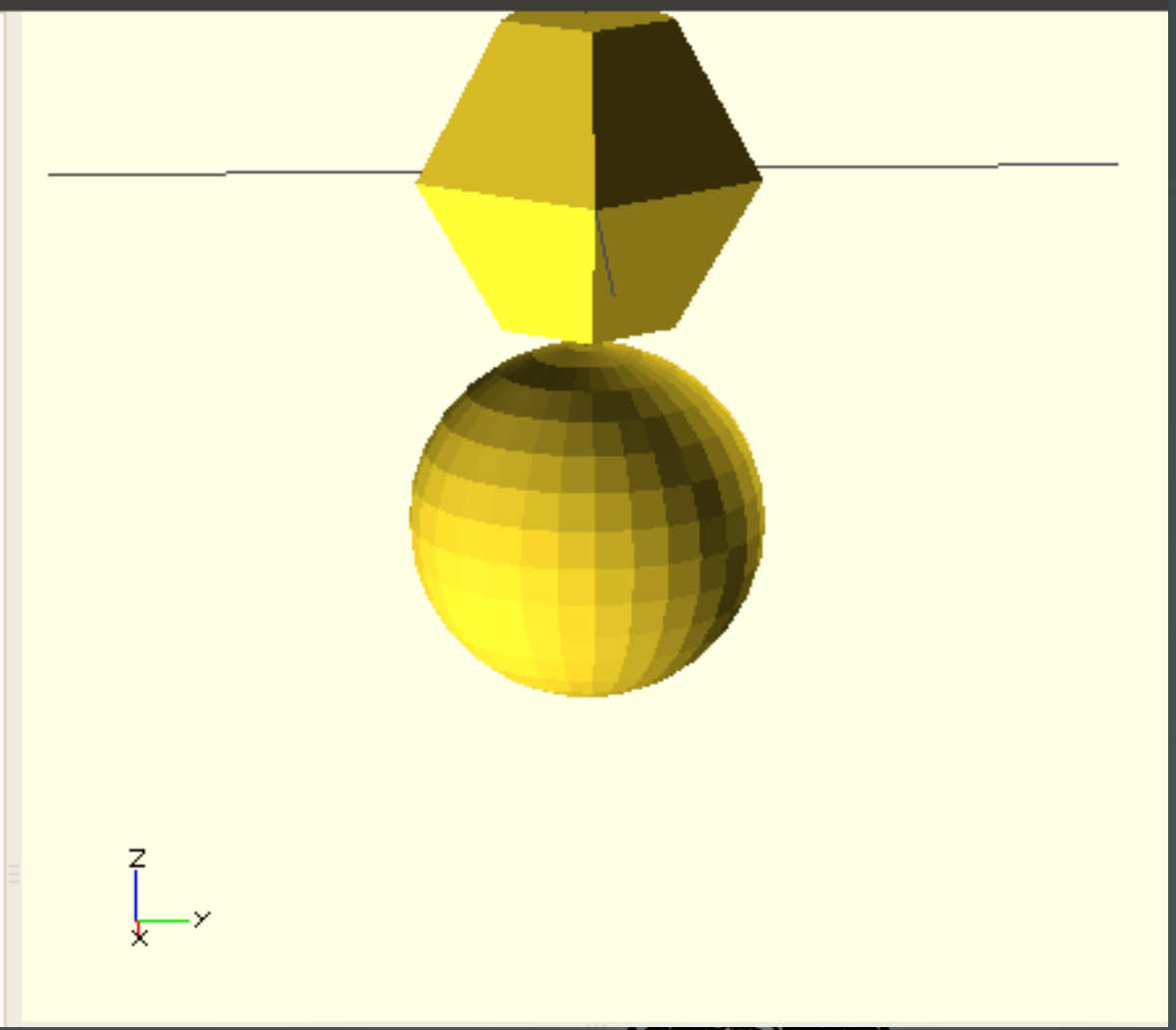
- $fn is the easiest to use – If you want circles, cylinders, and spheres to be smoother, increase $fn.
- The larger $fn is, the longer calculations take and the more vertices / file size your exported models will have.
- Can set globally, or pass to specific shapes individually.

$fn example

File Edit Design View Help

```
sphere(r=10, $fn=5);

translate([0,0,-20])
  sphere(r=10, $fn=30);
```



Modules

- Like functions, but can affect sub-trees that follow them, so can be used to implement complex transformations as well as objects.
- Allow you to reuse code.
- Can accept parameters.
- Use a curly-brackets to deliniate the sub-tree of code that is the module.
- Have local variable names.

Variable Scope

- root/global scope is different from within a module, so you can re-define a variable within a module without affecting its value outside of the module.
- Reminder: Because variables are set at compile time instead of run time, you can not re-assign a variable inside of an if sub-tree.

HexNut Module

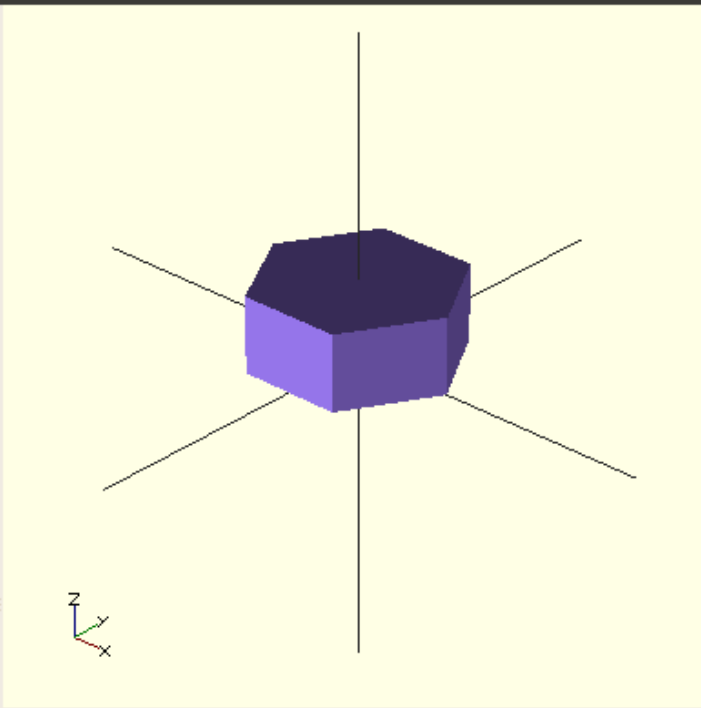
File Edit Design View Help

```
module hexNut( wrenchSize,thickness) {
 $fa = 0.5;
 $fs = 0.5;
 apothothem = wrenchSize / 2;    //Center to midpoint of side

 // Calculate the size of the hexagon side:
 side = 2 * apothothem * tan( 180 / 6) ;

 // Draw the hexagon as a union of 3 rectangular cubes
 color( [128/255,100/255,200/255] )
 union() {
  for (i= [0: 2] )
     rotate( [0,0,i*60] )
        translate( [-(side/2),-(wrenchSize/2),0] )
           cube( [side,wrenchSize,thickness] );
  } // end union
} // end module hexNut(wrenchSize, thickness)

// Call the module!
hexNut(5.9, 2.5);
```

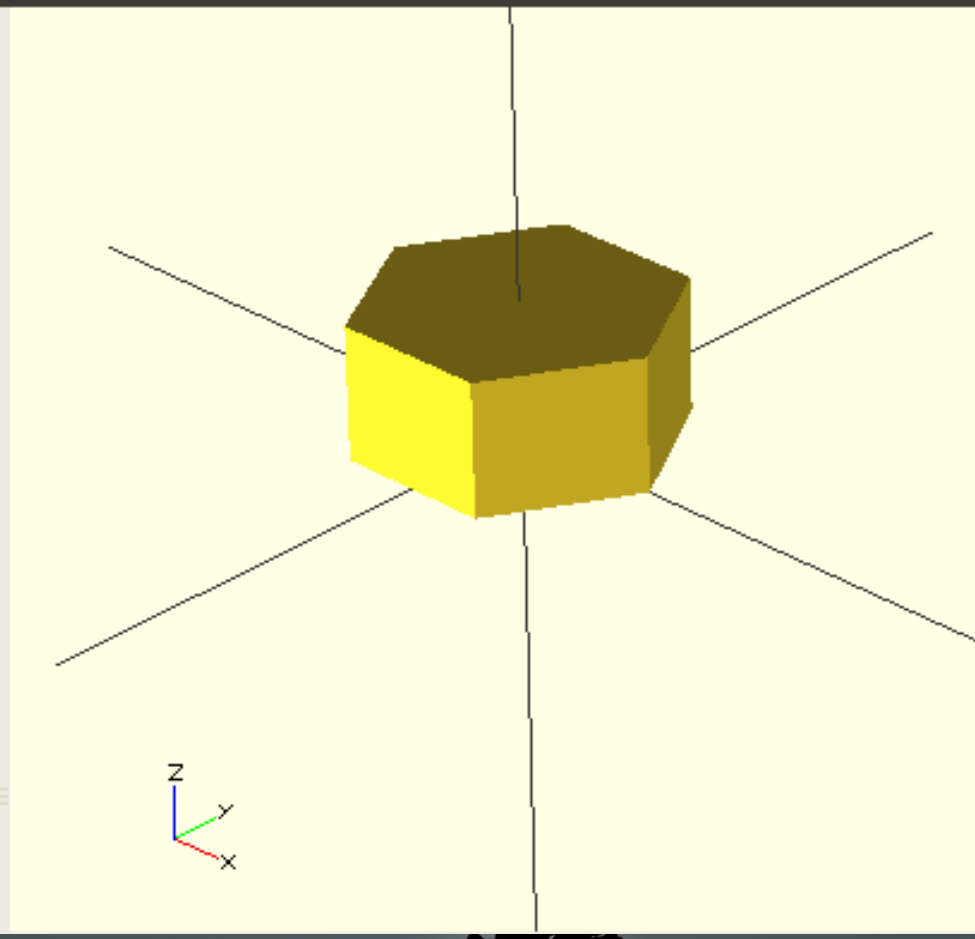



```
Parsing design (AST generation)...
Compiling design (CSG Tree generation)...
Compilation finished.
Compiling design (CSG Products generation)...
PolySets in cache: 3
Polygons in cache: 870
CGAL Polyhedrons in cache: 0
Vertices in cache: 0
Compiling design (CSG Products normalization)...
Normalized CSG tree has 3 elements
CSG generation finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
Saved design
```

Just use $fn

File Edit Design View Help

```
cylinder( h=2.5, r=5.9/2, $fn=6);
```



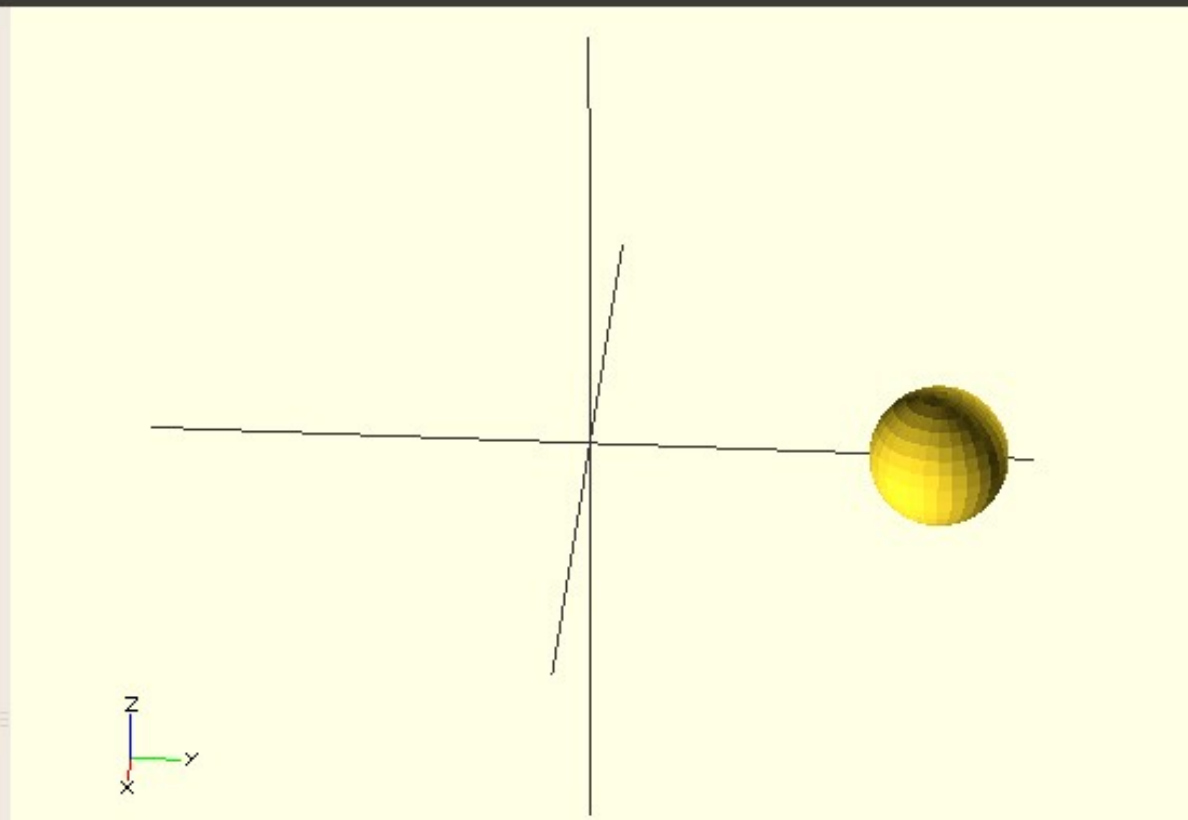
Make Spheres

File Edit Design View Help

```
module makeSpheres( howMany )
{
  for ( i = [0: howMany] )
  {
     rotate( i*360 / howMany )
        translate( [0,10,0] )
          sphere( r=2, $fn=30 );

  }// end for
} // end module

makeSpheres(1);
```



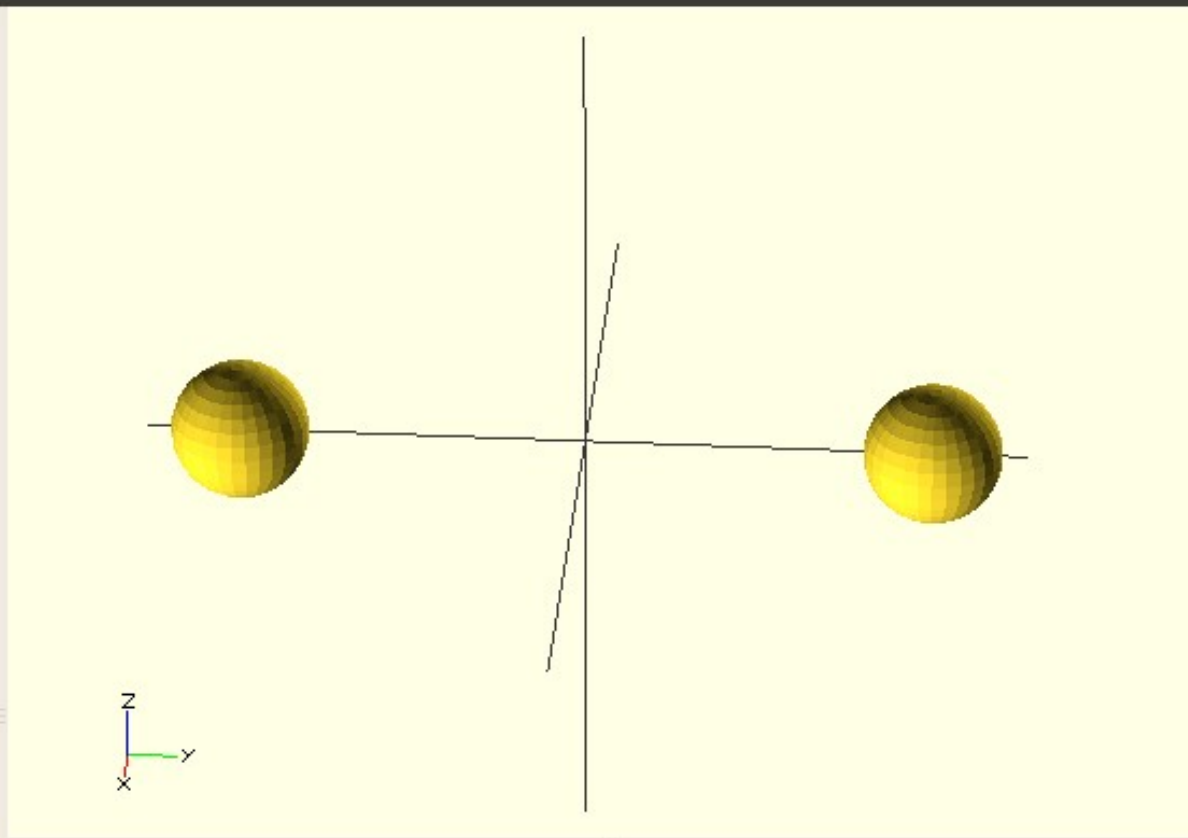
Make Spheres

File Edit Design View Help

```
module makeSpheres( howMany )
{
  for ( i = [0: howMany] )
  {
     rotate( i*360 / howMany )
        translate( [0,10,0] )
          sphere( r=2, $fn=30 );

  }// end for
} // end module

makeSpheres(2);
```



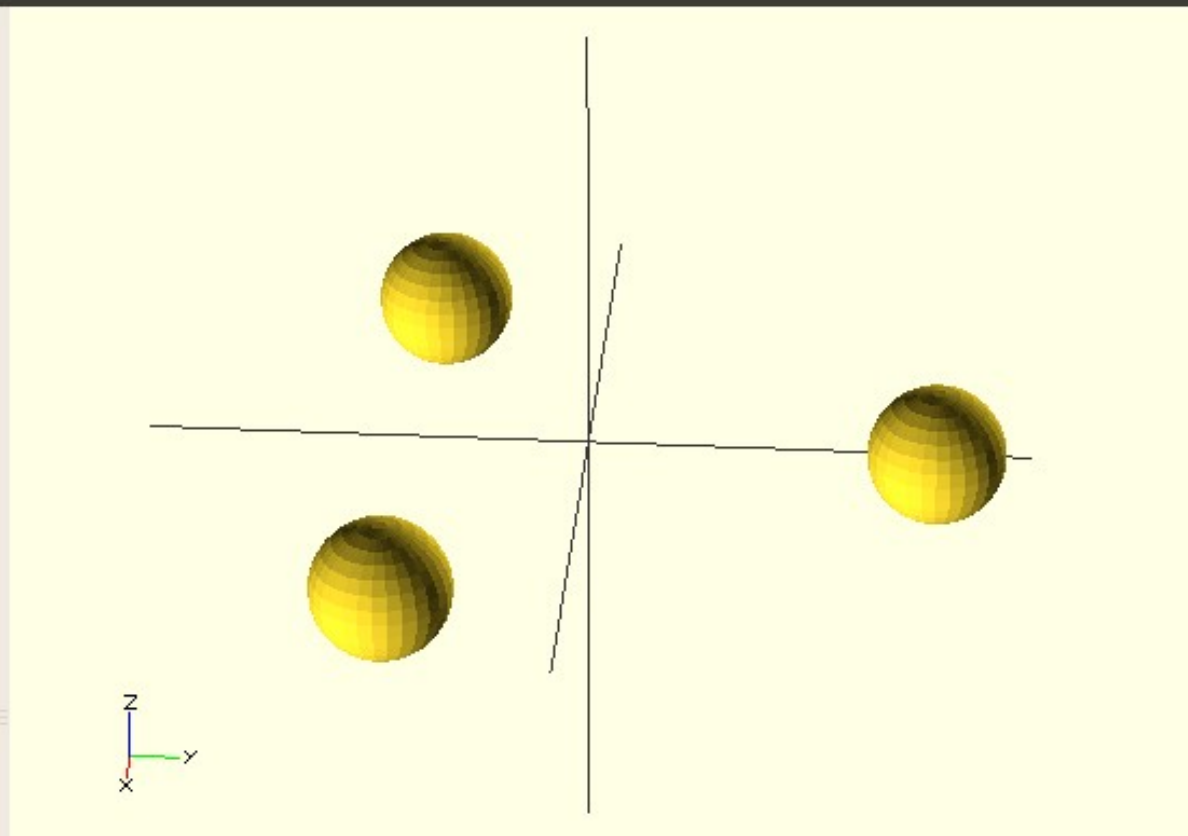
Make Spheres

File Edit Design View Help

```
module makeSpheres( howMany )
{
  for ( i = [0: howMany] )
  {
     rotate( i*360 / howMany )
        translate( [0,10,0] )
          sphere( r=2, $fn=30 );

  }// end for
} // end module

makeSpheres(3);
```



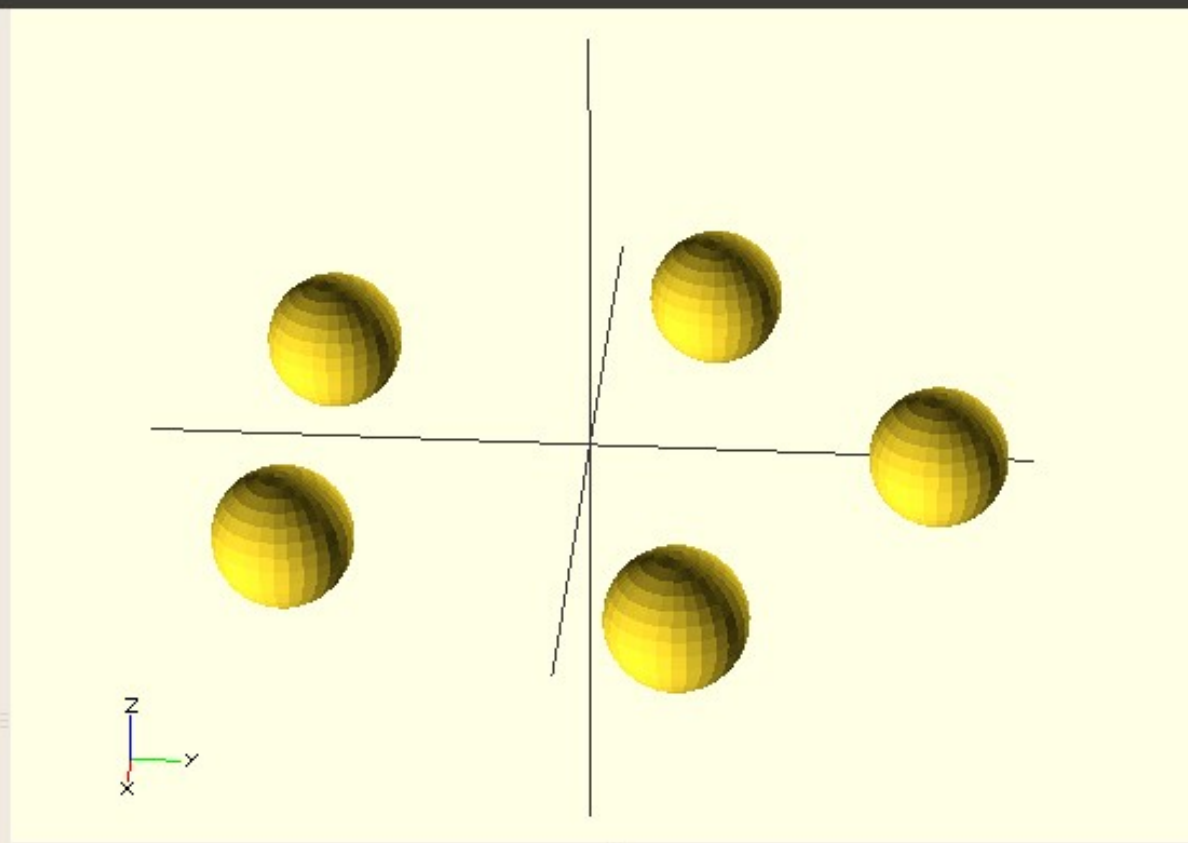
Make Spheres

File Edit Design View Help

```
module makeSpheres( howMany )
{
  for ( i = [0: howMany] )
  {
     rotate( i*360 / howMany )
        translate( [0,10,0] )
          sphere( r=2, $fn=30 );

  }// end for
} // end module

makeSpheres(5);
```



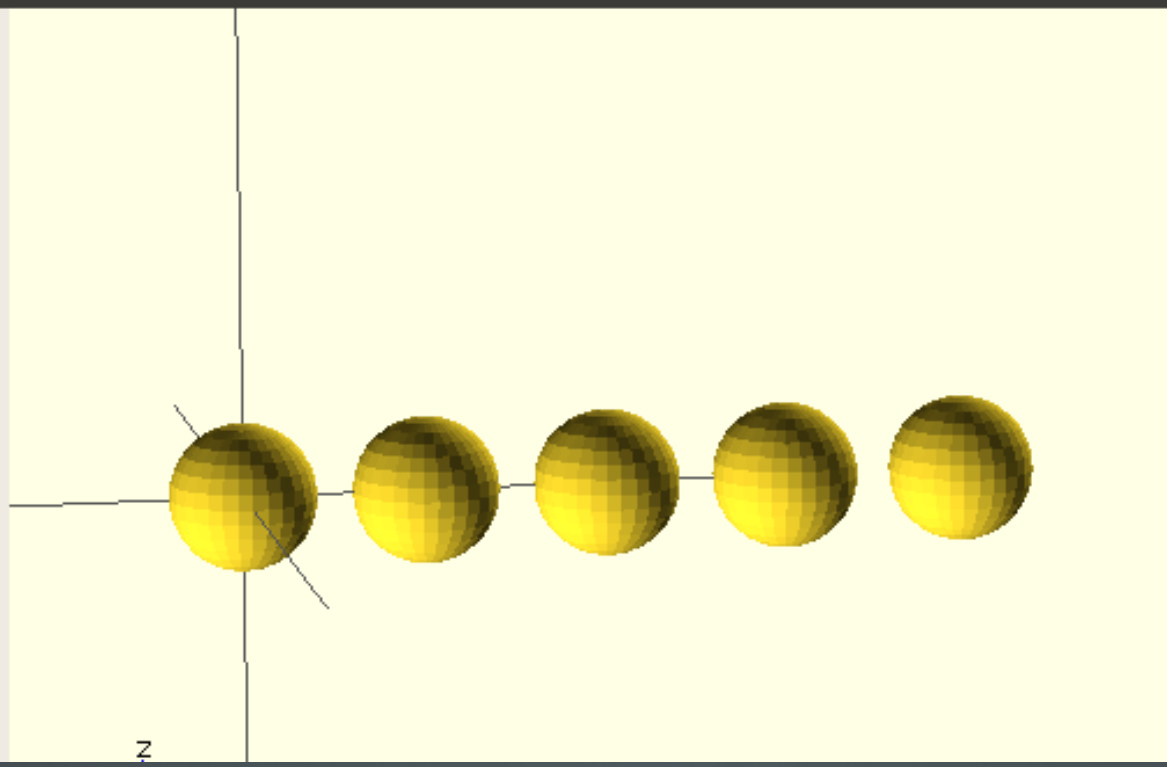
Iteration via for loops

- for (variable = <vector>) <sub-tree> - variable is assigned to each item in the vector and the sub-tree is executed.
- for (variable = <range>) <sub-tree>
  - Range = [<start> : < end>]
  - Range = [<start> : <increment> : <end>]
  - Note: Range syntax uses colons, and the resulting range includes the start and end points

for - Range

File Edit Design View Help

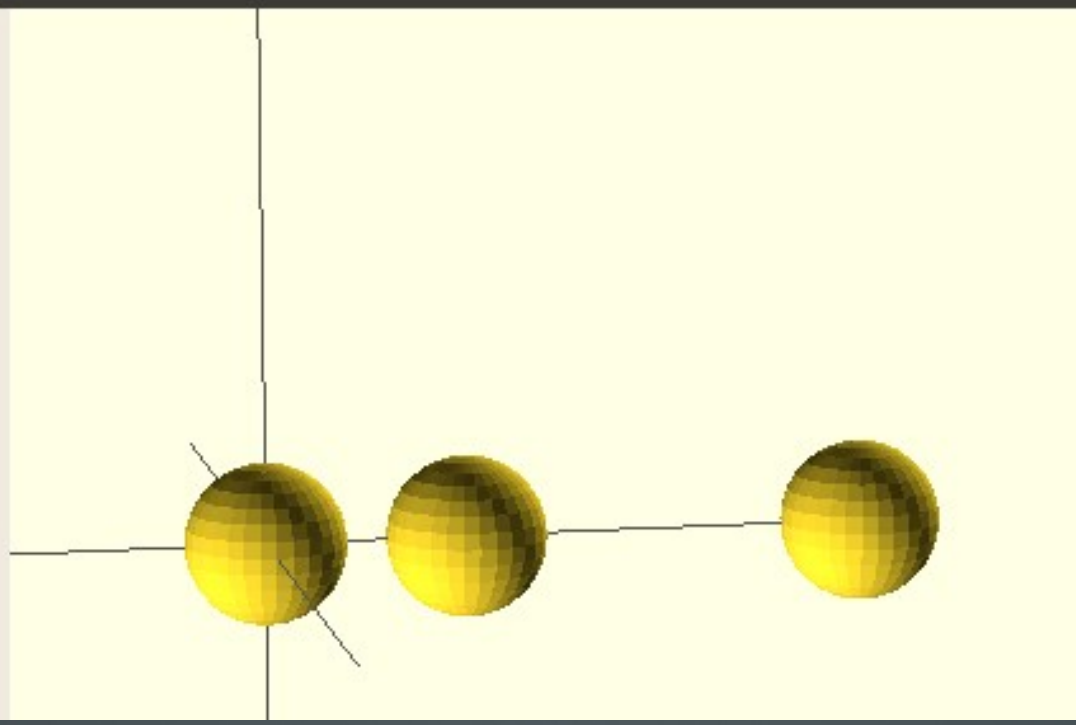
```
for ( i = [0 : 10 : 40] )
  {
    translate([0,i,0])
      sphere(r=4, $fn=30);
  }
// 0, 10, 20, 30, 40
```



for - Vector

File Edit Design View Help

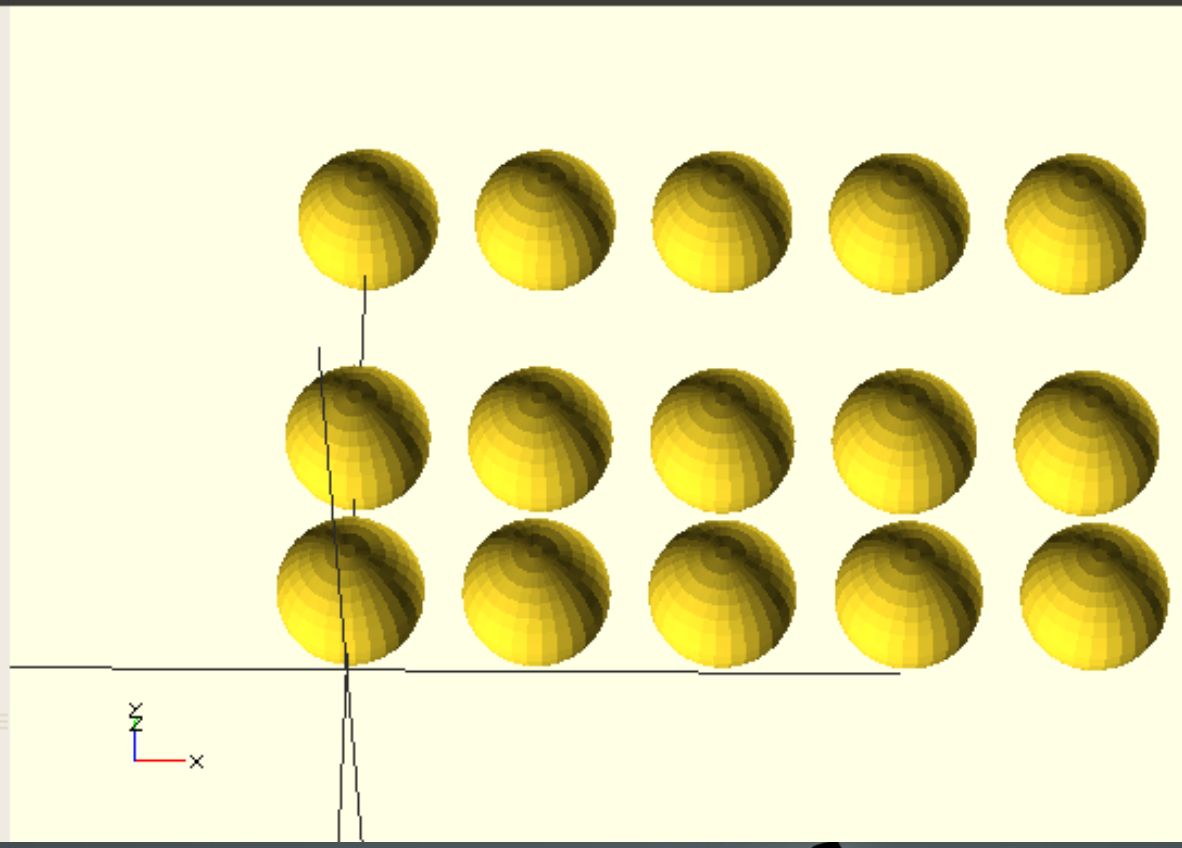
```
for ( i = [0,10,30] )
  {
    translate([0,i,0])
      sphere(r=4, $fn=30);
  }
// 0, 10, 30
```



for (multiple variables)
- range and vector

File Edit Design View Help

```
for ( x = [0 : 10 : 40],
      y = [5,15,30] )
  {
    translate([x,y,0])
      sphere(r=4, $fn=30);
  }
// Rows at 5, 15, 30
// Spheres on each column at:
// 0, 10, 20, 30, 40
```



if statements

- Syntax looks about how you would expect.
- Conditionally executed sub-tree based upon boolean expression with optional else clause.
- Come in useful if you want to have
  - two different versions of the object (different bolt patterns, adapter plates, etc..)
  - Different styles/shapes based upon user parameters.
  - ”debug” vs ”regular” mode.

if - syntax

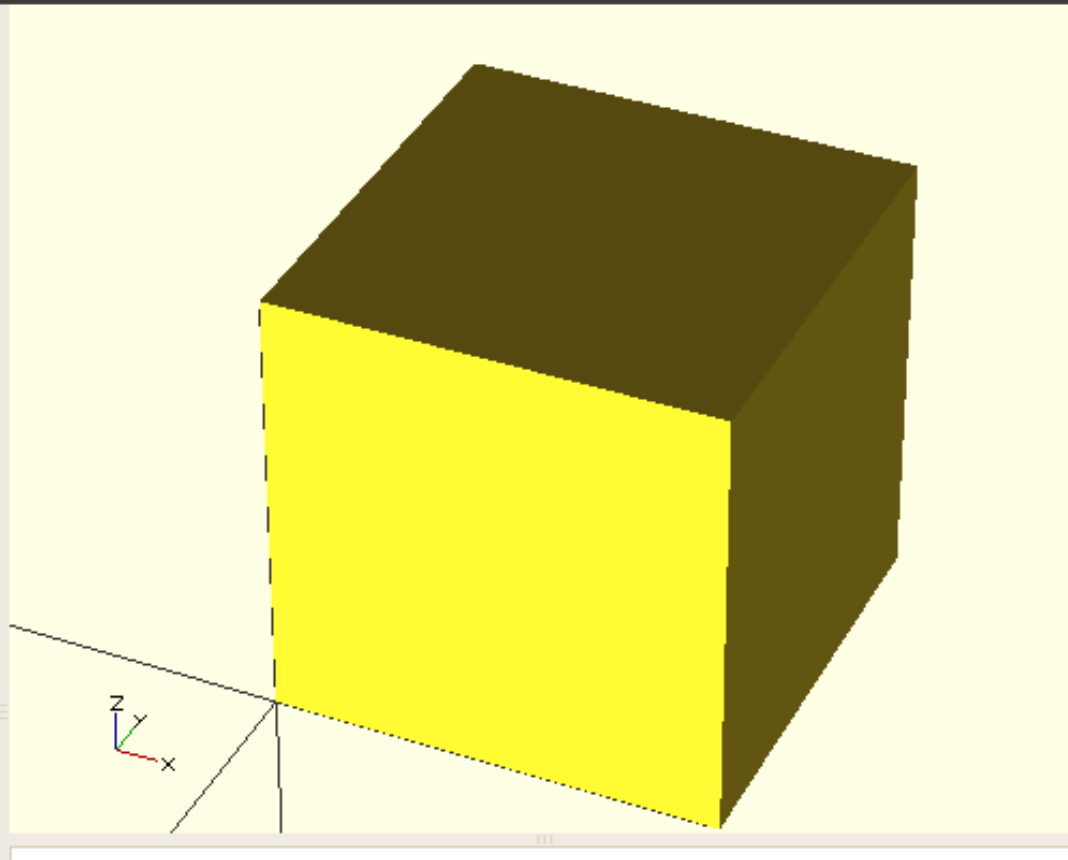
```
if ( <boolean Expression > )
{
   <executes on true>
} else {
   <executes on false>
}
```

if - example

File Edit Design View Help

```
BoxSize = 16;

if ( BoxSize > 15) {
  cube([BoxSize,BoxSize,BoxSize]);
} else {
  cylinder( r=BoxSize/2, h=BoxSize);
}
```

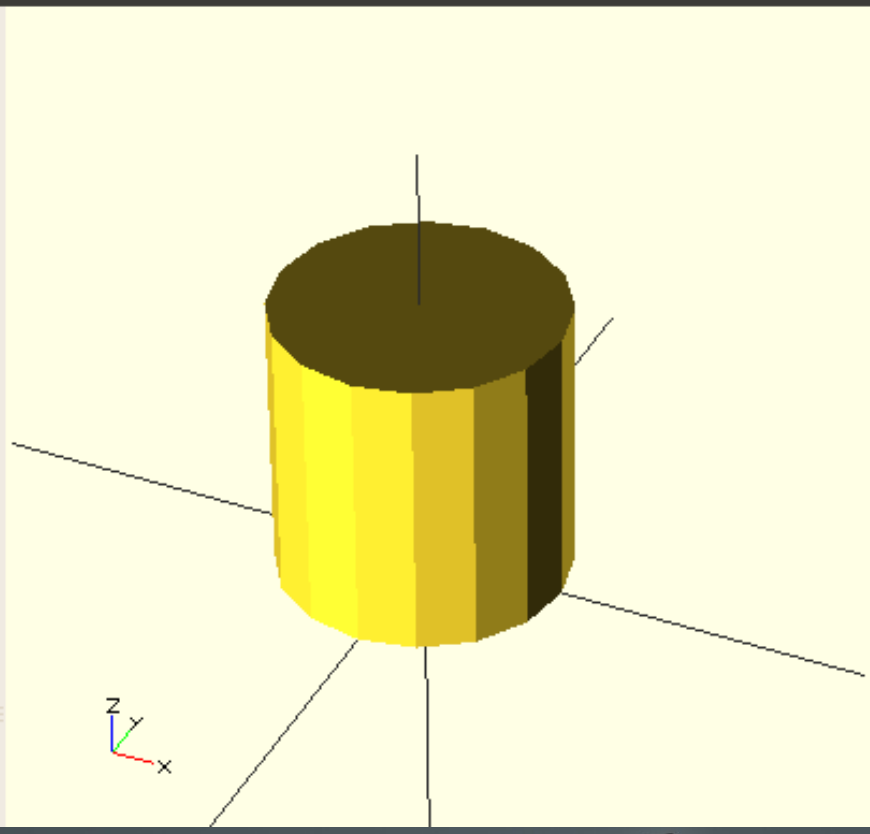


if - example

File Edit Design View Help

```
BoxSize = 10;

if ( BoxSize > 15) {
  cube([BoxSize,BoxSize,BoxSize]);
} else {
  cylinder( r=BoxSize/2, h=BoxSize);
}
```



Constructive Solid Geometry (CSG)

- Consists of modeling complex parts as unions, intersections, and differences of (relatively) simpler parts.
- The hull and minkowski transformations are also useful for creating compound objects.

Making a hollow box

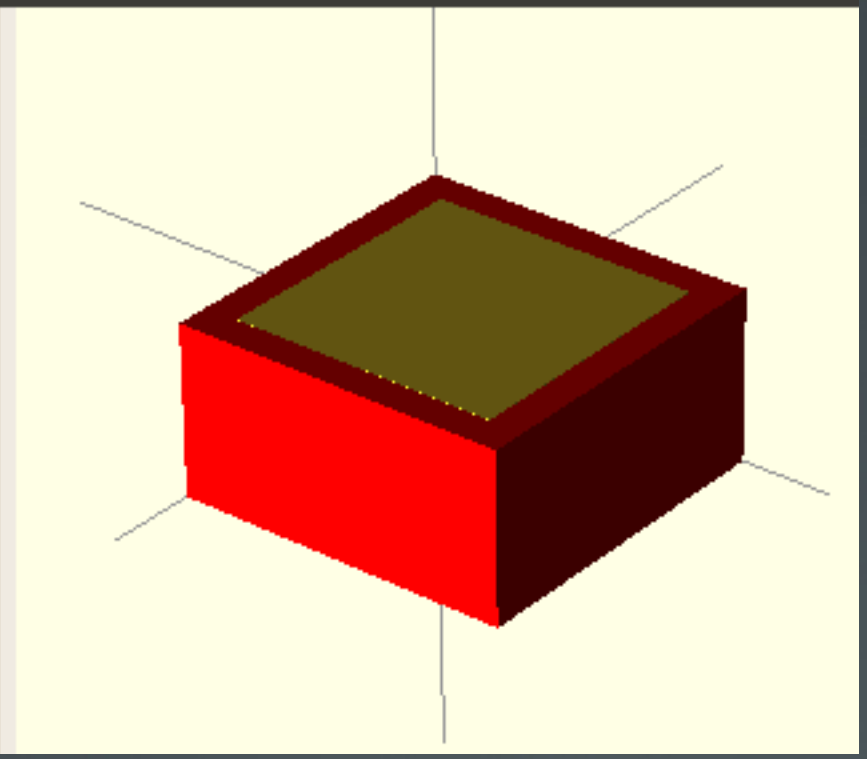
- Make a large cube for your outside dimensions.
- Make a smaller cube for your ”inside” dimensions.
- Translate the smaller cube inside the larger cube (and have it stick out the top by a very small amount such as 0.01) –
  - It has to ”poke out” of the top just a little bit so that the top face is definately open!

Two Cubes

File Edit Design View Help

```
color( [1,0,0] )
  cube( [10,10,5] );

translate( [1,1,2] )
  cube( [8,8,3.01] );
```

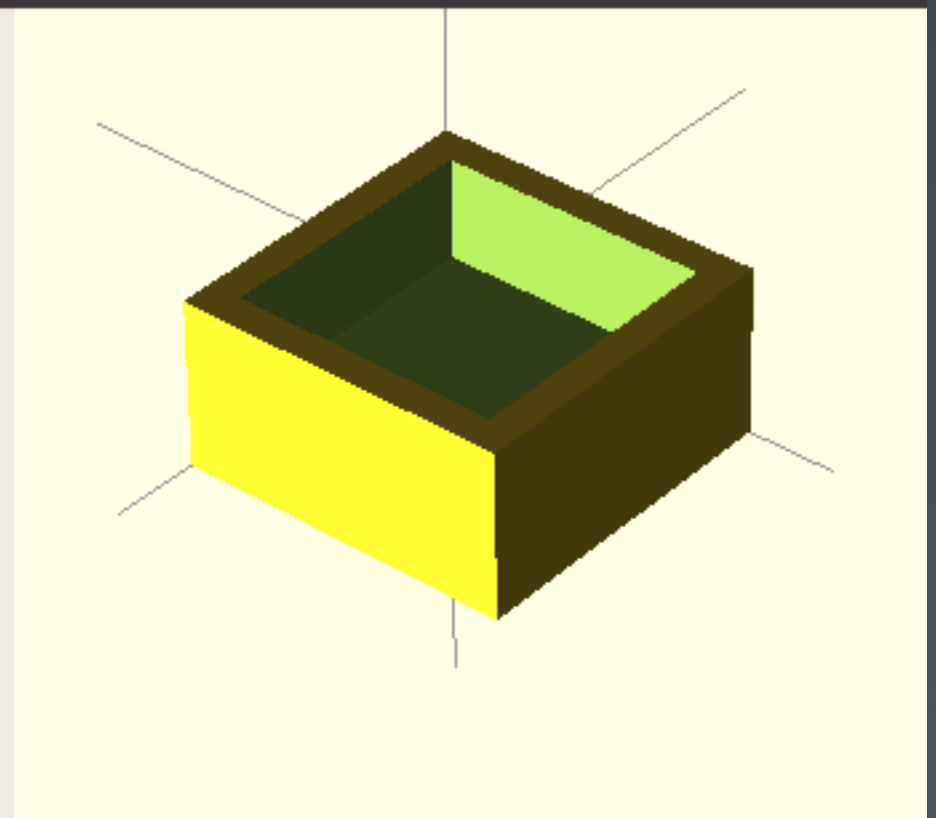


Hollow Box

File Edit Design View Help

```
difference() {
   cube( [10,10,5] );

 translate( [1,1,2] )
   cube( [8,8,3.01] );
} // end difference
```

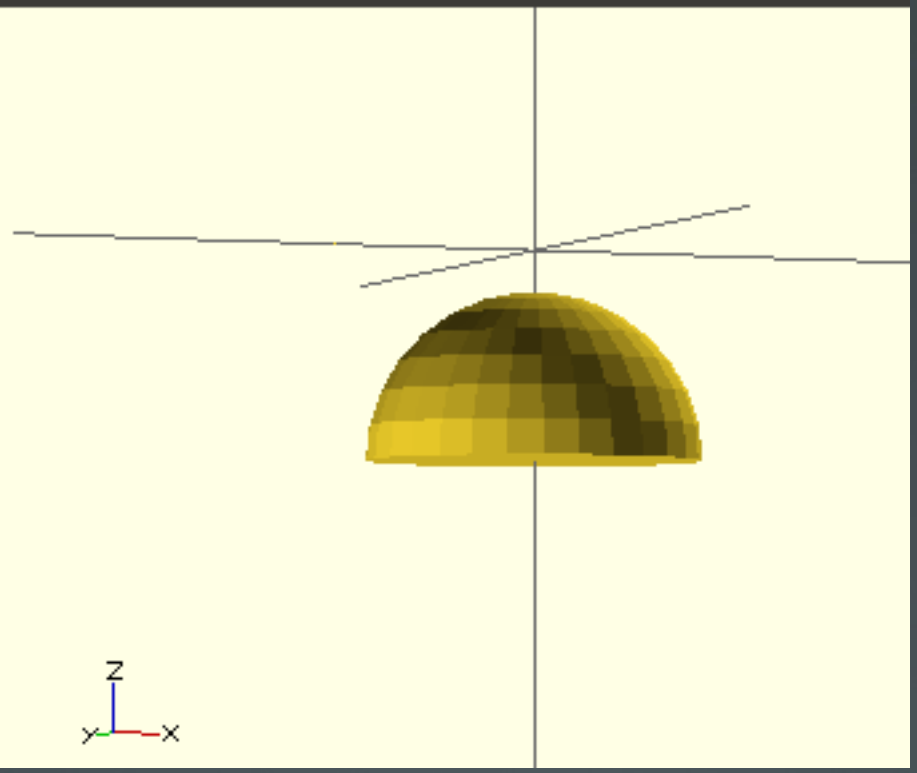


Half Sphere - intersection

File Edit Design View Help

```
intersection() {
   cube( [10,10,10],
        center=true);

 translate( [0,0,-5] )
   sphere( r=4, $fn=30 );
} // end difference
```



More than one way to skin a cat

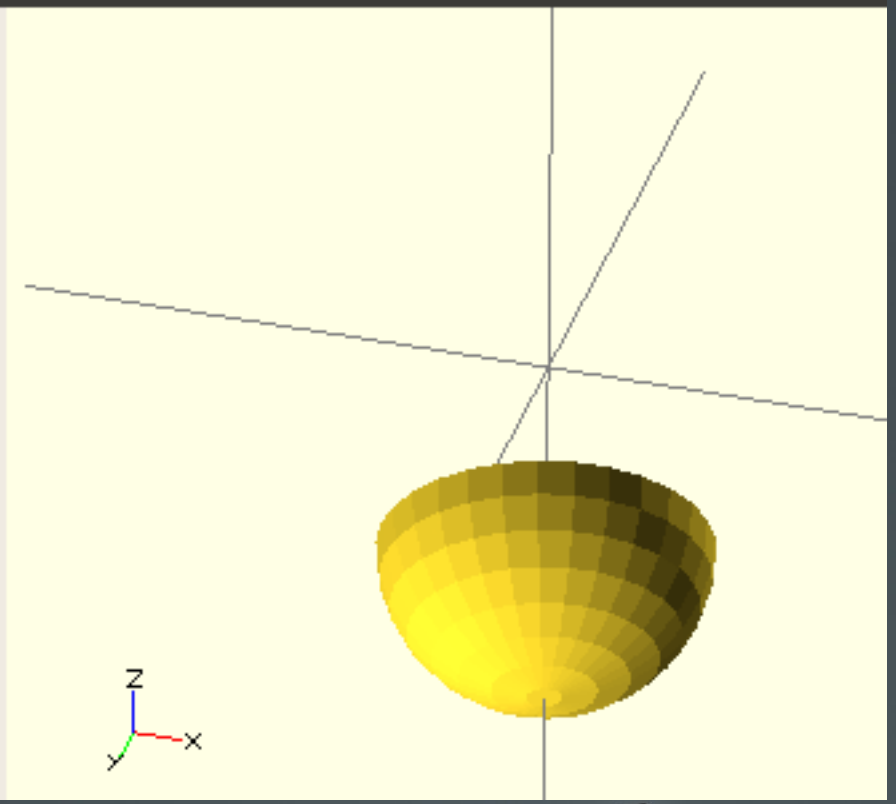
File Edit Design View Help

```
difference() {

 translate( [0,0,-5] )
   sphere( r=4, $fn=30 );

 cube( [10,10,10],
       center=true);

} // end difference
```

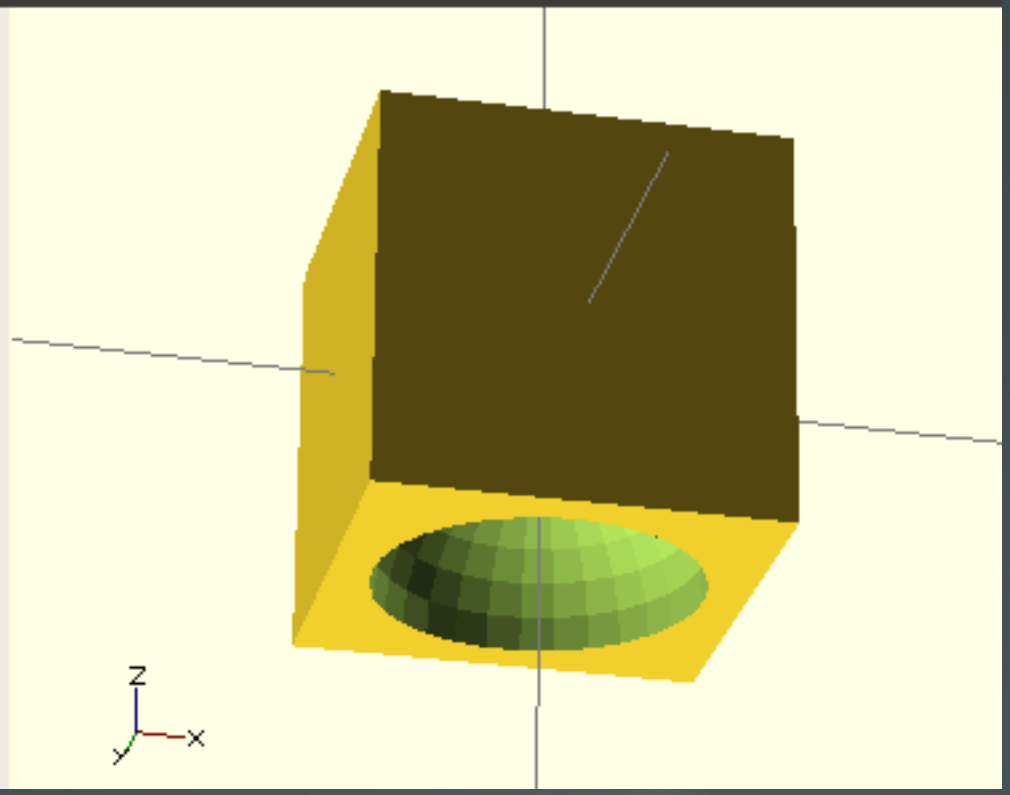


Order Maters for Difference!

File Edit Design View Help

```
difference() {
   cube( [10,10,10],
        center=true);

 translate( [0,0,-5] )
   sphere( r=4, $fn=30 );
} // end difference
```



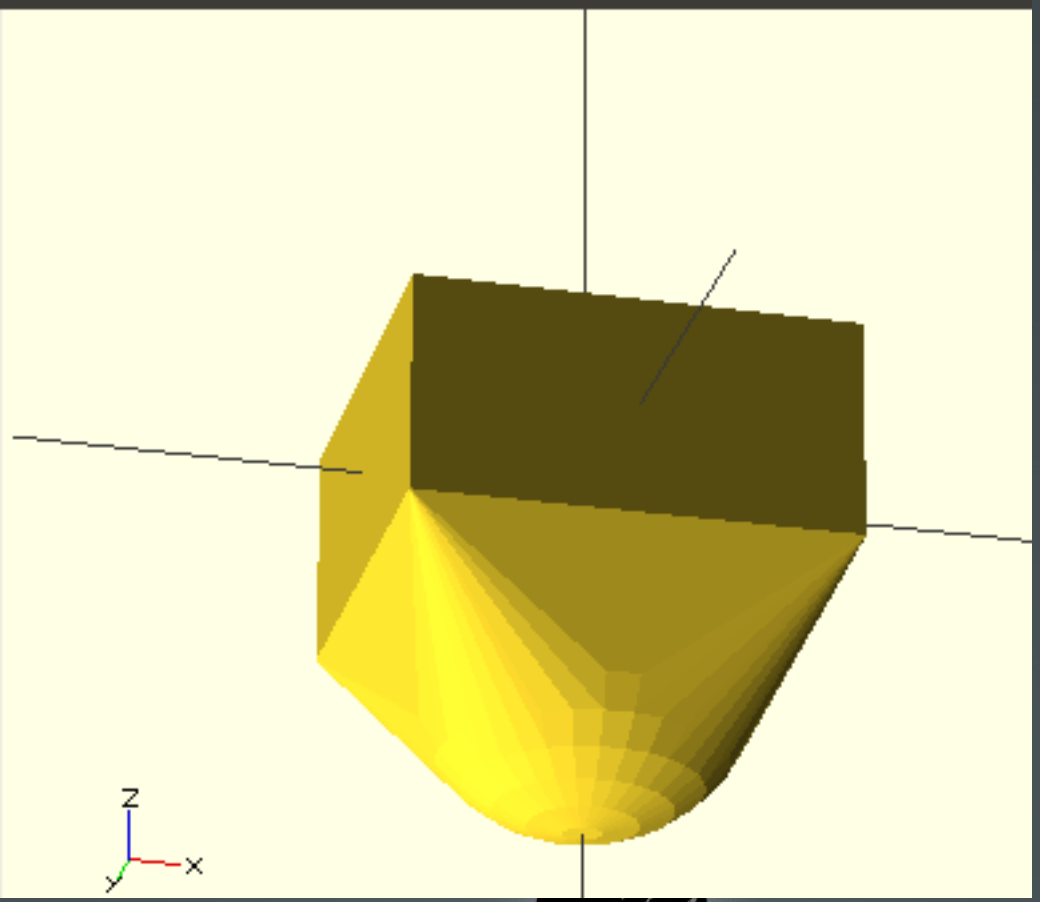
hull

File Edit Design View Help

```
hull() {
 translate( [0,0,-4] )
   sphere( r=4, $fn=30 );

 cube( [10,10,5],
       center=true);

} // end hull
```



Modifier characters

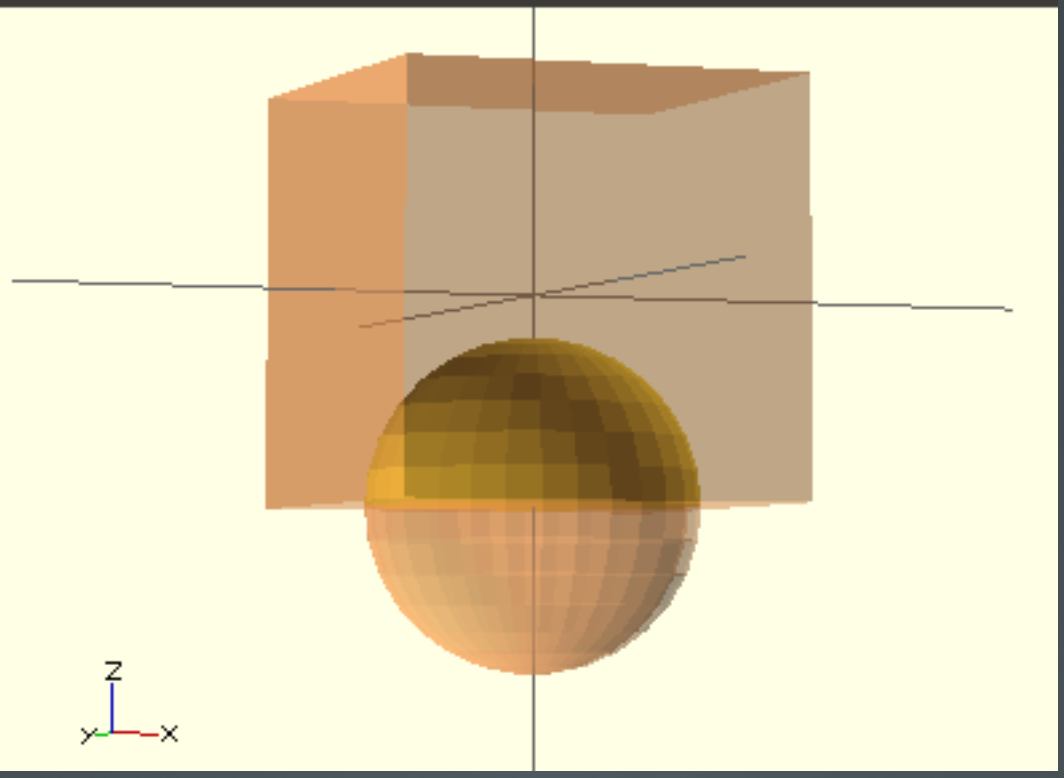
- Characters (#,!,%,*) prepended to objects to modify how they are processed.
- Generally used to visualize what is happening, try out a limited set of code without other code interfearing, etc.
- Most useful is the # or Debug Modifier, that draws objects in transparent pink for visualization purposes.

intersection – Debug View

File Edit Design View Help

```
intersection() {
   #cube( [10,10,10],
        center=true);

 translate( [0,0,-5] )
   #sphere( r=4, $fn=30 );
} // end difference
```



Other Modifier Characters

- The other modifier characters actually affect how your output is generated.
- % - background modifier - draws the sub-tree/object with transparent gray, but ignores it for all other rendering purposes.
- ! - root modifier – Ignore everything ELSE in your file, and only render this sub-tree!
- * - disable modifier – Disable/ignore this subtree.

Resource Links

- Downloads:
  http://www.openscad.org

- User Manual:
  http://en.wikibooks.org/wiki/OpenSCAD_User_Manual