Name	:	OS++; 80/20 Part A (filename=./2022doco/doco_A_80-20.pdf)
Purpose	:	80/20. ie 20% of OS++; to give 80% of functionality for a beginner
By / When	:	Stephen George / Winter 2022

Introduction

Oct 22 is the framework date.

The OS++ libraries come in “The_download”. To get the download, click on the icon similar to the one on the left. This icon should be at the top of every page on the website osplusplus.org.

NB It might have a different framework date. But will look very similar.

Once extracted “The download” will have a file called Learn_Hard1A_v1_0_0.scad, or something similar, it’s in the root directory. This is a list of examples and is the best way to learn in my view, and not this doco.

So, Open Learn_Hard1A*.scad in the openscad program (download from https://openscad.org/.) and work through the examples.

NB Learn_Hard1B_v1_0_0.scad will follow when the next set of Libs is ready

When all else fails, then “Read That Funky Manual”. This is part A of that manual. It’s aim is to give you 20% of the language, which you will use 80% of the time. When we hit a rabbit hole, which does not meet this 80/20 criteria, the issue is discussed in the doco_80-20_indepth.pdf.

Syntax

(Non standard features in this document)

***** text *****	- A rabbit hole, to continue down it see doco_80-20_indepth.pdf
Different colour	- A different personality, frame of reference, consider it another narrator

Chapter 0 – (Skip this goto Chapter 1)

Confused by the Chapter 0 preamble? Don't worry Just skip to "Chapter 1 – in the beginning"
"You are not expected to understand this" - Dennis Ritchie. (Well, not in the beginning anyway).
So why is it here? Because in a few pages, it describes most of the rest of the doco.

Terms

(Terms used in this document)

The Download	- Meaning the OS++ standard code library, all compressed
Anchor point	- Point that an object is rendered from or in relation to
Translate	- Command that moves an Anchor point to another point in space
Shape's Anchor point	- Point that the shape will be rendered from. Cube is x=0,y=0,z=0
Rcuboid	- Cuboid with Rounded corners and edges (useful in 3dprinting)
xyzVector	- Cartesian co-ord in format like [10,3,6] used in translate
Primary	- Short for Primary shape. An example of a primary would be a Cone
Definition datatype	- A datatype that gives you all the details of a primary
STL	- an STL is a generic interpretation of a model, used in 3dprinting
OS	- OpenScad the binary code base. Ie Standard_OpenScad
OS++	- OpenScad language frawework

Data definitions Terms

(Key for Data definitions)

+	- Term is positive or zero
-	- Term is negative
+-	- Term is positive, negative or zero
L	- Term is a length (can be in any direction)
X	- Term is a X value or magnitude. In the X Cartesian dimension
Y	- Term is a Y value or magnitude. In the Y Cartesian dimension
Z	- Term is a Z value or magnitude. In the Z Cartesian dimension
D	- Term is a Diameter
"example"	- Term is a string, contains an example of string

Data definitions

(Name,Syntax,Library,Notes)

Cuboid	[+X,+Y,+Z]	lib_cuboid	Corners and edges are sharp
Full Rcuboid	[[+X,+Y,+Z],+D]	lib_cuboid	Corners and edges are rounded by D
Semi Rcuboid	[[+X,+Y,+Z],+D,"-Z"]	lib_cuboid	Some Corners and edges are rounded
Cylinder	["Z",+L, +D]	lib_cone	D specifies both end circles of cylinder
Cone	["Z",+L, +D0,+DL]	lib_cone	D0 start circle D, DL last circle Diameter
Sphere	[+D]	lib_sphere	
xyzVector	[+-X,+-Y,+-Z]	*	Can be used to create a Cuboid if +

Face names

(Name,Library,NB,Notes)

Z0	lib_cuboid	Where Z=0;	Bottom of Cuboid
ZL	lib_cuboid	Where Z=L; Last bit of cuboid	Top of Cuboid
X0	lib_cuboid	Where X=0	
XL	lib_cuboid	Where X=L; Last bit of Cuboid	Opposite of X0
Y0	lib_cuboid	Where Y=0	
YL	lib_cuboid	Where Y=L; Last bit of Cubpoid	
Z	lib_cuboid	Short for Z0ZL	Top and Bottom
X	lib_cuboid	Short for X0XL	
Y	lib_cuboid	Short For Y0YL	
D0	lib_cone	Where D=0, ie first	Diameter of bottom circle (D1 in OS)
DL	lib_cone	Where D=Last	Diameter of top circle (D2 in OS)

Quick Framework

Confused by the framework? Don't worry Just skip to "Chapter 1 – in the beginning"

1) The libraries are stored in ./z_lib and their version is called by a master library called OS++.scad at the start of your code. eg see Template_minimal_v1_0_0.scad or similar for an example of it's usage.

2) An example of a library is ./z_lib/lib_cuboid_v1_0_3.scad. It deals with the primary type cuboids, ie Cubes, Cuboids and Rcuboids (Cuboid with at least 1 rounded edge or corner).

3) All code in lib_cuboid has a name that starts lib_cuboid*. Code starting with an _lib_cuboid is a private function.

4) lib_cuboid and friends are completely standalone and will render on their own. There will be a remmed out example at the top of the library. Do not leave it unremmed.

5) lib_cuboid friends are lib_cone (Which deals with cylinders and cones) & lib_sphere (which deals with spheres) and they follow the same basic rules.

6) lib cuboid and friends have 3 main parts :

A) **Datatype**. This defines the cuboid eg [1,2,3] means a cuboid with dimensions 1,2,3
B) **Render**. This renders the Datatype from a fixed point, known as an anchor point.
C) **Move**. This will move the anchor point relative to a primary of your choice. (grid=Cartesian)

7) OS++20221031.scad will load specific individual libraries and versions. Code tested on this will always work as the library has become canon. Each library has a "hello" statement which is displayed in the console. This "hello" outlines the library name, it's version and it's main help module. This help module can be run to display help in the console.

8) The library's specific main help module will echo a copy and pastable example of the library's user code in the console. Along with a list of sub help modules.

Example of the three main areas of the libraries

A) Define a cuboid datatype and load it into a variable

```
// X length is 10, Y length is 20, Z length is 30,
cuboid_YourDescription=[10,20,30];
```

B) render or display the datatype, using an anchor

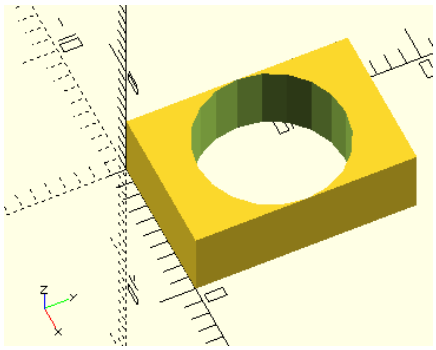
```
// 000 means use point x=0,Y=0, Z=0 as a fixed point to render the cube
lib_cuboid("000",cuboid_YourDescription);
```

NB if this does not copy and paste into openscad properly, delete the " and add them in again. " gets translated into a different char on openscad which looks like a double quote but is not, due to the font translation.

C) A move. Move the anchor point to the Middle of X,Y and keep Z=0 with reference to the cuboid known as cuboid_YourDescription

```
translate(
 lib_cuboid_mv("000","MM0", cuboid_YourDescription)
 ) // no ; as this is a translate
```

Pasteable example .



```
// copy and paste able into Template_minimal_v1_0_0.scad
// which can be found in “the download”

//http://osplusplus.org/2022standard/professional

include <z_lib/OS++20221031.scad>;
$fn=20;

cuboid_body=[10,15,5];

difference()
{
lib_cuboid("000",cuboid_body);
translate( lib_cuboid_mv("000","MM0", cuboid_body) )
cylinder(30,5,5);   // openscad internal cylinder
}
```

Confused? Don’t worry. Your not meant to understand the framework, right at the beginning. It’s just a skeleton to show you what is going on under the cover. Ie a broad map to hang ideas on.

Just skip to “Chapter 1 – in the beginning”

Chapter 1 - In the beginning

In the beginning, there was openscad and it is pretty awesome. It meant I could use C / bash / awk to generate scad/text code which could output into a 3dprint.

Openscad_standard has the primary shape called **cube([10,20,30]);** This means it generates a cuboid with dimensions of X=10, Y=20, Z=30. It generate that shape from an anchor point of X=0,Y=0,Z=0. Ie If you increase the value of X,Y or Z it appears to grow from the bottom left corner.

Openscad_standard has another primary called **cylinder(h=10,d1=5,d2=5);** This means it generates a cylinder with height of 10 and a D1 and D2 of 5 in the Z axis. If you increase these value it appears to grow out from the middle of the bottom circle, where Z=0;

I wanted to have code that rendered a cylinder from X=**0**,Y=**0**,Z=**0** when I was thinking in "cubes" or Cartesian coordinates. And so I wrote a module called.

cylinder_000(h=10,d1=5,d2=5); // "000" means render a cylinder from X=0,Y=0,Z=0 like a cube

This was great for thinking in cubes, but rendering cylinders. But when I was thinking in cylinders and I wanted to render a cube from the **M**iddle of it's bottom square. What would I call that? How about X=Middle, Y=Middle and Z=0. "MM0" for short. So the following was born

cube_MM0([10,20,30]); // MM0 means render from the Middle of it's bottom square, where Z=0

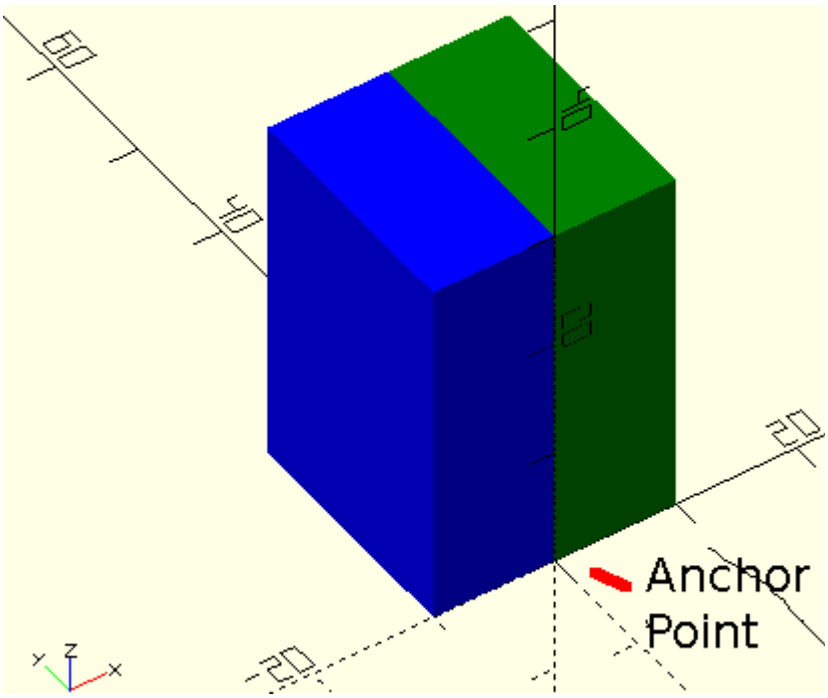
But if 0 is the start corner of the cuboid primary and M is the middle point of a primary. What is the end or **L**ast point? I decided **L**. L is the 12th character of the alphabet. This meant, that I had to write code for the following modules

```
cube_000([10,20,30]); // cuboid 1     - Identical to openscad_standard cube
cube_M00([10,20,30]); // cuboid 2     - Creates cuboid using X as the Mmiddle, Y=0start, Z=0start
cube_L00([10,20,30]); // cuboid 3
cube_0M0([10,20,30]); // cuboid 4
cube_0L0([10,20,30]); // cuboid 5
….
cube_MM0([10,20,30]); // cuboid ?    - Creates a cube, like a cylinder is created in openscad.
cube_MMM([10,20,30]); // cuboid ?    - Creates a cube, like a sphere ie from it's z=Mmiddle
…
cube_LLL([10,20,30]); // cuboid 27 ie 3^3=27
```

But what was this "000" thing to be called? After all, the path to wisdom starts by giving things their correct name. I decided on the term "Anchor point". After all if a ship is tied to an anchor, it might point one way when the tide comes in, and the other when the tide goes out. But the actual point would remain the same.

So, a rendered Cuboid with a shape anchor point of "000" would point one way (Quadrant 1). A cuboid with a shape anchor point of "L00" would point another (Quadrant 2). But the actual point that it's tied to is still X=0,Y=0,Z=0

What are you talking about? Perhaps show the user a picture? Okay.



Picture of two cuboids. The green cuboid grows out of X=0,Y=0,Z=0 as is standard with the openscad cube.

The blue cuboid is exactly the same cuboid. It has the same anchor point in space. But rather than growing out of “000” it grows out of “L00”.

Ie X is the (L)last point not the (0)start point of the cuboid.

- Same cuboid, same anchor point. Different tide direction or “shape render/anchor point”

So in the example, it’s the same Cuboid (in two different colours), with the same anchor point. But the cuboid is rendered in a different relative way.

Before we look at the code, let’s recap some Openscad stuff.

Let’s recap our Openscad knowledge (so we are on the same page)

- A Openscad model starts with an anchor point of X=0,Y=0,Z=0 on an euclidean plane/volume.
- You can move this anchor point using the translate command.
- When a openscad_standard cube is rendered from the anchor point, it is render from it’s bottom left hand corner.Ie from the cubes perspective the anchor point is X=0,Y=0, Z=0. “000” for short.
- When a openscad_standard cylinder (center=false) is rendered from the anchor point, it is rendered from it’s middle bottom circle. Ie from the cylinders perspective the anchor point is at the bottom middle of the shape ie X=Middle, Y=Middle, Z=0 or start
- When a openscad_standard sphere is rendered from the anchor point, it is render from it’s centre and grows out. Ie from the spheres perspective the anchor point is at the middle of the shape ie X=Middle, Y=Middle, Z=Middle.

All the above, is standard openscad.

- OS++; allows you to render a cuboid in the same way as a sphere (ie MMM) or a Cylinder Ie (MM0).
- OS++; Also introduces the idea of the L(last), as well as M(middle) and 0(start)

The actual os++ code for the example is

```
//http://osplusplus.org/2022standard/professional

include <z_lib/OS++20221031.scad>;
$fn=20;

// Define primaries
cuboid_example=[10,20,30]; // Xlength=10, Ylength=20, Zlength=30

main();

module main()
{
// render it green from the shapes anchor point "000" (with F5 to see color)
color("green")
lib_cuboid("000",cuboid_example);
```

```
// render it Blue from shapes anchor point "L00" (with F5 to see color)
color("blue")
lib_cuboid("L00",cuboid_example);

} // end of module
```

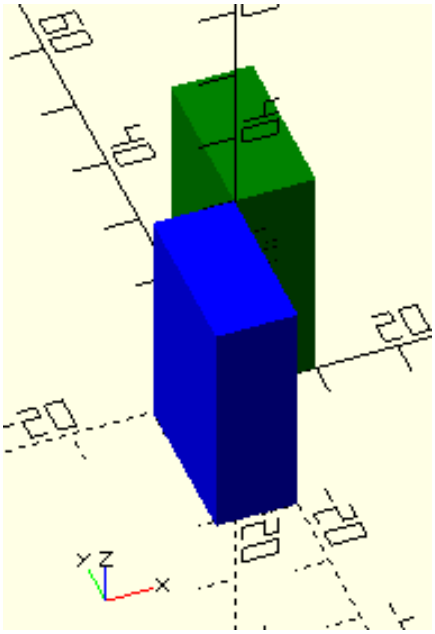
You might have noticed that “cube_000() is now lib_cuboid(). There are a couple of reasons for that.

1) It’s a cuboid, not a cube and I like to at least try and be accurate.

2) lib_cuboid is the name of the library. So you know the module code will be in lib_cuboid.scad in the Z_libs directory if you want to check it out.

3) The anchor has migrated from the name of the module to a parameter of the module. This means there is only one lib_cuboid module. Rather than 27 cube_XYZs. (But they once existed in a version a long long time ago. Unfortunately)

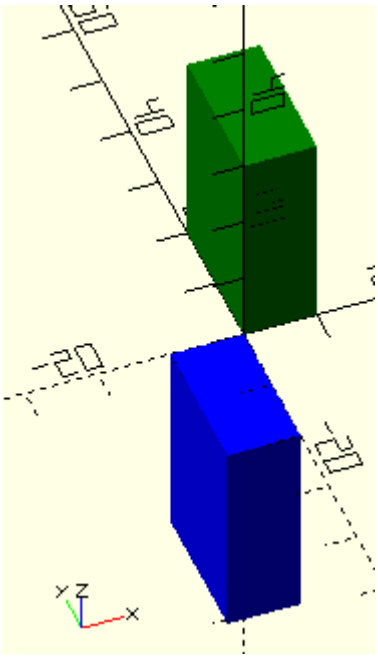
Let’s change the blue cuboids shape anchor point from L00 to LL0. What do we get?




```
// This is exactly the same code but for 1 letter change


//http://osplusplus.org/2022standard/professional

include <z_lib/OS++20221031.scad>;
$fn=20;

// Define primaries
cuboid_example=[10,20,30];

main();

module main()
{

color("green")
lib_cuboid("000",cuboid_example);

color("blue")
lib_cuboid("LL0",cuboid_example);

} // end of module
```

You can see that we still have the same anchor point of 000, the openscad_standard. We still have the same cuboid. But how it is rendered is different. Instead of rendering from the bottom left most corner (Quandrant1) as is standard for the cube command. We now render the cuboid from the perspective of a different corner. (Quandrant3)

Let’s change the blue cuboids anchor point from LL0 to LLL. What do we get?



```
// This is exactly the same code but for a new shape
render anchor


///http://osplusplus.org/2022standard/professional

include <z_lib/OS++20221031.scad>;
$fn=20;

// Define primaries
cuboid_example=[10,20,30];

main();

module main()
{
color("green")
lib_cuboid("000",cuboid_example);

color("blue")
lib_cuboid("LLL",cuboid_example);

} // end of module
```

So, the blue cube is rendered using (0,0,0) but from the shapes perspective that anchor is used to denote it top, right most corner. Rather than it's bottom left most corner as an openscad cube normally is.

My next problem was that I would design a model while thinking in cuboids and the next I would be thinking in cones (The super set name for cylinders – as a cylinder is just a cone with the same top and bottom circle diameter). In effect I needed to "translate or Move" the anchor point from "000" to "MM0" of the primary shape I was dealing with, and back, depending on what thought pattern of thought worked best for the problem at hand.

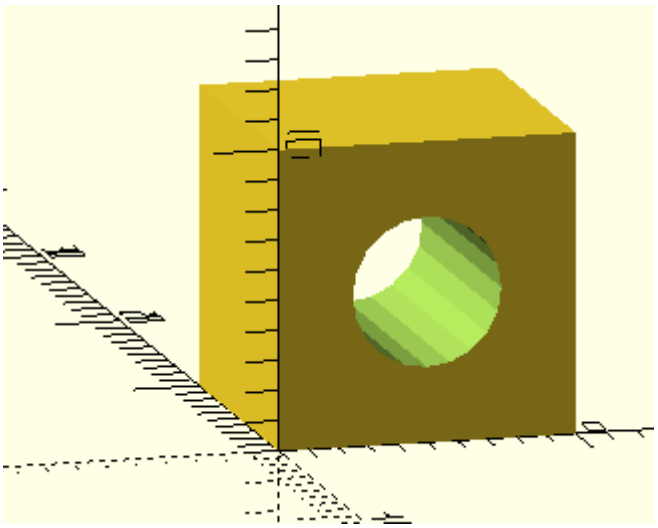
What?

Let me put it another way.

- Openscad start anchor point is X=0, Y=0, Z=0
- I render a cube using this anchor point. I call this cube. **cuboid_example1_plus**
-
I now want to remove a sphere from this cube's middle. Therefore I need to move the anchor point from the cubes's anchor point of 000 to MMM .
-
The command I do this with is a translate. But I need code to calculate the xyzVector.
-
translate(lib_cuboid_mv("000","MMM",cuboid_example1_plus)) // no ; on the end

Are you trying to melt their brains, What do you mean,exactly? Perhaps I should do another example with a cylinder? Okay.

Example (To show code result)



Code snippet (needs OS++ framework to run, explanation below)

```
//http://osplusplus.org/2022standard/professional

include <z_lib/OS++20221031.scad>;

$fn=20;

// Define primaries
cuboid_example1_plus      = [10,10,10];  // cube of X length=10mm etc
cone_example2_minus       = ["Y",10,5];  // cylinder pointing in Y, length=10,Diameter=3
//
// Render and put defined primaries
difference()
{
lib_cuboid("000",cuboid_example1_plus); // render cuboid data definition

// move the anchor point
translate(lib_cuboid_mv("000","M0M",cuboid_example1_plus)) // no ; on the end
 {
 lib_cone_rGrid("M0M",cone_example2_minus); // remove cylinder
 } // end of translate
//

// Show me the results of the magic function "lib_cuboid_mv" used by translate
 echo (
  "Debug show lib_cuboid_mv()", // Output should be in console
   lib_cuboid_mv("000","M0M",cuboid_example1_plus) // Output xyzVector to get 000 - MOM
    );

// Result in console is ECHO: "Debug show lib_cuboid_mv()", [5, 0, 5]


} // end of difference
```

Before I continue, I would like to point out that **if** there is an error in the user code, there seems to be no way to stop openscad from continuing “compilation” from the library code. As a result you can get a wall of red error messages in the console, which is quite intimidating.

If this happens to you, scowl to the top of the errors in the console. Now usually it is quite obvious what the problem is with the first error. Usually it’s a misspelled variable or module name.

Anyway back to the show.

Update: openscad has a “stop on first warning” tick box. See Menu edit, select Preferences, tab advanced, tick box at bottom.

Code explanation

The reader skip-able header is below. Line 1 has a Remark (Rem) that specifies the grade of the user code to follow. Line 2 loads up a specific version of the OS++ library. And Line 3 specifies the “resolution” for circles and cones.

```
//http://osplusplus.org/2022standard/professional

include <z_lib/OS++20221031.scad>;

$fn=20;
```

We then have defined a cuboid called **cuboid_example1_plus.** All we know is that it has a X dimension of 10, a Y dimension of 10 and a Z dimension of 10. ie

```
cuboid_example1_plus  = [10,10,10];
```

Next We have defined a cone called **cone_example2_minus.** All we know is that the Cone points in the Y direction. Has a length of 10mm and the bottom circle is 5mm. Since we don’t have another number we assume the top circle is also 5mm and therefore the cone is actually a cylinder (cone where D0 and DL just happen to match).
Ie

```
cone_example2_minus   = ["Y",10,5];
```

Next We call **“difference”** (which is standard openscad) and render the cuboid **cuboid_example1_plus.** We render this shape using the the shapes anchor point of 0,0,0. Which is the same as the openscad_standard cube command.
ie

```
difference()
{
lib_cuboid("000",cuboid_example1_plus);
```

Next we translate/move the anchor point, that we render the primaries from. To do the translate we use the the shape called cuboid_example1_plus as a guide or reference. We move from point 0,0,0 to where X is the (M) middle of the shape (cuboid_example1_plus). Y is still (0)start. And Z is the (M) middle of the shape (cuboid_example1_plus) .

ie

// move the anchor point
translate(lib_cuboid_mv("000","M0M",cuboid_example1_plus)) // no ; on the end

You will have noticed that the standard openscad translate command was fed by a function called lib_cuboid_mv. This means:

1)The code of this module can be found in the lib_cuboid library
2) It's command is mv or "**M**o**Ve**s" the anchor point.

If you decided to echo the command, instead of doing a translate you would get

ECHO: "Debug show lib_cuboid_mv()", [5, 0, 5]

[5, 0, 5] is a standard input for the "translate" command that comes as standard for openscad.

Since we are going from 0,0,0 to M,0,M; The calculation for [X,Y,Z] is

X=half or middle of cuboid X (ie 5)

Y=0, as we are going from 0 to 0 (ie 0)

Z= half or middle of cuboid Z (ie 5)

we get

[10/2 , 0 , 10/2]

or [5,0,5]

Whats your point?Well you can take that code and try any anchor combination to see what the echo results are and get a feel for what is going on.

Anyway back to the code

Next, in the code example, we remove the cone/cylinder called cone_example2_minus from the previously rendered Cuboid. The cylinder grows from our new translated anchor point and grows in the +Y direction as y=0 is the bottom.
ie

```
{
lib_cone_rGrid("M0M",cone_example2_minus);
} // end of translate
```

You will notice that the render is in a set of brackets ie {}, this means that the block enclosed will all render from that translate/anchor starting point. Once that block closes } we go back to the original anchor point. Hence the rem // end of translate.

This is insane, how can I remember all this?

1) There is a cheat sheet pdf that outlines all the useful code.in the download.
2) You have lib_cuboid_help(); module in the library itself.
3) You can read the code in the library directly and write a reference manual.
4) You can pay me oodles of money to write a reference manual. For the benefit of mankind.

Syntax recap

- An example of a cuboid definition is

cuboid_main=[10,20,30];

Where

10 is the Cuboid in the X dimension
20 is the Cuboid in the Y dimension
30 is the Cuboid in the Z dimension

Chapter recap

- Shape’s Anchor point is where the shape grows from
- The Anchor point can be moved using a translate command
- The translate command is fed by an mv command to move the anchor point.

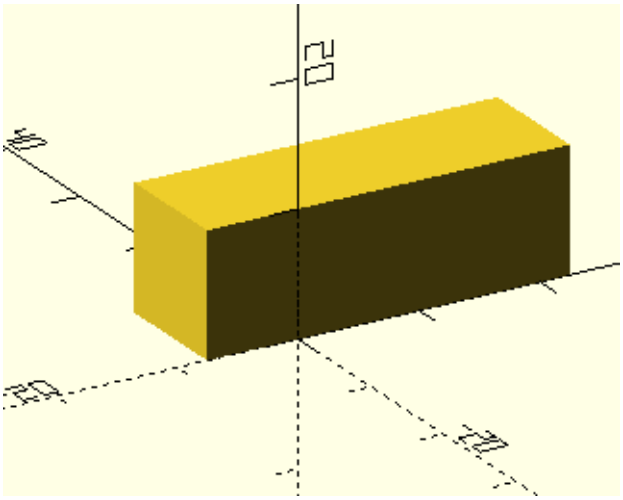
- The mv command uses the primary shape itself to calculate the translate
- When a shape is rendered from the anchor point. You need to specify how it is rendered.
- 000 means the anchor point is the most left, bottom corner.
- MMM means the centre. Or Middle,Middle,Middle
- LLL means the top most right corner. Or Last,Last,Last

Next Chapter – Base, how low can you go

So if 0,M,L is base 3. where 0 is zero, and M is ½ way through the shape. And L is the full way ie 1/1 or at the end. Depending where these letters are will in the string, will denote if it is in the "XYZ" dimension.

I know what you are thinking. What would base 12 look like? Ie 0 ,is well zero, A is 1/12. C is (3/12) or 1/4, D is 1/3 (4/12). F is 6/12 and L is 12/12 or the full way etc. Did I mention that L is the 12th character? And you thought L means Last bit of the shape? You have been in base 12 all this time.

So "C00" is 1/4th of X with Y=0 and Z=0. "C00"L huh? You need to get out more.

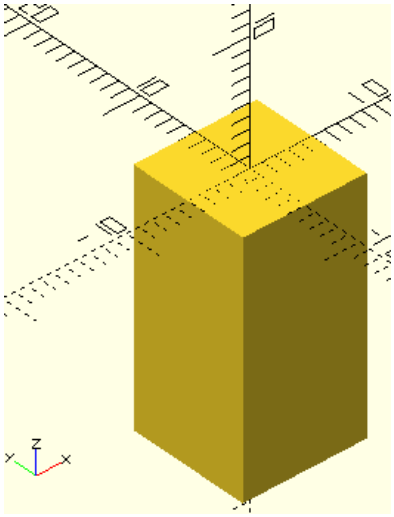


```
//////////////////
// Code snippet
//

// define the cubeoid
cuboid_example1 = [30,10,10];

// render it
lib_cuboid("C00",cuboid_example1);


// End
//////////////////
```

Base 10 - (deprecated numbering system – For people who don't wear shoes)

I have been lead to believe that there is still a small group of people out there, who like to think in base 10. I mean why? Can't represent 1/3 easily, can't represent 1/4. It's almost as if they decided to base their whole number system on something arbitrary, like the number of their toes. I suppose it's good for 1/5th. But who thinks in 1/5th? and what happens when they wear shoes?, they are stuffed then So, base 10 is supported for backward compatibility reasons. "200" means that X is 2/10 or 1/5th. Base10 is a deprecated numbering system, You should use base12 for geometry, base16 for code electronics, base4 for DNA. Unicode, what a waste of space! Character 7 is a literal bell sound, I mean really. Who "displays" a bell sound on a monitor? Can we get on??

Shape render anchor points "000" (cube), "MM0" (cone) and "MMM" (sphere) come up a lot. But a big one also seems to be "MML". It means "grow down". Perhaps a picture?

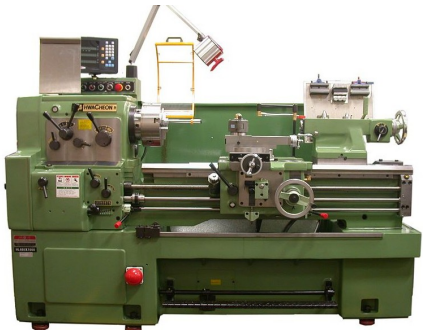


```
//////////////////
// Code snippet
//

// define the cuboid
cuboid_example1 = [10,10,20];

// render it
lib_cuboid("MML",cuboid_example1);

// End
////////////////////
```

Chapter recap
- Anchor point is where the shape grows from as you increase it's dimension values.
- When you render a shape you can specify where that anchor point is on the shape.
- You can use base3. Ie 0,M,L. Which corresponds to Beginning, Middle, Last
- You can use base 12 ie A-L. For 1/12, 2/12, etc
- You can use base 10 ie 0-9. For 1/10, 2/10 etc
- You can mix and match bases. "L0L", "AA1", "M0B"

Chapter 2A

Cones. In the past they were created by lathes.



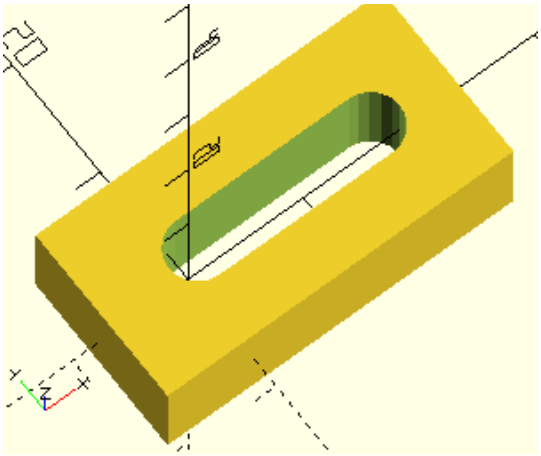
And what a cone they could make. Put a work piece in the chuck and spin it up. You now can cut it to the correct diameter. It is an art form. It's all about speeds and feeds.

And not leaving the chuck key in the chuck when you turn it on. Well ,not unless you want a hole in a wall.

Quite.

Slots were created by Mills (milling machine).

It's essentially a heavy duty drill press that can, drill a round hole. But then move the "drill bit" along to make a slot.



So a mill spins a "drill bit", called a mill bit, and you can draw square slots. But the corners are rounded to the size of the mill bit.

The lathe was the 3dprinter of it's own time, as it could be used to replicate itself. You can also mill in a lathe.

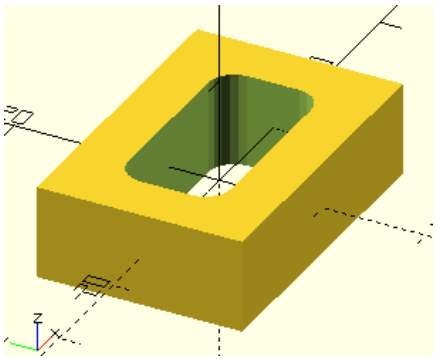
Right, Which leads us nicely on to the Rcuboid or rounded cuboid. A rounded cuboid is a cuboid with rounded edges and corners. A standard cuboid datatype would look like this

[20,10,30];

A rounded cuboid would look like this

[[20,10,30],5]; // where 5 is the diameter of the roundness or mill bit.

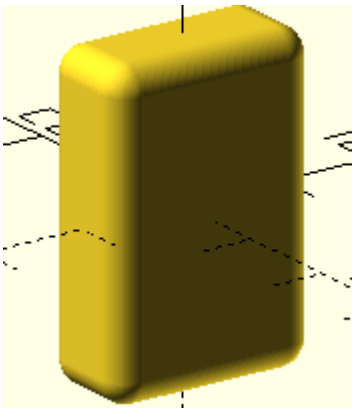
So how do we think in terms of a mill and slots? If I have a 5mm drill bit and I cut a square out of a bar of metal I would get as follows.



```
//////////////////
// Code snippet

// define the primaries
cuboid_bar  = [30,20,9];
cuboid_slot = [[20,10,30],5]; // 5=mill bit size

// render it
difference()
{
lib_cuboid("MMM",cuboid_bar);
// minus
lib_cuboid("MMM",cuboid_slot);
} // end of difference
```

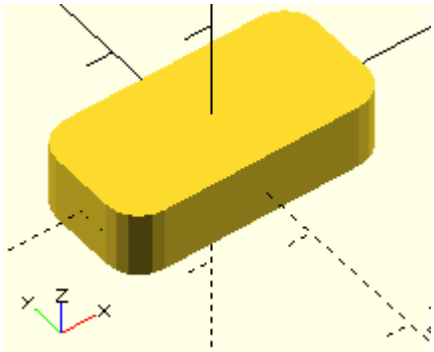
I have cheated slightly here. The data type [[20,10,30],5]; is more than the slot. It's a 3D shape where the corners are spheres of diameter 5mm. If I render the "minus" shape ,you see that it has spheres on the end. See below.

By removing the Rcuboid from the slot you only saw the negative cross section. But here you see the Rcuboid in all it's rounded glory.



```
//////////////////
// Code snippet

// define the primaries
cuboid_slot = [[20,10,30],5]; // 5=corner diameter

// render it // like a sphere ie MMM
lib_cuboid("MMM",cuboid_slot);
```

If I were to do the slot properly I would need to define an Rcuboid with a flat top and bottom. Ie the Z axis faces would be flat. Ie the Z bottom and Z top are flat. What's the Os++ code for that?




```
//////////////////
// Code snippet


// Z sides are flat. Not rounded
cuboid_biscuit = [[20,10,5],5,"Z"];

// render it
lib_cuboid("MMM",cuboid_biscuit);
```

```
// other examples
cuboid_biscuit = [[20,10,5],5,"X"];
cuboid_biscuit = [[20,10,5],5,"Y"];
```

Again I have cheated. I am detecting a pattern here. We need to protect them from the whole truth.

[[20,10,30],5]; is a Data type; where all the corners have a sphere on the corners with diameter of 5.

[[20,10,30],5,"Z"] is a Data type; where all the corners have a sphere on the corners with diameter of 5 except the top and bottom. But that is really not it's correct "name". "Z" is converted to "-Z" ie remove or make flat the Z bits. Ie top and bottom.

```
Which is actually short for [[20,10,30],5,"-Z"];
```

Ie remove the roundness on the Z planes.

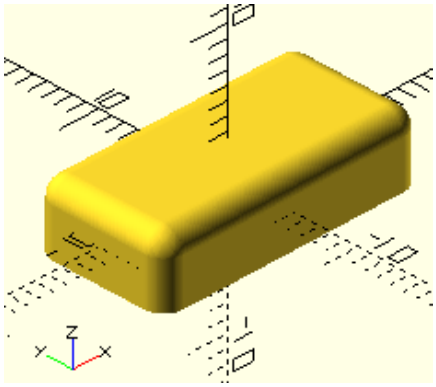
But what about having rounded sides except its bottom? You mean where Z=0? We could shorten it to Z0 What would the opposite of Z0? You mean the top, ie the **L**ast bit of the shape? Yes. ZL for the last square? What about X and Y?. X0, Y0 …

Essentially we need a name for each face of a cuboid, so we can tell OS++ which faces are to be rounded or not rounded.

X0 = X face where X=0 (Runs along the Y axis). The first square of the shape.
XL = Opposite side of X0, The Last square of the shape.
Y0 = Y face where Y=0 (Runs along the X axis)
YL = Opposite side of Y0
Z0 = Z face where Z=0 (The bottom)
ZL = Opposite side of Z0 (the top)

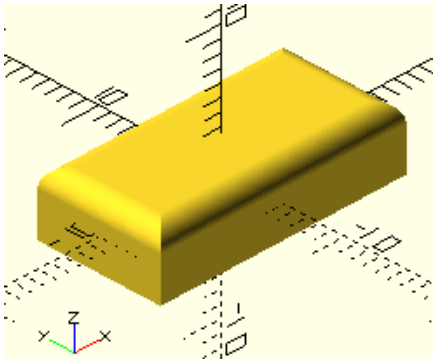
X means X0 and XL
Y means Y0 and YL
Z means Z0 and ZL

The order (or precedence) is "X0,XL,Y0,YL,Z0,ZL" so "X0XLZ0" is valid but can be shortened to "XZ0" while "Z0XL" is not valid as XL comes before Z0.



```
//////////////////
// Code snippet

// Z0 is a flat bottom
cuboid_biscuit = [[20,10,5],3,"Z0"];

// render it
lib_cuboid("MMM",cuboid_biscuit);


*** Why has the mill bit dropped to 3 from 5 ? ***
```

So, we have a Rcuboid. with all rounded corners. And we have a language to take some of those rounded corners away again and make them pointy. How about the opposite? Ie We start with a standard “square” cuboid with nice pointy sides and add a single rounded face. Because “-X0XLY0YLZ0” is going to get pretty ordinary, pretty fast. How about “+ZL”? means add a single rounded face to a “square” cube? And “-ZL” means take away a rounded side from a rounded side? And if a sign is not given it defaults to – so that all the code I have already written before this code came out, still works? Deal.



```
//////////////////
// Code snippet

// ZL is the only rounded top
cuboid_biscuit = [[20,10,5],3,"+ZL"];

// render it
lib_cuboid("MMM",cuboid_biscuit);
```

Note:

cuboid_biscuit1 = [[20,10,5],3,"-X0Y0Z0"]; // ie start with an Rcuboid and remove roundyness
is NOT the same as
cuboid_biscuit2 = [[20,10,5],3,"+XLYLZL"]; // ie start with a cuboid and add roundyness

Because an added rounded edge interferes with the surrounding edges, while a removed flat edge is it just cut away. And I can’t explain it any better than that. And I only discovered than while I was deep in the middle of writing this code.

Syntax recap

- An example of a Rcuboid definition is

cuboid_main=[[10,20,30],4,”-Z”];

Where
10 is the Cuboid in the X dimension
20 is the Cuboid in the Y dimension
30 is the Cuboid in the Z dimension
4 is the diameter of the Sphere on each corner
The string is a list of faces which are not rounded

Chapter recap
- A cuboid has 6 faces
- They are named X0,XL,Y0,YL,Z0,ZL. In that order
- When referring to a list of faces they have a precedence of X0,XL,Y0,YL,Z0,ZL.
- Often we need to name two faces on axis like Z0ZL, So you can just use Z
- An Rcuboid has rounded corners
- You can start with a standard cuboid and added rounded faces eg +Z
- You can start with a Full Rcuboid and remove rounded faces eg -Z
- For backwards compatibility. “Z” actually means “-Z” ie the default is -

Chapter 2B – Where Rcuboids get a little bit more complex

Remember anchor points? “000”, “MM0” and “MML” ? For the cuboid shape [10,20,30]

“000” is the bottom most left corner. IE where X=0,
“M00” is the same except where X=middle of shape; where X=5
“L00” is the same except where X=Last bit of shape, where X=10.

Ie we have “0”(start), “M”(middle), “L”(last) for each axis of X,Y,Z.

But an Rcuboid has spheres on the corners, not pointy corners. This means the Rcuboid is not physically present at X=0,Y=0 and Z=0. So where does the Rcuboid touch the axis? And what do we call it?

Let me introduce two more points for an Rcuboid anchor. These are p (lowecase p) and P (upper case P).

What? Let’s do an example

Example using the Rcuboid definition of [[10,20,30],2]

This is a rounded cuboid. Meaning it has rounded corners. These corners have spheres with a diameter of 2. Therefore the Radius of each sphere is 1. So, X=1,Y=1,Z=1 would be the centre of the sphere on the left, bottom most corner of the Rcuboid. X=9,Y=1,Z=1 would be the centre of the right bottom most corner.

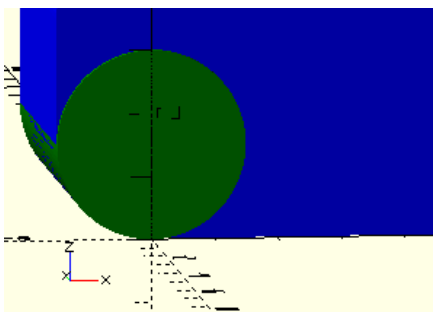
We have the following anchor points

“000” is the bottom most left corner where X=0,
“M00” is the same except where X=middle of shape; where X=5
“L00” is the same except where X=Last bit of shape, where X=10.
And
“p00” is the same except where X=the middle of the first circle. X=1 in the example
“P00” is the same except where X=the middle of the last circle. X=9

Therefore if I sliced the Rcuboid in half, down the middle of the Y. Middle being M. the anchor point “pM0” is the point where the Rcuboid touches the axis “PM0” is the point it leaves the axis.

In summary; Rcuboids have two extra points. p & P. and since we have 3 axis. That’s 8 more points (2^3). “ppp” to “PPP

What? Let’s do an example



Example (use F5 to render colours)

```
translate([-3/2,0,0])
{
// primaries
cuboid_biscuit1 = [[10,5,5],3,"-Y"];
cone_corner=["Y",5.2,3];

// render it
translate([0,0.1,0])
{
color("Blue") // press f5 for color
lib_cuboid("000",cuboid_biscuit1);
} // end of translate to show green cone
```

```
color("green") // press f5 for color
translate
 (lib_cuboid_mv("000","p0p",cuboid_biscuit1))
lib_cone_rGrid("M0M",cone_corner);

} // end of translate
```

Above is a close up of a blue Rcuboid corner. The green Mill bit size is diameter 3. ie Radius 1.5 and is represented by the green cylinder.

NB I have moved the whole shape along (translate([-3/2,0,0])) so you can see that the X is touching the axis.

NB the weird 0.1 and 5.2 numbers are used so that certain primaries stick out a little and so can easily be seen.

Anyway,

So, if we move along the Xaxis, we have start at 0. But because the side is rounded it does not actually touch the axis. “p” is the place it touches the axis. “P” is the place it leaves the axis before it gets to L. (At the end)

So “ppp” is the centre of the sphere on the bottom point of the Rcuboid. “Ppp” is the corner opposite it on the X axis. “pPp” is the sphere corner on the Y axis. And “PPp” is in the far corner on the bottom. “**P” is just a repeat but on the top.

As you can see “p0p” is a valid and useful anchor. But then again so is “P00”

Chapter recap

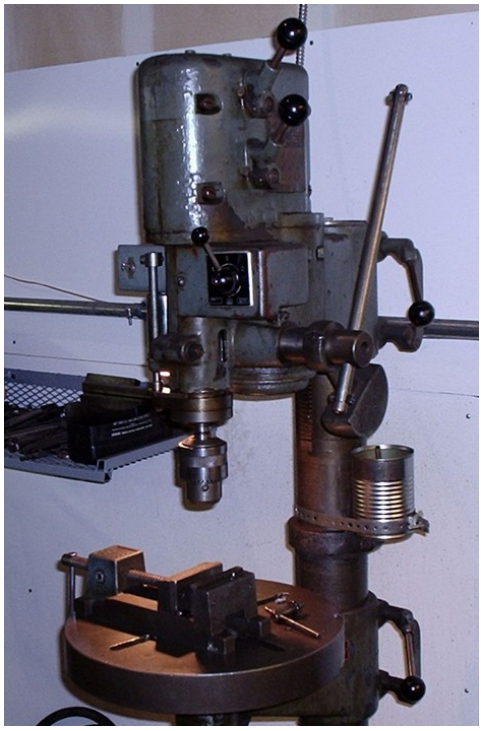
- A cuboid has 8 corners. Which are pointy.
- A Rcuboid has 8 rounded corners with spheres on the end.
- The centre of each sphere can be found with anchor point “ppp” to “PPP”
- “ppp”,”Ppp”,”pPp”,”PPp” are all on the bottom corners of the Rcuboid (Z=p)
- “ppP”,”PpP”,”pPP”,”PPP” are all on the top corners of the Rcuboid (Z=P)
- These letters can be mixed with base3, base12 and base 10
- “M0p” and “M0P” are valid
- You can move the anchor point from “000” to “ppp” using lib_cuboid_mv
- P and p only exist in the Cuboid world. They do not exist in lib_cone.

If this makes absolutely no sense to you it’s probably best that you try out some examples. The best examples can be found in Learn_Hard1A_v1_0_0.scad, or a later version. This code can be found in “the download“ of the OS++ framework.

http://osplusplus.org/2022standard/index.html

Chapter 3

In the metal workshop, “Drill holes” or negative cylinders, are made by a drill. In theory a go anywhere, drill any angle, hand drill with a good battery seems like the ultimate solution in the metal workshop. But it’s the fixed “pillar drill” which only drills holes in the Z axis where I seem to spend most of my time. Even more time than my beloved lathe. The fact of the matter is that vertical holes are much in demand in the metal workshop. lib_cones is the equivalent of the drill press in the 3d printing world.



Picture of a drill press / pillar drill

Pop the safety sunnies on. Switch in on. Slap the work on the plate and pull the bar, to drill a hole with a little cutting oil on the drill bit. In a few seconds you have a perfectly vertical hole in the Z axis.

I especially like the tin can bolted onto the side of the press. I think they call that an after market sale upgrade.

Awesome machine.

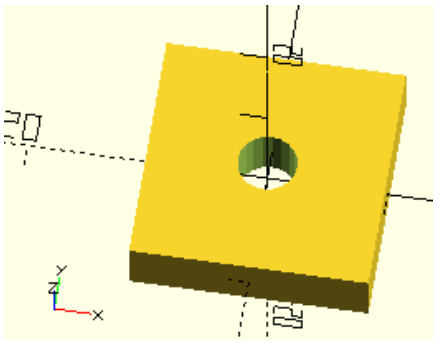
In summary, a pillar drill with an M8 drill bit, will drill a 8mm diameter hole. If the metal had a thickness of 10mm. That hole would have a height or length of 10mm. This whole paragraph could be condensed into

cone_hole= [“Z”,10,8]; // this is a cylinder

If we started with an M9 drill bit, which wore down to an M8 at the end of the drill. (ie M9 at the top with M8 at the bottom)

cone_hole= [“Z”,10,8,9]; // this is a cone

Unlike the drill press. Lib_cone can give you a hole in the “Z”, “X” and Y axis.



```
//////////////////
// Code snippet

cone_drillHole=["Z",10,5];   // cylinder
cuboid_workPiece=[20,20,5]; // cube

difference()
{
lib_cuboid_rGrid ("MMM",cuboid_workPiece);
  lib_cone_rGrid ("MMM",cone_drillHole);
} // end of difference
```

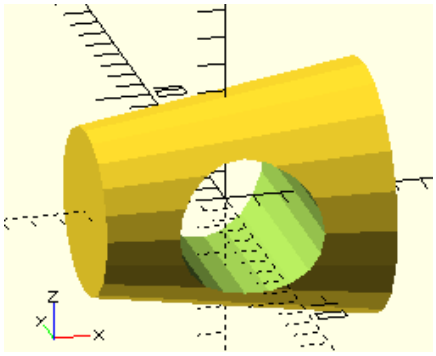
What is this lib_cuboid_rGrid? How is it different to lib_cuboid?

> What is lib_cuboid_rGrid? How is it different to lib_cuboid?

Answer=None. There isn’t a differences between the two. In fact lib_cuboid_rGrid calls lib_cuboid. R means render. Grid means use the Cartesian grid. It appears as an evolving naming standard.

***** What is lib_cuboid_rGrid? How is it different to lib_cuboid?? *****

Another example of a cone



```
//////////////////
// Code snippet

cone_work_piece=["X",10,6,9]; // len=10, D0=6, DL=9
cone_drillHole =["Y",10,5];    // len=10 D=5

difference()
{

lib_cone_rGrid ("MMM",cone_work_piece);
lib_cone_rGrid ("MMM",cone_drillHole);

} // end of difference
```

___

There is a problem. How does a cone fit in a cube centric Cartesian co-ordinates system like Grid (Cartesian)? It's a round peg, in a square hole problem. Ie What would "000" mean when dealing with a cone?

The way I deal with this is to wrap the cone in an imaginary cuboid box and use that to place and render the cone. But which circle, in the cone, decides the imaginary boxes dimensions? Do we just go with the biggest? By default, yes. But you can use other circles in the cone.

The 80/20 real world rule is to use MMM (or MM0 & MML with a Z axis cone) when rendering cones as it does not matter what size the circles are at the top or bottom. There are better examples in the Learn_Hard1A_v1_0_0.scad

***** But you can use other circles in the cone. *****

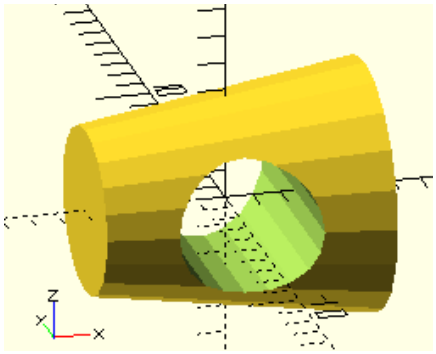
Syntax recap

- An example of a cone definition is

cone_main=["Z",10,3];

Where
"Z" is the direction the Cylinder points in
10 is the height or length of the cylinder
3 is the diameter of the Bottom circle
Since this is a cylinder, the Top Circle also has a 3mm diameter

Next Chapter



```
//////////////////
// Code snippet

cone_work_piece=["X",10,6,9]; // len=10, D0=6, DL=9
cone_drillHole =["Y",10,5];    // len=10 D=5

difference()
{

lib_cone_rGrid ("MMM",cone_work_piece);
lib_cone_rGrid ("MMM",cone_drillHole);

} // end of difference
```

The code example above, has two cones. One in the X direction and one in the Y direction. The code wisely uses “MMM” to reference it’s rendering. But you can use 000 if you want.

Openscad_standard provides the command cylinder. This produces a cone in the Z axis. You provide it’s height and D1 and D2. D1 is the circle at the bottom and D2 is the circle at the top. All nice and easy.

OS++ however produces cones in the X and Y axis. Ie no top and no bottom. So which circle is which, and what is the height?

- An example of a cone definition in OS++ is

cone_main=[“X”,10,3,2];

Where
“X” is the direction. Ie it points in the X direction
10 is the length (not height) of the cone.
3 is the diameter of the first circle where X=0. It’s called D0.
2 is the diameter of the Last circle where X=L. It’s called DL (last point)

So to translate to OS++; from Openscad_standard

Length instead of height,
D0 instead of D1
DL instead of D2

All good.

Well no, the OS++; render has a gotya. As stated a openscad_standard cylinder will create a cone in the Z axis and as you increase the height it grows up from Z=0. That is it grows from MM0 in OS++ speak.

eg

```
lib_cone_rGrid ("MM0",cone_inZaxis); // Z=0
```

But if you have a cone which points in the X axis. You probably want it to grow in the X direction not the Z. In this case, the shapes anchor point you will want is “0MM”. //X=0

ie The cone you want, will grow from X=0 instead of Z=0

eg

```
lib_cone_rGrid ("0MM",cone_inXaxis); // X=0
```

If you have a cone which points in the Y axis. Then the shapes anchor point you will want is “M0M” as the cone you want will grow from Y=0.

In summary:

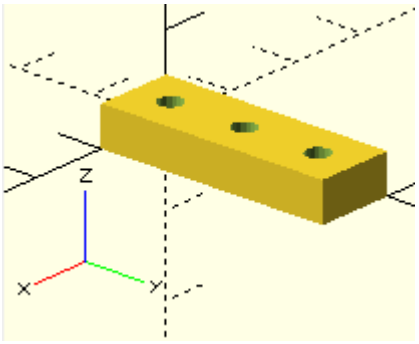
“0MM” (for X pointing cones – growing positively)
“M0M” (for Y pointing cones – growing positively)
“MM0” (for Z pointing cones – growing positively **or UP**)

Of course you could keep with the anchor render point MMM for all cones, pointing in different axis. Ie nice and easy as they are all the same. But MMM grows “out” from both ends as you increase the length. In generally want it to grow up or down, as I tweak the primaries, assuming it’s in the Z axis.

Therefore other anchor points for cones, that comes up a lot, are

“LMM” (for X pointing cones – growing negatively)
“MLM” (for Y pointing cones)
“MML” (for Z pointing cones – growing negatively **or down**)

Ie as you increase the length of the cone it grows down, instead of up. This is really handy if you have a bar and you want to drill holes from it’s top surface. (Much like a drill in a metal workshop)




```
//////////////////
// Code snippet - MML in red at bottom


// define the primaries

cuboid_bar=[10,30,5];     // cuboid
cone_drillHole=["Z",5,3];   // cylinder

// method

difference()
{

// openscad anchor point is at X=0,Y=0,Z=0;

// render the bar
lib_cuboid_rGrid ("000",cuboid_bar);

// Move to the top of the bar in the middle
translate(lib_cuboid_mv("000","MML",cuboid_bar) )
 {

 // openscad anchor point is at X=5,Y=15,Z=5;
 // ie MML of cuboid_bar
 translate([0,0,0]) //
 lib_cone_rGrid ("MML",cone_drillHole);
```

```
 // openscad anchor point is at X=5,Y=15,Z=5;
 // ie MML of cuboid_bar
 translate([0,10,0]) // local translate
 lib_cone_rGrid ("MML",cone_drillHole);

 // openscad anchor point is at X=5,Y=15,Z=5;
 // ie MML of cuboid_bar
 translate([0,-10,0]) // // local translate
 lib_cone_rGrid ("MML",cone_drillHole);

 } // end of translate to MML on cuboid_bar

// openscad anchor point is at X=0,Y=0,Z=0;

} // end of difference
__
```

Chapter recap
- Cones can be pointed in different directions
- D0 is the start circle diameter.
- DL is the last circle diameter
- M0M is a valid render anchor point for Cones pointing in the Y direction

Next Chapter

To move around a cuboid is as follows

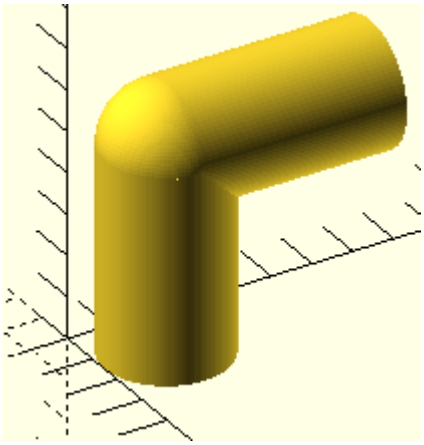
```
cuboid_bar=[10,30,5];
translate(lib_cuboid_mv("000","MML",cuboid_bar) )
// You could use  lib_cuboid_mvGrid instead of lib_cuboid_mv
```

What about a cone. What if I want to move to the top of a cone instead of a cuboid?

```
cone_Z=["Z",5,3];  // cylinder
translate(lib_cone_mvGrid("000","MML",cone_Z))
```

A few notes

1) The MV command states lib_cone. This means the code is in the lib_cone.scad file.
2) The MV command has Grid in it's name. This means it accepts Cartesian anchor points eg "000" to "LLL" in base 3, base 12 and base 10. (Does not accept p or P)
3) There is no such thing as p or P, as there is with Rcuboid. (Didn't I just say that?)



```
//////////////////
// Code snippet

// define the primaries
cone_Z=["Z",5,3];  // cylinder
cone_Y=["Y",5,3];  // cylinder
sphere=[3];

union()
{

// Render a cone like it was a cube using 000
lib_cone_rGrid ("000",cone_Z);

translate(lib_cone_mvGrid("000","MML",cone_Z))
 {
 lib_cone_rGrid ("M0M",cone_Y);
 lib_sphere_rGrid ("MMM",sphere);
 } // end of translate

} // end of union
```

Chapter recap
- You can move around a cone in the same way you can move round a cuboid with mvGrid
- Grid is short for Cartesian grid

Next chapter – autonomy of a OS++ user code

Specimen code. I will take each block and explain my current thinking.

```
//http://osplusplus.org/2022standard/professional
include <z_lib/OS++20221031.scad>;
$fn=20;

// Define primaries
cuboid_example1_plus      = [10,10,10];
cone_example2_minus       = ["Y",10,5];

// call code
main();

// code
module main()
{

 difference()
 {
 // render cuboid cuboid_example1_plus
 lib_cuboid("000",cuboid_example1_plus);

 // remove cone
 translate(lib_cuboid_mv("000","MOM",cuboid_example1_plus))
  {
  lib_cone_rGrid("MOM",cone_example2_minus); // remove cylinder data definition
  } // end of translate
 } // end of difference
} // end of module
```

The header block

```
//http://osplusplus.org/2022standard/professional
include <z_lib/OS++20221031.scad>;
$fn=20;
```

Line 1 ://http://osplusplus.org/2022standard/professional

This rem is possibly the most important line of the code. It declares 3 things

1) **The language** this code is written in. ie “OS++”

(NB OS++ is short for Open Scad Plus Plus)

2) The standard, tradition and customs of this code. Think dialect. ie “2022standard”

(NB 2022standard is to be expanded and updated in 2023 and hopefully 2024.)

3) **The grade** of the user code. Think formal vs pub or trade talk. eg “professional”

(NB In what capacity the user will be writing code and what the reader can expect)

This declaration is used to categorise your code. For example “**http://osplusplus.org/2022standard/Amateur**” code is unlikely to be allowed in production or critical infrastructure.

Examples of declaration (organisational defined, or user defined)

http://osplusplus.org/2022standard/Shop	- A companies own grade and interpretation.
http://... 22standard/user_defined	- A user defined grade and interpretation.

Examples of excuse declaration (ie Where standard are not wholly adhered to)

http:// ... 22standard/Amateur	- Someone who is learning or having fun
http:// ... 22standard/Kludge	- An emergency change or fix
http:// ... 22standard/Prehistoric	- Code written before standards
http:// ... 22standard/Privilege	- Code created by a DBA or Judge Dredd.
http:// ... 22standard/Throwaway	- Spitballing new ideas
http:// ... 22standard/Technical_deficit	- You know it's wrong, but....

***** Declaration (Language/Dialect/Grade) *****

```
Line2 :include <z_lib/OS++20221031.scad>;
```

This line calls all the OS++ libraries in a single command. Below is a simplified code extract from z_lib/OS++20221031.scad

```
echo ("using OS++; include <z_lib/OS++20221031.scad>; )");
include <./lib_cuboid_v1_0_3.scad>;
include <./lib_cone_v1_0_2.scad>;
include <./lib_sphere_v1_0_1.scad>;
```

Notice each library has it's own version. Once released the library becomes canon and is left unchanged, even in later releases of the OS++ code base. As a result your code will work even if it relies on a bug that has been fixed in later revisions.

```
include <z_lib/OS++.scad>;
```

This is a special os++ load. It is a straight copy of the latest and greatest dated include for the version you downloaded. This includes the "hello statement". For example

```
echo ("using OS++; include <z_lib/OS++20221031.scad>; )");
```

As a result you can find the latest set of version, of the OS++ framework you are using, by checking the console once the model is rendered. Once known you can then amend your include statement to the dated one and fix it to that current latest version, if you wish.

***** Include (Library load) *****

The primaries block

```
// Define primaries
cuboid_example1_plus      = [10,10,10];
cone_example2_minus       = ["Y",10,5]; // cylinder point in Y direction
```

These are simply primaries defined at the top of your code in global memory space. Therefore all modules have access to the primaries, to render or move around the model.

Each primary definition starts with it's library name. Eg cuboid is a definition used in lib_cuboid. And is followed by a human level description.

This allows easy Tweaking of another persons design, because the primaries are easy to find and you don't have to wade through the modules.

***** Primaries, in global memory *****

Future Chapter

Doco recap

-The Libs for OS++ are stored in ./z_libs
- Libs are activated by a singe line, like include <z_lib/OS++20221031.scad>;
- <z_lib/OS++20221031.scad>; is now fixed. So if your code runs. It runs forever
- ie you upgrade to the latest OS++ by copying over ./Z_libs, it still runs.

- An activated lib will say hello. Tell you it's version and help code in the console
- Libs are independent from each other and can run stand alone
- Each Lib contains an example remed out
- Each lib follows the same basic pattern and naming standard.
- All lib code starts with the name of the library. Eg liib_cuboid

-lib_cuboid has modules and functions that deal with cuboids
- It's simplest definition data type is [+X,+Y,+Z] eg [10,20,30]
- It can be rendered with lib_cuboid_rGrid()
- You can move the anchor point using lib_cuboid_mvGrid()
- You can "set" the definition data type to create new data types using lib_cuboid_set()
- You can "get" parts of the definition data using lib_cuboid_get()

- There is a lib called lib_sphere. Where is it's code stored?
- What is the move command, using Cartesian co-ords, from "000" to "MML"?
-
- If you said ./z_libs and lib_sphere_mvGrid("000","MML,sphere_mine); I am impressed. Because I don't remember including that lib in the Doco.

Last Chapter

I am afraid that's it. I would love to finish this doco off but I have run out of time. Centrelink has sent me on a forklift course and I have passed. YAAY for me!!! Out of 25 people I was the only one to pass. Despite having an argument, with the examiner, in the written test, about how the OHS question was wrong. I don't think that happens very often. You actually went on to describe how the text book, they had given you was "wrong", on a question marked "Mandatory". What does that mean? If you get it wrong, you don't pass the course, it's that important to get it right. To be fair ,he did see my point. An auspicious start to your blossoming future career as a forklift driver. Quite

Anyway it looks like I will be moving onto greener pastures. I hope you enjoyed this very weird doco. Here's the ending credits.

That's it folks. Curtain falls. The Audience cheers and boos. The curtains go up and the actors take a bow. They are, in no particular order.

Green	The voice of exasperation, advocacy for the reader
Blue	My thoughts, trying to explain.
Red	Annoying know it all, who likes a glass of red and a bit of a stirrer (Statler ?)
Purple	Old man, now get off my lawn. What? Speak up! I am not deaf. (Waldorf?)
Black/Yellow	Childlike (>35 years old) that points out the emperor has no clothes
Black	Only the facts

Curtain goes down. People start to leave.

Epilogue

Hopefully you enjoyed this silliness, or at the very least I stopped myself from falling asleep while I talked about something as exciting as rounded corner cubes and their location in Cartesian space in an euclidean volume. I did start with a dry and rather matter of fact instruction manual, but it put me to sleep. And he drinks a lot of coffee. So I went with a playfull style experiment. I had a lot of fun writing it. I hope you did. If not, I guess you get what you pay for.

All the best

Stephen George
Freeloaders of the world unite!
And stay off my lawn.

Page left intentionally blank

Page left intentionally blank.
(No it’s not. You wrote in it.)