omdl

v0.6

Generated by Doxygen 1.8.9.1

Tue Apr 4 2017 20:15:43

ii CONTENTS

Contents

1	omd		1
	1.1	Using	2
	1.2	Contributing	2
	1.3	Support	2
2	Con	ventions	2
	2.1	Data types	2
		2.1.1 Built-in	3
		2.1.2 Additional conventions	3
3	Valid	dation	7
	3.1	Datatypes	7
		3.1.1 Identification	7
		3.1.2 Operations	23
	3.2	Math	40
		3.2.1 Vector Algebra	40
		3.2.2 Bitwise	47
4	Insta	alling	53
5	Test	List	53
6	Todo	o List	54
7	Mod	lule Index	54
•	7.1		54
	7.1	induies	J
8	File	Index	56
	8.1	File List	56
9	Mod	lule Documentation	58
	9.1	2d Extrusions	58
		9.1.1 Detailed Description	59
		9.1.2 Function Documentation	59
	9.2	2d Shapes	77
		9.2.1 Detailed Description	78
		9.2.2 Function Documentation	78
	9.3	3d Shapes	92
		9.3.1 Detailed Description	92

9.5	Alignment 9.4.1 Detailed Description 9.4.2 Function Documentation Angles 9.5.1 Detailed Description 9.5.2 Function Documentation Bitwise 9.6.1 Detailed Description 9.6.2 Function Documentation	. 102 . 104 . 104 . 105 . 107
9.5	9.4.2 Function Documentation Angles	. 102 . 104 . 104 . 105 . 107
9.5	Angles	. 104 . 104 . 105 . 107 . 108
	9.5.1 Detailed Description	. 104 . 105 . 107 . 108
	9.5.2 Function Documentation	. 105 . 107 . 108
	Bitwise	. 107 . 108
	9.6.1 Detailed Description	. 108
9.6	·	
	9.6.2 Function Documentation	
		. 108
9.7	Component	. 112
9.8	Console	. 113
	9.8.1 Detailed Description	. 113
	9.8.2 Function Documentation	. 114
9.9	Constants	. 117
	9.9.1 Detailed Description	. 117
9.10	Coordinates	. 118
	9.10.1 Detailed Description	. 118
	9.10.2 Function Documentation	. 120
9.11	Database	. 122
	9.11.1 Detailed Description	. 122
9.12	Datatypes	. 123
	9.12.1 Detailed Description	. 124
9.13	Edge	. 125
	9.13.1 Detailed Description	. 125
	9.13.2 Function Documentation	. 125
9.14	Electrical	. 128
9.15	Euclidean	. 129
	9.15.1 Detailed Description	. 130
9.16	Extrude	. 131
	9.16.1 Detailed Description	. 131
	9.16.2 Function Documentation	. 131
9.17	General	. 135
	9.17.1 Detailed Description	. 135
9.18	Geometry	. 136
	9.18.1 Detailed Description	. 136
9.19	Identification	. 137

iv CONTENTS

	9.19.1 Detailed Description
9.20	Iterables
	9.20.1 Detailed Description
	9.20.2 Function Documentation
9.21	Iterables
	9.21.1 Detailed Description
	9.21.2 Function Documentation
9.22	Lengths
	9.22.1 Detailed Description
	9.22.2 Function Documentation
9.23	Linear Algebra
	9.23.1 Detailed Description
	9.23.2 Function Documentation
9.24	Lists
	9.24.1 Detailed Description
	9.24.2 Function Documentation
9.25	Lists
	9.25.1 Detailed Description
	9.25.2 Function Documentation
9.26	Maps
	9.26.1 Detailed Description
	9.26.2 Function Documentation
9.27	Material
9.28	Math
	9.28.1 Detailed Description
9.29	Operations
	9.29.1 Detailed Description
9.30	Other Shapes
	9.30.1 Detailed Description
	9.30.2 Function Documentation
9.31	Parts
9.32	Polyhedra
	9.32.1 Detailed Description
	9.32.2 Variable Documentation
9.33	Polytope
	9.33.1 Detailed Description
	9.33.2 Function Documentation

9.34	Polytopes	27
	9.34.1 Detailed Description	28
	9.34.2 Function Documentation	28
9.35	Repeat	13
	9.35.1 Detailed Description	ŀ3
	9.35.2 Function Documentation	ŀ3
9.36	Resolutions	ŀ6
	9.36.1 Detailed Description	ļ 7
	9.36.2 Function Documentation	ļ 7
9.37	Scalars	51
	9.37.1 Detailed Description	52
	9.37.2 Function Documentation	52
9.38	Scalars	59
	9.38.1 Detailed Description	59
	9.38.2 Function Documentation	59
9.39	Shapes	31
	9.39.1 Detailed Description	31
9.40	System	35
	9.40.1 Detailed Description	35
9.41	Tables	36
	9.41.1 Detailed Description	37
	9.41.2 Function Documentation	8
9.42	Tools	⁷ 4
	9.42.1 Detailed Description	′5
9.43	Triangles	⁷ 6
	9.43.1 Detailed Description	7
	9.43.2 Function Documentation	7
9.44	Units	32
	9.44.1 Detailed Description	33
9.45	Utilities	34
	9.45.1 Detailed Description	34
	9.45.2 Function Documentation	34
9.46	Utilities	36
	9.46.1 Detailed Description	36
9.47	Validation	37
	9.47.1 Detailed Description	37
	9.47.2 Function Documentation	38

vi CONTENTS

	9.48	Vector Algebra	289
		9.48.1 Detailed Description	290
		9.48.2 Function Documentation	290
10	File I	Documentation 2	296
10		console.scad File Reference	
	10.1	10.1.1 Detailed Description	
	10.2	constants.scad File Reference	
	10.2		
	10.0	10.2.1 Detailed Description	
	10.3	database/geometry/polyhedra/anti_prisms.scad File Reference	
		10.3.1 Detailed Description	
	10.4	database/geometry/polyhedra/archimedean.scad File Reference	
		10.4.1 Detailed Description	
	10.5	database/geometry/polyhedra/archimedean_duals.scad File Reference	
		10.5.1 Detailed Description	
	10.6	database/geometry/polyhedra/cupolas.scad File Reference	
		10.6.1 Detailed Description	
	10.7	database/geometry/polyhedra/dipyramids.scad File Reference	303
		10.7.1 Detailed Description	303
	10.8	database/geometry/polyhedra/johnson.scad File Reference	304
		10.8.1 Detailed Description	304
	10.9	database/geometry/polyhedra/platonic.scad File Reference	305
		10.9.1 Detailed Description	305
	10.10	Odatabase/geometry/polyhedra/polyhedra_all.scad File Reference	306
		10.10.1 Detailed Description	306
	10.11	1 database/geometry/polyhedra/prisms.scad File Reference	307
		10.11.1 Detailed Description	307
	10.12	2database/geometry/polyhedra/pyramids.scad File Reference	308
		10.12.1 Detailed Description	308
	10.13	3database/geometry/polyhedra/trapezohedron.scad File Reference	309
		10.13.1 Detailed Description	309
	10.14	4datatypes.scad File Reference	310
		10.14.1 Detailed Description	311
	10.15	5datatypes/datatypes_identify_iterable.scad File Reference	311
		10.15.1 Detailed Description	
	10.16	6datatypes/datatypes_identify_list.scad File Reference	
		10.16.1 Detailed Description	

CONTENTS vii

10.17datatypes/datatypes_identify_scalar.scad File Reference
10.17.1 Detailed Description
10.18datatypes/datatypes_map.scad File Reference
10.18.1 Detailed Description
10.19datatypes/datatypes_operate_iterable.scad File Reference
10.19.1 Detailed Description
10.20datatypes/datatypes_operate_list.scad File Reference
10.20.1 Detailed Description
10.21datatypes/datatypes_operate_scalar.scad File Reference
10.21.1 Detailed Description
10.22datatypes/datatypes_table.scad File Reference
10.22.1 Detailed Description
10.23mainpage.scad File Reference
10.23.1 Detailed Description
10.24math.scad File Reference
10.24.1 Detailed Description
10.25math/math_bitwise.scad File Reference
10.25.1 Detailed Description
10.26math/math_linalg.scad File Reference
10.26.1 Detailed Description
10.27math/math_oshapes.scad File Reference
10.27.1 Detailed Description
10.28math/math_polytope.scad File Reference
10.28.1 Detailed Description
10.29math/math_triangle.scad File Reference
10.29.1 Detailed Description
10.30math/math_utility.scad File Reference
10.30.1 Detailed Description
10.31 math/math_vecalg.scad File Reference
10.31.1 Detailed Description
10.32shapes/shapes2d.scad File Reference
10.32.1 Detailed Description
10.33shapes/shapes2de.scad File Reference
10.33.1 Detailed Description
10.34shapes/shapes3d.scad File Reference
10.34.1 Detailed Description
10.35tools/tools_align.scad File Reference

1 omdl 1

10.35.1 Detailed Description
10.36tools/tools_edge.scad File Reference
10.36.1 Detailed Description
10.37tools/tools_polytope.scad File Reference
10.37.1 Detailed Description
10.38tools/tools_utility.scad File Reference
10.38.1 Detailed Description
10.39units/units_angle.scad File Reference
10.39.1 Detailed Description
10.40units/units_coordinate.scad File Reference
10.40.1 Detailed Description
10.41 units/units_length.scad File Reference
10.41.1 Detailed Description
10.42units/units_resolution.scad File Reference
10.42.1 Detailed Description
10.43validation.scad File Reference
10.43.1 Detailed Description

1 omdl

Index

It is an OpenSCAD mechanical design library (omd1) that provides open-source high-level design primitives with coherent documentation generated by Doxygen using openscad-amu.

357

With Doxygen, the code documentation is written within the code itself, and is thus easy to keep current. Moreover, it provides a standard way to both write and present OpenSCAD design documentation, compilable to common output formats (html, pdf, etc). With omdl, all library primitives are *parametric* with minimal, mostly zero, global variable dependencies and all library API's include markups that describe its parameters, behavior, and use.

Validation scripts are used to verify that the core operations work as expected across evolving OpenSCAD versions (validation performed when building the documentation). The library uses a common set of conventions for specifying data types and is divided into individual component modules of functionality, organized into groups, that may be included as desired.

Example:

```
include <shapes/shapes2de.scad>;
include <shapes/shapes3d.scad>;

$fn = 36;

frame = triangle_lp2ls( [ [30,0], [0,40], [30,40] ] );
core = 2 * frame / 3;
vrnd = [1, 2, 4];

cone( h=20, r=10, vr=2 );
rotate([0, 0, 360/20])
radial_repeat( n=5, angle=true )
    etriangle_ls_c( vs=frame, vc=core, vr=vrnd, h=10 );
```

```
translate([0, -50,0])
linear_extrude(height=10)
text( text="omdl", size=20, halign="center", valign="center" );
```

Table 1: Example Result



1.1 Using

To use omd1, the library files must be copied to an OpenSCAD library location. This can be done manually or can be done using using openscad-amu.

The ladder has several advantages and is recommended. When using openscad-amu, the library documentation is installed together with the library source code. This documentation is also added to a local browsable index, which facilitates reference use. Moreover, with openscad-amu installed, one can develop documentation for other Open SCAD design scripts.

See the recommended installation method for more information. Library releases are periodically made available in the omdl repository under snapshots.

1.2 Contributing

omdl uses git for development tracking, and is hosted on GitHub following the usual practice of forking and submitting pull requests to the source repository.

As it is released under the GNU Lesser General Public License, any file you change should bear your copyright notice alongside the original authors' copyright notices typically located at the top of each file.

Ideas, requests, comments, contributions, and constructive criticism are welcome.

1.3 Support

In case you have any questions or would like to make feature requests, you can contact the maintainer of the project or file an issue.

2 Conventions

Data types

2.1.1 Built-in

omdl assumes a value is either a number, a boolean, a string, a list, a range, or the undefined value. What is called a vector in the OpenSCAD types documentation is refereed to as a *list* here in order to distinguish between sequential lists of values and Euclidean vectors.

type	description
value	any valid OpenSCAD storable datum
number	an arithmetic value
boolean	a binary logic value (true or false)
string	a sequential list of of character values
list	a sequential list of arbitrary values
range	an arithmetic sequence
undef	the undefined value

2.1.1.1 Special numerical values

value	description
nan	a numerical value which is not a number
inf	a numerical value which is infinite

2.1.2 Additional conventions

When a list has an expected number of elements 'n', the *count* is appended following a '-'. When there is a range of expected elements, the lower and upper bounds are separated by a ':' and appended (order of bounds may be reversed). When the elements values are of an expected type, that *type* is prepended. Combinations are used as needed as in the following table:

name	description
list-n	a list of of n elements
list-l:u	a list of I to u elements
type-list	a list of elements with an expected type
type-list-n	a list of n elements with an expected type

2.1.2.1 Distinctions

omdl make the following distinctions on variable types.

name	description
scalar	a single non-iterable value
iterable	a multi-part sequence of values
empty	an iterable value with zero elements
even	an even numerical value
odd	an odd numerical value

2.1.2.2 **General**

From the fixed built-in set of data types, omdl adds the following general type specifications and conventions.

name	description
bit	a binary numerical value (0 or 1)
integer	a positive, negative, or zero whole number
decimal	integer numbers with a fractional part

index	a list index sequence
datastruct	a defined data structure
data	an arbitrary data structure

2.1.2.2.1 Index sequence

The data type **index** refers to a specified sequence of list element indexes. A list index sequence may be specified in one of the following forms.

value / form	description	
true	All index positions of the list [0:size-1]	
false	No index positions	
"all"	All index positions of the list [0:size-1]	
"none"	No index positions	
"rands"	Random index selection of the list [0:size-1]	
"even"	The even index of the list [0:size-1]	
"odd"	The odd index of the list [0:size-1]	
<integer></integer>	The single position given by an <integer></integer>	
<range></range>	The range of positions given by a <range></range>	
<integer-list></integer-list>	The list of positions give in <integer-list></integer-list>	

The function get_index() can be used to convert a value of this data type into a sequence of list element indexes.

Example

```
// list
11 = [a,b,c,d,e,f]

// index sequence
get_index(11) = [0,1,2,3,4,5]
get_index(11, "rands") = [0,2,5]
```

2.1.2.3 Geometric

For geometric specifications and geometric algebra, omdl adds the following type specifications and conventions.

name	description
point	a list of numbers to identify a location in space
vector	a direction and magnitude in space
line	a start and end point in space (line wiki)
normal	a vector that is perpendicular to a given object
pnorm	a vector that is perpendicular to a plane
plane	a flat 2d infinite surface (plane wiki)
coords	a list of points in space
matrix	a rectangular array of values

When a particular dimension is expected, the dimensional expectation is appended to the end of the name after a '-' dash as in the following table.

name	description	
point-Nd	a point in an 'N' dimensional space	
vector-Nd	a vector in an 'N' dimensional space	
line-Nd	a line in an 'N' dimensional space	
coords-Nd	a coordinate list in an 'N' dimensional space	
matrix-MxN	a 'M' by 'N' matrix of values	

2.1.2.3.1 Lines and vectors

The data type **line** refers to a convention for specifying a line or a vector. A vector is a direction and magnitude in space. A line, too, has direction and magnitude, but also has location, as it starts at one point in space and ends at

another. Operators in omd1 make use of a common convention for specifying Euclidean vectors and straight lines as summarized in the following table:

Given two points 'p1' and 'p2', in space:

no.	form	description
1	p2	a line or vector from the origin to 'p2'
2	[p2]	a line or vector from the origin to 'p2'
3	[p1, p2]	line or vector from 'p1' to 'p2'

The functions get_line_dim(), get_line_tp(), get_line_ip(), and get_line2origin(), are available to identify the dimension of and convert a line into a vector or point.

Example

```
// points
p1 = [a,b,c]
p2 = [d,e,f]

// lines and vectors
v1 = p2 = [d,e,f]
v2 = [p2] = [[d,e,f]]
v3 = [p1, p2] = [[a,b,c], [d,e,f]]
v1 == v2
v1 == v2 == v3, iff p1 == origin3d
```

2.1.2.3.2 Planes

Operators in omdl use a common convention for specifying planes. A **plane** is identified by a point on its surface together with its normal vector specified by pnorm, which is discussed in the following section. A list with a point and normal together specify the plane as follows:

name	form
plane	[point, pnorm]

2.1.2.3.3 Planes' normal

The data type **pnorm** refers to a convention for specifying a direction vector that is perpendicular to a plane. Given three points 'p1', 'p2', 'p3', and three vectors 'v1', 'v2', 'vn', the planes' normal can be specified in any of the following forms:

no.	form	description
1	vn	the predetermined normal vector to
		the plane
2	[vn]	the predetermined normal vector to
		the plane
3	[v1, v2]	two distinct but intersecting vectors
4	[p1, p2, p3]	three (or more) non-collinear
		coplanar points

The function get_pnorm2nv() can be used to convert a value of this data type into a normal vector.

Example

```
// points
p1 = [a,b,c];
p2 = [d,e,f];
p3 = [g,h,i];

// lines and vectors
v1 = [p1, p2] = [[a,b,c], [d,e,f]]
v2 = [p1, p3] = [[a,b,c], [g,h,i]]
vn = cross_l1(v1, v2)

// planes' normal
```

3 Validation

- Datatypes
- Math

3.1 Datatypes

- Identification
- Operations

3.1.1 Identification

- Scalar
- · Iterables
- Lists

3.1.1.1 Scalar

- Script
- Results

3.1.1.1.1 Script

```
include <datatypes.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
show_passing = true;  // show passing tests
show_skipped = true;  // show skipped tests
echo( str("OpenSCAD Version ", version()) );

// test-values columns
test_c =
[
    ["id", "identifier"],
    ["td", "description"],
    ["tv", "test value"]
];

// test-values rows
test_r =
```

```
["t01", "The undefined value", ["t02", "An odd integer",
                                    undef],
  ["t03", "An small even integer",
                                    101,
  ["t04", "A large integer",
                                    1000000001,
  ["t05", "A small decimal (epsilon)", aeps],
  ["t06", "The max number",
                                    number max],
  ["t07", "The min number",
                                    number_min],
  ["t08", "The max number^2",
                                    number_max * number_max],
  ["t09", "The invalid number nan",
["t10", "The boolean true",
["t11", "The boolean false",
                                    0 / 01,
                                    truel.
                                    false],
  ["t12", "A character string", ["t13", "A string",
                                    "a"],
                                    "This is a longer string"],
  ["t14", "The empty string", ["t15", "The empty list",
                                    empty_str],
                                    empty_lst],
  ["t16", "A 1-tuple list of undef",
                                    [undef]],
  ["t17", "A 1-tuple list",
                                    [10]],
  ["t18", "A 3-tuple list",
                                    [1, 2, 3]],
  ["t19", "A list of lists",
                                    [[1,2,3], [4,5,6], [7,8,9]]],
 ["t20", "A shorthand range", ["t21", "A range",
                                    [0:911,
                                    [0:0.5:9]]
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 's' to skip test
t = true: // shortcuts
f = false;
u = undef;
          // skip test
s = -1;
good r =
                 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21
[ // function
  ["is_defined",
                 ["not_defined",
                ["is_nan",
                 f, f, f, f, f, f, f, t, f, f],
  ["is_inf",
                 ["is_scalar",
                  t, f, f, f, f, f, f, f, s, s],
  ["is_iterable",
                 f, t, t, t, t, t, t, t, s, s],
  ["is_empty",
                 f, t, t, t, f, f, f, f, f, f]
  ["is_number",
                  f, t, t, t, t, t, t, t, f, f],
  ["is_integer",
                 ["is_decimal",
                  ["is_boolean",
                 f, f, f, f, f, f, f, f, t, t, f, f, f, f, f, f, f, f, f, f]
  ["is_string",
                 f, f, f, f, f, f, f, f, f, t, t, t, f, f, f, f, f, f, f],
  ["is_list",
                  f, t, t, t, t, t, s, s],
  ["is_range",
                  ["is_even",
                  ["is_odd",
                  ["is_between_MM", f, t, t, t, t, t, f, f, t, t, f, f]
];
// sanity-test tables
table_check( test_r, test_c, false );
table_check( good_r, good_c, false );
// validate helper function and module
function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
module run_test( fname, fresult, vid )
 value_text = get_table_v(test_r, test_c, vid, "td");
 pass_value = get_table_v(good_r, good_c, fname, vid);
 test_pass = validate( cv=fresult, t=pass_value, pf=true );
test_text = validate( str(fname, "(", get_value(vid), ")=", pass_value), fresult, pass_value
 if ( pass_value != s )
   if ( !test_pass )
     log_warn( str(vid, "(", value_text, ") ", test_text) );
   else if ( show_passing )
  log_info( str(vid, " ", test_text) );
 else if ( show skipped )
   log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
```

```
// Indirect function calls would be very useful here!!!

for (vid=test_ids) run_test( "is_defined", is_defined(get_value(vid)), vid );

for (vid=test_ids) run_test( "not_defined", not_defined(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_nan", is_nan(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_inf", is_inf(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_scalar", is_scalar(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_iterable", is_iterable(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_empty", is_empty(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_inumber", is_number(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_integer", is_integer(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_decimal", is_decimal(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_string", is_string(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_list", is_list(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_range", is_range(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_even", is_renge(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_odd", is_odd(get_value(vid)), vid );

for (vid=test_ids) run_test( "is_between_MM", is_between(get_value(vid), number_min,number_max), vid );
```

3.1.1.1.2 Results

```
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'is_defined(undef)=false'"
3 ECHO: "[ INFO ] run_test(); t02 passed: 'is_defined(1)=true'"
4 ECHO: "[ INFO ]
                     run_test(); t03 passed: 'is_defined(10)=true'"
5 ECHO: "[ INFO ] run_test(); t04 passed: 'is_defined(1e+08)=true'"
                     run_test(); t05 passed: 'is_defined(0.001)=true'"
6 ECHO: "[ INFO ]
7 ECHO: "[ INFO ] run_test(); t06 passed: 'is_defined(1e+308)=true'"
8 ECHO: "[ INFO ] run_test(); t07 passed: 'is_defined(-1e+308)=true'"
9 ECHO: "[ INFO ] run_test(); t08 passed: 'is_defined(inf)=true'"
10 ECHO: "[ INFO ] run_test(); t09 passed: 'is_defined(nan)=true'"
11 ECHO: "[ INFO ] run_test(); t10 passed: 'is_defined(true)=true'"
12 ECHO: "[ INFO ] run_test(); t11 passed: 'is_defined(false)=true'"
13 ECHO: "[ INFO ] run_test(); t12 passed: 'is_defined(a)=true'"
15 ECHO: "[INFO] run_test(); t13 passed: 'is_defined()=true'"

15 ECHO: "[INFO] run_test(); t14 passed: 'is_defined()=true'"
16 ECHO: "[ INFO ] run_test(); t15 passed: 'is_defined([])=true'"
17 ECHO: "[ INFO ] run_test(); t16 passed: 'is_defined([undef])=true'"
18 ECHO: "[ INFO ] run_test(); t17 passed: 'is_defined([10])=true'"

19 ECHO: "[ INFO ] run_test(); t18 passed: 'is_defined([1, 2, 3])=true'"
20 ECHO: "[ INFO ] run_test(); t19 passed: 'is_defined([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=true'"
21 ECHO: "[ INFO ] run_test(); t20 passed: 'is_defined([0 : 1 : 9])=true'"
22 ECHO: "[ INFO ] run_test(); t21 passed: 'is_defined([0 : 0.5 : 9])=true'"
23 ECHO: "[ INFO ] run_test(); t01 passed: 'not_defined(undef)=true'"
24 ECHO: "[ INFO ] run_test(); t02 passed: 'not_defined(1)=false'"
25 ECHO: "[ INFO ] run_test(); t03 passed: 'not_defined(10)=false'"
26 ECHO: "[ INFO ] run_test(); t04 passed: 'not_defined(1e+08)=false'"
27 ECHO: "[ INFO ] run_test(); t05 passed: 'not_defined(0.001)=false'"
28 ECHO: "[ INFO ] run_test(); t06 passed: 'not_defined(1e+308)=false'"
29 ECHO: "[ INFO ] run_test(); t07 passed: 'not_defined(-le+308)=false'"
30 ECHO: "[ INFO ] run_test(); t08 passed: 'not_defined(inf)=false'"
31 ECHO: "[
              INFO ] run_test(); t09 passed: 'not_defined(nan)=false'
32 ECHO: "[ INFO ] run_test(); t10 passed: 'not_defined(true)=false'"
33 ECHO: "[ INFO ] run_test(); t11 passed: 'not_defined(false)=false'"
34 ECHO: "[ INFO ] run_test(); t12 passed: 'not_defined(a)=false'"
35 ECHO: "[ INFO ] run_test(); t13 passed: 'not_defined(This is a longer string)=false'"
36 ECHO: "[ INFO ] run_test(); t14 passed: 'not_defined()=false'"
37 ECHO: "[ INFO ] run_test(); t15 passed: 'not_defined([])=false'"
38 ECHO: "[ INFO ] run_test(); t16 passed: 'not_defined([undef])=false'"
39 ECHO: "[ INFO ] run_test(); t17 passed: 'not_defined([10])=false'
40 ECHO: "[ INFO ] run_test(); t18 passed: 'not_defined([1, 2, 3])=false'"
41 ECHO: "[ INFO ] run_test(); t19 passed: 'not_defined([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
42 ECHO: "[ INFO ] run_test(); t20 passed: 'not_defined([0 : 1 : 9]) = false'"
43 ECHO: "[ INFO ] run_test(); t21 passed: 'not_defined([0 : 0.5 : 9])=false'"
44 ECHO: "[ INFO ] run_test(); t01 passed: 'is_nan(undef)=false'"
45 ECHO: "[ INFO ] run_test(); t02 passed: 'is_nan(1)=false'
46 ECHO: "[ INFO ] run_test(); t03 passed: 'is_nan(10)=false'"
47 ECHO: "[ INFO ] run_test(); t04 passed: 'is_nan(1e+08)=false'"
48 ECHO: "[ INFO ] run_test(); t05 passed: 'is_nan(0.001)=false'"
49 ECHO: "[ INFO ] run_test(); t06 passed: 'is_nan(1e+308)=false'"
50 ECHO: "[ INFO ] run_test(); t07 passed: 'is_nan(-1e+308)=false'"
51 ECHO: "[ INFO ] run_test(); t08 passed: 'is_nan(inf)=false'
52 ECHO: "[ INFO ] run_test(); t09 passed: 'is_nan(nan)=true'"
53 ECHO: "[ INFO ] run_test(); t10 passed: 'is_nan(true)=false'"
```

```
54 ECHO: "[ INFO ] run_test(); t11 passed: 'is_nan(false)=false'"
55 ECHO: "[ INFO ] run_test(); t12 passed: 'is_nan(a)=false''
56 ECHO: "[ INFO ] run_test(); t13 passed: 'is_nan(This is a longer string)=false'"
57 ECHO: "[ INFO ] run_test(); t14 passed: 'is_nan()=false'"
58 ECHO: "[
            INFO ] run_test(); t15 passed: 'is_nan([])=false'"
59 ECHO: "[ INFO ] run_test(); t16 passed: 'is_nan([undef])=false'"
60 ECHO: "[
            INFO ] run_test(); t17 passed: 'is_nan([10])=false''
61 ECHO: "[ INFO ] run_test(); t18 passed: 'is_nan([1, 2, 3])=false'"
62 ECHO: "[ INFO ] run_test(); t19 passed: 'is_nan([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
63 ECHO: "[ INFO ] run_test(); t20 passed: 'is_nan([0 : 1 : 9])=false''
64 ECHO: "[ INFO ] run_test(); t21 passed: 'is_nan([0:0.5:9])=false'"
65 ECHO: "[ INFO ] run_test(); t01 passed: 'is_inf(undef)=false'"
66 ECHO: "[ INFO ] run_test(); t02 passed: 'is_inf(1) = false''
67 ECHO: "[ INFO ] run_test(); t03 passed: 'is_inf(10)=false'"
68 ECHO: "[ INFO ] run_test(); t04 passed: 'is_inf(1e+08)=false'"
69 ECHO: "[ INFO ] run_test(); t05 passed: 'is_inf(0.001)=false'"
70 ECHO: "[ INFO ] run_test(); t06 passed: 'is_inf(1e+308)=false''
71 ECHO: "[ INFO ] run_test(); t07 passed: 'is_inf(-1e+308)=false'"
72 ECHO: "[ INFO ] run_test(); t08 passed: 'is_inf(inf)=true'
73 ECHO: "[ INFO ] run_test(); t09 passed: 'is_inf(nan)=false'"
74 ECHO: "[ INFO ] run_test(); t10 passed: 'is_inf(true)=false'"
75 ECHO: "[ INFO ] run_test(); t11 passed: 'is_inf(false)=false'"
76 ECHO: "[ INFO ] run_test(); t12 passed: 'is_inf(a)=false'
77 ECHO: "[ INFO ] run_test(); t13 passed: 'is_inf(This is a longer string)=false'"
78 ECHO: "[ INFO ] run_test(); t14 passed: 'is_inf()=false'"
79 ECHO: "[ INFO ] run_test(); t15 passed: 'is_inf([])=false'"
80 ECHO: "[ INFO ] run_test(); t16 passed: 'is_inf([undef])=false'"
81 ECHO: "[ INFO ] run_test(); t17 passed: 'is_inf([10])=false'
82 ECHO: "[ INFO ] run_test(); t18 passed: 'is_inf([1, 2, 3])=false'"
83 ECHO: "[ INFO ] run_test(); t19 passed: 'is_inf([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
84 ECHO: "[ INFO ] run_test(); t20 passed: 'is_inf([0 : 1 : 9])=false'
85 ECHO: "[ INFO ] run_test(); t21 passed: 'is_inf([0 : 0.5 : 9])=false'"
86 ECHO: "[ INFO ] run_test(); t01 passed: 'is_scalar(undef)=true'"
87 ECHO: "[ INFO ] run_test(); t02 passed: 'is_scalar(1)=true'
88 ECHO: "[ INFO ] run_test(); t03 passed: 'is_scalar(10)=true'"
89 ECHO: "[ INFO ] run_test(); t04 passed: 'is_scalar(1e+08)=true'"
90 ECHO: "[ INFO ] run_test(); t05 passed: 'is_scalar(0.001)=true'"
91 ECHO: "[ INFO ] run_test(); t06 passed: 'is_scalar(1e+308)=true'"
92 ECHO: "[ INFO ] run_test(); t07 passed: 'is_scalar(-1e+308)=true'"
93 ECHO: "[ INFO ] run_test(); t08 passed: 'is_scalar(inf)=true'
94 ECHO: "[ INFO ] run_test(); t09 passed: 'is_scalar(nan)=true'"
95 ECHO: "[ INFO ] run_test(); t10 passed: 'is_scalar(true)=true'
96 ECHO: "[ INFO ] run_test(); t11 passed: 'is_scalar(false)=true'"
97 ECHO: "[ INFO ] run_test(); t12 passed: 'is_scalar(a)=false'"
98 ECHO: "[ INFO ] run_test(); t13 passed: 'is_scalar(This is a longer string)=false'"
99 ECHO: "[ INFO ] run_test(); t14 passed: 'is_scalar()=false'"
100 ECHO: "[ INFO ] run_test(); t15 passed: 'is_scalar([])=false'"
101 ECHO: "[ INFO ] run_test(); t16 passed: 'is_scalar([undef])=false'"
102 ECHO: "[ INFO ] run_test(); t17 passed: 'is_scalar([10])=false'"
103 ECHO: "[ INFO ] run_test(); t18 passed: 'is_scalar([1, 2, 3])=false'"
104 ECHO: "[ INFO ] run_test(); t19 passed: 'is_scalar([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
105 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_scalar(A shorthand range)''
106 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_scalar(A range)'"
107 ECHO: "[ INFO ] run_test(); t01 passed: 'is_iterable(undef)=false'"
108 ECHO: "[ INFO ] run_test(); t02 passed: 'is_iterable(1)=false'"
109 ECHO: "[ INFO ]
                    run_test(); t03 passed: 'is_iterable(10)=false'"
110 ECHO: "[ INFO ] run_test(); t04 passed: 'is_iterable(1e+08)=false'"
111 ECHO: "[ INFO ] run_test(); t05 passed: 'is_iterable(0.001)=false''
112 ECHO: "[ INFO ] run_test(); t06 passed: 'is_iterable(1e+308)=false'"
113 ECHO: "[ INFO ] run_test(); t07 passed: 'is_iterable(-1e+308)=false'"
114 ECHO: "[ INFO ] run_test(); t08 passed: 'is_iterable(inf)=false'"
115 ECHO: "[ INFO ] run_test(); t09 passed: 'is_iterable(nan)=false'"
116 ECHO: "[ INFO ] run_test(); t10 passed: 'is_iterable(true)=false'"
117 ECHO: "[ INFO ] run_test(); tl1 passed: 'is_iterable(false)=false'"
118 ECHO: "[ INFO ] run_test(); t12 passed: 'is_iterable(a)=true'"
119 ECHO: "[ INFO ] run_test(); t13 passed: 'is_iterable(This is a longer string)=true'"
120 ECHO: "[ INFO ] run_test(); t14 passed: 'is_iterable()=true'"
121 ECHO: "[ INFO ] run_test(); t15 passed: 'is_iterable([])=true'"
122 ECHO: "[ INFO ] run_test(); t16 passed: 'is_iterable([undef])=true'"
123 ECHO: "[ INFO ] run_test(); t17 passed: 'is_iterable([10])=true'
124 ECHO: "[ INFO ] run_test(); t18 passed: 'is_iterable([1, 2, 3])=true'"
125 ECHO: "[ INFO ] run_test(); t19 passed: 'is_iterable([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=true'"
126 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_iterable(A shorthand range)'"
127 ECHO: "[ INFO ]
                    run_test(); t21 *skip*: 'is_iterable(A range)'
128 ECHO: "[ INFO ] run_test(); t01 passed: 'is_empty(undef)=false'"
129 ECHO: "[ INFO ]
                    run_test(); t02 passed: 'is_empty(1)=false'
130 ECHO: "[ INFO ] run_test(); t03 passed: 'is_empty(10)=false'"
131 ECHO: "[ INFO ]
                    run_test(); t04 passed: 'is_empty(1e+08)=false'
132 ECHO: "[ INFO ] run_test(); t05 passed: 'is_empty(0.001)=false'
133 ECHO: "[ INFO ] run_test(); t06 passed: 'is_empty(1e+308)=false'"
134 ECHO: "[ INFO ] run_test(); t07 passed: 'is_empty(-1e+308)=false'"
```

```
135 ECHO: "[ INFO ] run_test(); t08 passed: 'is_empty(inf)=false'"
136 ECHO: "[ INFO ] run_test(); t09 passed: 'is_empty(nan)=false'"
                    run_test(); t10 passed: 'is_empty(true)=false'"
137 ECHO: "[ INFO ]
138 ECHO: "[ INFO ] run_test(); t11 passed: 'is_empty(false)=false'"
                    run_test(); t12 passed: 'is_empty(a)=false'"
139 ECHO: "[ INFO
                    run_test(); t13 passed: 'is_empty(This is a longer string)=false'"
140 ECHO: "[ INFO ]
141 ECHO: "[ INFO
                    run_test(); t14 passed: 'is_empty()=true'"
142 ECHO: "[ INFO ] run_test(); t15 passed: 'is_empty([])=true'"
143 ECHO: "[ INFO ]
                    run_test(); t16 passed: 'is_empty([undef])=false'"
144 ECHO: "[ INFO ] run_test(); t17 passed: 'is_empty([10])=false'"
                    run_test(); t18 passed: 'is_empty([1, 2, 3])=false'"
145 ECHO: "[ INFO
146 ECHO: "[ INFO ] run_test(); t19 passed: 'is_empty([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
147 ECHO: "[ INFO ]
                    run_test(); t20 passed: 'is_empty([0 : 1 : 9])=false'"
148 ECHO: "[ INFO ]
                    run_test(); t21 passed: 'is_empty([0 : 0.5 : 9])=false'"
149 ECHO: "[ INFO ]
                    run_test(); t01 passed: 'is_number(undef)=false'
150 ECHO: "[ INFO ] run_test(); t02 passed: 'is_number(1)=true'"
151 ECHO: "[ INFO ]
                    run_test(); t03 passed: 'is_number(10)=true'"
152 ECHO: "[ INFO ] run_test(); t04 passed: 'is_number(1e+08)=true'"
153 ECHO: "[ INFO ]
                    run_test(); t05 passed: 'is_number(0.001)=true'"
154 ECHO: "[ INFO ] run_test(); t06 passed: 'is_number(1e+308)=true'"
155 ECHO: "[ INFO ]
                    run_test(); t07 passed: 'is_number(-1e+308)=true'"
156 ECHO: "[ INFO ] run_test(); t08 passed: 'is_number(inf)=true''
157 ECHO: "[ INFO ]
                    run_test(); t09 passed: 'is_number(nan)=true'"
158 ECHO: "[ INFO ] run_test(); t10 passed: 'is_number(true)=false'"
159 ECHO: "[ INFO ]
                    run_test(); t11 passed: 'is_number(false)=false'"
160 ECHO: "[ INFO ] run_test(); t12 passed: 'is_number(a)=false'
161 ECHO: "[ INFO ]
                    run_test(); t13 passed: 'is_number(This is a longer string)=false'"
162 ECHO: "[ INFO ] run_test(); t14 passed: 'is_number()=false'"
163 ECHO: "[ INFO ]
                    run_test(); t15 passed: 'is_number([])=false'"
164 ECHO: "[ INFO ] run_test(); t16 passed: 'is_number([undef])=false'"
                    run_test(); t17 passed: 'is_number([10])=false'
165 ECHO: "[ INFO ]
166 ECHO: "[ INFO ] run_test(); t18 passed: 'is_number([1, 2, 3])=false'"
160 ECHO: [INFO] run_test(); t19 passed: 'is_number([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
168 ECHO: "[INFO] run_test(); t20 passed: 'is_number([0 : 1 : 9])=false'"
169 ECHO: "[ INFO ] run_test(); t21 passed: 'is_number([0 : 0.5 : 9])=false'"
170 ECHO: "[ INFO ] run_test(); t01 passed: 'is_integer(undef)=false'"
171 ECHO: "[ INFO ] run_test(); t02 passed: 'is_integer(1)=true'
172 ECHO: "[ INFO ] run_test(); t03 passed: 'is_integer(10)=true'"
173 ECHO: "[ INFO ] run_test(); t04 passed: 'is_integer(1e+08)=true'
174 ECHO: "[ INFO ] run_test(); t05 passed: 'is_integer(0.001)=false'"
175 ECHO: "[ INFO ] run_test(); t06 passed: 'is_integer(1e+308)=true'"
176 ECHO: "[ INFO ] run_test(); t07 passed: 'is_integer(-1e+308)=true'
177 ECHO: "[ INFO ] run_test(); t08 passed: 'is_integer(inf)=false'
178 ECHO: "[ INFO ] run_test(); t09 passed: 'is_integer(nan)=false'
179 ECHO: "[ INFO ] run_test(); t10 passed: 'is_integer(true)=false'"
180 ECHO: "[ INFO ] run_test(); t11 passed: 'is_integer(false)=false'"
181 ECHO: "[ INFO ] run_test(); t12 passed: 'is_integer(a)=false'"
182 ECHO: "[ INFO ] run_test(); t13 passed: 'is_integer(This is a longer string)=false'"
183 ECHO: "[ INFO ]
                    run_test(); t14 passed: 'is_integer()=false'"
184 ECHO: "[ INFO ]
                    run_test(); t15 passed: 'is_integer([])=false'"
185 ECHO: "[ INFO ]
                    run_test(); t16 passed: 'is_integer([undef]) = false'"
186 ECHO: "[ INFO ]
                    run_test(); t17 passed: 'is_integer([10])=false'"
187 ECHO: "[ INFO ] run_test(); t18 passed: 'is_integer([1, 2, 3])=false'"
188 ECHO: "[ INFO ]
                    run_test(); t19 passed: 'is_integer([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
189 ECHO: "[ INFO ] run_test(); t20 passed: 'is_integer([0 : 1 : 9])=false'"
190 ECHO: "[ INFO ]
                    run_test(); t21 passed: 'is_integer([0 : 0.5 : 9])=false'"
191 ECHO: "[ INFO ] run_test(); t01 passed: 'is_decimal(undef)=false'"
192 ECHO: "[ INFO ]
                    run_test(); t02 passed: 'is_decimal(1)=false'
193 ECHO: "[ INFO ] run_test(); t03 passed: 'is_decimal(10)=false'"
194 ECHO: "[ INFO ]
                    run_test(); t04 passed: 'is_decimal(1e+08)=false'"
195 ECHO: "[ INFO ] run_test(); t05 passed: 'is_decimal(0.001)=true'"
196 ECHO: "[ INFO
                    run_test(); t06 passed: 'is_decimal(1e+308)=false'"
197 ECHO: "[ INFO ] run_test(); t07 passed: 'is_decimal(-1e+308)=false'"
198 ECHO: "[ INFO ]
                    run_test(); t08 passed: 'is_decimal(inf)=false'
199 ECHO: "[ INFO ]
                    run_test(); t09 passed: 'is_decimal(nan)=false'"
200 ECHO: "[ INFO ]
                    run_test(); t10 passed: 'is_decimal(true)=false'"
201 ECHO: "[ INFO ] run_test(); t11 passed: 'is_decimal(false)=false'"
202 ECHO: "[ INFO ]
                    run_test(); t12 passed: 'is_decimal(a)=false'
203 ECHO: "[ INFO ] run_test(); t13 passed: 'is_decimal(This is a longer string) = false'"
204 ECHO: "[ INFO ]
                    run_test(); t14 passed: 'is_decimal()=false'"
205 ECHO: "[ INFO ] run_test(); t15 passed: 'is_decimal([])=false'"
206 ECHO: "[ INFO ]
                    run_test(); t16 passed: 'is_decimal([undef])=false'"
207 ECHO: "[ INFO ]
                    run_test(); t17 passed: 'is_decimal([10])=false'"
208 ECHO: "[ INFO ] run_test(); t18 passed: 'is_decimal([1, 2, 3])=false'"
209 ECHO: "[ INFO ] run_test(); t19 passed: 'is_decimal([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
210 ECHO: "[ INFO ]
                    run_test(); t20 passed: 'is_decimal([0 : 1 : 9])=false'
211 ECHO: "[ INFO ] run_test(); t21 passed: 'is_decimal([0 : 0.5 : 9])=false'"
212 ECHO: "[ INFO ]
                    run_test(); t01 passed: 'is_boolean(undef)=false'
213 ECHO: "[ INFO ] run_test(); t02 passed: 'is_boolean(1)=false'
214 ECHO: "[ INFO ] run_test(); t03 passed: 'is_boolean(10)=false'"
215 ECHO: "[ INFO ] run_test(); t04 passed: 'is_boolean(1e+08)=false'"
```

```
216 ECHO: "[ INFO ] run_test(); t05 passed: 'is_boolean(0.001)=false'"
217 ECHO: "[ INFO ] run_test(); t06 passed: 'is_boolean(1e+308)=false'"
                     run_test(); t07 passed: 'is_boolean(-1e+308)=false'"
218 ECHO: "[ INFO ]
219 ECHO: "[ INFO ] run_test(); t08 passed: 'is_boolean(inf)=false''
220 ECHO: "[ INFO ]
                     run_test(); t09 passed: 'is_boolean(nan)=false'"
221 ECHO: "[ INFO ] run_test(); t10 passed: 'is_boolean(true)=true'"
222 ECHO: "[ INFO
                     run_test(); t11 passed: 'is_boolean(false)=true'"
223 ECHO: "[ INFO ] run_test(); t12 passed: 'is_boolean(a)=false'"
224 ECHO: "[ INFO ]
                     run_test(); t13 passed: 'is_boolean(This is a longer string)=false'"
225 ECHO: "[ INFO ] run_test(); t14 passed: 'is_boolean()=false'"
                     run_test(); t15 passed: 'is_boolean([])=false'"
226 ECHO: "[ INFO ]
227 ECHO: "[ INFO ] run_test(); t16 passed: 'is_boolean([undef])=false'"
228 ECHO: "[ INFO ]
                     run_test(); t17 passed: 'is_boolean([10])=false'
229 ECHO: "[ INFO ] run_test(); t18 passed: 'is_boolean([1, 2, 3])=false'"
230 ECHO: "[ INFO ] run_test(); t19 passed: 'is_boolean([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
231 ECHO: "[ INFO ] run_test(); t20 passed: 'is_boolean([0 : 1 : 9])=false'"
232 ECHO: "[ INFO ] run_test(); t21 passed: 'is_boolean([0 : 0.5 : 9])=false'"
233 ECHO: "[ INFO ] run_test(); t01 passed: 'is_string(undef)=false'"
234 ECHO: "[ INFO ] run_test(); t02 passed: 'is_string(1)=false'
235 ECHO: "[ INFO ] run_test(); t03 passed: 'is_string(10)=false'"
236 ECHO: "[ INFO ] run_test(); t04 passed: 'is_string(1e+08)=false'
237 ECHO: "[ INFO ] run_test(); t05 passed: 'is_string(0.001)=false'"
238 ECHO: "[ INFO ] run_test(); t06 passed: 'is_string(1e+308)=false''
239 ECHO: "[ INFO ] run_test(); t07 passed: 'is_string(-le+308)=false'"
240 ECHO: "[ INFO ] run_test(); t08 passed: 'is_string(inf)=false'
241 ECHO: "[ INFO ] run_test(); t09 passed: 'is_string(nan)=false'"
242 ECHO: "[ INFO ]
                    run_test(); t10 passed: 'is_string(true)=false'"
243 ECHO: "[ INFO ] run_test(); t11 passed: 'is_string(false)=false'"
244 ECHO: "[ INFO ] run_test(); t12 passed: 'is_string(a)=true'
245 ECHO: "[ INFO ] run_test(); t13 passed: 'is_string(This is a longer string)=true'"
246 ECHO: "[ INFO ] run_test(); t14 passed: 'is_string()=true'"
247 ECHO: "[ INFO ] run_test(); t15 passed: 'is_string([])=false'"
248 ECHO: "[ INFO ] run_test(); t16 passed: 'is_string([undef])=false'"
249 ECHO: "[ INFO ] run_test(); t17 passed: 'is_string([10])=false'
250 ECHO: "[ INFO ] run_test(); t18 passed: 'is_string([1, 2, 3])=false'"
251 ECHO: "[ INFO ] run_test(); t19 passed: 'is_string([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
252 ECHO: "[ INFO ] run_test(); t20 passed: 'is_string([0 : 1 : 9])=false''
253 ECHO: "[ INFO ] run_test(); t21 passed: 'is_string([0 : 0.5 : 9])=false'"
254 ECHO: "[ INFO ] run_test(); t01 passed: 'is_list(undef)=false'"
255 ECHO: "[ INFO ] run_test(); t02 passed: 'is_list(1)=false'
256 ECHO: "[ INFO ] run_test(); t03 passed: 'is_list(10)=false'"
257 ECHO: "[ INFO ] run_test(); t04 passed: 'is_list(1e+08)=false'"
258 ECHO: "[ INFO ] run_test(); t05 passed: 'is_list(0.001)=false'"
259 ECHO: "[ INFO ] run_test(); t06 passed: 'is_list(1e+308)=false'
260 ECHO: "[ INFO ] run_test(); t07 passed: 'is_list(-1e+308)=false'"
261 ECHO: "[ INFO ] run_test(); t08 passed: 'is_list(inf)=false'
262 ECHO: "[ INFO ] run_test(); t09 passed: 'is_list(nan)=false'"
263 ECHO: "[ INFO ] run_test(); t10 passed: 'is_list(true)=false''
264 ECHO: "[ INFO ] run_test(); t11 passed: 'is_list(false)=false'"
265 ECHO: "[ INFO ]
                     run_test(); t12 passed: 'is_list(a)=false'"
266 ECHO: "[ INFO ] run_test(); t13 passed: 'is_list(This is a longer string)=false'"
267 ECHO: "[ INFO ]
                     run_test(); t14 passed: 'is_list()=false'"
268 ECHO: "[ INFO ] run_test(); t15 passed: 'is_list([])=true'"
269 ECHO: "[ INFO ]
                     run_test(); t16 passed: 'is_list([undef])=true'"
270 ECHO: "[ INFO ] run_test(); t17 passed: 'is_list([10])=true'"
271 ECHO: "[ INFO ]
                     run_test(); t18 passed: 'is_list([1, 2, 3])=true'"
272 ECHO: "[ INFO ] run_test(); t19 passed: 'is_list([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=true'"
273 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_list(A shorthand range)''
274 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_list(A range)'"
                     run_test(); t01 passed: 'is_range(undef)=false'"
275 ECHO: "[ INFO ]
276 ECHO: "[ INFO ] run_test(); t02 passed: 'is_range(1)=false'"
277 ECHO: "[ INFO ]
                     run_test(); t03 passed: 'is_range(10)=false'"
278 ECHO: "[ INFO ] run_test(); t04 passed: 'is_range(1e+08)=false'"
279 ECHO: "[ INFO ] run_test(); t05 passed: 'is_range(0.001)=false'"
280 ECHO: "[ INFO ] run_test(); t06 passed: 'is_range(1e+308)=false'"
281 ECHO: "[ INFO ]
                    run_test(); t07 passed: 'is_range(-1e+308)=false'"
282 ECHO: "[ INFO ] run_test(); t08 passed: 'is_range(inf)=false'"
283 ECHO: "[ INFO ] run_test(); t09 passed: 'is_range(nan)=false'"
284 ECHO: "[ INFO ] run_test(); t10 passed: 'is_range(true)=false'"
285 ECHO: "[ INFO ] run_test(); t11 passed: 'is_range(false)=false'"
286 ECHO: "[ INFO ] run_test(); t12 passed: 'is_range(a)=false'"
287 ECHO: "[ INFO ]
                    run_test(); t13 passed: 'is_range(This is a longer string)=false'"
288 ECHO: "[ INFO ] run_test(); t14 passed: 'is_range()=false'"
289 ECHO: "[ INFO ]
                     run_test(); t15 passed: 'is_range([])=false'"
290 ECHO: "[ INFO ] run_test(); t16 passed: 'is_range([undef])=false'"
291 ECHO: "[ INFO ]
                     run_test(); t17 passed: 'is_range([10])=false'
292 ECHO: "[ INFO ] run_test(); t18 passed: 'is_range([1, 2, 3])=false'"
293 ECHO: "[ INFO ] run_test(); t19 passed: 'is_range([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
294 ECHO: "[ INFO ] run_test(); t20 passed: 'is_range([[0 : 1 : 9])=true'"
295 ECHO: "[ INFO ] run_test(); t21 passed: 'is_range([0 : 0.5 : 9])=true'"
296 ECHO: "[ INFO ] run_test(); t01 *skip*: 'is_even(The undefined value)'"
```

```
297 ECHO: "[ INFO ] run_test(); t02 passed: 'is_even(1)=false'
298 ECHO: "[ INFO ] run_test(); t03 passed: 'is_even(10)=true'"
                   run_test(); t04 passed: 'is_even(1e+08)=true'"
299 ECHO: "[ INFO ]
300 ECHO: "[ INFO ] run_test(); t05 passed: 'is_even(0.001)=false'"
                   run_test(); t06 passed: 'is_even(1e+308)=true'"
301 ECHO: "[ INFO
                   run_test(); t07 passed: 'is_even(-1e+308)=true'"
302 ECHO: "[ INFO ]
303 ECHO: "[ INFO
                   run_test(); t08 *skip*: 'is_even(The max number^2)'"
304 ECHO: "[ INFO ] run_test(); t09 *skip*: 'is_even(The invalid number nan)'"
                   run_test(); t10 *skip*: 'is_even(The boolean true)'
305 ECHO: "[ INFO ]
306 ECHO: "[ INFO ] run_test(); t11 *skip*: 'is_even(The boolean false)'"
                   run_test(); t12 *skip*: 'is_even(A character string)'"
307 ECHO: "[ INFO
308 ECHO: "[ INFO ] run_test(); t13 *skip*: 'is_even(A string)'"
309 ECHO: "[ INFO ]
                   run_test(); t14 *skip*: 'is_even(The empty string)'"
310 ECHO: "[ INFO ]
                   run_test(); t15 *skip*: 'is_even(The empty list)'
311 ECHO: "[ INFO ]
                   run_test(); t16 *skip*: 'is_even(A 1-tuple list of undef)'"
312 ECHO: "[ INFO ] run_test(); t17 *skip*: 'is_even(A 1-tuple list)'"
313 ECHO: "[ INFO ]
                   run_test(); t18 *skip*: 'is_even(A 3-tuple list)'"
314 ECHO: "[ INFO ] run_test(); t19 *skip*: 'is_even(A list of lists)'"
315 ECHO: "[ INFO ]
                   run_test(); t20 *skip*: 'is_even(A shorthand range)'"
316 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_even(A range)'"
317 ECHO: "[ INFO ]
                   run_test(); t01 *skip*: 'is_odd(The undefined value)'"
318 ECHO: "[ INFO ] run_test(); t02 passed: 'is_odd(1)=true'
319 ECHO: "[ INFO ]
                   run_test(); t03 passed: 'is_odd(10)=false'"
320 ECHO: "[ INFO ] run_test(); t04 passed: 'is_odd(1e+08)=false'"
321 ECHO: "[ INFO ]
                   run_test(); t05 passed: 'is_odd(0.001)=false'"
322 ECHO: "[ INFO ] run_test(); t06 passed: 'is_odd(1e+308)=false'"
323 ECHO: "[ INFO ]
                   run_test(); t07 passed: 'is_odd(-1e+308)=false'"
324 ECHO: "[ INFO ] run_test(); t08 *skip*: 'is_odd(The max number^2)'"
325 ECHO: "[ INFO ]
                   run_test(); t09 *skip*: 'is_odd(The invalid number nan)'"
326 ECHO: "[ INFO ] run_test(); t10 *skip*: 'is_odd(The boolean true)
327 ECHO: "[ INFO ]
                   run_test(); t11 *skip*: 'is_odd(The boolean false)'"
328 ECHO: "[ INFO ] run_test(); t12 *skip*: 'is_odd(A character string)'
329 ECHO: "[ INFO ]
                   run_test(); t13 *skip*: 'is_odd(A string)'"
330 ECHO: "[ INFO ] run_test(); t14 *skip*: 'is_odd(The empty string)''
                   run_test(); t15 *skip*: 'is_odd(The empty list)'
331 ECHO: "[ INFO ]
332 ECHO: "[ INFO ] run_test(); t16 *skip*: 'is_odd(A 1-tuple list of undef)'"
333 ECHO: "[ INFO ] run_test(); t17 *skip*: 'is_odd(A 1-tuple list)'"
334 ECHO: "[ INFO ] run_test(); t18 *skip*: 'is_odd(A 3-tuple list)'"
335 ECHO: "[ INFO ]
                   run_test(); t19 *skip*: 'is_odd(A list of lists)'"
336 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_odd(A shorthand range)'"
337 ECHO: "[ INFO ]
                   run_test(); t21 *skip*: 'is_odd(A range)'"
338 ECHO: "[ INFO ] run_test(); t01 passed: 'is_between_MM(undef)=false'"
339 ECHO: "[ INFO ]
                   run_test(); t02 passed: 'is_between_MM(1)=true'
340 ECHO: "[ INFO ] run_test(); t03 passed: 'is_between_MM(10)=true'"
341 ECHO: "[ INFO ] run_test(); t04 passed: 'is_between_MM(1e+08)=true'"
342 ECHO: "[ INFO ] run_test(); t05 passed: 'is_between_MM(0.001)=true'"
343 ECHO: "[ INFO ] run_test(); t06 passed: 'is_between_MM(1e+308)=true'"
344 ECHO: "[ INFO ] run_test(); t07 passed: 'is_between_MM(-1e+308)=true'"
345 ECHO: "[ INFO ] run_test(); t08 passed: 'is_between_MM(inf)=false'"
346 ECHO: "[ INFO ]
                   run_test(); t09 passed: 'is_between_MM(nan)=false''
347 ECHO: "[ INFO ] run_test(); t10 passed: 'is_between_MM(true)=true'"
348 ECHO: "[ INFO ]
                   run_test(); t11 passed: 'is_between_MM(false)=true'"
349 ECHO: "[ INFO ] run_test(); t12 passed: 'is_between_MM(a)=false'"
350 ECHO: "[ INFO ]
                   run_test(); t13 passed: 'is_between_MM(This is a longer string)=false'"
351 ECHO: "[ INFO ] run_test(); t14 passed: 'is_between_MM()=false''
352 ECHO: "[ INFO ]
                   run_test(); t15 passed: 'is_between_MM([])=false'"
353 ECHO: "[ INFO ] run_test(); t16 passed: 'is_between_MM([undef])=false'"
354 ECHO: "[ INFO
                   run_test(); t17 passed: 'is_between_MM([10])=false'
355 ECHO: "[ INFO ] run_test(); t18 passed: 'is_between_MM([1, 2, 3])=false'"
356 ECHO: "[ INFO ]
                   run_test(); t19 passed: 'is_between_MM([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
357 ECHO: "[ INFO ] run_test(); t20 passed: 'is_between_MM([0 : 1 : 9])=false'"
358 ECHO: "[ INFO ] run_test(); t21 passed: 'is_between_MM([0 : 0.5 : 9])=false'"
```

3.1.1.2 Iterables

- Script
- Results

3.1.1.2.1 Script

```
include <datatypes.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
show_passing = true;  // show passing tests
```

```
show_skipped = true;  // show skipped tests
echo( str("OpenSCAD Version ", version()) );
// test-values columns
test c =
  ["id", "identifier"],
  ["td", "description"],
 ["tv", "test value"]
// test-values rows
test_r =
  ["t01", "The undefined value",
                                   undef],
  ["t02", "An odd integer",
                                   11.
  ["t03", "The boolean true",
                                    truel,
  ["t04", "The boolean false",
                                    false],
  ["t05", "A character string",
                                    "a"],
 ["t06", "A string", ["t07", "The empty string",
                                    "This is a longer string"],
                                   empty_str],
  ["t08", "The empty list",
["t09", "A shorthand range",
                                    empty_lst],
                                   [0:9]],
  ["t10", "A range",
                                    [0:0.5:91],
  ["t11", "Test list 01", ["t12", "Test list 02",
                                    [undef]],
                                    [1]],
  ["t13", "Test list 03",
                                    [1, 2, 3]],
 ["t14", "Test list 04", ["t15", "Test list 05",
                                   [[1], [2], [3], [4], [5]]], [[1,2], [2,3]]],
                                   [[1,2], [2,3], [4,5], "ab"]],
[[1,2,3], [4,5,6], [7,8,9], ["a", "b", "c"]]],
  ["t16", "Test list 06",
  ["t17", "Test list 07",
  ["t18", "Test list 08",
                                   [1, 2, 3, undef]],
  ["t19", "Test list 09", ["t20", "Test list 10",
                                    [undef, undef, undef, undef]],
                                   [[undef], [undef], [undef]]],
 ["t21", "Test list 10",
["t21", "Test list 11",
["t22", "Test list 12",
["t23", "Test list 13",
                                    [true, true, true, true, false]],
                                    [true, false, false, false, false]],
                                   [true, true, true, true]]
1;
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 's' to skip test
t = true; // shortcuts
f = false:
u = undef;
s = -1;
          // skip test
good_r =
[ // function
                     01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
  ["all_equal_T",
                     ["all_equal_F",
                     ["all_equal_U",
                     ["any_equal_T",
                     "any_equal_F",
                     ["any_equal_U",
                     t, f, f, f, f, f, f, f, f, t, f, f, f, f, f, f, t, t, f, f, f, f],
  ["all defined",
                     ["any_undefined",
                     t, f, f, f, f, f, f, f, f, t, f, f, f, f, f, f, t, t, f, f, f, f],
  ["all_scalars",
                     u, t, t, t, f, f, s, s, s, s, t, t, t, f, f, f, f, t, t, f, t, t, t],
  ["all_lists",
                   u, f, f, f, f, t, t, f, f, f, f, t, t, f, t, f, f, t, f, f, f],
  ["all_strings",
                    ["all_numbers",
                     u, t, f, f, f, s, s, f, f, f, t, t, f, f],
  ["all_len_1",
                     ["all_len_2",
                    u, f, f, f, f, s, s, f, f, f, f, f, t, t, t, f, f, f, f, f, f, f],
  ["all_len_3",
                    u, f, f, f, f, s, s, f, f, f, f, f, f, f, t, f, f, f, f, f, f]
// sanity-test tables
table_check( test_r, test_c, false );
table_check( good_r, good_c, false );
// validate helper function and module
function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
module run_test( fname, fresult, vid )
 value_text = get_table_v(test_r, test_c, vid, "td");
 pass_value = get_table_v(good_r, good_c, fname, vid);
```

```
test_pass = validate( cv=fresult, t="equals", ev=pass_value, pf=true );
   test_text = validate( str(fname, "(", get_value(vid), ")=", pass_value), fresult, "equals",
   pass_value );
   if ( pass value != s )
      if ( !test_pass )
          log_warn( str(vid, "(", value_text, ") ", test_text) );
      else if ( show_passing )
  log_info( str(vid, " ", test_text) );
   else if ( show_skipped )
      log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
// Indirect function calls would be very useful here!!!
for (vid=test_ids) run_test( "all_equal_T", all_equal(get_value(vid),t), vid );
for (vid=test_ids) run_test( "all_equal_F", all_equal(get_value(vid),f), vid );
for (vid=test_ids) run_test( "all_equal_U", all_equal(get_value(vid),u), vid );
for (vid=test_ids) run_test( "any_equal_T", any_equal(get_value(vid),t), vid );
for (vid=test_ids) run_test( "any_equal_F", any_equal(get_value(vid),f), vid);
for (vid=test_ids) run_test( "any_equal_U", any_equal(get_value(vid),u), vid);
for (vid=test_ids) run_test( "all_defined", all_defined(get_value(vid)), vid);
for (vid=test_ids) run_test( "any_undefined", any_undefined(get_value(vid)), vid );
for (vid=test_ids) run_test( "all_scalars", all_scalars(get_value(vid)), vid);
for (vid=test_ids) run_test( "all_lists", all_lists(get_value(vid)), vid);
for (vid=test_ids) run_test( "all_strings", all_strings(get_value(vid)), vid );
for (vid=test_ids) run_test( "all_numbers", all_numbers(get_value(vid)), vid );
for (vid=test_ids) run_test( "all_len_1", all_len(get_value(vid),1), vid );
for (vid=test_ids) run_test( "all_len_2", all_len(get_value(vid),2), vid );
for (vid=test_ids) run_test( "all_len_3", all_len(get_value(vid),3), vid );
// end-of-tests
```

3.1.1.2.2 Results

```
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'all_equal_T(undef)=false'"
3 ECHO: "[ INFO ] run_test(); t02 passed: 'all_equal_T(1)=false'"
4 ECHO: "[ INFO ] run_test(); t03 passed: 'all_equal_T(true)=true'"
5 ECHO: "[ INFO ] run_test(); t04 passed: 'all_equal_T(false)=false'"
6 ECHO: "[ INFO ] run_test(); t05 passed: 'all_equal_T(a)=false'"
7 ECHO: "[ INFO ] run_test(); t06 passed: 'all_equal_T(This is a longer string)=false'"
8 ECHO: "[ INFO ] run_test(); t07 passed: 'all_equal_T()=true'"
9 ECHO: "[ INFO ] run_test(); t08 passed: 'all_equal_T([])=true'"
10 ECHO: "[ INFO ] run_test(); t09 passed: 'all_equal_T([0 : 1 : 9])=false'"
11 ECHO: "[ INFO ] run_test(); t10 passed: 'all_equal_T([0 : 0.5 : 9]) = false'"
12 ECHO: "[ INFO ] run_test(); t11 passed: 'all_equal_T([undef]) = false'"
13 ECHO: "[ INFO ] run_test(); t12 passed: 'all_equal_T([1])=false'"
14 ECHO: "[ INFO ] run_test(); t13 passed: 'all_equal_T([1, 2, 3])=false'"
15 ECHO: "[ INFO ] run_test(); t14 passed: 'all_equal_T([[1], [2], [3], [4], [5]])=false'"
16 ECHO: "[ INFO ] run_test(); t15 passed: 'all_equal_T([[1, 2], [2, 3]])=false'"

17 ECHO: "[ INFO ] run_test(); t16 passed: 'all_equal_T([[1, 2], [2, 3], [4, 5], "ab"])=false'"
18 ECHO: "[ INFO ] run_test(); t17 passed: 'all_equal_T([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]])=false'"
19 ECHO: "[ INFO ] run_test(); t18 passed: 'all_equal_T([1, 2, 3, undef])=false'"
20 ECHO: "[ INFO ] run_test(); t19 passed: 'all_equal_T([undef, undef, undef, undef])=false'"
21 ECHO: "[ INFO ] run_test(); t20 passed: 'all_equal_T([[undef], [undef]])=false'"
             INFO ] run_test(); t21 passed: 'all_equal_T([true, true, true, true, false])=false'"
22 ECHO: "[
23 ECHO: "[ INFO ] run_test(); t22 passed: 'all_equal_T([true, false, false, false, false])=false'"
24 ECHO: "[ INFO ] run_test(); t23 passed: 'all_equal_T([true, true, true, true])=true'"
25 ECHO: "[ INFO ] run_test(); t01 passed: 'all_equal_F(undef)=false'"
26 ECHO: "[ INFO ] run_test(); t02 passed: 'all_equal_F(1)=false'
27 ECHO: "[ INFO ] run_test(); t03 passed: 'all_equal_F(true)=false'"
28 ECHO: "[ INFO ] run_test(); t04 passed: 'all_equal_F(false)=true'"
29 ECHO: "[ INFO ] run_test(); t05 passed: 'all_equal_F(a)=false'"
30 ECHO: "[ INFO ] run_test(); t06 passed: 'all_equal_F(This is a longer string)=false'"
31 ECHO: "[ INFO ] run_test(); t07 passed: 'all_equal_F()=true'"
32 ECHO: "[ INFO ] run_test(); t08 passed: 'all_equal_F([])=true'"
33 ECHO: "[ INFO ] run_test(); t09 passed: 'all_equal_F([0 : 1 : 9])=false'"
34 ECHO: "[ INFO ] run_test(); t10 passed: 'all_equal_F([0 : 0.5 : 9])=false'"
35 ECHO: "[ INFO ] run_test(); tll passed: 'all_equal_F([undef])=false'"
36 ECHO: "[ INFO ] run_test(); t12 passed: 'all_equal_F([1])=false'"
37 ECHO: "[ INFO ] run_test(); t13 passed: 'all_equal_F([1, 2, 3])=false'"
38 ECHO: "[INFO] run_test(); t14 passed: 'all_equal_F([[1], [2], [3], [4], [5]])=false'"

39 ECHO: "[INFO] run_test(); t15 passed: 'all_equal_F([[1, 2], [2, 3]])=false'"
40 ECHO: "[ INFO ] run_test(); t16 passed: 'all_equal_F([[1, 2], [2, 3], [4, 5], "ab"])=false'"
41 ECHO: "[ INFO ] run_test(); t17 passed: 'all_equal_F([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]]) = false'
```

```
42 ECHO: "[ INFO ] run_test(); t18 passed: 'all_equal_F([1, 2, 3, undef])=false'"
43 ECHO: "[ INFO ] run_test(); t19 passed: 'all_equal_F([undef, undef, undef, undef])=false'"
44 ECHO: "[ INFO ] run_test(); t20 passed: 'all_equal_F([[undef], [undef], [undef]])=false'
45 ECHO: "[ INFO ] run_test(); t21 passed: 'all_equal_F([true, true, true, true, false])=false'"
             INFO ] run_test(); t22 passed: 'all_equal_F([true, false, false, false, false])=false'"
46 ECHO: "[
47 ECHO: "[ INFO ] run_test(); t23 passed: 'all_equal_F([true, true, true, true])=false'"
48 ECHO: "[
             INFO ] run_test(); t01 passed: 'all_equal_U(undef)=true'"
49 ECHO: "[ INFO ] run_test(); t02 passed: 'all_equal_U(1) = false'"
50 ECHO: "[ INFO ] run_test(); t03 passed: 'all_equal_U(true)=false'"
51 ECHO: "[ INFO ] run_test(); t04 passed: 'all_equal_U(false)=false'"
52 ECHO: "[ INFO ] run_test(); t05 passed: 'all_equal_U(a)=false'
53 ECHO: "[ INFO ] run_test(); t06 passed: 'all_equal_U(This is a longer string)=false'"
54 ECHO: "[ INFO ] run_test(); t07 passed: 'all_equal_U()=true'"
55 ECHO: "[ INFO ] run_test(); t08 passed: 'all_equal_U([])=true'"
56 ECHO: "[ INFO ] run_test(); t09 passed: 'all_equal_U([0 : 1 : 9])=false'"
57 ECHO: "[ INFO ] run_test(); t10 passed: 'all_equal_U([0 : 0.5 : 9]) = false'"
58 ECHO: "[ INFO ] run_test(); t11 passed: 'all_equal_U([undef]) = true'
59 ECHO: "[ INFO ] run_test(); t12 passed: 'all_equal_U([1])=false'"
60 ECHO: "[ INFO ] run_test(); t13 passed: 'all_equal_U([1, 2, 3])=false'"
61 ECHO: "[ INFO ] run_test(); t14 passed: 'all_equal_U([[1], [2], [3], [4], [5]])=false'"
62 ECHO: "[ INFO ] run_test(); t15 passed: 'all_equal_U([[1, 2], [2, 3]])=false'"
63 ECHO: "[ INFO ] run_test(); t16 passed: 'all_equal_U([[1, 2], [2, 3], [4, 5], "ab"])=false'"
64 ECHO: "[ INFO ] run_test(); t17 passed: 'all_equal_U([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
65 ECHO: "[ INFO ] run_test(); t18 passed: 'all_equal_U([1, 2, 3, undef])=false'"
66 ECHO: "[ INFO ] run_test(); t19 passed: 'all_equal_U([undef, undef, undef, undef])=true'"
67 ECHO: "[ INFO ] run_test(); t20 passed: 'all_equal_U([[undef], [undef]])=false'"
68 ECHO: "[ INFO ] run_test(); t21 passed: 'all_equal_U([true, true, true, true, false])=false'"
69 ECHO: "[ INFO ] run_test(); t22 passed: 'all_equal_U([true, false, false, false, false])=false'"
70 ECHO: "[ INFO ] run_test(); t23 passed: 'all_equal_U([true, true, true, true])=false'
71 ECHO: "[ INFO ] run_test(); t01 passed: 'any_equal_T(undef)=false'"
72 ECHO: "[ INFO ] run_test(); t02 passed: 'any_equal_T(1)=false'"
73 ECHO: "[ INFO ] run_test(); t03 passed: 'any_equal_T(true)=true'
74 ECHO: "[ INFO ] run_test(); t04 passed: 'any_equal_T(false)=false'"
75 ECHO: "[ INFO ] run_test(); t05 passed: 'any_equal_T(a)=false'
76 ECHO: "[ INFO ] run_test(); t06 passed: 'any_equal_T(This is a longer string)=false'"
77 ECHO: "[ INFO ] run_test(); t07 passed: 'any_equal_T()=false'"
78 ECHO: "[ INFO ] run_test(); t08 passed: 'any_equal_T([])=false'"
79 ECHO: "[ INFO ] run_test(); t09 passed: 'any_equal_T([0 : 1 : 9])=false'"
80 ECHO: "[ INFO ] run_test(); t10 passed: 'any_equal_T([0 : 0.5 : 9])=false'"
81 ECHO: "[ INFO ] run_test(); tll passed: 'any_equal_T([undef])=false'"
82 ECHO: "[ INFO ] run_test(); t12 passed: 'any_equal_T([1])=false'
83 ECHO: "[ INFO ] run_test(); t13 passed: 'any_equal_T([1, 2, 3])=false'"
84 ECHO: "[ INFO ] run_test(); t14 passed: 'any_equal_T([[1], [2], [3], [4], [5]])=false'"
85 ECHO: "[ INFO ] run_test(); t15 passed: 'any_equal_T([[1, 2], [2, 3]])=false'"
86 ECHO: "[ INFO ] run_test(); t16 passed: 'any_equal_T([[1, 2], [2, 3], [4, 5], "ab"])=false'"
87 ECHO: "[ INFO ] run_test(); t17 passed: 'any_equal_T([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
88 ECHO: "[ INFO ] run_test(); t18 passed: 'any_equal_T([1, 2, 3, undef])=false'"
89 ECHO: "[ INFO ] run_test(); t19 passed: 'any_equal_T([undef, undef, undef, undef])=false'"
90 ECHO: "[ INFO ] run_test(); t20 passed: 'any_equal_T([[undef], [undef], [undef]])=false'"
91 ECHO: "[ INFO ] run_test(); t21 passed: 'any_equal_T([true, true, true, true, false])=true'"
92 ECHO: "[ INFO ] run_test(); t22 passed: 'any_equal_T([true, false, false, false, false])=true'"
93 ECHO: "[ INFO ] run_test(); t23 passed: 'any_equal_T([true, true, true, true])=true'
94 ECHO: "[ INFO ] run_test(); t01 passed: 'any_equal_F (undef) = false'"
95 ECHO: "[ INFO ] run_test(); t02 passed: 'any_equal_F(1)=false'"
96 ECHO: "[ INFO ] run_test(); t03 passed: 'any_equal_F(true)=false'"
97 ECHO: "[ INFO ] run_test(); t04 passed: 'any_equal_F(false)=true'"
98 ECHO: "[ INFO ] run_test(); t05 passed: 'any_equal_F(a)=false''
99 ECHO: "[ INFO ] run_test(); t06 passed: 'any_equal_F(This is a longer string)=false'"
100 ECHO: "[ INFO ] run_test(); t07 passed: 'any_equal_F()=false'"
101 ECHO: "[ INFO ] run_test(); t08 passed: 'any_equal_F([])=false'"
102 ECHO: "[ INFO ] run_test(); t09 passed: 'any_equal_F([0 : 1 : 9])=false'"
103 ECHO: "[ INFO ] run_test(); t10 passed: 'any_equal_F([0 : 0.5 : 9])=false'"
104 ECHO: "[ INFO ] run_test(); t11 passed: 'any_equal_F([undef])=false'"
105 ECHO: "[ INFO ] run_test(); t12 passed: 'any_equal_F([1])=false'"
106 ECHO: "[ INFO ] run_test(); t13 passed: 'any_equal_F([1, 2, 3])=false'"
107 ECHO: "[ INFO ] run_test(); t14 passed: 'any_equal_F([[1], [2], [3], [4], [5]])=false'"
108 ECHO: "[ INFO ] run_test(); t15 passed: 'any_equal_F([[1, 2], [2, 3])]=false'"

109 ECHO: "[ INFO ] run_test(); t16 passed: 'any_equal_F([[1, 2], [2, 3], [4, 5], "ab"])=false'"

110 ECHO: "[ INFO ] run_test(); t17 passed: 'any_equal_F([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
111 ECHO: "[ INFO ] run_test(); t18 passed: 'any_equal_F([1, 2, 3, undef])=false'"
112 ECHO: "[ INFO ] run_test(); t19 passed: 'any_equal_F([undef, undef, undef])=false'"
113 ECHO: "[ INFO ] run_test(); t20 passed: 'any_equal_F([[undef], [undef], [undef]])=false'"
114 ECHO: "[ INFO ] run_test(); t21 passed: 'any_equal_F([true, true, true, true, false])=true'"
115 ECHO: "[ INFO ] run_test(); t22 passed: 'any_equal_F([true, false, false, false, false])=true'"
116 ECHO: "[ INFO ] run_test(); t23 passed: 'any_equal_F([true, true, true, true])=false'
117 ECHO: "[ INFO ] run_test(); t01 passed: 'any_equal_U(undef)=true'"
118 ECHO: "[ INFO ] run_test(); t02 passed: 'any_equal_U(1)=false'"
119 ECHO: "[ INFO ] run_test(); t03 passed: 'any_equal_U(true)=false'"
```

```
120 ECHO: "[ INFO ] run_test(); t04 passed: 'any_equal_U(false)=false'"
121 ECHO: "[ INFO ] run_test(); t05 passed: 'any_equal_U(a)=false'"
                      run_test(); t06 passed: 'any_equal_U(This is a longer string)=false'"
122 ECHO: "[ INFO ]
123 ECHO: "[ INFO ] run_test(); t07 passed: 'any_equal_U()=false'"
                      run_test(); t08 passed: 'any_equal_U([])=false'"
124 ECHO: "[ INFO ]
125 ECHO: "[ INFO ]
                     run_test(); t09 passed: 'any_equal_U([0 : 1 : 9])=false'"
126 ECHO: "[ INFO
                     run_test(); t10 passed: 'any_equal_U([0 : 0.5 : 9]) = false'"
127 ECHO: "[ INFO ] run_test(); t11 passed: 'any_equal_U([undef])=true'"
128 ECHO: "[ INFO ]
                     run_test(); t12 passed: 'any_equal_U([1])=false'"
129 ECHO: "[ INFO ] run_test(); t13 passed: 'any_equal_U([1, 2, 3])=false'"
130 ECHO: "[ INFO ]
                     run_test(); t14 passed: 'any_equal_U([[1], [2], [3], [4], [5]])=false'"
131 ECHO: "[ INFO ] run_test(); t15 passed: 'any_equal_U([[1, 2], [2, 3]])=false'"
132 ECHO: "[ INFO ] run_test(); t16 passed: 'any_equal_U([[1, 2], [2, 3], [4, 5], "ab"])=false'"
133 ECHO: "[ INFO ] run_test(); t17 passed: 'any_equal_U([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]])=false'"
134 ECHO: "[ INFO ] run_test(); t18 passed: 'any_equal_U([1, 2, 3, undef])=true'"
135 ECHO: "[ INFO ] run_test(); t19 passed: 'any_equal_U([undef, undef, undef, undef])=true'"
136 ECHO: "[ INFO ] run_test(); t20 passed: 'any_equal_U([[undef], [undef], [undef]])=false'"
137 ECHO: "[ INFO ]
                     run_test(); t21 passed: 'any_equal_U([true, true, true, true, false])=false'"
138 ECHO: "[ INFO ] run_test(); t22 passed: 'any_equal_U([true, false, false, false, false)]=false'"
139 ECHO: "[ INFO ] run_test(); t23 passed: 'any_equal_U([true, true, true, true])=false'
140 ECHO: "[ INFO ] run_test(); t01 passed: 'all_defined(undef)=false'"
141 ECHO: "[ INFO ]
                     run_test(); t02 passed: 'all_defined(1)=true'"
142 ECHO: "[ INFO ] run_test(); t03 passed: 'all_defined(true)=true'"
143 ECHO: "[ INFO ]
                     run_test(); t04 passed: 'all_defined(false)=true'"
144 ECHO: "[ INFO ] run_test(); t05 passed: 'all_defined(a)=true'"
145 ECHO: "[ INFO ]
                     run_test(); t06 passed: 'all_defined(This is a longer string)=true'"
146 ECHO: "[ INFO ] run_test(); t07 passed: 'all_defined()=true'"
147 ECHO: "[ INFO ] run_test(); t08 passed: 'all_defined([])=true'"
148 ECHO: "[ INFO ] run_test(); t09 passed: 'all_defined([0 : 1 : 9])=true'"
149 ECHO: "[ INFO ] run_test(); t10 passed: 'all_defined([0 : 0.5 : 9])=true'"
150 ECHO: "[ INFO ] run_test(); t11 passed: 'all_defined([undef])=false'"
151 ECHO: "[ INFO ] run_test(); t12 passed: 'all_defined([1])=true'"
152 ECHO: "[ INFO ] run_test(); t13 passed: 'all_defined([1, 2, 3])=true'"
153 ECHO: "[ INFO ] run_test(); t14 passed: 'all_defined([[1], [2], [3], [4], [5]])=true'"
154 ECHO: "[ INFO ] run_test(); t15 passed: 'all_defined([[1, 2], [2, 3]])=true'"
154 ECHO: [INFO] run_test(); t16 passed: 'all_defined([[1, 2], [2, 3], [4, 5], "ab"])=true'"
156 ECHO: "[INFO] run_test(); t17 passed: 'all_defined([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]])=true'"
157 ECHO: "[ INFO ] run_test(); t18 passed: 'all_defined([1, 2, 3, undef])=false'"
158 ECHO: "[ INFO ] run_test(); t19 passed: 'all_defined([undef, undef, undef])=false'"
159 ECHO: "[ INFO ] run_test(); t20 passed: 'all_defined([[undef], [undef], [undef]])=true'"
160 ECHO: "[ INFO ] run_test(); t21 passed: 'all_defined([true, true, true, true, false])=true'"
161 ECHO: "[ INFO ] run_test(); t22 passed: 'all_defined([true, false, false, false, false])=true'"
162 ECHO: "[ INFO ] run_test(); t23 passed: 'all_defined([true, true, true, true])=true'"
163 ECHO: "[ INFO ] run_test(); t01 passed: 'any_undefined(undef)=true'
164 ECHO: "[ INFO ] run_test(); t02 passed: 'any_undefined(1)=false'"
165 ECHO: "[ INFO ] run_test(); t03 passed: 'any_undefined(true)=false'"
166 ECHO: "[ INFO ] run_test(); t04 passed: 'any_undefined(false)=false'"
167 ECHO: "[ INFO ]
                     run_test(); t05 passed: 'any_undefined(a)=false'"
168 ECHO: "[ INFO ] run_test(); t06 passed: 'any_undefined(This is a longer string)=false'"
169 ECHO: "[ INFO ]
                     run_test(); t07 passed: 'any_undefined()=false'"
170 ECHO: "[ INFO ] run_test(); t08 passed: 'any_undefined([])=false'"
171 ECHO: "[ INFO ]
                     run_test(); t09 passed: 'any_undefined([0 : 1 : 9])=false'"
172 ECHO: "[ INFO ] run_test(); t10 passed: 'any_undefined([0 : 0.5 : 9])=false'"
173 ECHO: "[ INFO ]
                     run_test(); t11 passed: 'any_undefined([undef])=true'"
174 ECHO: "[ INFO ] run_test(); t12 passed: 'any_undefined([1])=false'"
175 ECHO: "[ INFO ]
                     run_test(); t13 passed: 'any_undefined([1, 2, 3])=false'"
176 ECHO: "[ INFO ]
                     run_test(); t14 passed: 'any_undefined([[1], [2], [3], [4], [5]])=false'"
177 ECHO: "[ INFO ] run_test(); t15 passed: 'any_undefined([[1, 2], [2, 3]])=false'"
178 ECHO: "[ INFO ] run_test(); t16 passed: 'any_undefined([[1, 2], [2, 3], [4, 5], "ab"])=false'"
179 ECHO: "[ INFO ] run_test(); t17 passed: 'any_undefined([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]])=false'"
180 ECHO: "[ INFO ] run_test(); t18 passed: 'any_undefined([1, 2, 3, undef])=true'"
181 ECHO: "[ INFO ]
                     run_test(); t19 passed: 'any_undefined([undef, undef, undef, undef])=true'"
                     run_test(); t20 passed: 'any_undefined([[undef], [undef], [undef]])=false'"
182 ECHO: "[ INFO ]
183 ECHO: "[ INFO ] run_test(); t21 passed: 'any_undefined([true, true, true, true, false])=false'"
184 ECHO: "[ INFO ]
                     run_test(); t22 passed: 'any_undefined([true, false, false, false, false])=false'"
185 ECHO: "[ INFO ] run_test(); t23 passed: 'any_undefined([true, true, true, true])=false'"
186 ECHO: "[ INFO ]
                     run_test(); t01 passed: 'all_scalars(undef) = undef'
187 ECHO: "[ INFO ] run_test(); t02 passed: 'all_scalars(1)=true'"
188 ECHO: "[ INFO ]
                     run_test(); t03 passed: 'all_scalars(true)=true'
189 ECHO: "[ INFO ] run_test(); t04 passed: 'all_scalars(false)=true'"
190 ECHO: "[ INFO ]
                     run_test(); t05 passed: 'all_scalars(a)=false'"
191 ECHO: "[ INFO ] run_test(); t06 passed: 'all_scalars(This is a longer string) = false'"
                     run_test(); t07 *skip*: 'all_scalars(The empty string)'"
192 ECHO: "[ INFO ]
193 ECHO: "[ INFO ] run_test(); t08 *skip*: 'all_scalars(The empty list)'"
194 ECHO: "[ INFO ]
                     run_test(); t09 *skip*: 'all_scalars(A shorthand range)'"
195 ECHO: "[ INFO ] run_test(); t10 *skip*: 'all_scalars(A range)'"
196 ECHO: "[ INFO ] run_test(); t11 passed: 'all_scalars([undef])=true'"
197 ECHO: "[ INFO ] run_test(); t12 passed: 'all_scalars([1])=true'"
```

```
198 ECHO: "[ INFO ] run_test(); t13 passed: 'all_scalars([1, 2, 3])=true'"
199 ECHO: "[ INFO ] run_test(); t14 passed: 'all_scalars([[1], [2], [3], [4], [5]])=false'"
200 ECHO: "[ INFO ] run_test(); t15 passed: 'all_scalars([[1, 2], [2, 3]])=false'"
201 ECHO: "[ INFO ] run_test(); t16 passed: 'all_scalars([[1, 2], [2, 3], [4, 5], "ab"])=false'"
202 ECHO: "[ INFO ] run_test(); t17 passed: 'all_scalars([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
203 ECHO: "[ INFO ] run_test(); t18 passed: 'all_scalars([1, 2, 3, undef])=true'"
204 ECHO: "[ INFO ] run_test(); t19 passed: 'all_scalars([undef, undef, undef, undef])=true'"
205 ECHO: "[ INFO ] run_test(); t20 passed: 'all_scalars([[undef], [undef], [undef]])=false'"
206 ECHO: "[ INFO ] run_test(); t21 passed: 'all_scalars([true, true, true, true, false])=true'"
207 ECHO: "[ INFO ]
                       run_test(); t22 passed: 'all_scalars([true, false, false, false, false])=true'"
208 ECHO: "[ INFO ] run_test(); t23 passed: 'all_scalars([true, true, true, true])=true'"
209 ECHO: "[ INFO ] run_test(); t01 passed: 'all_lists(undef)=undef'"
210 ECHO: "[ INFO ] run_test(); t02 passed: 'all_lists(1)=false'"
211 ECHO: "[ INFO ] run_test(); t03 passed: 'all_lists(true)=false'
212 ECHO: "[ INFO ] run_test(); t04 passed: 'all_lists(false)=false'"
213 ECHO: "[ INFO ] run_test(); t05 passed: 'all_lists(a)=false'"
214 ECHO: "[ INFO ] run_test(); t06 passed: 'all_lists(This is a longer string)=false'"
215 ECHO: "[ INFO ] run_test(); t07 passed: 'all_lists()=true'"
216 ECHO: "[ INFO ] run_test(); t08 passed: 'all_lists([])=true'"
217 ECHO: "[ INFO ] run_test(); t09 passed: 'all_lists([0 : 1 : 9])=false'"
218 ECHO: "[ INFO ] run_test(); t10 passed: 'all_lists([0 : 0.5 : 9])=false'"
219 ECHO: "[ INFO ] run_test(); t11 passed: 'all_lists([undef])=false''
220 ECHO: "[ INFO ] run_test(); t12 passed: 'all_lists([1])=false'"
221 ECHO: "[ INFO ] run_test(); t13 passed: 'all_lists([1, 2, 3])=false'"
222 ECHO: "[ INFO ] run_test(); t14 passed: 'all_lists([[1], [2], [3], [4], [5]])=true'"
223 ECHO: "[ INFO ] run_test(); t15 passed: 'all_lists([[1, 2], [2, 3]))=true'"
224 ECHO: "[ INFO ] run_test(); t16 passed: 'all_lists([[1, 2], [2, 3], [4, 5], "ab"])=false'"
225 ECHO: "[ INFO ] run_test(); t17 passed: 'all_lists([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=true'"
226 ECHO: "[ INFO ] run_test(); t18 passed: 'all_lists([1, 2, 3, undef])=false'"
227 ECHO: "[ INFO ] run_test(); t19 passed: 'all_lists([undef, undef, undef, undef])=false'"
228 ECHO: "[ INFO ] run_test(); t20 passed: 'all_lists([[undef], [undef], [undef]])=true'
229 ECHO: "[ INFO ] run_test(); t21 passed: 'all_lists([true, true, true, true, false])=false'"
230 ECHO: "[ INFO ] run_test(); t22 passed: 'all_lists([true, false, false, false, false])=false'"
231 ECHO: "[ INFO ] run_test(); t23 passed: 'all_lists([true, true, true, true])=false'
232 ECHO: "[ INFO ] run_test(); t01 passed: 'all_strings(undef)=undef'"
233 ECHO: "[ INFO ] run_test(); t02 passed: 'all_strings(1)=false'
234 ECHO: "[ INFO ] run_test(); t03 passed: 'all_strings(true)=false'
235 ECHO: "[ INFO ] run_test(); t04 passed: 'all_strings(false)=false'"
236 ECHO: "[ INFO ] run_test(); t05 passed: 'all_strings(a)=true'"
237 ECHO: "[ INFO ] run_test(); t06 passed: 'all_strings(This is a longer string)=true'"
238 ECHO: "[ INFO ] run_test(); t07 passed: 'all_strings()=true'"
239 ECHO: "[ INFO ] run_test(); t08 *skip*: 'all_strings(The empty list)'"
240 ECHO: "[ INFO ] run_test(); t09 passed: 'all_strings([0 : 1 : 9])=false'"
241 ECHO: "[ INFO ] run_test(); t10 passed: 'all_strings([0 : 0.5 : 9])=false'"
242 ECHO: "[ INFO ] run_test(); t11 passed: 'all_strings([undef])=false'"
243 ECHO: "[ INFO ] run_test(); t12 passed: 'all_strings([1])=false'"
244 ECHO: "[ INFO ] run_test(); t13 passed: 'all_strings([1, 2, 3])=false'"
245 ECHO: "[ INFO ] run_test(); t14 passed: 'all_strings([[1], [2], [3], [4], [5]])=false'" 246 ECHO: "[ INFO ] run_test(); t15 passed: 'all_strings([[1, 2], [2, 3]])=false'"
247 ECHO: "[ INFO ] run_test(); t16 passed: 'all_strings([[1, 2], [2, 3], [4, 5], "ab"])=false'"
248 ECHO: "[ INFO ] run_test(); t17 passed: 'all_strings([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
249 ECHO: "[ INFO ] run_test(); t18 passed: 'all_strings([1, 2, 3, undef])=false'"
250 ECHO: "[ INFO ] run_test(); t19 passed: 'all_strings([undef, undef, undef, undef])=false'"
251 ECHO: "[ INFO ] run_test(); t20 passed: 'all_strings([[undef], [undef], [undef]])=false'"
252 ECHO: "[ INFO ] run_test(); t21 passed: 'all_strings([true, true, true, true, false])=false'"
253 ECHO: "[ INFO ] run_test(); t22 passed: 'all_strings([true, false, false, false, false])=false'"
254 ECHO: "[ INFO ] run_test(); t23 passed: 'all_strings([true, true, true, true])=false''
255 ECHO: "[ INFO ] run_test(); t01 passed: 'all_numbers(undef) = undef'"
256 ECHO: "[ INFO ] run_test(); t02 passed: 'all_numbers(1)=true'"
257 ECHO: "[ INFO ] run_test(); t03 passed: 'all_numbers(true)=false'"
258 ECHO: "[ INFO ] run_test(); t04 passed: 'all_numbers(false)=false'"
259 ECHO: "[ INFO ] run_test(); t05 passed: 'all_numbers(a)=false'"
260 ECHO: "[ INFO ] run_test(); t06 passed: 'all_numbers(This is a longer string)=false'"
261 ECHO: "[ INFO ] run_test(); t07 *skip*: 'all_numbers(The empty string)'"
262 ECHO: "[ INFO ] run_test(); t08 *skip*: 'all_numbers(The empty list)'"
263 ECHO: "[ INFO ] run_test(); t09 passed: 'all_numbers([0 : 1 : 9])=false'"
264 ECHO: "[ INFO ] run_test(); t10 passed: 'all_numbers([0 : 0.5 : 9])=false'"
265 ECHO: "[ INFO ] run_test(); t11 passed: 'all_numbers([undef])=false'"
266 ECHO: "[ INFO ] run_test(); t12 passed: 'all_numbers([1])=true'"
267 ECHO: "[ INFO ] run_test(); t13 passed: 'all_numbers([1, 2, 3])=true'"
268 ECHO: "[ INFO ] run_test(); t14 passed: 'all_numbers([[1], [2], [3], [4], [5]])=false'"
269 ECHO: "[ INFO ] run_test(); t15 passed: 'all_numbers([[1, 2], [2, 3]])=false'"
270 ECHO: "[ INFO ] run_test(); t16 passed: 'all_numbers([[1, 2], [2, 3], [4, 5], "ab"])=false'"
271 ECHO: "[ INFO ] run_test(); t17 passed: 'all_numbers([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
272 ECHO: "[ INFO ] run_test(); t18 passed: 'all_numbers([1, 2, 3, undef])=false'
273 ECHO: "[ INFO ] run_test(); t19 passed: 'all_numbers([undef, undef, undef, undef])=false'"
274 ECHO: "[ INFO ] run_test(); t20 passed: 'all_numbers([[undef], [undef], [undef]])=false'"
```

```
275 ECHO: "[ INFO ] run_test(); t21 passed: 'all_numbers([true, true, true, true, false])=false'"
276 ECHO: "[ INFO ] run_test(); t22 passed: 'all_numbers([true, false, false, false, false])=false'"
277 ECHO: "[ INFO ]
                        run_test(); t23 passed: 'all_numbers([true, true, true, true])=false'
278 ECHO: "[ INFO ] run_test(); t01 passed: 'all_len_1(undef)=undef'"
                        run_test(); t02 passed: 'all_len_1(1)=false'"
279 ECHO: "[ INFO
                        run_test(); t03 passed: 'all_len_1(true)=false'"
280 ECHO: "[ INFO ]
281 ECHO: "[ INFO
                        run_test(); t04 passed: 'all_len_1(false)=false'"
282 ECHO: "[ INFO ] run_test(); t05 passed: 'all_len_1(a)=true'"
283 ECHO: "[ INFO ]
                        run_test(); t06 passed: 'all_len_1(This is a longer string)=true'"
284 ECHO: "[ INFO ] run_test(); t07 *skip*: 'all_len_1(The empty string)'"
                        run_test(); t08 *skip*: 'all_len_1(The empty list)'
285 ECHO: "[ INFO
286 ECHO: "[ INFO ] run_test(); t09 passed: 'all_len_1([0 : 1 : 9])=false'"
287 ECHO: "[ INFO ]
                        run_test(); t10 passed: 'all_len_1([0 : 0.5 : 9])=false'"
288 ECHO: "[ INFO ] run_test(); t11 passed: 'all_len_1([undef])=false'"
289 ECHO: "[ INFO 1
                        run_test(); t12 passed: 'all_len_1([1])=false'"
290 ECHO: "[ INFO ] run_test(); t13 passed: 'all_len_1([1, 2, 3])=false'"
291 ECHO: "[ INFO ]
                        run_test(); t14 passed: 'all_len_1([[1], [2], [3], [4], [5]])=true'"
292 ECHO: "[ INFO ] run_test(); t15 passed: 'all_len_1([[1, 2], [2, 3]])=false'"
293 ECHO: "[ INFO ] run_test(); t16 passed: 'all_len_1([[1, 2], [2, 3], [4, 5], "ab"])=false'"
294 ECHO: "[ INFO ] run_test(); t17 passed: 'all_len_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false''
295 ECHO: "[ INFO ] run_test(); t18 passed: 'all_len_1([1, 2, 3, undef])=false'"
296 ECHO: "[ INFO ] run_test(); t19 passed: 'all_len_1([undef, undef, undef])=false'"
297 ECHO: "[ INFO ] run_test(); t20 passed: 'all_len_1([[undef], [undef], [undef]))=true'"
298 ECHO: "[ INFO ]
                        run_test(); t21 passed: 'all_len_1([true, true, true, true, false])=false'"
299 ECHO: "[ INFO ] run_test(); t22 passed: 'all_len_1([true, false, false, false, false])=false'"
300 ECHO: "[ INFO ] run_test(); t23 passed: 'all_len_1([true, true, true, true])=false' 301 ECHO: "[ INFO ] run_test(); t01 passed: 'all_len_2(undef)=undef'"
                        run_test(); t02 passed: 'all_len_2(1)=false'
302 ECHO: "[ INFO ]
303 ECHO: "[ INFO ] run_test(); t03 passed: 'all_len_2(true)=false'"
                        run_test(); t04 passed: 'all_len_2(false)=false'"
304 ECHO: "[ INFO ]
305 ECHO: "[ INFO ] run_test(); t05 passed: 'all_len_2(a)=false'"
306 ECHO: "[ INFO ]
                        run_test(); t06 passed: 'all_len_2(This is a longer string)=false'"
307 ECHO: "[ INFO ] run_test(); t07 *skip*: 'all_len_2(The empty string)'
308 ECHO: "[ INFO ] run_test(); t08 *skip*: 'all_len_2(The empty list)'
309 ECHO: "[ INFO ] run_test(); t09 passed: 'all_len_2([0 : 1 : 9])=false'
310 ECHO: "[ INFO ] run_test(); t10 passed: 'all_len_2([0 : 0.5 : 9])=false'"
311 ECHO: "[ INFO ] run_test(); t11 passed: 'all_len_2([undef])=false'
312 ECHO: "[ INFO ] run_test(); t12 passed: 'all_len_2([1])=false'"
313 ECHO: "[ INFO ] run_test(); t13 passed: 'all_len_2([1, 2, 3])=false'"
314 ECHO: "[ INFO ] run_test(); t14 passed: 'all_len_2([[1], [2], [3], [4], [5]])=false'"
315 ECHO: "[ INFO ] run_test(); t15 passed: 'all_len_2([[1, 2], [2, 3]])=true'"
316 ECHO: "[ INFO ] run_test(); t15 passed: 'all_ten_2([[1, 2], [2, 3]) = true'"
316 ECHO: "[ INFO ] run_test(); t16 passed: 'all_ten_2([[1, 2], [2, 3], [4, 5], "ab"]) = true'"
317 ECHO: "[ INFO ] run_test(); t17 passed: 'all_ten_2([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
318 ECHO: "[ INFO ] run_test(); t18 passed: 'all_len_2([1, 2, 3, undef])=false'"
319 ECHO: "[ INFO ] run_test(); t19 passed: 'all_len_2([undef, undef, undef, undef])=false'"
320 ECHO: "[ INFO ] run_test(); t20 passed: 'all_len_2([[undef], [undef], [undef]])=false'"
321 ECHO: "[ INFO ] run_test(); t21 passed: 'all_len_2([true, true, true, true, false])=false'"
322 ECHO: "[ INFO ] run_test(); t22 passed: 'all_len_2([true, false, false, false, false)]=false'"
323 ECHO: "[ INFO ] run_test(); t23 passed: 'all_len_2([true, true, true, true])=false'"
324 ECHO: "[ INFO ]
                        run_test(); t01 passed: 'all_len_3(undef) = undef'"
325 ECHO: "[ INFO ] run_test(); t02 passed: 'all_len_3(1)=false'"
326 ECHO: "[ INFO ]
                        run_test(); t03 passed: 'all_len_3(true)=false'"
327 ECHO: "[ INFO ] run_test(); t04 passed: 'all_len_3(false)=false'"
328 ECHO: "[ INFO ]
                        run_test(); t05 passed: 'all_len_3(a)=false'"
329 ECHO: "[ INFO ] run_test(); t06 passed: 'all_len_3(This is a longer string)=false'"
330 ECHO: "[ INFO ] run_test(); t07 *skip*: 'all_len_3(The empty string)'"
331 ECHO: "[ INFO ] run_test(); t08 *skip*: 'all_len_3(The empty list)'"
332 ECHO: "[ INFO ]
                        run_test(); t09 passed: 'all_len_3([0 : 1 : 9])=false'"
333 ECHO: "[ INFO ] run_test(); t10 passed: 'all_len_3([0 : 0.5 : 9])=false'"
334 ECHO: "[ INFO ]
                        run_test(); t11 passed: 'all_len_3([undef])=false''
335 ECHO: "[ INFO ] run_test(); t12 passed: 'all_len_3([1])=false'"
336 ECHO: "[ INFO ] run_test(); t13 passed: 'all_len_3([1, 2, 3])=false'"
337 ECHO: "[ INFO ] run_test(); t14 passed: 'all_len_3([[1], [2], [3], [4], [5]])=false'"
338 ECHO: "[ INFO ] run_test(); t15 passed: 'all_len_3([[1, 2], [2, 3]))=false'"
339 ECHO: "[ INFO ] run_test(); t16 passed: 'all_len_3([[1, 2], [2, 3], [4, 5], "ab"])=false'"
340 ECHO: "[ INFO ] run_test(); t17 passed: 'all_len_3([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=true'"
341 ECHO: "[ INFO ] run_test(); t18 passed: 'all_len_3([1, 2, 3, undef])=false'"
342 ECHO: "[ INFO ] run_test(); t19 passed: 'all_len_3([undef, undef, undef, undef])=false'"
343 ECHO: "[ INFO ] run_test(); t20 passed: 'all_len_3([[undef], [undef], [undef]])=false'"
344 ECHO: "[ INFO ] run_test(); t21 passed: 'all_len_3([true, true, true, true, false])=false'"
345 ECHO: "[ INFO ] run_test(); t22 passed: 'all_len_3([true, false, false, false, false)]=false'"
346 ECHO: "[ INFO ] run_test(); t23 passed: 'all_len_3([true, true, true, true])=false'"
```

3.1.1.3 Lists

Script

· Results

3.1.1.3.1 Script

```
include <datatypes.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
                         // show passing tests
show_passing = true;
show_skipped = true;
                          // show skipped tests
echo( str("OpenSCAD Version ", version()) );
// test-values columns
test c =
  ["id", "identifier"],
  ["td", "description"],
["tv", "test value"]
// test-values rows
test r =
  ["t01", "The undefined value",
["t02", "An odd integer",
["t03", "The boolean true",
                                            undef],
                                            1],
                                             truel
  ["t04", "The boolean false", ["t05", "A character string",
                                             falsel,
                                             "a"],
  ["t06", "A string",
["t07", "The empty string",
["t08", "The empty list",
["t09", "A shorthand range",
                                            "This is a longer string"],
                                             empty_str],
                                             empty_lst],
                                            10:911,
  ["t10", "A range",
["t11", "Test list 01",
["t12", "Test list 02",
["t13", "Test list 03",
                                             [0:0.5:911,
                                             [undef]],
                                             [1]],
                                             [1, 2, 3]],
  ["t14", "Test list 04",
["t15", "Test list 05",
["t16", "Test list 05",
["t16", "Test list 06",
["t17", "Test list 07",
                                            [[1], [2], [3], [4], [5]]],
                                            [[1,2], [2,3]]],
[[1,2], [2,3], [4,5], "ab"]],
[[1,2,3], [4,5,6], [7,8,9], ["a", "b", "c"]]],
  ["t18", "Test list 08",
["t19", "Test list 09",
["t20", "Test list 10",
["t21", "Test list 11",
                                            [1, 2, 3, undef]],
                                            [undef, undef, undef, undef]],
                                            [[undef], [undef], [undef]]],
                                             [true, true, true, true, false]],
  ["t22", "Test list 12", ["t23", "Test list 13",
                                            [true, false, false, false, false]],
                                            [true, true, true, true]]
1;
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 's' to skip test
t = true; // shortcuts
f = false;
u = undef;
s = -1;
            // skip test
good_r =
[ // function
                         01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
  ["almost_equal_T",
  ["almost_equal_F",
  ["almost_equal_U",
  ["compare_AA",
// sanity-test tables
table_check( test_r, test_c, false );
table_check( good_r, good_c, false );
// validate helper function and module
function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
module run_test( fname, fresult, vid )
  value text = get table v(test r, test c, vid, "td");
```

```
pass_value = get_table_v(good_r, good_c, fname, vid);
  test_pass = validate( cv=fresult, t="equals", ev=pass_value, pf=true );
  test_text = validate( str(fname, "(", get_value(vid), ")=", pass_value), fresult, "equals",
  pass value );
  if ( pass_value != s )
    if ( !test_pass )
      log_warn( str(vid, "(", value_text, ") ", test_text) );
    else if ( show_passing )
  log_info( str(vid, " ", test_text) );
  else if ( show_skipped )
    log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
// Indirect function calls would be very useful here!!!
for (vid=test_ids) run_test( "n_almost_equal_AA", n_almost_equal(get_value(vid),get_value
 (vid)), vid );
for (vid=test ids) run test( "almost equal AA", almost equal (get value(vid), get value(vid))
 , vid );
for (vid=test_ids) run_test( "almost_equal_T", almost_equal(get_value(vid),t), vid );
for (vid=test_ids) run_test( "almost_equal_F", almost_equal(get_value(vid),f), vid);
for (vid=test_ids) run_test( "almost_equal_U", almost_equal(get_value(vid),u), vid);
for (vid=test_ids) run_test( "compare_AA", compare(get_value(vid), get_value(vid)) == 0, vid );
// end-of-tests
```

3.1.1.3.2 Results

```
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'n_almost_equal_AA(undef)=false'"
3 ECHO: "[ INFO ] run_test(); t02 passed: 'n_almost_equal_AA(1)=true'"
4 ECHO: "[ INFO ] run_test(); t03 passed: 'n_almost_equal_AA(true)=false'"
5 ECHO: "[ INFO ] run_test(); t04 passed: 'n_almost_equal_AA(false)=false'"
6 ECHO: "[ INFO ] run_test(); t05 passed: 'n_almost_equal_AA(a)=false'
7 ECHO: "[ INFO ] run_test(); t06 passed: 'n_almost_equal_AA(This is a longer string)=false'"
8 ECHO: "[ INFO ] run_test(); t07 passed: 'n_almost_equal_AA()=false'
9 ECHO: "[ INFO ] run_test(); t08 passed: 'n_almost_equal_AA([])=false'"
10 ECHO: "[ INFO ] run_test(); t09 passed: 'n_almost_equal_AA([0 : 1 : 9])=false'"
11 ECHO: "[ INFO ] run_test(); t10 passed: 'n_almost_equal_AA([0 : 0.5 : 9])=false'"
12 ECHO: "[ INFO ] run_test(); t11 passed: 'n_almost_equal_AA([undef]) = false'"
13 ECHO: "[ INFO ] run_test(); t12 passed: 'n_almost_equal_AA([1])=true'"
14 ECHO: "[ INFO ] run_test(); t13 passed: 'n_almost_equal_AA([1, 2, 3]) = true'"
15 ECHO: "[ INFO ] run_test(); t14 passed: 'n_almost_equal_AA([[1], [2], [3], [4], [5]])=false'"
16 ECHO: "[ INFO ] run_test(); t15 passed: 'n_almost_equal_AA([[1, 2], [2, 3]])=false'"
17 ECHO: "[ INFO ] run_test(); t16 passed: 'n_almost_equal_AA([[1, 2], [2, 3], [4, 5], "ab"])=false'"
18 ECHO: "[ INFO ] run_test(); t17 passed: 'n_almost_equal_AA([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]])=false'"
19 ECHO: "[ INFO ] run_test(); t18 passed: 'n_almost_equal_AA([1, 2, 3, undef])=false'"
20 ECHO: "[ INFO ] run_test(); t19 passed: 'n_almost_equal_AA([undef, undef, undef, undef]) = false'"
21 ECHO: "[ INFO ] run_test(); t20 passed: 'n_almost_equal_AA([[undef], [undef], [undef]))=false'"
22 ECHO: "[ INFO ] run_test(); t21 passed: 'n_almost_equal_AA([true, true, true, true, false])=false'"
23 ECHO: "[ INFO ] run_test(); t22 passed: 'n_almost_equal_AA([true, false, false, false, false])=false'"
24 ECHO: "[ INFO ] run_test(); t23 passed: 'n_almost_equal_AA([true, true, true, true])=false'"
25 ECHO: "[ INFO ] run_test(); t01 passed: 'almost_equal_AA(undef)=true'"
26 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_AA(1)=true'"
27 ECHO: "[ INFO ] run_test(); t03 passed: 'almost_equal_AA(true)=true'"
28 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_AA(false)=true'"
29 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_AA(a)=true'"
30 ECHO: "[ INFO ] run_test(); t06 passed: 'almost_equal_AA(This is a longer string)=true'"
31 ECHO: "[ INFO ] run_test(); t07 passed: 'almost_equal_AA()=true'"
32 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_AA([])=true'"
33 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_AA([0 : 1 : 9])=true'"
34 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_AA([0 : 0.5 : 9]) = true'"
35 ECHO: "[ INFO ] run_test(); t11 passed: 'almost_equal_AA([undef])=true'"
36 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_AA([1])=true'"
37 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_AA([1, 2, 3])=true'"
38 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_AA([[1], [2], [3], [4], [5]])=true'"
39 ECHO: "[ INFO ] run_test(); t15 passed: 'almost_equal_AA([[1, 2], [2, 3])]=true'"
40 ECHO: "[ INFO ] run_test(); t16 passed: 'almost_equal_AA([[1, 2], [2, 3], [4, 5], "ab"])=true'"
41 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_AA([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]]) = true' "
42 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_AA([1, 2, 3, undef])=true'"
43 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_AA([undef, undef, undef])=true'"
44 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_AA([[undef], [undef]])=true'"
45 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_AA([true, true, true, true, false])=true'"
46 ECHO: "[ INFO ] run_test(); t22 passed: 'almost_equal_AA([true, false, false, false, false])=true'"
```

```
47 ECHO: "[ INFO ] run_test(); t23 passed: 'almost_equal_AA([true, true, true, true])=true'"
48 ECHO: "[ INFO ] run_test(); t01 passed: 'almost_equal_T(undef)=false'
49 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_T(1)=false'"
50 ECHO: "[ INFO ] run_test(); t03 passed: 'almost_equal_T(true) = true'"
             INFO ] run_test(); t04 passed: 'almost_equal_T(false)=false'"
51 ECHO: "[
52 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_T(a)=false'"
53 ECHO: "[
             INFO ] run_test(); t06 passed: 'almost_equal_T(This is a longer string)=false'"
54 ECHO: "[ INFO ] run_test(); t07 passed: 'almost_equal_T()=false'"
55 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_T([])=false'"
56 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_T([0 : 1 : 9])=false'"
57 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_T([0 : 0.5 : 9]) = false'"
58 ECHO: "[ INFO ] run_test(); t11 passed: 'almost_equal_T([undef])=false'"
59 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_T([1])=false''
60 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_T([1, 2, 3])=false'"
61 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_T([[1], [2], [3], [4], [5]])=false'"
62 ECHO: "[ INFO ] run_test(); t15 passed: 'almost_equal_T([[1, 2], [2, 3]])=false'"
63 ECHO: "[ INFO ] run_test(); t16 passed: 'almost_equal_T([[1, 2], [2, 3], [4, 5], "ab"])=false'"
64 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_T([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
65 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_T([1, 2, 3, undef])=false'"
66 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_T([undef, undef, undef])=false' 67 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_T([[undef], [undef], [undef]))=false'"
68 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_T([true, true, true, true, false])=false'"
69 ECHO: "[ INFO ] run_test(); t22 passed: 'almost_equal_T([true, false, false, false, false])=false'"
70 ECHO: "[ INFO ] run_test(); t23 passed: 'almost_equal_T([true, true, true, true])=false' 71 ECHO: "[ INFO ] run_test(); t01 passed: 'almost_equal_F(undef)=false'"
72 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_F(1)=false'
73 ECHO: "[ INFO ] run_test(); t03 passed: 'almost_equal_F(true)=false'"
74 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_F(false)=true'
75 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_F(a)=false'
76 ECHO: "[ INFO ] run_test(); t06 passed: 'almost_equal_F(This is a longer string)=false'"
77 ECHO: "[ INFO ] run_test(); t07 passed: 'almost_equal_F()=false'"
78 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_F([])=false'"
79 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_F([0 : 1 : 9])=false'"
80 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_F([0 : 0.5 : 9])=false'"
81 ECHO: "[ INFO ] run_test(); t11 passed: 'almost_equal_F([undef])=false'"
82 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_F([1])=false''
83 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_F([1, 2, 3])=false'"
84 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_F([[1], [2], [3], [4], [5]])=false'"
85 ECHO: "[ INFO ] run_test(); t15 passed: 'almost_equal_F([[1, 2], [2, 3]])=false'"
86 ECHO: "[ INFO ] run_test(); t16 passed: 'almost_equal_F([[1, 2], [2, 3], [4, 5], "ab"])=false'"
87 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_F([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
88 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_F([1, 2, 3, undef])=false'"
90 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_F([undef, undef, undef, undef])=false'"
90 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_F([[undef], [undef], [undef]])=false'"
91 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_F([true, true, true, true, false])=false'"
92 ECHO: "[ INFO ] run_test(); t22 passed: 'almost_equal_F([true, false, false, false, false])=false'"
93 ECHO: "[ INFO ] run_test(); t23 passed: 'almost_equal_F([true, true, true])=false'"
94 ECHO: "[ INFO ] run_test(); t01 passed: 'almost_equal_U(undef)=true'"
95 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_U(1)=false'"
96 ECHO: "[ INFO ] run_test(); t03 passed: 'almost_equal_U(true)=false'"
97 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_U(false)=false'"
98 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_U(a)=false'"
99 ECHO: "[ INFO ] run_test(); t06 passed: 'almost_equal_U(This is a longer string)=false'"
100 ECHO: "[ INFO ] run_test(); t07 passed: 'almost_equal_U()=false'"
101 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_U([])=false'"
102 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_U([0 : 1 : 9]) = false'"
103 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_U([0 : 0.5 : 9]) = false'"
104 ECHO: "[ INFO ] run_test(); tl1 passed: 'almost_equal_U([undef])=false'"
105 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_U([1])=false'"
106 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_U([1, 2, 3])=false'"
107 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_U([[1], [2], [3], [4], [5]])=false'"
108 ECHO: "[ INFO ] run_test(); t15 passed: 'almost_equal_U(([1, 2], [2, 3]))=false'"
109 ECHO: "[ INFO ] run_test(); t16 passed: 'almost_equal_U(([1, 2], [2, 3], [4, 5], "ab"])=false'"
110 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_U([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
111 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_U([1, 2, 3, undef])=false'
112 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_U([undef, undef, undef, undef]) = false'"
113 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_U([[undef], [undef], [undef]])=false'"
114 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_U([true, true, true, true, false])=false'"
115 ECHO: "[ INFO ] run_test(); t22 passed: 'almost_equal_U([true, false, false, false, false])=false'"
116 ECHO: "[ INFO ] run_test(); t23 passed: 'almost_equal_U([true, true, true, true])=false'"
117 ECHO: "[ INFO ]
                      run_test(); t01 passed: 'compare_AA(undef)=true'"
118 ECHO: "[ INFO ] run_test(); t02 passed: 'compare_AA(1)=true'"
119 ECHO: "[ INFO ] run_test(); t03 passed: 'compare_AA(true)=true'
120 ECHO: "[ INFO ] run_test(); t04 passed: 'compare_AA(false)=true'"
121 ECHO: "[ INFO ] run_test(); t05 passed: 'compare_AA(a)=true'
122 ECHO: "[ INFO ] run_test(); t06 passed: 'compare_AA(This is a longer string)=true'"
123 ECHO: "[ INFO ] run_test(); t07 passed: 'compare_AA()=true'"
124 ECHO: "[ INFO ] run_test(); t08 passed: 'compare_AA([])=true'"
```

```
125 ECHO: "[ INFO ] run_test(); t09 passed: 'compare_AA([0 : 1 : 9])=true'"
126 ECHO: "[ INFO ] run_test(); t10 passed: 'compare_AA([0 : 0.5 : 9])=true'"
127 ECHO: "[ INFO ] run_test(); t11 passed: 'compare_AA([undef])=true'"
128 ECHO: "[ INFO ] run_test(); t12 passed: 'compare_AA([undef])=true'"
129 ECHO: "[ INFO ] run_test(); t13 passed: 'compare_AA([1])=true'"
130 ECHO: "[ INFO ] run_test(); t14 passed: 'compare_AA([1], [2], [3], [4], [5]])=true'"
131 ECHO: "[ INFO ] run_test(); t15 passed: 'compare_AA([[1], [2], [2, 3]])=true'"
132 ECHO: "[ INFO ] run_test(); t16 passed: 'compare_AA([[1, 2], [2, 3], [4, 5], "ab"])=true'"
133 ECHO: "[ INFO ] run_test(); t17 passed: 'compare_AA([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=true'"
134 ECHO: "[ INFO ] run_test(); t18 passed: 'compare_AA([1, 2, 3, undef])=true'"
135 ECHO: "[ INFO ] run_test(); t19 passed: 'compare_AA([undef, undef, undef, undef])=true'"
136 ECHO: "[ INFO ] run_test(); t20 passed: 'compare_AA([undef], [undef], [undef]])=true'"
138 ECHO: "[ INFO ] run_test(); t21 passed: 'compare_AA([true, true, true, false])=true'"
138 ECHO: "[ INFO ] run_test(); t22 passed: 'compare_AA([true, true, true, false])=true'"
139 ECHO: "[ INFO ] run_test(); t23 passed: 'compare_AA([true, true, true, true])=true'"
```

3.1.2 Operations

- Scalar
- Iterables
- Lists

3.1.2.1 Scalar

- Script
- Results

3.1.2.1.1 Script

```
include <datatypes.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
                         // show passing tests
show_passing = true;
show_skipped = true;
                          // show skipped tests
echo( str("OpenSCAD Version ", version()) );
// test-values columns
test_c =
  ["id", "identifier"],
  ["td", "description"],
["tv", "test value"]
// test-values rows
test_r =
  ["t01", "The undefined value",
                                              undef],
  ["t02", "The empty list",
                                              empty_lst],
  ["t03", "A range",
["t04", "A string",
["t05", "Test list 01",
                                               [0:0.5:9]],
                                              "A string"],
                                              ["orange", "apple", "grape", "banana"]],
["b", "a", "n", "a", "n", "a", "s"]],
  ["t06", "Test list 02", ["t07", "Test list 03",
                                              [undef]],
  ["t08", "Test list 04", ["t09", "Test list 05",
                                               [[1,2],[2,3]]],
                                              ["ab",[1,2],[2,3],[4,5]]],
  ["t10", "Test list 06",
                                               [[1,2,3],[4,5,6],[7,8,9],["a","b","c"]]],
  ["t11", "Vector of integers 0 to 15", [for (i=[0:15]) i]]
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 's' to skip test
```

```
skip = -1; // skip test
     good_r =
     [ // function
        ["defined_or_D",
          "default",
                                                                             // t01
          empty_lst,
                                                                             // t02
          [0:0.5:9],
                                                                             // t03
          "A string",
                                                                             // 04
          ["orange", "apple", "grape", "banana"],
["b", "a", "n", "a", "n", "a", "s"],
                                                                             // t05
                                                                             // t06
                                                                             // t07
          [undef],
          [[1,2],[2,3]],
                                                                             // t08
          ["ab",[1,2],[2,3],[4,5]],
[[1,2,3],[4,5,6],[7,8,9],["a","b","c"]],
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                             // t09
                                                                            // t10
                                                                            // t11
       1
     ];
     // sanity-test tables
     table_check( test_r, test_c, false );
     table_check( good_r, good_c, false );
     // validate helper function and module
     function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
     module run_test( fname, fresult, vid )
       value_text = get_table_v(test_r, test_c, vid, "td");
       pass_value = get_table_v(good_r, good_c, fname, vid);
       test_pass = validate( cv=fresult, t="equals", ev=pass_value, pf=true );
test_text = validate( str(fname, "(", get_value(vid), ")=", pass_value), fresult, "equals",
       pass_value );
        if ( pass_value != skip )
          if ( !test_pass )
            log_warn( str(vid, "(", value_text, ") ", test_text) );
          else if ( show_passing )
  log_info( str(vid, " ", test_text) );
       else if ( show_skipped )
  log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
     // Indirect function calls would be very useful here!!!
     for (vid=test_ids) run_test( "defined_or_D", defined_or(get_value(vid), "default"), vid );
     // circular_index() not tested
     // end-of-tests
3.1.2.1.2 Results
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'defined_or_D(undef)=default'"
3 ECHO: "[ INFO ] run_test(); t02 passed: 'defined_or_D([])=[]'"
4 ECHO: "[ INFO ] run_test(); t03 passed: 'defined_or_D([0: 0.5: 9])=[0: 0.5: 9]'"
5 ECHO: "[ INFO ] run_test(); t04 passed: 'defined_or_D(A string)=A string'"
6 ECHO: "[ INFO ] run_test(); t05 passed: 'defined_or_D(["orange", "apple", "grape", "banana"])=["orange",
         "apple", "grape", "banana"]'"
7 ECHO: "[ INFO ] run_test(); t06 passed: 'defined_or_D(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
         "a", "n", "a", "s"]''
8 ECHO: "[ INFO ] run_test(); t07 passed: 'defined_or_D([undef])=[undef]'"
9 ECHO: "[ INFO ] run_test(); t08 passed: 'defined_or_D([[1, 2], [2, 3])]=[[1, 2], [2, 3]]'"

10 ECHO: "[ INFO ] run_test(); t09 passed: 'defined_or_D(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3])
3], [4, 5]]'"

11 ECHO: "[ INFO ] run_test(); t10 passed: 'defined_or_D([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
"c"]]]=[[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]'"

12 ECHO: "[ INFO ] run_test(); t11 passed: 'defined_or_D([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
         15])=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]''
```

3.1.2.2 Iterables

- Script
- Results

3.1.2.2.1 Script

```
include <datatypes.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
echo( str("OpenSCAD Version ", version()) );
// test-values columns
test_c =
  ["id", "identifier"],
["td", "description"],
   ["tv", "test value"]
// test-values rows
test_r =
  ["t01", "The undefined value",
["t02", "The empty list",
                                                   undef],
                                                   empty_lst],
  ["t02", "The empty list"
["t03", "A range",
["t04", "A string",
["t05", "Test list 01",
["t06", "Test list 02",
["t07", "Test list 03",
["t08", "Test list 04",
["t09", "Test list 05",
["t10", "Test list 06",
["t11", "Vector of integrals
                                                   [0:0.5:9]],
                                                   "A string"],
                                                  ["orange", "apple", "grape", "banana"]],
["b", "a", "n", "a", "n", "a", "s"]],
                                                   [undef]],
                                                  [[1,2],[2,3]]],
["ab",[1,2],[2,3],[4,5]]],
[[1,2,3],[4,5,6],[7,8,9],["a","b","c"]]],
   ["t11", "Vector of integers 0 to 15", [for (i=[0:15]) i]]
1;
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 's' to skip test
skip = -1; // skip test
good_r =
[ // function
   ["edefined_or_DE3",
     "default",
                                                                        // t01
     "default",
                                                                        // t02
     "default",
                                                                        // t03
                                                                        // t04
     "t",
     "banana",
                                                                        // t05
// t06
     "a",
     "default",
                                                                        // t07
     "default",
                                                                        // t08
     [4,5],
["a","b","c"],
                                                                        // t09
                                                                        // t10
     3
                                                                        // t11
   ],
["find_12",
     empty_lst,
                                                                        // t01
     empty_lst,
                                                                        // t02
     empty_lst,
                                                                        // t03
     empty_lst,
                                                                        // t04
     empty_lst,
                                                                        // t05
     empty_lst,
                                                                        // t06
     empty_lst,
                                                                        // t07
     [0],
                                                                        // t08
                                                                        // t09
     [1],
     empty_lst,
                                                                        // t10
                                                                        // t11
     empty_lst
   ["count_S1",
     Ο,
                                                                        // t01
     Ο,
                                                                        // t02
                                                                        // t03
     0,
                                                                        // t04
     0,
                                                                        // t05
     Ο,
                                                                        // t06
     0.
                                                                        // t07
     0,
                                                                        // t08
     1.
```

```
// t09
  1,
                                                                         // t10
                                                                         // t11
  1
["exists_S1",
                                                                         // t01
// t02
  false,
  false,
  false,
                                                                         // t03
                                                                        // t04
// t05
// t06
  false,
  false,
  false,
                                                                         // t07
  false,
  true,
                                                                         // t08
                                                                         // t09
// t10
  true,
  true,
                                                                         // t11
  true
],
["first",
  undef,
                                                                         // t01
                                                                         // t02
  undef,
                                                                         // t03
  0,
"A",
                                                                         // t04
                                                                        // t05
// t06
  "orange",
   "b",
  undef,
                                                                         // t07
  [1,2],
                                                                         // t08
   "ab",
                                                                        // t09
// t10
  [1,2,3],
                                                                         // t11
  0
],
["second",
  undef,
                                                                         // t01
                                                                        // t02
// t03
  undef,
  0.5,
                                                                        // t04
// t05
  "apple",
                                                                        // t05
// t06
// t07
// t08
// t09
  undef.
   [2,3],
  [1,2],
                                                                         // t10
// t11
  [4,5,6],
  1
],
["third",
                                                                        // t01
// t02
// t03
// t04
  undef,
  undef,
  9,
"s",
                                                                        // t05
// t06
// t07
// t08
  "grape",
  "n",
  undef,
  undef,
                                                                         // t09
// t10
// t11
   [2,3],
   [7,8,9],
  2
["last",
                                                                         // t01
  undef,
                                                                         // t02
  undef,
  undef,
                                                                         // t03
                                                                         // t04
   "g",
                                                                        // t05
// t06
  "banana",
  "s",
  undef,
                                                                         // t07
  [2,3],
                                                                         // t08
  [4,5],
["a","b","c"],
                                                                         // t09
                                                                         // t10
  15
                                                                         // t11
],
["nfirst_1",
  undef,
                                                                         // t01
                                                                        // t02
// t03
  undef,
  undef,
                                                                        // t04
// t05
  ["A"],
   ["orange"],
   ["b"],
                                                                         // t06
                                                                         // t07
   [undef],
  [[1,2]],
["ab"],
                                                                         // t08
                                                                         // t09
                                                                         // t10
// t11
   [[1,2,3]],
   F 0 1
```

```
],
["nlast_1",
                                                                             // t01
   undef,
   undef,
                                                                             // t02
   undef,
                                                                             // t03
   ["g"],
                                                                             // t04
   ["banana"],
                                                                             // t05
   ["s"],
                                                                             // t06
   [undef],
                                                                             // t07
   [[2,3]],
                                                                             // t08
                                                                             // t09
   [[4,5]],
   [["a","b","c"]],
                                                                             // t10
   [15]
                                                                             // t11
["nhead_1",
  undef,
                                                                             // t01
                                                                             // t02
   undef,
  undef,
["A"," ","s","t","r","i","n"],
["orange","apple","grape"],
["b","a","n","a","n","a"],
                                                                             // t03
                                                                             // t04
                                                                             // t05
                                                                             // t06
   empty_lst.
                                                                             // t07
                                                                             // t08
   [[1,2]],
   ["ab",[1,2],[2,3]],
                                                                             // t09
   [[1,2,3],[4,5,6],[7,8,9]],
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]
                                                                             // t10
                                                                             // t11
["ntail_1",
  undef,
                                                                             // t01
                                                                             // t02
   undef.
   undef,
[" ","s","t","r","i","n","g"],
                                                                             // t03
                                                                             // t04
   ["apple", "grape", "banana"], ["a", "n", "a", "n", "a", "s"],
                                                                             // t05
// t06
   empty_lst,
                                                                             // t07
                                                                             // t08
   [[2,3]],
   [[1,2],[2,3],[4,5]],
[[4,5,6],[7,8,9],["a","b","c"]],
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                             // t09
                                                                             // t10
                                                                             // t11
],
["reverse",
                                                                             // t01
  undef,
                                                                             // t01
// t02
// t03
   empty_lst,
   undef,
   ["g","n","i","r","t","s"," ","A"],
["banana","grape","apple","orange"],
["s","a","n","a","n","a","b"],
                                                                            // t04
// t05
// t06
// t07
   [undef],
                                                                             // t08
   [[2,3],[1,2]],
   [[4,5],[2,3],[1,2],"ab"],
                                                                             // t09
                                                                            // t10
// t11
   [["a", "b", "c"], [7,8,9], [4,5,6], [1,2,3]],
   [15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
],
["rselect_02",
   undef,
                                                                             // t01
   empty_lst,
                                                                             // t02
  empty_ist,
undef,
["A"," ","s"],
["orange","apple","grape"],
["b","a","n"],
                                                                             // t03
                                                                             // t04
                                                                             // t05
                                                                             // t06
   undef,
                                                                             // t07
   undef,
                                                                             // t08
   ["ab",[1,2],[2,3]],
                                                                             // t09
   [[1,2,3],[4,5,6],[7,8,9]],
                                                                             // t10
   [0,1,2]
                                                                             // t11
],
["nssequence_31",
  empty_lst,
                                                                             // t01
                                                                             // t02
   empty_lst,
                                                                             // t03
   empty_lst,
     ["A"," ","s"],[" ","s","t"],["s","t","r"],
["t","r","i"],["r","i","n"],["i","n","g"]
                                                                             // t04
     ["orange", "apple", "grape"],
["apple", "grape", "banana"]
                                                                             // t05
     ["b", "a", "n"], ["a", "n", "a"], ["n", "a", "n"], ["a", "n", "a"], ["n", "a", "s"]
```

```
// t06
  empty_lst,
                                                                        // t07
                                                                        // t08
  empty_lst,
  [["ab",[1,2],[2,3]],[[1,2],[2,3],[4,5]]],
                                                                        // t09
     [[1,2,3],[4,5,6],[7,8,9]],
[[4,5,6],[7,8,9],["a","b","c"]]
                                                                        // t10
     [0,1,2],[1,2,3],[2,3,4],[3,4,5],[4,5,6],[5,6,7],
[6,7,8],[7,8,9],[8,9,10],[9,10,11],[10,11,12],
     [11,12,13],[12,13,14],[13,14,15]
                                                                        // t11
  1
["eappend_T0",
  undef,
                                                                        // t01
  [[0]],
                                                                        // t02
                                                                        // t03
  undef,
     ["A",0],[" ",0],["s",0],["t",0],
     ["r",0],["i",0],["n",0],["g",0]
                                                                        // t04
     ["orange", 0], ["apple", 0],
     ["grape", 0], ["banana", 0]
                                                                        // t05
     ["b",0],["a",0],["n",0],["a",0],
     ["n",0],["a",0],["s",0]
                                                                        // t06
  [[undef, 0]],
                                                                        // t07
  [[1,2,0],[2,3,0]],
                                                                        // t08
  [["ab",0],[2,3,6],
[[1,2,3,0],[4,5,6,0],[7,8,9,0],["a","b","c",0]],
                                                                        // t09
                                                                        // t10
     [0,0],[1,0],[2,0],[3,0],[4,0],[5,0],
     [6,0],[7,0],[8,0],[9,0],[10,0],[11,0],
     [12,0],[13,0],[14,0],[15,0]
  ]
                                                                        // t11
["insert_T0",
                                                                        // t01
  undef,
                                                                        // t02
// t03
  undef.
  undef.
                                                                        // t04
  undef,
  ["orange",0,"apple","grape","banana"],
["b","a","n","a","n","a",0,"s"],
                                                                        // t05
                                                                        // t06
// t07
  undef,
  [[1,2],0,[2,3]],
                                                                        // t08
   ["ab",[1,2],0,[2,3],[4,5]],
                                                                        // t09
  undef,
                                                                        // t10
                                                                       // t11
  [0,1,2,3,4,0,5,6,7,8,9,10,11,12,13,14,15]
["delete_T0",
  undef,
                                                                        // t01
  empty_lst,
                                                                        // t02
  empty_los,
undef,
["A"," ","s","t","r","i","n","g"],
["orange","grape","banana"],
["b","a","n","a","n","a"],
                                                                        // t03
                                                                        // t04
                                                                        // t05
                                                                        // t06
                                                                        // t07
   [[1,2]],
                                                                        // t08
  ["ab",[1,2],[4,5]],
                                                                        // t09
  [[1,2,3],[4,5,6],[7,8,9],["a","b","c"]],
[0,1,2,3,4,6,7,8,9,10,11,12,13,14,15]
                                                                        // t10
                                                                        // t11
],
["strip",
  undef,
                                                                        // t01
                                                                        // t02
  empty_lst,
  undef,
undef,
["A"," ","s","t","r","i","n","g"],
["orange", "apple", "grape", "banana"],
["b","a","n","a","n","a","s"],
                                                                        // t03
                                                                        // t04
                                                                        // t05
                                                                        // t06
                                                                        // t07
  [undef],
  [[1,2],[2,3]],
                                                                        // t08
  ["ab",[1,2],[2,3],[4,5]],
[[1,2,3],[4,5,6],[7,8,9],["a","b","c"]],
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                        // t09
                                                                        // t10
                                                                        // t11
["unique",
                                                                        // t01
  undef.
```

```
// t02
           empty_lst,
                                                                                     // t03
            -----,
["A"," ","s","t","r","i","n","g"],
["orange","apple","grape","banana"],
                                                                                     // t04
                                                                                     // t05
            ["b", "a", "n", "s"],
                                                                                     // t06
            [undef],
                                                                                     // t07
            [[1,2],[2,3]],
                                                                                     // t08
            ["ab",[1,2],[2,3],[4,5]],
                                                                                     // t09
           [[1,2,3],[4,5,6],[7,8,9],["a","b","c"]],
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                                     // t10
     1;
      // sanity-test tables
      table_check( test_r, test_c, false );
      table_check( good_r, good_c, false );
      // validate helper function and module
      function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
     module run_test( fname, fresult, vid )
        value_text = get_table_v(test_r, test_c, vid, "td");
        pass_value = get_table_v(good_r, good_c, fname, vid);
        test_pass = validate( cv=fresult, t="equals", ev=pass_value, pf=true );
test_text = validate( str(fname, "(", get_value(vid), ")=", pass_value), fresult, "equals",
        pass value );
        if ( pass value != skip )
           if ( !test_pass )
              log_warn( str(vid, "(", value_text, ") ", test_text) );
           else if ( show_passing )
  log_info( str(vid, " ", test_text) );
        else if ( show_skipped )
           log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
      // Indirect function calls would be very useful here!!!
      for (vid=test_ids) run_test( "edefined_or_DE3", edefined_or(get_value(vid), 3, "default"), vid
      for (vid=test_ids) run_test( "find_12", find([1,2],get_value(vid)), vid );
for (vid=test_ids) run_test( "count_S1", count(1,get_value(vid),true), vid );
for (vid=test_ids) run_test( "exists_S1", exists(1,get_value(vid),true), vid );
      for (vid=test_ids) run_test( "first", first(get_value(vid)), vid);
for (vid=test_ids) run_test( "second", second(get_value(vid)), vid);
for (vid=test_ids) run_test( "third", third(get_value(vid)), vid);
      for (vid=test_ids) run_test( "last", last(get_value(vid)), vid );
      for (vid=test_ids) run_test( "nfirst_1", nfirst(get_value(vid),n=1), vid );
      for (vid=test_ids) run_test( "nlast_1", nlast(get_value(vid),n=1), vid );
for (vid=test_ids) run_test( "nhead_1", nhead(get_value(vid),n=1), vid );
      for (vid=test_ids) run_test( "ntail_1", ntail(get_value(vid), n=1), vid );
for (vid=test_ids) run_test( "reverse", reverse(get_value(vid)), vid );
      for (vid=test_ids) run_test( "rselect_02", rselect(get_value(vid),i=[0:2]), vid );
      for (vid=test_ids) run_test( "nssequence_31", nssequence(get_value(vid),n=3,s=1), vid );
      for (vid=test_ids) run_test( "eappend_T0", eappend(0,get_value(vid)), vid );
      for (vid=test_ids) run_test( "insert_T0", insert(0,get_value(vid),mv=["x","r","apple","s",[2,3],5
       ]), vid );
      for (vid=test ids) run test( "delete TO", delete(get value(vid),mv=["x","r","apple","s",[2,3],5]), vid
       );
     for (vid=test_ids) run_test( "strip", strip(get_value(vid)), vid );
for (vid=test_ids) run_test( "unique", unique(get_value(vid)), vid );
      // end-of-tests
3.1.2.2.2 Results
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'edefined_or_DE3(undef) = default'"
3 ECHO: "[ INFO ] run_test(); t02 passed: 'edefined_or_DE3([])=default'"
4 ECHO: "[ INFO ] run_test(); t03 passed: 'edefined_or_DE3([0 : 0.5 : 9]) = default'"
5 ECHO: "[ INFO ]
                         run_test(); t04 passed: 'edefined_or_DE3(A string)=t'"
6 ECHO: "[ INFO ] run_test(); t05 passed: 'edefined_or_DE3(["orange", "apple", "grape", "banana"])=banana'"
7 ECHO: "[ INFO ] run_test(); t06 passed: 'edefined_or_DE3(["b", "a", "n", "a", "n", "a", "s"])=a'"
8 ECHO: "[ INFO ] run_test(); t07 passed: 'edefined_or_DE3([undef])=default'"
9 ECHO: "[ INFO ] run_test(); t08 passed: 'edefined_or_DE3([[1, 2], [2, 3]])=default'"

10 ECHO: "[ INFO ] run_test(); t09 passed: 'edefined_or_DE3([[ab", [1, 2], [2, 3], [4, 5]])=[4, 5]'"

11 ECHO: "[ INFO ] run_test(); t10 passed: 'edefined_or_DE3([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
```

```
"c"]])=["a", "b", "c"]'"
12 ECHO: "[ INFO ] run_test(); t11 passed: 'edefined_or_DE3([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          15])=3'"
13 ECHO: "[ INFO ] run_test(); t01 passed: 'find_12(undef)=[]'"
14 ECHO: "[ INFO ] run_test(); t02 passed: 'find_12([])=[]'"
15 ECHO: "[ INFO ] run_test(); t03 passed: 'find_12([0:0.5:9])=[]'"
16 ECHO: "[ INFO ] run_test(); t04 passed: 'find_12(["orange", "apple", "grape", "banana"])=[]'"

17 ECHO: "[ INFO ] run_test(); t05 passed: 'find_12(["orange", "apple", "grape", "banana"])=[]'"

18 ECHO: "[ INFO ] run_test(); t06 passed: 'find_12(["b", "a", "n", "a", "n", "a", "s"])=[]'"
19 ECHO: "[ INFO ] run_test(); t07 passed: 'find_12([undef])=[]'"
20 ECHO: "[ INFO ] run_test(); t08 passed: 'find_12([[1, 2], [2, 3]])=[0]'"
21 ECHO: "[ INFO ] run_test(); t09 passed: 'find_12(["ab", [1, 2], [2, 3], [4, 5]])=[1]'"
22 ECHO: "[ INFO ] run_test(); t10 passed: 'find_12([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[]'"
23 ECHO: "[ INFO ] run_test(); t11 passed: 'find_12([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          15])=[]'"
24 ECHO: "[ INFO ] run_test(); t01 passed: 'count_S1(undef)=0'"
25 ECHO: "[ INFO ] run_test(); t02 passed: 'count_S1([])=0'"
26 ECHO: "[ INFO ] run_test(); t03 passed: 'count_S1([0 : 0.5 : 9])=0'"
27 ECHO: "[ INFO ] run_test(); t04 passed: 'count_S1(A string)=0'
28 ECHO: "[ INFO ] run_test(); t05 passed: 'count_S1(["orange", "apple", "grape", "banana"])=0'"
29 ECHO: "[ INFO ] run_test(); t06 passed: 'count_S1(["b", "a", "n", "a", "n", "a", "s"])=0'"
30 ECHO: "[ INFO ] run_test(); t07 passed: 'count_S1([undef])=0'"
31 ECHO: "[ INFO ] run_test(); t08 passed: 'count_S1([[1, 2], [2, 3]])=1'"
32 ECHO: "[ INFO ] run_test(); t09 passed: 'count_S1(["ab", [1, 2], [2, 3], [4, 5]])=1'"
33 ECHO: "[ INFO ] run_test(); t10 passed: 'count_S1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=1'"
34 ECHO: "[ INFO ] run_test(); t11 passed: 'count_S1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
         15])=1'"
35 ECHO: "[ INFO ] run_test(); t01 passed: 'exists_S1(undef)=false'
36 ECHO: "[ INFO ] run_test(); t02 passed: 'exists_S1([])=false'"
37 ECHO: "[ INFO ] run_test(); t03 passed: 'exists_S1([0 : 0.5 : 9])=false'"
38 ECHO: "[ INFO ] run_test(); t04 passed: 'exists_S1(A string)=false'
38 ECHO: "[ INFO ] run_test(); t04 passed: exists_S1(["orange", "apple", "grape", "banana"])=false'"
40 ECHO: "[ INFO ] run_test(); t06 passed: 'exists_S1(["b", "a", "n", "a", "n", "a", "s"])=false'"
41 ECHO: "[ INFO ] run_test(); t07 passed: 'exists_S1([undef])=false'"
41 ECHO: [ INFO ] run_test(), to Passed: 'exists_S1([[1, 2], [2, 3]])=true'"
42 ECHO: "[ INFO ] run_test(); t08 passed: 'exists_S1([[ab", [1, 2], [2, 3], [4, 5]])=true'"
43 ECHO: "[ INFO ] run_test(); t09 passed: 'exists_S1(["ab", [1, 2], [2, 3], [4, 5]])=true'"
44 ECHO: "[ INFO ] run_test(); t10 passed: 'exists_S1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=true'"
45 ECHO: "[ INFO ] run_test(); t11 passed: 'exists_S1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]) = true'"
46 ECHO: "[ INFO ] run_test(); t01 passed: 'first(undef)=undef'"
47 ECHO: "[ INFO ] run_test(); t02 passed: 'first([])=undef'"
48 ECHO: "[ INFO ] run_test(); t03 passed: 'first([0 : 0.5 : 9])=0'"
49 ECHO: "[ INFO ] run_test(); t04 passed: 'first(A string)=A'"
50 ECHO: "[ INFO ] run_test(); t05 passed: 'first(["orange", "apple", "grape", "banana"])=orange'"
51 ECHO: "[ INFO ] run_test(); t06 passed: 'first(["b", "a", "n", "a", "n", "a", "s"])=b'"
52 ECHO: "[ INFO ] run_test(); t07 passed: 'first([undef])=undef'"
53 ECHO: "[ INFO ] run_test(); t08 passed: 'first([[1, 2], [2, 3]])=[1, 2]'"
54 ECHO: "[ INFO ] run_test(); t09 passed: 'first(["ab", [1, 2], [2, 3], [4, 5]])=ab'"
55 ECHO: "[ INFO ] run_test(); t10 passed: 'first([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1, 2,
         3],"
56 ECHO: "[ INFO ] run_test(); t11 passed: 'first([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=0'"
57 ECHO: "[ INFO ] run_test(); t01 passed: 'second(undef)=undef'"
58 ECHO: "[ INFO ] run_test(); t02 passed: 'second([])=undef'"
59 ECHO: "[ INFO ] run_test(); t03 passed: 'second([0 : 0.5 : 9])=0.5'"
60 ECHO: "[ INFO ] run_test(); t04 passed: 'second(A string) = '"
61 ECHO: "[ INFO ] run_test(); t05 passed: 'second(("orange", "apple", "grape", "banana"])=apple'"
62 ECHO: "[ INFO ] run_test(); t06 passed: 'second(("b", "a", "n", "a", "n", "a", "s"])=a'"
63 ECHO: "[ INFO ] run_test(); t07 passed: 'second([undef])=undef'"
64 ECHO: "[ INFO ] run_test(); t08 passed: 'second([[1, 2], [2, 3]])=[2, 3]'"
65 ECHO: "[ INFO ] run_test(); t09 passed: 'second(["ab", [1, 2], [2, 3], [4, 5]])=[1, 2]'"
66 ECHO: "[ INFO ] run_test(); t10 passed: 'second([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[4, 5,
         6]′"
67 ECHO: "[ INFO ] run_test(); t11 passed: 'second([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=1'" 68 ECHO: "[ INFO ] run_test(); t01 passed: 'third(undef)=undef'"
69 ECHO: "[ INFO ] run_test(); t02 passed: 'third([]) = undef'"
70 ECHO: "[ INFO ] run_test(); t03 passed: 'third([0:0.5:9])=9'"
71 ECHO: "[ INFO ] run_test(); t04 passed: 'third(A string) = s'"
72 ECHO: "[ INFO ] run_test(); t05 passed: 'third(["orange", "apple", "grape", "banana"])=grape'"
73 ECHO: "[ INFO ] run_test(); t06 passed: 'third(["b", "a", "n", "a", "n", "a", "s"])=n'"
74 ECHO: "[ INFO ] run_test(); t07 passed: 'third([undef])=undef'"
75 ECHO: "[ INFO ] run_test(); t08 passed: 'third([[1, 2], [2, 3]])=undef'"
76 ECHO: "[ INFO ] run_test(); t09 passed: 'third(["ab", [1, 2], [2, 3], [4, 5]])=[2, 3]'"
77 ECHO: "[ INFO ] run_test(); t10 passed: 'third([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[7, 8,
78 ECHO: "[ INFO ] run_test(); t11 passed: 'third([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=2'"
79 ECHO: "[ INFO ] run_test(); t01 passed: 'last(undef)=undef'"
80 ECHO: "[ INFO ] run_test(); t02 passed: 'last([])=undef'"
81 ECHO: "[ INFO ] run_test(); t03 passed: 'last([0:0.5:9])=undef'
82 ECHO: "[ INFO ] run_test(); t04 passed: 'last(A string)=g'"
83 ECHO: "[ INFO ] run_test(); t05 passed: 'last(["orange", "apple", "grape", "banana"])=banana'"
```

```
84 ECHO: "[ INFO ] run_test(); t06 passed: 'last(["b", "a", "n", "a", "n", "a", "s"])=s'"
85 ECHO: "[ INFO ] run_test(); t07 passed: 'last([undef])=undef'"
86 ECHO: "[ INFO ] run_test(); t08 passed: 'last([[1, 2], [2, 3]])=[2, 3]'"
87 ECHO: "[ INFO ] run_test(); t09 passed: 'last(["ab", [1, 2], [2, 3], [4, 5]])=[4, 5]'"
88 ECHO: "[INFO] run_test(); t10 passed: 'last([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=["a", "b", "c"]'"

89 ECHO: "[INFO] run_test(); t11 passed: 'last([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=15'"

90 ECHO: "[INFO] run_test(); t01 passed: 'nfirst_1(undef)=undef'"
91 ECHO: "[ INFO ] run_test(); t02 passed: 'nfirst_1([])=undef'"
92 ECHO: "[ INFO ] run_test(); t03 passed: 'nfirst_1([0 : 0.5 : 9]) = undef'"
93 ECHO: "[ INFO ] run_test(); t04 passed: 'nfirst_1(A string)=["A"]'"
94 ECHO: "[ INFO ] run_test(); t05 passed: 'nfirst_1(["orange", "apple", "grape", "banana"])=["orange"]'"
95 ECHO: "[ INFO ] run_test(); t06 passed: 'nfirst_1(["b", "a", "n", "a", "n", "a", "s"])=["b"]'"
96 ECHO: "[ INFO ] run_test(); t07 passed: 'nfirst_1([undef])=[undef]'"
97 ECHO: "[ INFO ] run_test(); t08 passed: 'nfirst_1([[1, 2], [2, 3]])=[[1, 2]]'"
98 ECHO: "[ INFO ] run_test(); t09 passed: 'nfirst_1(["ab", [1, 2], [2, 3], [4, 5]])=["ab"]'"
99 ECHO: "[ INFO ] run_test(); t10 passed: 'nfirst_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1,
2, 3]]'"

100 ECHO: "[ INFO ] run_test(); t11 passed: 'nfirst_1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          15])=[0]'"
101 ECHO: "[ INFO ] run_test(); t01 passed: 'nlast_1(undef) = undef'
102 ECHO: "[ INFO ] run_test(); t02 passed: 'nlast_1([]) = undef'"
103 ECHO: "[ INFO ] run_test(); t03 passed: 'nlast_1([0 : 0.5 : 9]) = undef'"
104 ECHO: "[ INFO ] run_test(); t04 passed: 'nlast_1(A string)=["q"]'"
105 ECHO: "[ INFO ] run_test(); t05 passed: 'nlast_1(("orange", "apple", "grape", "banana"])=["banana"]'"
106 ECHO: "[ INFO ] run_test(); t06 passed: 'nlast_1(("b", "a", "n", "a", "n", "a", "s"])=["s"]'"
107 ECHO: "[ INFO ] run_test(); t07 passed: 'nlast_1(([undef])=[undef]'"
108 ECHO: "[ INFO ] run_test(); t08 passed: 'nlast_1([[1, 2], [2, 3]])=[[2, 3]]'"
109 ECHO: "[INFO] run_test(); t09 passed: 'nlast_1([[1, 2], [2, 3]])=[[2, 3]])=[[4, 5]]'"
110 ECHO: "[INFO] run_test(); t10 passed: 'nlast_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[["a", "b", "c"]]'"
111 ECHO: "[INFO] run_test(); t11 passed: 'nlast_1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
           151)=[15]'"
112 ECHO: "[ INFO ] run_test(); t01 passed: 'nhead_1(undef)=undef'"
113 ECHO: "[ INFO ] run_test(); t02 passed: 'nhead_1([])=undef'
114 ECHO: "[ INFO ] run_test(); t03 passed: 'nhead_1([0 : 0.5 : 9])=undef'"
115 ECHO: "[ INFO ] run_test(); t04 passed: 'nhead_1(A string)=["A", " ", "s", "t", "r", "i", "n"]'"
116 ECHO: "[ INFO ] run_test(); t05 passed: 'nhead_1(["orange", "apple", "grape", "banana"])=["orange",
"apple", "grape")'"
117 ECHO: "[ INFO ] run_test(); t06 passed: 'nhead_1(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
           "n",
                   "a"]/"
"n", "a"]'"

118 ECHO: "[ INFO ] run_test(); t07 passed: 'nhead_1([undef])=[]'"

119 ECHO: "[ INFO ] run_test(); t08 passed: 'nhead_1([[1, 2], [2, 3]])=[[1, 2]]'"

120 ECHO: "[ INFO ] run_test(); t09 passed: 'nhead_1(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3]]'"

120 ECHO: "[ INFO ] run_test(); t09 passed: 'nhead_1(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3]]'"
121 ECHO: "[ INFO ] run_test(); t10 passed: 'nhead_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]'"
122 ECHO: "[ INFO ] run_test(); tll passed: 'nhead_1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]) = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]'"
123 ECHO: "[ INFO ] run_test(); t01 passed: 'ntail_1(undef)=undef'"
124 ECHO: "[ INFO ] run_test(); t02 passed: 'ntail_1([])=undef'"

125 ECHO: "[ INFO ] run_test(); t03 passed: 'ntail_1([])=undef'"

126 ECHO: "[ INFO ] run_test(); t04 passed: 'ntail_1(A string)=[" ", "s", "t", "r", "i", "n", "g"]'"

127 ECHO: "[ INFO ] run_test(); t05 passed: 'ntail_1(["orange", "apple", "grape", "banana"])=["apple", "grape",
           "banana"]'"
130 ECHO: "[INFO] run_test(); t08 passed: 'ntail_1([[1, 2], [2, 3]])=[[2, 3]]'"
131 ECHO: "[INFO] run_test(); t09 passed: 'ntail_1(["ab", [1, 2], [2, 3], [4, 5]])=[[1, 2], [2, 3], [4, 5]]'"
132 ECHO: "[INFO] run_test(); t10 passed: 'ntail_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[4,
           5, 6], [7, 8, 9], ["a", "b", "c"]]'"
133 ECHO: "[ INFO ] run_test(); t11 passed: 'ntail_1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"
134 ECHO: "[ INFO ] run_test(); t01 passed: 'reverse(undef)=undef'"
135 ECHO: "[ INFO ] run_test(); t02 passed: 'reverse([])=[]'"
136 ECHO: "[ INFO ] run_test(); t03 passed: 'reverse([0:0.5:9])=undef'"
137 ECHO: "[ INFO ] run_test(); t04 passed: 'reverse(A string)=["g", "n", "i", "r", "t", "s", " ", "A"]'"
138 ECHO: "[ INFO ] run_test(); t05 passed: 'reverse(["orange", "apple", "grape", "banana"])=["banana",
           "grape", "apple", "orange"]'"
139 ECHO: "[ INFO ] run_test(); t06 passed: 'reverse(["b", "a", "n", "a", "n", "a", "s"])=["s", "a", "n", "a",
           "n", "a", "b"]'"
140 ECHO: "[ INFO ] run_test(); t07 passed: 'reverse([undef])=[undef]'"
141 ECHO: "[ INFO ] run_test(); t08 passed: 'reverse([[1, 2], [2, 3]])=[[2, 3], [1, 2]]'"
142 ECHO: "[INFO] run_test(); t09 passed: 'reverse(["ab", [1, 2], [2, 3], [4, 5]])=[[4, 5], [2, 3], [1, 2],
            "ab"]'"
143 ECHO: "[ INFO ] run_test(); t10 passed: 'reverse([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[["a",
"b", "c"], [7, 8, 9], [4, 5, 6], [1, 2, 3]]'"

144 ECHO: "[ INFO ] run_test(); t11 passed: 'reverse([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]'"
145 ECHO: "[INFO] run_test(); t01 passed: 'rselect_02(undef)=undef'"
146 ECHO: "[INFO] run_test(); t02 passed: 'rselect_02([])=[]'"
```

3.1 Datatypes 31

```
147 ECHO: "[ INFO ] run_test(); t03 passed: 'rselect_02([0 : 0.5 : 9])=undef'"
148 ECHO: "[ INFO ] run_test(); t04 passed: 'rselect_02(A string)=["A", " ", "s"]'"
149 ECHO: "[ INFO ] run_test(); t05 passed: 'rselect_02(["orange", "apple", "grape", "banana"])=["orange",
            "apple", "grape"]'"
150 ECHO: "[ INFO ] run_test(); t06 passed: 'rselect_02(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n"]'"
151 ECHO: "[ INFO ] run_test(); t07 passed: 'rselect_02([undef])=undef'"
152 ECHO: "[ INFO ] run_test(); t08 passed: 'rselect_02([[1, 2], [2, 3]])=undef'"
153 ECHO: "[ INFO ] run_test(); t09 passed: 'rselect_02(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2,
            311/"
154 ECHO: "[ INFO ] run_test(); t10 passed: 'rselect_02([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
"c"]])=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]'"
155 ECHO: "[ INFO ] run_test(); t11 passed: 'rselect_02([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            15])=[0, 1, 2]'"
156 ECHO: "[ INFO ] run_test(); t01 passed: 'nssequence_31(undef)=[]'"
157 ECHO: "[ INFO ] run_test(); t02 passed: 'nssequence_31([])=[]'"
158 ECHO: "[ INFO ] run_test(); t03 passed: 'nssequence_31([0 : 0.5 : 9])=[]'"
159 ECHO: "[ INFO ] run_test(); t04 passed: 'nssequence_31(A string)=[["A", " ", "s"], [" ", "s", "t"], ["s", "t", "r"], ["t", "r", "i"], ["r", "i", "n"], ["i", "n", "g"]]'"
"t", "r"], ["t", "r", "i"], ["r", "i", "n"], ["i", "n"], ["i", "n"], "g"]]/"

160 ECHO: "[ INFO ] run_test(); t05 passed: 'nssequence_31(["orange", "apple", "grape", "banana"])=[["orange", "apple", "grape"], ["apple", "grape", "banana"]]'"
apple, glape ], [apple, glape", "Bandha"]] "

161 ECHO: "[ INFO ] run_test(); t06 passed: 'nssequence_31(["b", "a", "n", "a", "n", "a", "s"])=[["b", "a", "n"], ["a", "n", "a"], ["n", "a", "n"], ["a", "n", "a"], ["n", "a", "s"]]'"

162 ECHO: "[ INFO ] run_test(); t07 passed: 'nssequence_31([undef])=[]'"

163 ECHO: "[ INFO ] run_test(); t08 passed: 'nssequence_31([[1, 2], [2, 3]])=[]'"
164 ECHO: "[ INFO ] run_test(); t09 passed: 'nssequence_31(["ab", [1, 2], [2, 3], [4, 5]])=[["ab", [1, 2], [2,
164 ECHO: "[ INFO ] run_test(); t09 passed: 'nssequence_31(["ab", [1, 2], [2, 3], [4, 5]])=[["ab", [1, 2], [2, 3]], [[1, 2], [2, 3]], [[1, 2], [2, 3]], [[1, 2], [2, 3]], [4, 5]])"

165 ECHO: "[ INFO ] run_test(); t10 passed: 'nssequence_31([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]]'"

166 ECHO: "[ INFO ] run_test(); t11 passed: 'nssequence_31([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[[0, 1, 2], [1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6], [5, 6, 7], [6, 7, 8], [7, 8, 9], [8, 9, 10], [9, 10, 11], [10, 11, 12], [11, 12, 13], [12, 13, 14], [13, 14, 15]]'"

167 ECHO: "[ INFO ] run_test(); t01 passed: 'eappend_TO(undef)=undef'"

168 ECHO: "[ INFO ] run_test(): t02 passed: 'eappend_TO([])=[[0]]]'"
168 ECHO: "[ INFO ] run_test(); t02 passed: 'eappend_T0([])=[[0]]'
169 ECHO: "[ INFO ] run_test(); t03 passed: 'eappend_T0([0 : 0.5 : 9]) = undef'"
170 ECHO: "[ INFO ] run_test(); t04 passed: 'eappend_T0(A string)=[["A", 0], [" ", 0], ["s", 0], ["t", 0], ["r", 0], ["i", 0], ["g", 0]]'"
[2, 3, 0], [4, 5, 0]]'"
176 ECHO: "[ INFO ] run_test(); t10 passed: 'eappend_T0([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 2, 3, 0], [4, 5, 6, 0], [7, 8, 9, 0], ["a", "b", "c", 0]]'"

177 ECHO: "[ INFO ] run_test(); t11 passed: 'eappend_T0([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            15])=[[0, 0], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 0], [8, 0], [9, 0], [10, 0], [11, 0], [12, 0],
            [13, 0], [14, 0], [15, 0]]'"
178 ECHO: "[ INFO ] run_test(); t01 passed: 'insert_T0(undef)=undef'"
179 ECHO: "[ INFO ] run_test(); t02 passed: 'insert_T0([])=undef'"
180 ECHO: "[ INFO ] run_test(); t03 passed: 'insert_T0([0 : 0.5 : 9]) = undef'"

181 ECHO: "[ INFO ] run_test(); t04 passed: 'insert_T0(A string) = undef'"
182 ECHO: "[ INFO ] run_test(); t05 passed: 'insert_T0(["orange", "apple", "grape", "banana"])=["orange", 0,
            "apple", "grape", "banana"]'"
183 ECHO: "[ INFO ] run_test(); t06 passed: 'insert_T0(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
"a", "n", "a", 0, "s"]'"

184 ECHO: "[ INFO ] run_test(); t07 passed: 'insert_T0([undef])=undef'"

185 ECHO: "[ INFO ] run_test(); t08 passed: 'insert_T0([[1, 2], [2, 3]])=[[1, 2], 0, [2, 3]]'"
186 ECHO: "[ INFO ] run_test(); t09 passed: 'insert_T0(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], 0, [2,
3], [4, 5]]'"

187 ECHO: "[ INFO ] run_test(); t10 passed: 'insert_TO([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
            "c"]])=undef'"
188 ECHO: "[ INFO ] run_test(); t11 passed: 'insert_TO([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 0, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"
189 ECHO: "[ INFO ] run_test(); t01 passed: 'delete_T0(undef)=undef'"
190 ECHO: "[ INFO ] run_test(); t02 passed: 'delete_T0([])=[]'"
191 ECHO: "[ INFO ] run_test(); t03 passed: 'delete_T0([0 : 0.5 : 9]) = undef'"
192 ECHO: "[ INFO ] run_test(); t04 passed: 'delete_T0(A string) = ["A", " ", "s", "t", "r", "i", "n", "g"]'"
193 ECHO: "[ INFO ] run_test(); t05 passed: 'delete_T0(["orange", "apple", "grape", "banana"])=["orange",
            "grape", "banana"]'"
194 ECHO: "[ INFO ] run_test(); t06 passed: 'delete_T0(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
                    "n", "a"]/"
            "a",
195 ECHO: "[ INFO ] run_test(); t07 passed: 'delete_T0([undef])=[undef]'"
196 ECHO: "[ INFO ] run_test(); t08 passed: 'delete_T0([[1, 2], [2, 3]])=[[1, 2]]'"
197 ECHO: "[ INFO ] run_test(); t09 passed: 'delete_T0(["ab", [1, 2], [2, 3], [4, 5])]=["ab", [1, 2], [4, 5]]'"
198 ECHO: "[ INFO ] run_test(); t10 passed: 'delete_T0([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1,
2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]

2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]'"

199 ECHO: "[ INFO ] run_test(); t11 passed: 'delete_TO([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"

200 ECHO: "[ INFO ] run_test(); t01 passed: 'strip(undef)=undef'"
```

```
201 ECHO: "[ INFO ] run_test(); t02 passed: 'strip([])=[]'"
202 ECHO: "[ INFO ] run_test(); t03 passed: 'strip([0 : 0.5 : 9])=undef'"
203 ECHO: "[ INFO ] run_test(); t03 passed: strip([0.0.3.3])-minder
204 ECHO: "[ INFO ] run_test(); t04 passed: 'strip(8 string)=["ah, " ", "s", "t", "r", "i", "n", "g"]'"
204 ECHO: "[ INFO ] run_test(); t05 passed: 'strip(["orange", "apple", "grape", "banana"])=["orange", "apple",
                  "grape", "banana"]'"
205 ECHO: "[ INFO ] run_test(); t06 passed: 'strip(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
"n", "a", "s"]'"

206 ECHO: "[ INFO ] run_test(); t07 passed: 'strip([undef])=[undef]'"
207 ECHO: "[ INFO ] run_test(); t08 passed: 'strip([1, 2], [2, 3])=[[1, 2], [2, 3]]'"
208 ECHO: "[ INFO ] run_test(); t09 passed: 'strip(["ab", [1, 2], [2, 3], [4, 5])=["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3]])=["ab", [2, 3]]
                  5]]/"
209 ECHO: "[ INFO ] run_test(); t10 passed: 'strip([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 2,
3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]'"
210 ECHO: "[ INFO ] run_test(); t11 passed: 'strip([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"
211 ECHO: "[ INFO ] run_test(); t01 passed: 'unique(undef)=undef'"
212 ECHO: "[ INFO ] run_test(); t02 passed: 'unique([])=[]'"
213 ECHO: "[ INFO ] run_test(); t03 passed: 'unique([0 : 0.5 : 9])=undef'"
214 ECHO: "[ INFO ] run_test(); t04 passed: 'unique([a string)=["A", " ", "s", "t", "r", "i", "n", "g"]'"
215 ECHO: "[ INFO ] run_test(); t05 passed: 'unique(["orange", "apple", "grape", "banana"])=["orange", "apple",
                   "grape", "banana"]'"
216 ECHO: "[ INFO ] run_test(); t06 passed: 'unique(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "s"])"
217 ECHO: "[ INFO ] run_test(); t07 passed: 'unique([undef])=[undef]'"
218 ECHO: "[ INFO ] run_test(); t08 passed: 'unique([[1, 2], [2, 3]])=[[1, 2], [2, 3]]'"
219 ECHO: "[ INFO ] run_test(); t09 passed: 'unique(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3], [4,
                 511'"
220 ECHO: "[ INFO ] run_test(); t10 passed: 'unique([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]'"
221 ECHO: "[INFO] run_test(); t11 passed: 'unique([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"
```

3.1.2.3 Lists

- Script
- Results

3.1.2.3.1 Script

```
include <datatypes.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
                         // show passing tests
show_passing = true;
                        // show skipped tests
show_skipped = true;
echo( str("OpenSCAD Version ", version()) );
// test-values columns
test_c =
  ["id", "identifier"],
  ["td", "description"],
  ["tv", "test value"]
1;
// test-values rows
test r =
  ["t01", "The undefined value",
                                            undef],
  ["t02", "The empty list",
                                            empty_lst],
  ["t03", "A range",
                                             [0:0.5:9]],
  ["t04", "A string",
                                             "A string"],
                                            ["orange", "apple", "grape", "banana"]],
["b", "a", "n", "a", "n", "a", "s"]],
  ["t05", "Test list 01",
  ["t06", "Test list 02",
  ["t07", "Test list 03",
                                             [undef]],
  ["t08", "Test list 04",
                                             [[1,2],[2,3]]],
  ["t09", "Test list 05", ["t10", "Test list 06",
                                            [["ab",[1,2],[2,3],[4,5]]],
[[1,2,3],[4,5,6],[7,8,9],["a","b","c"]]],
  ["t11", "Vector of integers 0 to 15", [for (i=[0:15]) i]]
1;
test ids = get table ridl( test r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
```

3.1 Datatypes 33

```
// expected rows: ("golden" test results), use 's' to skip test
skip = -1; // skip test
good_r =
[ // function
     ["lstr",
          undef,
                                                                                                                                                      // t01
                                                                                                                                                      // t02
          empty_str,
          "[0:0.5:9]",
                                                                                                                                                      // t03
                                                                                                                                                      // t04
          "A string",
                                                                                                                                                      // t05
          "orangeapplegrapebanana",
           "bananas",
                                                                                                                                                      // t06
                                                                                                                                                      // t07
           "undef",
           "[1, 2][2, 3]",
                                                                                                                                                      // t08
          "0123456789101112131415"
                                                                                                                                                     // t11
     ["lstr_html_B",
           "<b>undef</b>",
                                                                                                                                                      // t01
                                                                                                                                                      // t02
           emptv str.
           "<b>[0 : 0.5 : 9]</b>",
                                                                                                                                                      // t03
           "<b>A string</b>",
                                                                                                                                                      // t04
           "<b>orange</b><b>apple</b><b>grape</b><b>banana</b>",
           "<b>b</b><b>a</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n</b><b>n<
           "<b>undef</b>",
                                                                                                                                                   // t07
           "<b>[1, 2]</b><b>[2, 3]</b>",
                                                                                                                                                    // t08
          "<b>ab</b><b:[1, 2]</b><b>[2, 3]</b><b>[4, 5]</b>", // t09
"<b>[1, 2, 3]</b><b>[4, 5, 6]</b><b>[7, 8, 9]</b><b>[\"a\", \"b\", \"c\"]</b>",
     <br/><b>0</b><b>1</b><b>2</b><b>3</b><b>4</b><b>5</b><b>6</b><b>7</b><b>8</b><b>9</b><b>10</b><b>11</b><b>11</b><b>12</b><b>13</b><b>14</b><b>14</b><b>12</b><b>14</b><b>15</b><b>14</b><b>15</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16<b>16<b>16<b>16<b>16<b>16161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616<t
     ],
["consts",
          empty_lst,
                                                                                                                                                      // t01
                                                                                                                                                      // t02
          empty_lst,
                                                                                                                                                      // t03
// t04
          empty_lst,
          empty_lst,
                                                                                                                                                      // t05
          empty_lst,
                                                                                                                                                      // t06
          empty_lst,
                                                                                                                                                      // t07
          empty_lst,
                                                                                                                                                      // ±08
          empty_lst,
                                                                                                                                                      // t09
// t10
          empty_lst,
          empty_lst,
          empty_lst
                                                                                                                                                      // t11
     ["get_index",
                                                                                                                                                     // t01
          empty_lst,
           empty_lst,
                                                                                                                                                      // t02
           empty_lst,
                                                                                                                                                      // t03
           [0,1,2,3,4,5,6,7],
                                                                                                                                                      // t04
                                                                                                                                                     // t05
           [0,1,2,3],
           [0,1,2,3,4,5,6],
                                                                                                                                                      // t06
           [0],
                                                                                                                                                      // t07
           [0,1],
                                                                                                                                                      // t08
           [0,1,2,3],
                                                                                                                                                      // t09
           [0,1,2,3],
                                                                                                                                                      // t10
           [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
     ["pad_9",
                                                                                                                                                     // t01
           [undef, 0, 0, 0, 0, 0, 0, 0, 0],
           [0,0,0,0,0,0,0,0,0],
                                                                                                                                                      // t02
           [[0:0.5:9],0,0,0,0,0,0,0,0],
                                                                                                                                                     // t03
                                                                                                                                                     // t04
           ["A string", 0],
           ["orange", "apple", "grape", "banana", 0, 0, 0, 0, 0], ["b", "a", "n", "a", "n", "a", "s", 0, 0],
                                                                                                                                                     // t05
                                                                                                                                                     // t06
           [undef, 0, 0, 0, 0, 0, 0, 0, 0],
                                                                                                                                                     // t07
           [[1,2],[2,3],0,0,0,0,0,0,0,0],
                                                                                                                                                     // t08
           [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                                                                                                      // t11
    ],
["limit_12",
          undef,
                                                                                                                                                      // t01
          empty_lst,
                                                                                                                                                     // t02
           [0:0.5:9],
                                                                                                                                                      // t03
                                                                                                                                                      // t04
            "A string"
          ["orange", "apple", "grape", "banana"], ["b", "a", "n", "a", "n", "a", "s"],
                                                                                                                                                     // t05
                                                                                                                                                     // t06
           [undef],
                                                                                                                                                      // ±07
           [[1,2],[2,2]],
                                                                                                                                                      // t08
```

```
["ab",[1,2],[2,2],[2,2]],
                                                                      // t09
  [[1,2,2],[2,2],[2,2,2],["a","b","c"]],
[1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2]
                                                                       // t10
                                                                      // t11
["sum",
  undef,
                                                                       // t01
                                                                       // t02
  85.5,
                                                                       // t03
  undef,
                                                                       // t04
                                                                       // t05
  undef,
  undef,
                                                                       // t06
                                                                       // t07
  [3,5],
                                                                      // t08
                                                                      // t09
  undef,
   [undef, undef, undef],
                                                                       // t10
  120
                                                                      // t11
["mean",
  undef,
                                                                      // t01
                                                                      // t02
  Ο,
                                                                      // t03
  4.5,
  undef,
                                                                      // t04
                                                                      // t05
  undef,
  undef,
                                                                      // t06
  undef,
                                                                      // t07
                                                                      // t08
  [1.5,2.5],
                                                                      // t09
  undef.
                                                                      // t10
  [undef, undef, undef],
                                                                      // t11
  7.5
],
["eselect F",
                                                                      // t01
  undef,
                                                                      // t02
// t03
  empty_lst,
  undef,
["A"," ","s","t","r","i","n","g"],
["o","a","g","b"],
["b","a","n","a","n","a","s"],
                                                                      // t04
                                                                      // t05
                                                                      // t06
// t07
                                                                      // t08
   [1,2],
   ["a",1,2,4],
                                                                      // t09
                                                                      // t10
// t11
   [1,4,7,"a"],
  skip
],
["eselect_L",
                                                                      // t01
  undef,
                                                                      // t02
// t03
// t04
  empty_lst,
  empry_se,
undef,
["A"," ","s","t","r","i","n","g"],
["e","e","e","a"],
["b","a","n","a","n","a","s"],
                                                                      // t05
// t06
                                                                      // t07
// t08
   [undef],
   [2,3],
   ["b",2,3,5],
                                                                       // t09
                                                                       // t10
   [3,6,9,"c"],
  skip
                                                                       // t11
["eselect_1",
  undef,
                                                                      // t01
  empty_lst,
                                                                       // t02
  undef,
                                                                       // t03
  skip,
                                                                       // t04
   ["r", "p", "r", "a"],
                                                                      // t05
                                                                      // t06
   skip,
                                                                      // t07
   [undef],
                                                                      // t08
   [2,3],
   ["b",2,3,5],
                                                                      // t09
   [2,5,8,"b"],
                                                                      // t10
                                                                      // t11
  skip
],
["smerge",
  undef,
                                                                      // t01
  empty_lst,
                                                                      // t02
   [[0:0.5:9]],
                                                                      // t03
                                                                      // t04
   ["A string"],
   ["orange", "apple", "grape", "banana"],
["b", "a", "n", "a", "n", "a", "s"],
                                                                      // t05
                                                                      // t06
                                                                      // t07
   [undef],
                                                                      // t08
   [1,2,2,3],
   ["ab",1,2,2,3,4,5],
                                                                      // t09
  [1,2,3,4,5,6,7,8,9,"a","b","c"],
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                      // t10
                                                                      // t11
```

3.1 Datatypes 35

```
["pmerge",
    undef,
                                                                // t01
    empty_lst,
                                                                // t02
    undef,
                                                                // t03
    ["A string"],
                                                                 // t04
       ["o", "a", "g", "b"], ["r", "p", "r", "a"],
      ["a","p","a","n"],["n","l","p","a"],
["g","e","e","n"]
                                                                // t05
    [["b", "a", "n", "a", "n", "a", "s"]],
                                                                // t06
    undef,
                                                                // t07
    [[1,2],[2,3]],
                                                                // t08
    [["a",1,2,4],["b",2,3,5]],
                                                                // t09
    [[1,4,7,"a"],[2,5,8,"b"],[3,6,9,"c"]],
                                                                // t10
                                                                // t11
    undef
  ],
["qsort",
                                                                // t01
    undef,
    empty_lst,
                                                                // t02
    undef,
                                                                // t03
                                                                // t04
    undef,
    ["apple", "banana", "grape", "orange"],
                                                                // t05
    ["a", "a", "a", "b", "n", "n", "s"],
                                                                // t06
                                                                // t07
    [undef],
                                                                // t08
    skip,
                                                                // t09
    skip,
                                                                // t10
    skip,
    [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                // t11
  ],
["qsort_1R",
    undef,
                                                                // t01
                                                                // t02
    empty_lst,
                                                                // t03
    undef.
                                                                // t04
    undef.
                                                                // t05
    ["orange", "grape", "apple", "banana"],
                                                                // t06
    skip,
                                                                // t07
    skip,
                                                                // t08
    [[2,3],[1,2]],
    [[4,5],[2,3],[1,2],"ab"],
                                                                // t09
    [[7,8,9],[4,5,6],[1,2,3],["a","b","c"]],
                                                                // ±10
                                                                // t11
    skip
  ["qsort2_1R",
                                                                // t01
    undef.
                                                                // t02
// t03
    empty_lst,
    undef.
                                                                // t04
    undef,
                                                                // t05
    ["orange", "grape", "apple", "banana"],
                                                                // t06
// t07
    skip,
    skip,
    [[2,3],[1,2]],
                                                                // t08
     ["ab",[4,5],[2,3],[1,2]],
                                                                // t09
    [["a", "b", "c"], [7,8,9], [4,5,6], [1,2,3]],
                                                                // t10
    skip
                                                                // t11
  ["qsort2_HR",
    undef,
                                                                // t01
    empty_lst,
                                                                // t02
                                                                // t03
                                                                // t04
    ["orange", "grape", "banana", "apple"], ["s", "n", "n", "b", "a", "a", "a"],
                                                                // t05
                                                                // t06
                                                                // t07
    [undef],
    [[3,2],[2,1]],
                                                                // t08
    [[5,4],[3,2],[2,1],"ab"],
[["c","b","a"],[9,8,7],[6,5,4],[3,2,1]],
[15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
                                                                // t09
                                                                // t10
                                                                // t11
 ]
];
// sanity-test tables
table_check( test_r, test_c, false );
table_check( good_r, good_c, false );
// validate helper function and module
function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
module run_test( fname, fresult, vid )
  value_text = get_table_v(test_r, test_c, vid, "td");
```

```
pass_value = get_table_v(good_r, good_c, fname, vid);
   test_pass = validate( cv=fresult, t="equals", ev=pass_value, pf=true );
   test_text = validate( str(fname, "(", get_value(vid), ")=", pass_value), fresult, "equals",
   pass value );
   if ( pass_value != skip )
      if ( !test_pass )
         log_warn( str(vid, "(", value_text, ") ", test_text) );
      else if ( show_passing )
  log_info( str(vid, " ", test_text) );
   else if ( show_skipped )
      log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
// Indirect function calls would be very useful here!!!
for (vid=test_ids) run_test( "lstr", lstr(get_value(vid)), vid );
for (vid=test_ids) run_test( "lstr_html_B", lstr_html(get_value(vid),p="b"), vid );
for (vid=test_ids) run_test( "consts", consts(get_value(vid)), vid );
for (vid=test_ids) run_test( "get_index", get_index(get_value(vid)), vid );
for (vid=test_ids) run_test( "pad_9", pad(get_value(vid), w=9), vid );
log_info( "not testing: dround()" );
log_info( "not testing: sround()" );
for (vid=test_ids) run_test( "limit_12", limit(get_value(vid),1,2), vid );
for (vid=test_ids) run_test( "sum", sum(get_value(vid)), vid);
for (vid=test_ids) run_test( "mean", mean(get_value(vid)), vid);
log_info( "not testing: ciselect()" );
log_info( "not testing: cmvselect()" );
for (vid=test_ids) run_test( "eselect_F", eselect(get_value(vid), f=true), vid );
for (vid=test_ids) run_test( "eselect_L", eselect(get_value(vid), l=true), vid );
for (vid=test_ids) run_test( "eselect_l", eselect(get_value(vid), i=1), vid );
for (vid=test_ids) run_test( "smerge", smerge(get_value(vid)), vid );
for (vid=test_ids) run_test( "pmerge", pmerge(get_value(vid)), vid );
for (vid=test_ids) run_test( "pmerge", pmerge(get_value(vid)), vid );
for (vid=test_ids) run_test( "qsort", qsort(get_value(vid)), vid );
for (vid=test_ids) run_test( "qsort_1R", qsort(get_value(vid), i=1, r=true), vid );
for (vid=test_ids) run_test( "qsort2_1R", qsort2(get_value(vid), i=1, r=true), vid );
for (vid=test_ids) run_test( "qsort2_1R", qsort2(get_value(vid), d=5, r=true), vid );
// end-of-tests
```

3.1.2.3.2 Results

```
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'lstr(undef)=undef'"
3 ECHO: "[ INFO ] run_test(); t02 passed: 'lstr([])='"
4 ECHO: "[ INFO ] run_test(); t03 passed: 'lstr([0: 0.5: 9])=[0: 0.5: 9]'"
5 ECHO: "[ INFO ] run_test(); t04 passed: 'lstr(A string) = A string'
6 ECHO: "[ INFO ] run_test(); t05 passed: 'lstr(["orange", "apple", "grape",
             "banana"])=orangeapplegrapebanana'"
7 ECHO: "[ INFO ] run_test(); t06 passed: 'lstr(["b", "a", "n", "a", "n", "a", "s"])=bananas'"

8 ECHO: "[ INFO ] run_test(); t07 passed: 'lstr([undef])=undef'"

9 ECHO: "[ INFO ] run_test(); t08 passed: 'lstr([[1, 2], [2, 3]])=[1, 2][2, 3]'"

10 ECHO: "[ INFO ] run_test(); t09 passed: 'lstr(["ab", [1, 2], [2, 3], [4, 5]])=ab[1, 2][2, 3][4, 5]'"

11 ECHO: "[ INFO ] run_test(); t10 passed: 'lstr([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1, 2,
             3][4, 5, 6][7, 8, 9]["a", "b", "c"]'"
12 ECHO: "[ INFO ] run_test(); t11 passed: 'lstr([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            15])=0123456789101112131415'"
13 ECHO: "[ INFO ] run_test(); t01 passed: 'lstr_html_B(undef)=<b>undef</b>'"
14 ECHO: "[ INFO ] run_test(); t02 passed: 'lstr_html_B([])='"
15 ECHO: "[ INFO ] run_test(); t03 passed: 'lstr_html_B([0 : 0.5 : 9]) = <b > [0 : 0.5 : 9] < /b > ""
16 ECHO: "[ INFO ] run_test(); t04 passed: 'lstr_html_B(A string) = <b>A string</b>
17 ECHO: "[ INFO ] run_test(); t05 passed: 'lstr_html_B(["orange", "apple", "grape",
             "banana"])=<b>orange</b><b>apple</b><b>grape</b><b>banana</b>
18 ECHO: "[ INFO ] run_test(); t06 passed: 'lstr_html_B(["b", "a", "n", "a", "n", "a",
             19 ECHO: "[ INFO ] run_test(); t07 passed: 'lstr_html_B([undef]) = <b>undef </b>'"
20 ECHO: "[ INFO ] run_test(); t08 passed: 'lstr_html_E([[1, 2], [2, 3]])=<b>[1, 2]</b><b>[2, 3]</b>'"
21 ECHO: "[ INFO ] run_test(); t09 passed: 'lstr_html_B(["ab", [1, 2], [2, 3], [4, 5]]) = <b > ab < /b > (1, 2), [2, 3], [4, 5]])
            2]</b><b>[2, 3]</b><b>[4, 5]</b>'"
22 ECHO: "[ INFO ] run_test(); t10 passed: 'lstr_html_B([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=<b>[1, 2, 3]</b><b>[4, 5, 6]</b><b>[7, 8, 9]</b><b>["a", "b", "c"]</b>'"
23 ECHO: "[ INFO ] run_test(); t11 passed: 'lstr_html_B([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            151)
           =<b>0</b><b>1</b><b>2</b><b>3</b><b>4</b><b>5</b><b>6</b><b>7</b><b>8</b><b>9</b><b>10</b><b>11</b><b>11</b><b>12</b><b>13</b><b>14</b><b>14</b><b>15</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16</b><b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<b>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<bb>16<br/>16<bb>16<br/>1616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616161616<
24 ECHO: "[ INFO ] run_test(); t01 passed: 'consts(undef)=[]'"
25 ECHO: "[ INFO ] run_test(); t02 passed: 'consts([])=[]'"
26 ECHO: "[ INFO ] run_test(); t03 passed: 'consts([0 : 0.5 : 9])=[]'"
```

3.1 Datatypes 37

```
27 ECHO: "[ INFO ] run_test(); t04 passed: 'consts(A string)=[]'"
28 ECHO: "[ INFO ] run_test(); t05 passed: 'consts(["orange", "apple", "grape", "banana"])=[]'"
29 ECHO: "[ INFO ] run_test(); t06 passed: 'consts(["b", "a", "n", "a", "n", "a", "s"])=[]'"
30 ECHO: "[ INFO ] run_test(); t07 passed: 'consts([undef])=[]'"
31 ECHO: "[ INFO ] run_test(); t08 passed: 'consts([[1, 2], [2, 3]])=[]'"
32 ECHO: "[ INFO ] run_test(); t09 passed: 'consts(["ab", [1, 2], [2, 3], [4, 5]])=[]'"
33 ECHO: "[ INFO ] run_test(); t10 passed: 'consts([0, 1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[]'"
34 ECHO: "[ INFO ] run_test(); t11 passed: 'consts([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          15])=[]'"
35 ECHO: "[ INFO ] run_test(); t01 passed: 'get_index(undef)=[]'"
36 ECHO: "[ INFO ] run_test(); t02 passed: 'get_index([])=[]
37 ECHO: "[ INFO ] run_test(); t03 passed: 'get_index([0 : 0.5 : 9])=[]'"
38 ECHO: "[ INFO ] run_test(); t04 passed: 'get_index(A string)=[0, 1, 2, 3, 4, 5, 6, 7]'"
39 ECHO: "[ INFO ] run_test(); t05 passed: 'get_index(["orange", "apple", "grape", "banana"])=[0, 1, 2, 3]'"
40 ECHO: "[ INFO ] run_test(); t06 passed: 'get_index(["b", "a", "n", "a", "n", "a", "s"])=[0, 1, 2, 3, 4, 5,
          6]′"
41 ECHO: "[ INFO ] run_test(); t07 passed: 'get_index([undef])=[0]'
41 ECHO: "[ INFO ] run_test(); t08 passed: 'get_index([[1, 2], [2, 3]])=[0, 1]'"

42 ECHO: "[ INFO ] run_test(); t08 passed: 'get_index([[1, 2], [2, 3]])=[0, 1]'"

43 ECHO: "[ INFO ] run_test(); t09 passed: 'get_index(["ab", [1, 2], [2, 3], [4, 5]])=[0, 1, 2, 3]'"

44 ECHO: "[ INFO ] run_test(); t10 passed: 'get_index([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[0, 1, 2, 3]'"
1, 2, 3]'"

45 ECHO: "[ INFO ] run_test(); t11 passed: 'get_index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"

46 ECHO: "[ INFO ] run_test(); t01 passed: 'pad_9(undef)=[undef, 0, 0, 0, 0, 0, 0, 0, 0]'"
47 ECHO: "[ INFO ] run_test(); t02 passed: 'pad_9([])=[0, 0, 0, 0, 0, 0, 0, 0, 0]'"
48 ECHO: "[ INFO ] run_test(); t03 passed: 'pad_9([0:0.5:9])=[[0:0.5:9], 0, 0, 0, 0, 0, 0, 0]'"
49 ECHO: "[ INFO ] run_test(); t04 passed: 'pad_9(A string)=["A string", 0]'"
50 ECHO: "[ INFO ] run_test(); t05 passed: 'pad_9(["orange", "apple", "grape", "banana"])=["orange", "apple",
           "grape", "banana", 0, 0, 0, 0, 0]'"
51 ECHO: "[INFO] run_test(); t06 passed: 'pad_9(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
          "n", "a", "s", 0, 0]'"
52 ECHO: "[ INFO ] run_test(); t07 passed: 'pad_9([undef])=[undef, 0, 0, 0, 0, 0, 0, 0, 0]'"
53 ECHO: "[ INFO ] run_test(); t08 passed: 'pad_9(["ab", [1, 2], [2, 3]) = ["l, 2], [2, 3], 0, 0, 0, 0, 0, 0, 0]'"
54 ECHO: "[ INFO ] run_test(); t09 passed: 'pad_9(["ab", [1, 2], [2, 3], [4, 5]]) = ["ab", [1, 2], [2, 3], [4,
54 ECHO: "[INFO] run_test(); to9 passed: 'pad_9(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3], [4, 5], [3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"], 0, 0, 0, 0, 0]'"

56 ECHO: "[INFO] run_test(); t11 passed: 'pad_9([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"

57 ECHO: "[INFO] root(); not testing: dround()"
58 ECHO: "[ INFO ] root(); not testing: sround()"
59 ECHO: "[ INFO ] run_test(); t01 passed: 'limit_12(undef)=undef'"
60 ECHO: "[ INFO ] run_test(); t02 passed: 'limit_12([])=[]'
61 ECHO: "[ INFO ] run_test(); t03 passed: 'limit_12([0 : 0.5 : 9])=[0 : 0.5 : 9]'"
62 ECHO: "[INFO] run_test(); t04 passed: 'limit_12(0 . 0.5 . 9])-[0 . 0.5 : 9]
63 ECHO: "[INFO] run_test(); t05 passed: 'limit_12(["orange", "apple", "grape", "banana"])=["orange", "apple", "grape", "banana"]'"
64 ECHO: "[ INFO ] run_test(); t06 passed: 'limit_12(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
"n", "a", "s"]'"

65 ECHO: "[ INFO ] run_test(); t08 passed: 'limit_12([undef])=[undef]'"

66 ECHO: "[ INFO ] run_test(); t08 passed: 'limit_12([[1, 2], [2, 3]])=[[1, 2], [2, 2]]'"

67 ECHO: "[ INFO ] run_test(); t09 passed: 'limit_12(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 2],
[2, 2]]'"

68 ECHO: "[ INFO ] run_test(); t10 passed: 'limit_12([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1,
2, 2], [2, 2, 2], [2, 2, 2], ["a", "b", "c"]]'"

69 ECHO: "[ INFO ] run_test(); t11 passed: 'limit_12([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          70 ECHO: "[ INFO ] run_test(); t01 passed: 'sum(undef)=undef'"
71 ECHO: "[ INFO ] run_test(); t02 passed: 'sum([])=0'"
72 ECHO: "[ INFO ] run_test(); t03 passed: 'sum([0:0.5:9])=85.5'"
73 ECHO: "[ INFO ] run_test(); t04 passed: 'sum(A string)=undef'"
74 ECHO: "[ INFO ] run_test(); t05 passed: 'sum(["orange", "apple", "grape", "banana"])=undef'"
75 ECHO: "[ INFO ] run_test(); t06 passed: 'sum(["b", "a", "n", "a", "n", "a", "s"])=undef'"
76 ECHO: "[ INFO ] run_test(); t07 passed: 'sum([undef])=undef'"
77 ECHO: "[ INFO ] run_test(); t08 passed: 'sum([[1, 2], [2, 3]])=[3, 5]'"
78 ECHO: "[ INFO ] run_test(); t09 passed: 'sum(["ab", [1, 2], [2, 3], [4, 5]])=undef'"
79 ECHO: "[ INFO ] run_test(); t10 passed: 'sum([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[undef,
          undef, undef]'"
80 ECHO: "[ INFO ] run_test(); t11 passed: 'sum([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=120'"
81 ECHO: "[ INFO ] run_test(); t01 passed: 'mean(undef)=undef'"
82 ECHO: "[ INFO ] run_test(); t02 passed: 'mean([])=0'"
83 ECHO: "[ INFO ] run_test(); t03 passed: 'mean([0 : 0.5 : 9])=4.5'"
84 ECHO: "[ INFO ] run_test(); t04 passed: 'mean(A string)=undef'"
85 ECHO: "[ INFO ] run_test(); t05 passed: 'mean(["orange", "apple", "grape", "banana"])=undef'"
86 ECHO: "[ INFO ] run_test(); t06 passed: 'mean(["b", "a", "n", "a", "n", "a", "s"])=undef'"
87 ECHO: "[ INFO ] run_test(); t07 passed: 'mean([undef])=undef'"
88 ECHO: "[INFO] run_test(); t08 passed: 'mean([[1, 2], [2, 3]])=[1.5, 2.5]'"
89 ECHO: "[INFO] run_test(); t09 passed: 'mean(["ab", [1, 2], [2, 3], [4, 5]])=undef'"
90 ECHO: "[INFO] run_test(); t10 passed: 'mean([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[undef,
          undef, undef]'"
91 ECHO: "[ INFO ] run_test(); tl1 passed: 'mean([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=7.5'"
```

```
92 ECHO: "[ INFO ] root(); not testing: ciselect()"
93 ECHO: "[ INFO ] root(); not testing: cmvselect()"
94 ECHO: "[ INFO ] run_test(); t01 passed: 'eselect_F (undef) = undef'"
95 ECHO: "[ INFO ] run_test(); t02 passed: 'eselect_F([])=[]'"
96 ECHO: "[ INFO ] run_test(); t03 passed: 'eselect_F([0:0.5:9])=undef'"
97 ECHO: "[ INFO ] run_test(); t04 passed: 'eselect_F(A string)=["A", " ", "s", "t", "r", "i", "n", "g"]'"
98 ECHO: "[ INFO ] run_test(); t05 passed: 'eselect_F(["orange", "apple", "grape", "banana"])=["o", "a", "g",
           "b"]'"
99 ECHO: "[ INFO ] run_test(); t06 passed: 'eselect_F(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
"a", "n", "a", "s"]'"

100 ECHO: "[ INFO ] run_test(); t07 passed: 'eselect_F([undef])=[undef]'
101 ECHO: "[ INFO ] run_test(); t08 passed: 'eselect_F([[1, 2], [2, 3]])=[1, 2]'"
102 ECHO: "[ INFO ] run_test(); t09 passed: 'eselect_F(["ab", [1, 2], [2, 3], [4, 5]])=["a", 1, 2, 4]'"
103 ECHO: "[ INFO ] run_test(); t10 passed: 'eselect_F([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1,
           4, 7, "a"]'
104 ECHO: "[ INFO ] run_test(); tll *skip*: 'eselect_F(Vector of integers 0 to 15)'"
105 ECHO: "[ INFO ] run_test(); t01 passed: 'eselect_L(undef)=undef'
106 ECHO: "[ INFO ] run_test(); t02 passed: 'eselect_L([])=[]'"
107 ECHO: "[ INFO ] run_test(); t03 passed: 'eselect_L([0 : 0.5 : 9]) = undef'"
108 ECHO: "[ INFO ] run_test(); t04 passed: 'eselect_L(A string)=["A", " ", "s", "t", "r", "i", "n", "g"]'"
109 ECHO: "[ INFO ] run_test(); t05 passed: 'eselect_L(["orange", "apple", "grape", "banana"])=["e", "e", "e",
            "a"]'"
110 ECHO: "[ INFO ] run_test(); t06 passed: 'eselect_L(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
           "a", "n", "a", "s"]'"
111 ECHO: "[ INFO ] run_test(); t07 passed: 'eselect_L([undef])=[undef]'
111 ECHO: [INFO] run_test(); t08 passed: 'eselect_L([[1, 2], [2, 3]))=[2, 3]'"
112 ECHO: "[INFO] run_test(); t08 passed: 'eselect_L([[1, 2], [2, 3], [4, 5]])=["b", 2, 3, 5]'"
113 ECHO: "[INFO] run_test(); t10 passed: 'eselect_L([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[3,
6, 9, "c"]'"

115 ECHO: "[ INFO ] run_test(); t11 *skip*: 'eselect_L(Vector of integers 0 to 15)'"
116 ECHO: "[ INFO ] run_test(); t01 passed: 'eselect_1(undef)=undef'
117 ECHO: "[ INFO ] run_test(); t02 passed: 'eselect_1([])=[]
118 ECHO: "[ INFO ] run_test(); t03 passed: 'eselect_1([0 : 0.5 : 9])=undef'"
119 ECHO: "[ INFO ] run_test(); t04 *skip*: 'eselect_1(A string)'"
120 ECHO: "[ INFO ] run_test(); t05 passed: 'eselect_1(["orange", "apple", "grape", "banana"])=["r", "p", "r",
          "a"]'"
121 ECHO: "[ INFO ] run_test(); t06 *skip*: 'eselect_1(Test list 02)'"
122 ECHO: "[ INFO ] run_test(); t07 passed: 'eselect_1([undef])=[undef]'"
5, 8, "b"]'"

126 ECHO: "[ INFO ] run_test(); t11 *skip*: 'eselect_1(Vector of integers 0 to 15)'"

127 ECHO: "[ INFO ] run_test(); t01 passed: 'smerge(undef' "
128 ECHO: "[ INFO ] run_test(); t02 passed: smerge([])=[]'"
129 ECHO: "[ INFO ] run_test(); t03 passed: 'smerge([] : 0.5 : 9]]'"
130 ECHO: "[ INFO ] run_test(); t04 passed: 'smerge([0 .0.5 .5])"|" 131 ECHO: "[ INFO ] run_test(); t05 passed: 'smerge(["orange", "apple", "grape", "banana"])=["orange", "apple", "grape", "apple", "apple", "grape", "apple", "ap
           "grape", "banana"]'"
132 ECHO: "[ INFO ] run_test(); t06 passed: 'smerge(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
133 ECHO: [INFO] run_test(); t08 passed: 'smerge([[1, 2], [2, 3]])=[1, 2, 2, 3]'"

135 ECHO: "[INFO] run_test(); t08 passed: 'smerge([[1, 2], [2, 3], [4, 5]])=["ab", 1, 2, 2, 3, 4, 5]'"

136 ECHO: "[INFO] run_test(); t10 passed: 'smerge([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1, 2, 3, 4, 5, 6, 7, 8, 9, "a", "b", "c"]'"
137 ECHO: "[ INFO ] run_test(); t11 passed: 'smerge([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0,
           1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]"
138 ECHO: "[ INFO ] run_test(); t01 passed: 'pmerge(undef) = undef'"
139 ECHO: "[ INFO ] run_test(); t02 passed: 'pmerge([])=[]'"
140 ECHO: "[ INFO ] run_test(); t03 passed: 'pmerge([0 : 0.5 : 9])=undef'"
141 ECHO: "[ INFO ] run_test(); t04 passed: 'pmerge(A string)=["A string"]'"
"n", "a", "s"]]'"

144 ECHO: "[ INFO ] run_test(); t07 passed: 'pmerge([undef]) = undef'"
145 ECHO: "[ INFO ] run_test(); t08 passed: 'pmerge([[1, 2], [2, 3]])=[[1, 2], [2, 3]]'"
146 ECHO: "[ INFO ] run_test(); t09 passed: 'pmerge(["ab", [1, 2], [2, 3], [4, 5]))=[["a", 1, 2, 4], ["b", 2,
           3, 5]]'"
147 ECHO: "[ INFO ] run_test(); t10 passed: 'pmerge([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1, 4,
7, "a"], [2, 5, 8, "b"], [3, 6, 9, "c"]]'"

148 ECHO: "[ INFO ] run_test(); tll passed: 'pmerge([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
           151) = undef'
149 ECHO: "[ INFO ] run_test(); t01 passed: 'qsort(undef)=undef'"
150 ECHO: "[ INFO ] run_test(); t02 passed: 'qsort([])=[]'"
151 ECHO: "[ INFO ] run_test(); t03 passed: 'qsort([0:0.5:9])=undef'
152 ECHO: "[ INFO ] run_test(); t04 passed: 'qsort(A string)=undef'
153 ECHO: "[ INFO ] run_test(); t05 passed: 'qsort(["orange", "apple", "grape", "banana"])=["apple", "banana",
           "grape", "orange"]'
154 ECHO: "[ INFO ] run_test(); t06 passed: 'qsort(["b", "a", "n", "a", "n", "a", "s"])=["a", "a", "a", "b",
```

3.2 Math 39

```
"n", "n", "s"]'"
155 ECHO: "[ INFO ] run_test(); t07 passed: 'qsort([undef])=[undef]'"
156 ECHO: "[ INFO ] run_test(); t08 *skip*: 'qsort(Test list 04)'"
157 ECHO: "[ INFO ] run_test(); t09 *skip*: 'qsort(Test list 05)'"
158 ECHO: "[INFO] run_test(); t10 *skip*: 'qsort(Test list 06)'"
159 ECHO: "[INFO] run_test(); t10 *skip*: 'qsort(Test list 06)'"
159 ECHO: "[INFO] run_test(); t11 passed: 'qsort([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]'"
160 ECHO: "[INFO] run_test(); t01 passed: 'qsort_1R(undef)=undef'"
161 ECHO: "[ INFO ] run_test(); t02 passed: 'qsort_1R([])=[]'"
162 ECHO: "[ INFO ] run_test(); t03 passed: 'qsort_1R([0 : 0.5 : 9]) = undef'"
163 ECHO: "[ INFO ] run_test(); t04 passed: 'qsort_1R(A string)=undef'"
164 ECHO: "[ INFO ] run_test(); t05 passed: 'qsort_lR(["orange", "apple", "grape", "banana"])=["orange",
         "grape", "apple", "banana"]'"
165 ECHO: "[ INFO ] run_test(); t06 *skip*: 'qsort_1R(Test list 02)'"
166 ECHO: "[ INFO ] run_test(); t07 *skip*: 'qsort_1R(Test list 03)'"
167 ECHO: "[ INFO ] run_test(); t08 passed: 'qsort_lR([[1, 2], [2, 3]])=[[2, 3], [1, 2]]'"
168 ECHO: "[ INFO ] run_test(); t09 passed: 'qsort_1R(["ab", [1, 2], [2, 3], [4, 5]])=[[4, 5], [2, 3], [1, 2],
        "ab"]'"
169 ECHO: "[ INFO ] run_test(); t10 passed: 'qsort_1R([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[7,
        8, 9], [4, 5, 6], [1, 2, 3], ["a", "b", "c"]]'"
170 ECHO: "[ INFO ] run_test(); t11 *skip*: 'qsort_1R(Vector of integers 0 to 15)'"
171 ECHO: "[ INFO ] run_test(); t01 passed: 'qsort2_1R(undef)=undef'
172 ECHO: "[ INFO ] run_test(); t02 passed: 'qsort2_1R([])=[]'"
173 ECHO: "[ INFO ] run_test(); t03 passed: 'qsort2_1R([0 : 0.5 : 9])=undef'
174 ECHO: "[ INFO ] run_test(); t04 passed: 'qsort2_1R(A string)=undef'"
175 ECHO: "[ INFO ] run_test(); t05 passed: 'qsort2_1R(["orange", "apple", "grape", "banana"])=["orange",
         "grape", "apple", "banana"]'"
176 ECHO: "[ INFO ] run_test(); t06 *skip*: 'qsort2_1R(Test list 02)'
177 ECHO: "[ INFO ] run_test(); t07 *skip*: 'qsort2_1R(Test list 03)'"
178 ECHO: "[ INFO ] run_test(); t08 passed: 'qsort2_1R([[1, 2], [2, 3]])=[[2, 3], [1, 2]]'"
179 ECHO: "[ INFO ] run_test(); t09 passed: 'qsort2_1R(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [4, 5], [2, 3],
        [1, 211'"
180 ECHO: "[INFO] run_test(); t10 passed: 'qsort2_1R([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[["a", "b", "c"], [7, 8, 9], [4, 5, 6], [1, 2, 3]]'"

181 ECHO: "[INFO] run_test(); t11 *skip*: 'qsort2_1R(Vector of integers 0 to 15)'"
182 ECHO: "[ INFO ] run_test(); t01 passed: 'qsort2_HR(undef)=undef'
183 ECHO: "[ INFO ] run_test(); t02 passed: 'qsort2_HR([])=[]'"
184 ECHO: "[ INFO ] run_test(); t03 passed: 'qsort2_HR([0 : 0.5 : 9])=undef'"
185 ECHO: "[ INFO ] run_test(); t04 passed: 'qsort2_HR(A string)=undef'"
186 ECHO: "[ INFO ] run_test(); t05 passed: 'qsort2_HR(["orange", "apple", "grape", "banana"])=["orange",
         "grape", "banana", "apple"]'"
187 ECHO: "[ INFO ] run_test(); t06 passed: 'qsort2_HR(["b", "a", "n", "a", "n", "a", "s"])=["s", "n", "n", "b", "a", "a", "a", "a"]'"
"ab"]'"
191 ECHO: "[ INFO ] run_test(); t10 passed: 'qsort2_HR([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=[["c", "b", "a"], [9, 8, 7], [6, 5, 4], [3, 2, 1]]'"
192 ECHO: "[INFO] run_test(); tl1 passed: 'qsort2_HR([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=[15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]'"
```

3.2 Math

- Vector Algebra
- Bitwise

3.2.1 Vector Algebra

- Script
- Results

3.2.1.1 Script

```
include <math.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>;
use <validation.scad>;
show passing = true; // show passing tests
```

```
show_skipped = true;  // show skipped tests
echo( str("OpenSCAD Version ", version()) );
// test-values columns
test_c =
  ["id", "identifier"],
  ["td", "description"],
["tv", "test value"]
// test-values rows
test_r =
  ["fac", "Function argument count",
                                          undef],
  ["crp", "Result precision", ["t01", "All undefined",
                                           undef],
                                            [undef, undef, undef, undef, undef]],
  ["t02", "All empty lists",
                                            [empty_lst,empty_lst,
  empty_lst,empty_lst,empty_lst,empty_lst]],
                                            [60, 50, 40, 30, 20, 10]],
[[99], [58], [12], [42], [15], [1]]],
  ["t03", "All scalars",
["t04", "All 1d vectors",
["t05", "All 2d vectors",
                                              [99,2], [58,16], [12,43],
                                              [42,13], [15,59], [1,85]
                                            11,
  ["t06", "All 3d vectors",
                                              [199,20,55], [158,116,75], [12,43,90],
                                              [42,13,34], [15,59,45], [62,33,69]
                                            ]],
  ["t07", "All 4d vectors",
                                              [169,27,35,10], [178,016,25,20], [12,43,90,30],
                                              [42,13,34,60], [15,059,45,50], [62,33,69,40]
                                            11,
  ["t08", "Orthogonal vectors",
                                              +x_axis3d_uv, +
  v axis3d uv, +z axis3d uv,
                                              -x axis3d uv. -
  y_axis3d_uv, -z_axis3d_uv,
                                            ]],
  ["t09", "Coplanar vectors",
                                              +x_axis3d_uv, +
 y_axis3d_uv, [2,2,0],
                                              origin3d, origin3d,
 origin3d,
                                            11
];
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 'skip' to skip test
skip = -1; // skip test
[ // function
  ["distance_pp",
    2,
                                                              // fac
                                                              // crp
    undef,
                                                              // t01
    undef,
                                                              // t02
                                                              // t03
    10,
                                                              // t04
    43.3244,
                                                              // t05
    106.2873,
                                                              // t06
    20.0499,
                                                              // t07
                                                              // t08
    1.4142,
                                                              // t09
    1.4142
  ["is_left_ppp",
    3.
                                                              // fac
                                                              // crp
    4,
                                                              // t01
    undef,
    undef,
                                                              // t02
                                                              // t03
    undef,
    undef,
                                                              // t04
                                                              // t05
    -463,
    17009.
                                                              // t06
                                                              // t07
    -1583.
```

3.2 Math 41

```
// t08
  -3
                                                                  // t09
["dimension_2to3_v",
  1,
                                                                  // fac
                                                                  // crp
// t01
  [undef,undef,0],
  [undef,undef,0],
                                                                  // t02
  [undef,undef,0],
                                                                  // t03
                                                                  // t04
  [99, undef, 0],
  [99,2,0],
                                                                  // t05
                                                                  // t06
  [199,20,55],
  [169,27,0],
                                                                  // t07
                                                                  // t08
// t09
  x_axis3d_uv,
  x_axis3d_uv
["get_line_dim",
                                                                  // fac
// crp
// t01
  2,
  4,
  2,
                                                                  // t02
  0,
  2,
                                                                  // t03
                                                                  // t04
  1,
  2,
                                                                  // t05
                                                                  // t06
  3.
                                                                  // t07
  4,
                                                                  // t08
// t09
  3.
  3
["get_line_tp",
  2,
                                                                  // fac
                                                                  // crp
  4.
                                                                  // t01
// t02
  [undef,undef],
  empty_lst,
[60,50],
                                                                  // t03
// t04
  [58],
                                                                  // t04
// t05
// t06
// t07
// t08
  [58,16],
  [158, 116, 75],
  [178,16,25,20],
  y_axis3d_uv,
  y_axis3d_uv
                                                                  // t09
],
["get_line_ip",
                                                                  // fac
  2,
                                                                  // crp
// t01
// t02
// t03
  4,
  origin2d,
  empty_lst,
  origin2d,
                                                                  // t04
// t05
  [99],
  [99,2],
                                                                  // t06
// t07
  [199,20,55],
  [169,27,35,10],
  x_axis3d_uv,
                                                                  // t08
                                                                  // t09
  x_axis3d_uv
["get_line2origin",
  2,
                                                                  // fac
                                                                  // crp
                                                                  // t01
  [undef, undef],
  empty_lst,
                                                                  // t02
  [60,50],
                                                                  // t03
                                                                  // t04
  [-41],
                                                                  // t05
  [-41,14],
  [-41,96,20],
                                                                  // t06
                                                                  // t07
// t08
  [9,-11,-10,10],
  [-1, 1, 0],
  [-1,1,0]
                                                                  // t09
["dot_11",
  4,
                                                                  // fac
  4,
                                                                  // crp
  undef,
                                                                  // t01
                                                                  // t02
// t03
// t04
  undef,
  3900,
  -1230,
  -1650,
                                                                  // t05
                                                                  // t06
  -5230,
                                                                  // t07
  1460,
                                                                  // t08
 1,
  0
                                                                  // t09
],
```

```
["cross_ll",
  4,
                                                                        // fac
                                                                        // crp
// t01
  4,
  skip,
  skip,
                                                                        // t02
                                                                       // t03
// t04
  skip,
  skip,
  810,
                                                                       // t05
   [-4776, -1696, -1650],
                                                                       // t06
  skip, [-1,-1,1],
                                                                        // t07
                                                                       // t08
// t09
  [0,0,4]
],
["striple_lll",
  6,
                                                                        // fac
  4,
                                                                        // crp
                                                                       // t01
// t02
  skip,
  skip,
                                                                        // t03
  skip,
                                                                        // t04
  skip,
  [-14760,5040],
                                                                        // t05
  -219976,
                                                                       // t06
                                                                       // t07
// t08
  skip,
  -2,
                                                                       // t09
  0
["angle_ll",
                                                                       // fac
  4,
                                                                       // crp
// t01
  4,
  undef,
  undef, -2.9357,
                                                                       // t02
                                                                       // t03
                                                                       // t04
// t05
  undef.
  153.8532,
                                                                       // t06
// t07
  134.4573,
  undef,
                                                                       // t08
// t09
  60,
  90
],
["angle_lll",
                                                                       // fac
// crp
// t01
// t02
  6,
  4,
  skip,
  skip,
                                                                       // t02
// t03
// t04
// t05
// t06
  skip,
  skip,
  skip,
  -91.362,
                                                                       // t07
// t08
  skip,
  -63.4349,
  0
                                                                        // t09
["unit_l",
                                                                       // fac
  2,
                                                                       // crp
// t01
  4,
  undef,
                                                                       // t02
// t03
  undef,
   [.7682,0.6402],
                                                                       // t04
   [-1],
   [-0.9464,0.3231],
                                                                       // t05
  [-0.3857,0.9032,0.1882],
[0.44888,-0.5486,-0.4988,0.4988],
                                                                       // t06
                                                                       // t07
                                                                        // t08
   [-0.7071,0.7071,0],
  [-0.7071, 0.7071, 0]
                                                                       // t09
["are_coplanar_lll",
  6,
                                                                        // fac
                                                                       // crp
// t01
  4,
  skip,
                                                                       // t01
// t02
// t03
  skip,
  skip,
  skip,
                                                                        // t04
  skip,
                                                                        // t05
                                                                       // t06
// t07
  false,
  skip,
                                                                       // t08
// t09
  false.
  true
["get_pnorm2nv",
                                                                       // fac
// crp
  2,
  4,
```

3.2 Math 43

```
// t01
       skip,
       skip,
                                                                                                   // t02
       [60,50,0],
                                                                                                   // t03
       skip,
                                                                                                   // t04
       [0,0,1468],
                                                                                                   // t05
       [-4880,-6235,19924],
                                                                                                   // t06
       skip,
                                                                                                   // t07
       z_axis3d_uv,
                                                                                                   // t08
       z_axis3d_uv
                                                                                                   // t09
   ]
1:
// sanity-test tables
table_check( test_r, test_c, false );
table_check( good_r, good_c, false );
// validate helper function and module
function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
function qv( vid, e ) = get_value( vid )[e];
module run( fname, vid )
   value_text = get_table_v(test_r, test_c, vid, "td");
   if ( get_table_v(good_r, good_c, fname, vid) != skip )
      children();
   else if ( show skipped )
      log_info( str("*skip*: ", vid, " '", fname, "(", value_text, ")'") );
module test (fname, fresult, vid, pair )
   value_text = get_table_v(test_r, test_c, vid, "td");
   fname_argc = get_table_v(good_r, good_c, fname, "fac");
comp_prcsn = get_table_v(good_r, good_c, fname, "crp");
   pass_value = get_table_v(good_r, good_c, fname, vid);
   test_pass = validate(cv=fresult, t="almost", ev=pass_value, p=comp_prcsn, pf=true);
   j=false, l=false));
   test_text = validate(str(fname, "(", farg_text, ")=~", pass_value), fresult, "almost",
   pass_value, comp_prcsn);
   if ( pass_value != skip )
      if ( !test_pass )
          log_warn( str(vid, "(", value_text, ") ", test_text) );
      else if ( show_passing )
  log_info( str(vid, " ", test_text) );
   else if ( show_skipped )
      log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
// Indirect function calls would be very useful here!!!
run_ids = delete( test_ids, mv=["fac", "crp"] );
// group 1: point
for (vid=run_ids) run("distance_pp",vid) test( "distance_pp", distance_pp(gv(vid,0),gv(vid,1)
   )), vid, false );
for (vid=run_ids) run("is_left_ppp", vid) test( "is_left_ppp", is_left_ppp(gv(vid,0),gv(vid,1
  ),gv(vid,2)), vid, false);
for (vid=run_ids) run("dimension_2to3_v", vid) test( "dimension_2to3_v",
   dimension_2to3_v(gv(vid,0)), vid, false );
// group 3: line (or vector)
for (vid=run_ids) run("get_line_dim",vid) test( "get_line_dim", get_line_dim([gv(vid,0),gv(
  vid, 1)]), vid, true);
for (vid=run_ids) run("get_line_tp", vid) test( "get_line_tp", get_line_tp([gv(vid,0),gv(vid,
   1) 1), vid, true );
for (vid=run_ids) run("get_line_ip",vid) test( "get_line_ip", get_line_ip([gv(vid,0),gv(vid,
  1)1), vid, true );
for (vid=run_ids) run("get_line2origin",vid) test( "get_line2origin", get_line2origin([
   gv(vid,0),gv(vid,1)]), vid, true);
for (vid=run_ids) run("dot_ll",vid) test( "dot_ll", dot_ll([gv(vid,0),gv(vid,1)],[gv(vid,2),gv(
   vid, 3) 1), vid, true );
for \ (vid=run\_ids) \ run("cross\_11",vid) \ test( "cross\_11", cross\_11([gv(vid,0),gv(vid,1)],[gv(vid,0),gv(vid,1)]), \\ for \ (vid=run\_ids) \ run("cross\_11",vid) \ test( "cross\_11",vid) \ for \ (vid=run\_ids) \ run("cross\_11",vid) \ test( "cross\_11",vid) \ for \ (vid=run\_ids) \ run("cross\_11",vid) \ for \ (vid=run\_ids) \ for \ (
   2), gv(vid, 3)]), vid, true);
```

```
for (vid=run_ids) run("striple_lll",vid) test( "striple_lll", striple_lll([gv(vid,0),gv(vid,
    1)],[gv(vid,2),gv(vid,3)],[gv(vid,4),gv(vid,5)]), vid, true );
for (vid=run_ids) run("angle_ll",vid) test( "angle_ll", angle_ll([gv(vid,0),gv(vid,1)],[gv(vid,
    2),gv(vid,3)]), vid, true );
for (vid=run_ids) run("angle_lll",vid) test( "angle_lll", angle_lll([gv(vid,0),gv(vid,1)],[gv(vid,2),gv(vid,3)],[gv(vid,4),gv(vid,5)]), vid, true );
for (vid=run_ids) run("unit_l",vid) test( "unit_l", unit_l([gv(vid,0),gv(vid,1)]), vid, true );
for (vid=run_ids) run("are_coplanar_lll",vid) test( "are_coplanar_lll",
    are_coplanar_lll([gv(vid,0),gv(vid,1)],[gv(vid,2),gv(vid,3)],[gv(vid,4),gv(vid,5)]), vid, true );

// group 4: plane and pnorm
for (vid=run_ids) run("get_pnorm2nv",vid) test( "get_pnorm2nv", get_pnorm2nv([gv(vid,0),gv(vid,1)]), vid, true );

// end-of-tests
```

3.2.1.2 Results

```
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[ INFO ] run(): test(); t01 passed: 'distance_pp(undef, undef)=~undef'"
3 ECHO: "[ INFO ] run(): test(); t02 passed: 'distance_pp([], [])=~undef'"
                        run(): test(); t03 passed: 'distance_pp(60, 50)=~10'"
4 ECHO: "[ INFO ]
5 ECHO: "[ INFO ] run(): test(); t03 passed: distance_pp(09), [58])=~41'"
6 ECHO: "[ INFO ] run(): test(); t05 passed: 'distance_pp([99], [58])=~41'"
6 ECHO: "[ INFO ] run(): test(); t05 passed: 'distance_pp([99, 2], [58, 16])=~43.3244'"
7 ECHO: "[ INFO ] run(): test(); t06 passed: 'distance_pp([199, 20, 55], [158, 116, 75])=~106.287'"
8 ECHO: "[ INFO ] run(): test(); t07 passed: 'distance_pp([169, 27, 35, 10], [178, 16, 25, 20])=~20.0499'"
9 ECHO: "[ INFO ] run(): test(); t08 passed: 'distance_pp([1, 0, 0], [0, 1, 0])=~1.4142'"
10 ECHO: "[ INFO ] run(): test(); t09 passed: 'distance_pp([1, 0, 0], [0, 1, 0])=~1.4142'"
11 ECHO: "[ INFO ] run(): test(); t01 passed: 'is_left_ppp(undef, undef) =~undef'"
12 ECHO: "[ INFO ] run(): test(); t02 passed: 'is_left_ppp([], [], [])=~undef'"
13 ECHO: "[ INFO ] run(): test(); t03 passed: 'is_left_ppp(60, 50, 40)=~undef'"
14 ECHO: "[ INFO ] run(): test(); t04 passed: 'is_left_ppp([99], [58], [12])=~undef'"

15 ECHO: "[ INFO ] run(): test(); t05 passed: 'is_left_ppp([99, 2], [58, 16], [12, 43])=~-463'"
16 ECHO: "[INFO] run(): test(); t06 passed: 'is_left_ppp([199, 20, 55], [158, 116, 75], [12, 43, 90])=~17009'"
17 ECHO: "[ INFO ] run(): test(); t07 passed: 'is_left_ppp([169, 27, 35, 10], [178, 16, 25, 20], [12, 43, 90, 30])=~-1583'"
18 ECHO: "[ INFO ] run(): test(); t08 passed: 'is_left_ppp([1, 0, 0], [0, 1, 0], [0, 0, 1])=~1'"
19 ECHO: "[ INFO ] run(): test(); t09 passed: 'is_left_ppp([1, 0, 0], [0, 1, 0], [2, 2, 0])=~-3'"
19 ECHO: "[ INFO ] run(): test(); to9 passed: 'Is_lett_ppp([1, 0, 0], [0, 1, 0], [2, 2, 0])
20 ECHO: "[ INFO ] run(): test(); t01 passed: 'dimension_2to3_v(undef) =~ [undef, undef, 0]'"
21 ECHO: "[ INFO ] run(): test(); t02 passed: 'dimension_2to3_v([]) =~ [undef, undef, 0]'"
22 ECHO: "[ INFO ] run(): test(); t03 passed: 'dimension_2to3_v(60)=~[undef, undef, 0]'"
23 ECHO: "[ INFO ] run(): test(); t04 passed: 'dimension_2to3_v([99])=~[99, undef, 0]'
24 ECHO: "[INFO] run(): test(); t05 passed: 'dimension_2to3_v([99, 2])=~[99, 2, 0]'"
25 ECHO: "[INFO] run(): test(); t06 passed: 'dimension_2to3_v([199, 20, 55])=~[199, 20, 55]'"
26 ECHO: "[ INFO ] run(): test(); t07 passed: 'dimension_2to3_v([169, 27, 35, 10])=~[169, 27, 0]'"
27 ECHO: "[ INFO ] run(): test(); t08 passed: 'dimension_2to3_v([1, 0, 0])=\sim[1, 0, 0]'" 28 ECHO: "[ INFO ] run(): test(); t09 passed: 'dimension_2to3_v([1, 0, 0])=\sim[1, 0, 0]'"
29 ECHO: "[ INFO ] run(): test(); t01 passed: 'get_line_dim([undef, undef]) = \sim 2'"
30 ECHO: "[ INFO ] run(): test(); t02 passed: 'get_line_dim([[], []])=~0'"
31 ECHO: "[ INFO ] run(): test(); t03 passed: 'get_line_dim([60, 50])=~2'"
32 ECHO: "[INFO] run(): test(); t04 passed: 'get_line_dim([[99], [58]])=~1'"
33 ECHO: "[INFO] run(): test(); t05 passed: 'get_line_dim([[99, 2], [58, 16]])=~2'"
34 ECHO: "[ INFO ] run(): test(); t06 passed: 'get_line_dim([[199, 20, 55], [158, 116, 75]])=~3'"
35 ECHO: "[ INFO ] run(): test(); t07 passed: 'get_line_dim([[169, 27, 35, 10], [178, 16, 25, 20]])=~4'"
36 ECHO: "[ INFO ] run(): test(); t08 passed: 'get_line_dim([[1, 0, 0], [0, 1, 0]])=~3'"
37 ECHO: "[ INFO ] run(): test(); t09 passed: 'get_line_dim([[1, 0, 0], [0, 1, 0]])=~3'"
38 ECHO: "[ INFO ] run(): test(); t01 passed: 'get_line_tp([undef, undef]) =~ [undef, undef]'"
39 ECHO: "[ INFO ] run(): test(); t02 passed: 'get_line_tp([[], []])=~[]'"
40 ECHO: "[ INFO ] run(): test(); t03 passed: 'get_line_tp([60, 50])=~[60, 50]'"
41 ECHO: "[ INFO ] run(): test(); t04 passed: 'get_line_tp([[99], [58]])=~[58]'"
42 ECHO: "[ INFO ] run(): test(); t05 passed: 'get_line_tp([[99, 2], [58, 16]])=~[58, 16]'"
43 ECHO: "[ INFO ] run(): test(); t06 passed: 'get_line_tp([[199, 20, 55], [158, 116, 75]])=~[158, 116, 75]'"
44 ECHO: "[ INFO ] run(): test(); t07 passed: 'get_line_tp([[169, 27, 35, 10], [178, 16, 25, 20]])=~[178, 16,
         25, 20]'"
45 ECHO: "[ INFO ] run(): test(); t08 passed: 'get_line_tp([[1, 0, 0], [0, 1, 0]])=~[0, 1, 0]'"
46 ECHO: "[ INFO ] run(): test(); t09 passed: 'get_line_tp([[1, 0, 0], [0, 1, 0]])=~[0, 1, 0]'"
47 ECHO: "[ INFO ] run(): test(); t01 passed: 'get_line_ip([undef, undef]) =~[0, 0]'"
48 ECHO: "[ INFO ] run(): test(); t02 passed: 'get_line_ip([[], []])=~[]'
49 ECHO: "[ INFO ] run(): test(); t03 passed: 'qet_line_ip([60, 50])=~[0, 0]'"
50 ECHO: "[ INFO ] run(): test(); t04 passed: 'get_line_ip([[99], [58]])=~[99]'"
51 ECHO: "[ INFO ] run(): test(); t05 passed: 'get_line_ip([[99, 2], [58, 16]])=~[99, 2]'"
52 ECHO: "[ INFO ] run(): test(); t06 passed: 'get_line_ip([[199, 20, 55], [158, 116, 75]])=~[199, 20, 55]'"
53 ECHO: "[ INFO ] run(): test(); t07 passed: 'get_line_ip([[169, 27, 35, 10], [178, 16, 25, 20]])=~[169, 27,
         35, 10]'"
54 ECHO: "[ INFO ] run(): test(); t08 passed: 'get_line_ip([[1, 0, 0], [0, 1, 0]])=~[1, 0, 0]'
55 ECHO: "[ INFO ] run(): test(); t09 passed: 'get_line_ip([[1, 0, 0], [0, 1, 0]])-~[1, 0, 0]'"
56 ECHO: "[ INFO ] run(): test(); t01 passed: 'get_line2origin([undef, undef])-~[undef, undef]'"
57 ECHO: "[ INFO ] run(): test(); t02 passed: 'get_line2origin([[], []])=~[]''
```

3.2 Math 45

```
58 ECHO: "[ INFO ] run(): test(); t03 passed: 'get_line2origin([60, 50])=~[60, 50]'"
59 ECHO: "[ INFO ] run(): test(); t04 passed: 'get_line2origin([[99], [58]])=~[-41]'"
60 ECHO: "[ INFO ] run(): test(); t05 passed: 'get_line2origin([[99, 2], [58, 16]])=~[-41, 14]'"
61 ECHO: "[ INFO ] run(): test(); t06 passed: 'get_line2origin([[199, 20, 55], [158, 116, 75]])=~[-41, 96,
         201'"
62 ECHO: "[ INFO ] run(): test(); t07 passed: 'get_line2origin([[169, 27, 35, 10], [178, 16, 25, 20]])=~[9,
         -11, -10, 10]'"
63 ECHO: "[ INFO ] run(): test(); t08 passed: 'get_line2origin([[1, 0, 0], [0, 1, 0]])=~[-1, 1, 0]'"
64 ECHO: "[ INFO ] run(): test(); t09 passed: 'get_line2origin([[1, 0, 0], [0, 1, 0]])=~[-1, 1, 0]'"
65 ECHO: "[ INFO ] run(): test(); t01 passed: 'dot_ll([undef, undef], [undef, undef])=~undef'"
66 ECHO: "[ INFO ] run(): test(); t02 passed: 'dot_ll([[], []], [[], []])=~undef'"
67 ECHO: "[ INFO ] run(): test(); t03 passed: 'dot_l1([60, 50], [40, 30])=~3900'"
68 ECHO: "[ INFO ] run(): test(); t04 passed: 'dot_l1([[99], [58]], [[12], [42]])=~-1230'"
69 ECHO: "[ INFO ] run(): test(); t05 passed: 'dot_l1([[99, 2], [58, 16]], [[12, 43], [42, 13]])=~-1650'"
70 ECHO: "[ INFO ] run(): test(); t06 passed: 'dot_l1([[199, 20, 55], [158, 116, 75]], [[12, 43, 90], [42, 13,
        34]])=~-5230'"
71 ECHO: "[ INFO ] run(): test(); t07 passed: 'dot_11([[169, 27, 35, 10], [178, 16, 25, 20]], [[12, 43, 90,
        30], [42, 13, 34, 60]])=~1460'"
72 ECHO: "[ INFO ] run(): test(); t08 passed: 'dot_l1([[1, 0, 0], [0, 1, 0]], [[0, 0, 1], [-1, 0, 0]])=~1'"
73 ECHO: "[ INFO ] run(): test(); t09 passed: 'dot_l1([[1, 0, 0], [0, 1, 0]], [[2, 2, 0], [0, 0, 0]])=~0'"
74 ECHO: "[ INFO ] run(); *skip*: t01 'cross_l1(All undefined)'"
75 ECHO: "[ INFO ] run(); *skip*: t02 'cross_11(All empty lists)'"
76 ECHO: "[ INFO ] run(); *skip*: t03 'cross_11(All scalars)'
77 ECHO: "[ INFO ] run(); *skip*: t04 'cross_11(All 1d vectors)'"
78 ECHO: "[ INFO ] run(): test(); t05 passed: 'cross_ll([[99, 2], [58, 16]], [[12, 43], [42, 13]])=~810'"
79 ECHO: "[ INFO ] run(): test(); t06 passed: 'cross_ll([[199, 20, 55], [158, 116, 75]], [[12, 43, 90], [42,
        13, 34]])=~[-4776, -1696, -1650]'"
80 ECHO: "[ INFO ] run(); *skip*: t07 'cross_11(All 4d vectors)'"
81 ECHO: "[ INFO ] run(): test(); t08 passed: 'cross_11([[1, 0, 0], [0, 1, 0]], [[0, 0, 1], [-1, 0, 0]])=~[-1,
        -1, 1]'"
82 ECHO: "[ INFO ] run(): test(); t09 passed: 'cross_ll([[1, 0, 0], [0, 1, 0]], [[2, 2, 0], [0, 0, 0]])=~[0,
        0, 4]'"
83 ECHO: "[ INFO ] run(); *skip*: t01 'striple_lll(All undefined)'"
84 ECHO: "[ INFO ] run(); *skip*: t02 'striple_ll1(All empty lists)'"
85 ECHO: "[ INFO ] run(); *skip*: t03 'striple_lll(All scalars)'
86 ECHO: "[ INFO ] run(); *skip*: t04 'striple_lll(All 1d vectors)'"
87 ECHO: "[ INFO ] run(): test(); t05 passed: 'striple_lll([[99, 2], [58, 16]], [[12, 43], [42, 13]], [[15, 59], [1, 85]])=~[-14760, 5040]'"
88 ECHO: "[ INFO ] run(): test(); t06 passed: 'striple_lll([[199, 20, 55], [158, 116, 75]], [[12, 43, 90], [42, 13, 34]], [[15, 59, 45], [62, 33, 69]])=~-219976'"
89 ECHO: "[ INFO ] run(); *skip*: t07 'striple_lll(All 4d vectors)'"
90 ECHO: "[ INFO ] run(): test(); t08 passed: 'striple_lll([[1, 0, 0], [0, 1, 0]], [[0, 0, 1], [-1, 0, 0]],
         [[0, -1, 0], [0, 0, -1]])=~-2'"
91 ECHO: "[ INFO ] run(): test(); t09 passed: 'striple_lll([[1, 0, 0], [0, 1, 0]], [[2, 2, 0], [0, 0, 0]],
         [[0, 0, 0], [0, 0, 0]])=~0'"
92 ECHO: "[ INFO ] run(): test(); t01 passed: 'angle_ll([undef, undef], [undef, undef]) =~undef'"
93 ECHO: "[ INFO ] run(): test(); t02 passed: 'angle_ll([[], []], [[], []]) =~undef'"
94 ECHO: "[ INFO ] run(): test(); t03 passed: 'angle_ll([60, 50], [40, 30]) =~-2.9357'"
95 ECHO: "[ INFO ] run(): test(); t04 passed: 'angle_l1([[99], [58]], [[12], [42]])=~undef'"
96 ECHO: "[ INFO ] run(): test(); t05 passed: 'angle_l1([[99, 2], [58, 16]], [[12, 43], [42, 13]])=~153.853'"
97 ECHO: "[ INFO ] run(): test(); t06 passed: 'angle_l1([[199, 20, 55], [158, 116, 75]], [[12, 43, 90], [42,
        13, 34]])=~134.457'"
98 ECHO: "[ INFO ] run(): test(); t07 passed: 'angle_l1([[169, 27, 35, 10], [178, 16, 25, 20]], [[12, 43, 90,
         30], [42, 13, 34, 60]])=~undef'"
99 ECHO: "[ INFO ] run(): test(); t08 passed: 'angle_ll([[1, 0, 0], [0, 1, 0]], [[0, 0, 1], [-1, 0, 0]])=~60'"
100 ECHO: "[ INFO ] run(): test(); t09 passed: 'angle_ll([[1, 0, 0], [0, 1, 0]], [[2, 2, 0], [0, 0, 0]])=~90'"
101 ECHO: "[ INFO ] run(); *skip*: t01 'angle_lll(All undefined)'"
102 ECHO: "[ INFO ] run(); *skip*: t02 'angle_111(All empty lists)'
103 ECHO: "[ INFO ] run(); *skip*: t03 'angle_111(All scalars)'"
104 ECHO: "[ INFO ] run(); *skip*: t04 'angle_lll(All 1d vectors)'"
105 ECHO: "[ INFO ] run(); *skip*: t05 'angle_111(All 2d vectors)'"
106 ECHO: "[ INFO ] run(): test(); t06 passed: 'angle_lll([[199, 20, 55], [158, 116, 75]], [[12, 43, 90], [42,
        13, 34]], [[15, 59, 45], [62, 33, 69]])=~-91.362'"
107 ECHO: "[ INFO ] run(); *skip*: t07 'angle_lll(All 4d vectors)'"
108 ECHO: "[ INFO ] run(): test(); t08 passed: 'angle_lll([[1, 0, 0], [0, 1, 0]], [[0, 0, 1], [-1, 0, 0]], [[0,
         -1, 0], [0, 0, -1]])=~-63.4349'"
109 ECHO: "[ INFO ] run(): test(); t09 passed: 'angle_lll([[1, 0, 0], [0, 1, 0]], [[2, 2, 0], [0, 0, 0]], [[0,
         [0, 0], [0, 0, 0]]) = \sim 0'
110 ECHO: "[ INFO ] run(): test(); t01 passed: 'unit_1([undef, undef]) =~undef'"
111 ECHO: "[ INFO ] run(): test(); t02 passed: 'unit_1([[], []])=~undef'"
112 ECHO: "[ INFO ] run(): test(); t03 passed: 'unit_1([60, 50]) =~ [0.7682, 0.6402]'"
113 ECHO: "[ INFO ] run(): test(); t04 passed: 'unit_1([[99], [58]])=~[-1]'"
114 ECHO: "[ INFO ] run(): test(); t05 passed: 'unit_1([[99, 2], [58, 16]])=~[-0.9464, 0.3231]'"
115 ECHO: "[ INFO ] run(): test(); t06 passed: 'unit_1([[199, 20, 55], [158, 116, 75]])=~[-0.3857, 0.9032,
         0.1882]'"
116 ECHO: "[INFO] run(): test(); t07 passed: 'unit_1([[169, 27, 35, 10], [178, 16, 25, 20]])=~[0.44888, -0.5486, -0.4988, 0.4988]'"
117 ECHO: "[ INFO ] run(): test(); t08 passed: 'unit_1([[1, 0, 0], [0, 1, 0]])=~[-0.7071, 0.7071, 0]'"
118 ECHO: "[ INFO ] run(): test(); t09 passed: 'unit_1([[1, 0, 0], [0, 1, 0]])=~[-0.7071, 0.7071, 0]'"
119 ECHO: "[ INFO ] run(); *skip*: t01 'are_coplanar_lll(All undefined)'"
120 ECHO: "[ INFO ] run(); *skip*: t02 'are_coplanar_lll(All empty lists)'"
```

```
121 ECHO: "[ INFO ] run(); *skip*: t03 'are_coplanar_lll(All scalars)'"
122 ECHO: "[ INFO ] run(); *skip*: t04 'are_coplanar_lll(All 1d vectors)'"
123 ECHO: "[ INFO ] run(); *skip*: t05 'are_coplanar_111(All 2d vectors)'"
124 ECHO: "[ INFO ] run(): test(); t06 passed: 'are_coplanar_lll([[199, 20, 55], [158, 116, 75]], [[12, 43,
         90], [42, 13, 34]], [[15, 59, 45], [62, 33, 69]])=~false'"
125 ECHO: "[ INFO ] run(); *skip*: t07 'are_coplanar_lll(All 4d vectors)'"
126 ECHO: "[ INFO ] run(): test(); t08 passed: 'are_coplanar_lll([[1, 0, 0], [0, 1, 0]], [[0, 0, 1], [-1, 0,
0]], [[0, -1, 0], [0, 0, -1]])=-false'"

127 ECHO: "[ INFO ] run(): test(); t09 passed: 'are_coplanar_lll([[1, 0, 0], [0, 1, 0]], [[2, 2, 0], [0, 0,
         0]], [[0, 0, 0], [0, 0, 0]])=~true'"
128 ECHO: "[ INFO ] run(); *skip*: t01 'get_pnorm2nv(All undefined)'"
129 ECHO: "[ INFO ] run(); *skip*: t02 'get_pnorm2nv(All empty lists)'"
130 ECHO: "[ INFO ] run(): test(); t03 passed: 'get_pnorm2nv([60, 50])=~[60, 50, 0]'"
131 ECHO: "[ INFO ] run(); *skip*: t04 'get_pnorm2nv(All ld vectors)'"
132 ECHO: "[ INFO ] run(): test(); t05 passed: 'get_pnorm2nv([[99, 2], [58, 16]])=~[0, 0, 1468]'"
133 ECHO: "[ INFO ] run(): test(); t06 passed: 'get_pnorm2nv([[199, 20, 55], [158, 116, 75]])=~[-4880, -6235,
        19924]'"
134 ECHO: "[ INFO ] run(); *skip*: t07 'get_pnorm2nv(All 4d vectors)'"
135 ECHO: "[ INFO ] run(): test(); t08 passed: 'get_pnorm2nv([[1, 0, 0], [0, 1, 0]])=~[0, 0, 1]'"
136 ECHO: "[ INFO ] run(): test(); t09 passed: 'get_pnorm2nv([[1, 0, 0], [0, 1, 0]])=~[0, 0, 1]'"
```

3.2.2 Bitwise

- Script
- Results

3.2.2.1 Script

```
include <math/math_bitwise.scad>;
use <datatypes/datatypes_table.scad>;
use <console.scad>:
use <validation.scad>;
show_passing = true;
                         // show passing tests
show_skipped = true;
                         // show skipped tests
echo( str("OpenSCAD Version ", version()) );
// test-values columns
  ["id", "identifier"],
  ["td", "description"],
["tv", "test value"]
// test-values rows
test r =
  ["fac", "Function argument count",
  ["t01", "All undefined",
                                          [undef,undef]],
  ["t02", "All empty lists",
                                          [empty_lst,empty_lst]],
  ["t03", "test value 1",
                                          [254, 01],
  ["t04", "test value 2", ["t05", "test value 3",
                                         [254, 1]],
                                          [255, 011,
  ["t06", "test value 4",
                                         [0, 255]],
  ["t07", "test value 5",
                                          [126, 63]],
  ["t08", "test value 6",
                                         [25, 10]],
  ["t09", "test value 7",
                                          [1024, 512]],
  ["t10", "test value 8",
                                          [4253, 315]],
  ["t11", "test value 9",
                                          [835, 769]],
  ["t12", "test value 10",
                                          [856, 625]]
test_ids = get_table_ridl( test_r );
// expected columns: ("id" + one column for each test)
good_c = pmerge([concat("id", test_ids), concat("identifier", test_ids)]);
// expected rows: ("golden" test results), use 'skip' to skip test
skip = -1; // skip test
good_r =
[ // function
  ["bitwise_is_equal_0", 2,
```

3.2 Math 47

```
false,
                                           // t01
                                           // t02
// t03
  false,
  true,
                                           // t04
  false,
  false,
                                           // t05
  true,
                                           // t06
                                          // t07
// t08
  true,
  true,
  true,
                                          // t09
                                          // t10
  true,
  true,
                                           // t11
                                           // t12
  true
["bitwise_is_equal_1", 2,
  false,
                                           // t01
                                           // t02
                                          // t03
// t04
  false,
  true,
                                          // t05
// t06
  true,
  false,
                                          // t07
// t08
  false,
  false,
                                          // t09
  false,
  false,
                                          // t10
                                          // t11
// t12
  false,
  false
["bitwise_i2v", 1,
  undef,
                                           // t01
                                          // t02
  undef,
  [1,1,1,1,1,1,0],
[1,1,1,1,1,1,1,0],
                                          // t03
                                          // t04
                                         // t05
// t06
  [1,1,1,1,1,1,1,1],
  [0],
[1,1,1,1,1,1,0],
                                          // t07
// t08
  [1,1,0,0,1],
  [1,1,0,0,1],
[1,0,0,0,0,0,0,0,0,0],
[1,0,0,0,0,1,0,0,1,1,1,0,1],
[1,1,0,1,0,0,0,0,1,1],
[1,1,0,1,0,1,1,0,0,0]
                                          // t09
// t10
                                           // t11
                                           // t12
],
["bitwise_i2v_v2i", 1,
                                           // t01
// t02
  undef,
  undef,
                                          // t02
// t03
// t04
// t05
// t06
  254,
  254.
  255,
  Ο,
                                          // t07
// t08
// t09
  126,
  25,
  1024.
                                          // t10
  4253,
  835,
                                          // t11
                                          // t12
  856
],
["bitwise_i2s", 1,
  undef,
                                          // t01
                                          // t02
// t03
  undef,
  "11111110",
  "11111110",
                                          // t04
                                          // t05
  "11111111",
  "0",
"1111110",
                                         // t06
// t07
  "11001",
                                          // t08
  "10000000000",
                                          // t09
  "1000010011101",
                                          // t10
  "1101000011",
                                          // t11
  "1101011000"
                                          // t12
["bitwise_i2s_s2i", 1,
  undef,
                                           // t01
                                          // t02
// t03
  undef,
  254,
                                          // t04
// t05
  254,
  255,
                                           // t06
// t07
  0.
  126,
                                           // t08
  25,
  1024,
                                           // t09
  4253.
                                           // t10
                                           // t11
  835.
```

```
// t12
   856
["bitwise_imi_32", 1,
                                                    // t01
  undef,
                                                   // t01
// t02
// t03
// t04
// t05
   undef,
   7,
   7,
                                                   // t06
// t07
   0,
   7,
                                                    // t08
// t09
   0,
                                                   // t10
// t11
   Ο,
   6
                                                    // t12
["bitwise_and", 2,
                                                    // t01
// t02
// t03
  undef,
  undef,
   Ο,
                                                    // t04
// t05
   Ο,
   Ο,
                                                   // t06
// t07
   0,
   62,
                                                   // t08
// t09
   8,
   0,
                                                   // t10
// t11
   25,
   769,
                                                    // t12
  592
],
["bitwise_or", 2,
  undef,
                                                    // t01
                                                   // t02
// t03
   undef,
  254,
                                                   // t04
// t05
   255,
  255,
                                                   // t05
// t06
// t07
// t08
// t09
   255,
   127,
  27,
1536,
                                                   // t10
// t11
// t12
   4543,
   835,
   889
["bitwise_xor", 2,
                                                   // t01
// t02
// t03
// t04
// t05
// t06
// t07
// t08
// t09
// t10
// t11
// t12
  undef,
   undef,
  254,
  255,
   255,
   255,
   65,
   19,
   1536,
   4518,
   66,
   297
],
["bitwise_not", 1,
  undef,
                                                    // t01
                                                    // t02
   undef,
                                                    // t03
// t04
   1,
                                                   // t05
// t06
   Ο,
   1,
                                                   // t07
// t08
   1,
                                                   // t09
// t10
   1023,
   3938,
                                                   // t11
// t12
   188,
  167
],
["bitwise_lsh", 1,
                                                    // t01
// t02
  undef,
   undef,
                                                    // t03
// t04
   508,
   508,
                                                    // t05
// t06
   510,
   Ο,
                                                    // t07
// t08
   252,
   50,
```

3.2 Math 49

```
2048,
                                      // t09
    8506,
                                      // t10
    1670.
                                      // t11
    1712
                                      // t12
  ["bitwise_rsh", 1,
    undef,
                                      // t01
                                      // t02
    undef,
                                      // t03
    127.
                                      // t04
    127,
    127,
                                      // t05
                                      // t06
    0,
                                      // t07
    63,
                                      // t08
    12.
    512.
                                      // t09
    2126.
                                      // t10
    417,
                                      // t11
                                      // t12
    428
  ]
];
// sanity-test tables
table_check( test_r, test_c, false );
table_check( good_r, good_c, false );
// validate helper function and module
function get_value( vid ) = get_table_v(test_r, test_c, vid, "tv");
function gv( vid, e ) = get_value( vid )[e];
module run( fname, vid )
  value_text = get_table_v(test_r, test_c, vid, "td");
  if ( get_table_v(good_r, good_c, fname, vid) != skip )
   children();
  else if ( show_skipped )
   log_info( str("*skip*: ", vid, " '", fname, "(", value_text, ")'") );
module test (fname, fresult, vid )
  value_text = get_table_v(test_r, test_c, vid, "td");
  fname_argc = get_table_v(good_r, good_c, fname, "fac");
  pass_value = get_table_v(good_r, good_c, fname, vid);
  test_pass = validate(cv=fresult, t="equals", ev=pass_value, pf=true);
  farg_text = lstr(eappend(", ", rselect(get_value(vid), [0:fname_argc-1]), r=false,
  i=false, l=false));
  test_text = validate(str(fname, "(", farg_text, ")=", pass_value), fresult, "equals",
  pass_value);
  if ( pass_value != skip )
      log_warn( str(vid, "(", value_text, ") ", test_text) );
    else if ( show_passing )
  log_info( str(vid, " ", test_text) );
  else if ( show_skipped )
    log_info( str(vid, " *skip*: '", fname, "(", value_text, ")'") );
// Indirect function calls would be very useful here!!!
run_ids = delete( test_ids, mv=["fac", "crp"] );
for (vid=run_ids) run("bitwise_is_equal_0", vid) test( "bitwise_is_equal_0",
bitwise_is_equal(gv(vid,0),gv(vid,1),0), vid);
for (vid=run_ids) run("bitwise_is_equal_1",vid) test( "bitwise_is_equal_1",
 bitwise_is_equal(gv(vid,0),gv(vid,1),1), vid);
for (vid=run_ids) run("bitwise_i2v",vid) test( "bitwise_i2v", bitwise_i2v(gv(vid,0)), vid );
for (vid=run_ids) run("bitwise_i2v_v2i", vid) test( "bitwise_i2v_v2i",
 bitwise_v2i(bitwise_i2v(gv(vid,0))), vid );
for (vid=run_ids) run("bitwise_i2s", vid) test( "bitwise_i2s", bitwise_i2s(gv(vid,0)), vid);
for (vid=run_ids) run("bitwise_i2s_s2i", vid) test( "bitwise_i2s_s2i",
bitwise_s2i(bitwise_i2s(gv(vid,0))), vid);
for (vid=run_ids) run("bitwise_imi_32",vid) test( "bitwise_imi_32",
 bitwise_imi(gv(vid,0),3,2), vid);
for (vid=run_ids) run("bitwise_and", vid) test( "bitwise_and", bitwise_and(qv(vid,0),qv(vid,1
 )), vid );
for (vid=run ids) run("bitwise or", vid) test( "bitwise or", bitwise or(qv(vid,0),qv(vid,1)),
 vid );
for (vid=run_ids) run("bitwise_xor", vid) test( "bitwise_xor", bitwise_xor(gv(vid,0),gv(vid,1
 )), vid );
for (vid=run_ids) run("bitwise_not", vid) test( "bitwise_not", bitwise_not(qv(vid,0)), vid);
```

```
for (vid=run_ids) run("bitwise_lsh",vid) test( "bitwise_lsh", bitwise_lsh(gv(vid,0)), vid );
for (vid=run_ids) run("bitwise_rsh",vid) test( "bitwise_rsh", bitwise_rsh(gv(vid,0)), vid );
// end-of-tests
```

3.2.2.2 Results

```
1 ECHO: "OpenSCAD Version [2017, 2, 19]"
2 ECHO: "[INFO] run(): test(); t01 passed: 'bitwise_is_equal_0(undef, undef)=false'"
3 ECHO: "[ INFO ]
                   run(): test(); t02 passed: 'bitwise_is_equal_0([], [])=false'
                   run(): test(); t03 passed: 'bitwise_is_equal_0(254, 0)=true'"
5 ECHO: "[ INFO ]
                   run(): test(); t04 passed: 'bitwise_is_equal_0(254, 1)=false'"
                   run(): test(); t05 passed: 'bitwise_is_equal_0(255, 0)=false'"
7 ECHO: "[ INFO ]
                   run(): test(); t06 passed: 'bitwise_is_equal_0(0, 255)=true'"
8 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_is_equal_0(126, 63)=true'"
9 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_is_equal_0(25, 10)=true'"
10 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_is_equal_0(1024, 512)=true'"
11 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_is_equal_0(4253, 315)=true'"
12 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_is_equal_0(835, 769)=true'"
13 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_is_equal_0(856, 625)=true'"
14 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_is_equal_1(undef, undef)=false'"
15 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_is_equal_1([], [])=false'"
16 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_is_equal_1(254, 0)=false'"
17 ECHO: "[INFO] run(): test(); t04 passed: 'bitwise_is_equal_1(254, 1)=true'"

18 ECHO: "[INFO] run(): test(); t05 passed: 'bitwise_is_equal_1(255, 0)=true'"
19 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_is_equal_1(0, 255)=false'"
20 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_is_equal_1(126, 63)=false'"
21 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_is_equal_1(25, 10)=false'"
22 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_is_equal_1(1024, 512)=false'"
23 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_is_equal_1(4253, 315)=false'"
24 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_is_equal_1(835, 769)=false'"
25 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_is_equal_1(856, 625)=false'"
26 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_i2v(undef)=undef'
27 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_i2v([])=undef'"
28 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_i2v(254)=[1, 1, 1, 1, 1, 1, 1, 0]'"
29 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_i2v(254)=[1, 1, 1, 1, 1, 1, 1, 0]'"
30 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_i2v(255)=[1, 1, 1, 1, 1, 1, 1, 1]'"
31 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_i2v(0)=[0]'"
32 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_i2v(126)=[1, 1, 1, 1, 1,
33 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_i2v(25)=[1, 1, 0, 0, 1]'"
34 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_i2v(1024)=[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'"
35 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_i2v(4253)=[1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1]'"
36 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_i2v(835)=[1, 1, 0, 1, 0, 0, 0, 0, 1, 1]'"
37 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_i2v(856)=[1, 1, 0, 1, 0, 1, 1, 0, 0, 0]'"
38 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_i2v_v2i(undef)=undef'"
39 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_i2v_v2i([])=undef'"
40 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_i2v_v2i(254)=254''
41 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_i2v_v2i(254)=254'"
42 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_i2v_v2i(255)=255'"
43 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_i2v_v2i(0)=0'"
44 ECHO: "[ INFO
                    run(): test(); t07 passed: 'bitwise_i2v_v2i(126)=126'"
45 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_i2v_v2i(25)=25'"
46 ECHO: "[ INFO
                  ] run(): test(); t09 passed: 'bitwise_i2v_v2i(1024)=1024'"
47 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_i2v_v2i(4253)=4253'"
48 ECHO: "[
             INFO
                  ] run(): test(); t11 passed: 'bitwise_i2v_v2i(835)=835'"
49 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_i2v_v2i(856)=856'"
50 ECHO: "[
                  run(): test(); t01 passed: 'bitwise_i2s(undef)=undef'"
             INFO
51 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_i2s([])=undef'"
52 ECHO: "[
            INFO
                  ] run(): test(); t03 passed: 'bitwise_i2s(254)=111111110'"
53 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_i2s(254)=111111110'"
54 ECHO: "[
            INFO
                    run(): test(); t05 passed: 'bitwise_i2s(255)=111111111'"
55 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_i2s(0)=0'"
56 ECHO: "[ INFO
                  | run(): test(); t07 passed: 'bitwise_i2s(126)=11111110'"
57 ECHO: "[ INFO
                  ] run(): test(); t08 passed: 'bitwise_i2s(25)=11001'"
58 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_i2s(1024)=10000000000""
59 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_i2s(4253)=1000010011101'"
60 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_i2s(835)=1101000011'
61 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_i2s(856)=1101011000'"
62 ECHO: "[ INFO ]
                    run(): test(); t01 passed: 'bitwise_i2s_s2i(undef)=undef'"
63 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_i2s_s2i([])=undef'"
64 ECHO: "[ INFO ]
                    run(): test(); t03 passed: 'bitwise_i2s_s2i(254)=254'"
65 ECHO: "[ INFO
                  run(): test(); t04 passed: 'bitwise_i2s_s2i(254)=254'"
66 ECHO: "[ INFO ]
                    run(): test(); t05 passed: 'bitwise_i2s_s2i(255)=255'"
67 ECHO: "[
                  ] run(): test(); t06 passed: 'bitwise_i2s_s2i(0)=0'"
            INFO
68 ECHO: "[
                    run(): test(); t07 passed: 'bitwise_i2s_s2i(126)=126'"
            INFO 1
69 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_i2s_s2i(25)=25'"
70 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_i2s_s2i(1024)=1024'
71 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_i2s_s2i(4253)=4253'"
72 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_i2s_s2i(835)=835'
```

3.2 Math 51

```
73 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_i2s_s2i(856)=856'"
74 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_imi_32(undef)=undef'"
                   run(): test(); t02 passed: 'bitwise_imi_32([])=undef'
75 ECHO: "[
            INFO
76 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_imi_32(254)=7'"
77 ECHO: "[
            INFO
                   run(): test(); t04 passed: 'bitwise_imi_32(254)=7'"
78 ECHO: "[ INFO
                  ] run(): test(); t05 passed: 'bitwise_imi_32(255)=7'"
79 ECHO: "[
            INFO
                   run(): test(); t06 passed: 'bitwise_imi_32(0)=0'"
                  run(): test(); t07 passed: 'bitwise_imi_32(126)=7'"
80 ECHO: "[ INFO
81 ECHO: "[
            INFO
                  run(): test(); t08 passed: 'bitwise_imi_32(25)=6'"
82 ECHO: "[ INFO
                 ] run(): test(); t09 passed: 'bitwise_imi_32(1024)=0'"
83 ECHO: "[
            INFO
                  run(): test(); t10 passed: 'bitwise_imi_32(4253)=7'"
84 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_imi_32(835)=0'"
85 ECHO: "[ INFO
                 ] run(): test(); t12 passed: 'bitwise_imi_32(856)=6'"
86 ECHO: "[ INFO
                 ] run(): test(); t01 passed: 'bitwise_and(undef, undef)=undef'"
87 ECHO: "[
            TNFO
                 ] run(): test(); t02 passed: 'bitwise_and([], [])=undef'
                 ] run(): test(); t03 passed: 'bitwise_and(254, 0)=0'"
89 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_and(254, 1)=0'"
90 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_and(255, 0)=0'"
91 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_and(0, 255)=0'"
92 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_and(126, 63)=62'"
93 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_and(25, 10)=8''
94 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_and(1024, 512)=0'"
95 ECHO: "[
            INFO ]
                   run(): test(); t10 passed: 'bitwise_and(4253, 315)=25'"
96 ECHO: "[ INFO ] run(): test(); tll passed: 'bitwise_and(835, 769)=769'"
97 ECHO: "[ INFO ]
                   run(): test(); t12 passed: 'bitwise_and(856, 625)=592'"
98 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_or(undef, undef)=undef'"
99 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_or([], [])=undef'
100 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_or(254, 0)=254'"
                    run(): test(); t04 passed: 'bitwise_or(254, 1)=255'"
101 ECHO: "[ INFO ]
102 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_or(255, 0)=255'"
103 ECHO: "[ INFO ]
                    run(): test(); t06 passed: 'bitwise_or(0, 255)=255'"
104 ECHO: "[ INFO ]
                    run(): test(); t07 passed: 'bitwise_or(126, 63)=127'"
                    run(): test(); t08 passed: 'bitwise_or(25, 10)=27'
105 ECHO: "[ INFO ]
106 ECHO: "[ INFO ]
                    run(): test(); t09 passed: 'bitwise_or(1024, 512)=1536'"
                    run(): test(); t10 passed: 'bitwise_or(4253, 315)=4543''
run(): test(); t11 passed: 'bitwise_or(835, 769)=835'"
107 ECHO: "[ INFO ]
108 ECHO: "[ INFO ]
109 ECHO: "[ INFO ]
                    run(): test(); t12 passed: 'bitwise_or(856, 625)=889'"
110 ECHO: "[ INFO ]
                    run(): test(); t01 passed: 'bitwise_xor(undef, undef)=undef'"
111 ECHO: "[ INFO ]
                    run(): test(); t02 passed: 'bitwise_xor([], [])=undef'
112 ECHO: "[ INFO ]
                    run(): test(); t03 passed: 'bitwise_xor(254, 0)=254'
113 ECHO: "[ INFO ]
                    run(): test(); t04 passed: 'bitwise_xor(254, 1)=255'"
114 ECHO: "[ TNFO ]
                    run(): test(); t05 passed: 'bitwise_xor(255, 0)=255'"
115 ECHO: "[ INFO ]
                    run(): test(); t06 passed: 'bitwise_xor(0, 255)=255'"
116 ECHO: "[ TNFO ]
                    run(): test(); t07 passed: 'bitwise_xor(126, 63)=65'"
117 ECHO: "[ INFO ]
                    run(): test(); t08 passed: 'bitwise_xor(25, 10)=19'"
118 ECHO: "[ INFO ]
                    run(): test(); t09 passed: 'bitwise_xor(1024, 512)=1536'"
119 ECHO: "[ INFO ]
                    run(): test(); t10 passed: 'bitwise_xor(4253, 315)=4518'"
120 ECHO: "[ INFO ]
                    run(): test(); t11 passed: 'bitwise_xor(835, 769)=66''
121 ECHO: "[ INFO ]
                    run(): test(); t12 passed: 'bitwise_xor(856, 625)=297'"
122 ECHO: "[ INFO ]
                    run(): test(); t01 passed: 'bitwise_not(undef) = undef'"
123 ECHO: "[ INFO ]
                    run(): test(); t02 passed: 'bitwise_not([])=undef'"
124 ECHO: "[ INFO ]
                    run(): test(); t03 passed: 'bitwise_not(254)=1'"
125 ECHO: "[ INFO ]
                    run(): test(); t04 passed: 'bitwise_not(254)=1'"
126 ECHO: "[ INFO
                    run(): test(); t05 passed: 'bitwise_not(255)=0'"
127 ECHO: "[ INFO ]
                    run(): test(); t06 passed: 'bitwise_not(0)=1'"
                    run(): test(); t07 passed: 'bitwise_not(126)=1'"
128 ECHO: "[ INFO
129 ECHO: "[ INFO ]
                    run(): test(); t08 passed: 'bitwise_not(25)=6'"
                    run(): test(); t09 passed: 'bitwise_not(1024)=1023'"
130 ECHO: "[ INFO
131 ECHO: "[ INFO ]
                    run(): test(); t10 passed: 'bitwise_not(4253)=3938'"
132 ECHO: "[ INFO
                    run(): test(); t11 passed: 'bitwise_not(835)=188'"
133 ECHO: "[ INFO ]
                    run(): test(); t12 passed: 'bitwise_not(856)=167'"
134 ECHO: "[ INFO
                    run(): test(); t01 passed: 'bitwise_lsh(undef) = undef'"
135 ECHO: "[ INFO ]
                    run(): test(); t02 passed: 'bitwise_lsh([])=undef'"
136 ECHO: "[ INFO
                    run(): test(); t03 passed: 'bitwise_lsh(254)=508'"
137 ECHO: "[ INFO
                    run(): test(); t04 passed: 'bitwise_lsh(254)=508'"
138 ECHO: "[ INFO
                    run(): test(); t05 passed: 'bitwise_lsh(255)=510'"
                    run(): test(); t06 passed: 'bitwise_lsh(0)=0'"
140 ECHO: "[ INFO ]
                    run(): test(); t07 passed: 'bitwise_lsh(126)=252'"
141 ECHO: "[ INFO ]
                    run(): test(); t08 passed: 'bitwise_lsh(25)=50'"
142 ECHO: "[ INFO ]
                    run(): test(); t09 passed: 'bitwise_lsh(1024)=2048'"
143 ECHO: "[ INFO ]
                    run(): test(); t10 passed: 'bitwise_lsh(4253)=8506'"
144 ECHO: "[ INFO ]
                    run(): test(); t11 passed: 'bitwise_lsh(835)=1670'"
                    run(): test(); t12 passed: 'bitwise_lsh(856)=1712'"
145 ECHO: "[ INFO
146 ECHO: "[ INFO ]
                    run(): test(); t01 passed: 'bitwise_rsh(undef)=undef'"
147 ECHO: "[ INFO ]
                    run(): test(); t02 passed: 'bitwise_rsh([])=undef'"
                    run(): test(); t03 passed: 'bitwise_rsh(254)=127'"
148 ECHO: "[ INFO ]
          "[ INFO ]
                    run(): test(); t04 passed: 'bitwise rsh(254)=127'"
149 ECHO:
150 ECHO: "[ INFO ]
                    run(): test(); t05 passed: 'bitwise_rsh(255)=127'"
                    run(): test(); t06 passed: 'bitwise_rsh(0)=0'"
151 ECHO: "[ INFO ]
152 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_rsh(126)=63'"
153 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_rsh(25)=12'"
```

```
154 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_rsh(1024)=512'"
155 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_rsh(4253)=2126'"
156 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_rsh(835)=417'"
157 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_rsh(856)=428'"
```

4 Installing

First install openscad—amu. More information can be found at amu on Thingiverse and in the GitHib amu repository where the source is maintained.

A build script exists for *Linux* and *Cygwin* (pull requests for *macos* are welcome). If wget is not available, here is a downloadable link to the bootstrap script.

```
$ mkdir tmp && cd tmp

$ wget https://raw.githubusercontent.com/royasutton/openscad-amu/master/snapshots/bootstrap.{bash,conf} .
$ chmod +x bootstrap.bash
$ ./bootstrap.bash --yes --install
$ openscad-seam -v -V
```

If the last step reports the tool version, then the install most likely completed successfully and the temporary directory may be removed as desired.

Now omdl can be compiled, verified, and installed. First download the source from omdl on Thingiverse or clone the GitHub omdl repository and install as follows:

```
$ git clone https://github.com/royasutton/omdl.git
$ cd omdl
$ make install
```

The library and documentation should now have been installed to the OpenSCAD *built-in* library location along with the omd1 reference documentation that can be views with a web browser.

Have a look in:

- Linux: \$HOME/.local/share/OpenSCAD/libraries
- Windows: My Documents\OpenSCAD\libraries

You may include the desired library component from your project as follows, replacing the version number as needed:

```
include <omdl-v0.4/shapes2de.scad>;
include <omdl-v0.4/shapes3d.scad>;
...
```

5 Test List

File units_resolution.scad

Review model for accuracy.

6 Todo List 53

6 Todo List

globalScope> Global linear_extrude_uls (h, center=false)

This function should be rewritten to use the built-in scaling provided by linear_extrude() in the upper and lower scaling zones.

globalScope> Global pyramid_q (size, center=false)

Support vertex rounding radius.

globalScope> Global pyramid_t (size, center=false)

Support vertex rounding radius.

File shapes3d.scad

Complete rounded cylinder.

globalScope> Global triangle_ppp (v1, v2, v3, vr, v1r, v2r, v3r, centroid=false, incenter=false)

Replace the hull() operation with calculated tangential intersection of the rounded vertexes.

Remove the all or nothing requirement for vertex rounding.

7 Module Index

7.1 Modules

Here is a list of all modules:

Constants	117
Euclidean	129
General	135
System	265
Database	122
Component	112
Electrical	128
Geometry	136
Polyhedra	186
Material	177
Datatypes	123
Identification	137
Iterables	139
Lists	159
Scalars	251
Maps	173

Operations	180
Iterables	143
Lists	163
Scalars	259
Tables	266
Math	178
Bitwise	107
Linear Algebra	156
Other Shapes	182
Polytopes	227
Triangles	276
Utilities	284
Vector Algebra	289
Parts	185
Shapes	261
2d Extrusions	58
2d Shapes	77
3d Shapes	92
Tools	274
Alignment	102
Edge	125
Extrude	131
Polytope	224
Repeat	243
Units	282
Angles	104
Coordinates	118
Lengths	153
Resolutions	246
Utilities	286

8 File Index 55

	Console	113
	Validation	287
8	File Index	
8.1	File List	
Hei	re is a list of all documented files with brief descriptions:	
	console.scad Message logging functions	296
	constants.scad Design constant definitions	297
	datatypes.scad Data type identification and operations	310
	mainpage.scad Documentation main page	325
	math.scad Mathematical function primitives	325
	validation.scad Function validation methods	355
	database/geometry/polyhedra/anti_prisms.scad Table of polyhedra data group: anti_prisms	299
	database/geometry/polyhedra/archimedean.scad Table of polyhedra data group: archimedean	300
	database/geometry/polyhedra/archimedean_duals.scad Table of polyhedra data group: archimedean_duals	301
	database/geometry/polyhedra/cupolas.scad Table of polyhedra data group: cupolas	302
	database/geometry/polyhedra/dipyramids.scad Table of polyhedra data group: dipyramids	303
	database/geometry/polyhedra/johnson.scad Table of polyhedra data group: johnson	304
	database/geometry/polyhedra/platonic.scad Table of polyhedra data group: platonic	305
	database/geometry/polyhedra/polyhedra_all.scad Table of polyhedra data group: polyhedra_all	306
	database/geometry/polyhedra/prisms.scad Table of polyhedra data group: prisms	307

database/geometry/polyhedra/pyramids.scad Table of polyhedra data group: pyramids	308
database/geometry/polyhedra/trapezohedron.scad Table of polyhedra data group: trapezohedron	309
datatypes/datatypes_identify_iterable.scad Iterable data type identification	311
datatypes/datatypes_identify_list.scad List data type identification	313
List data type identification	313
datatypes/datatypes_identify_scalar.scad Scalar data type identification	315
datatypes/datatypes_map.scad Map data type operations	317
datatypes/datatypes_operate_iterable.scad Iterable data type operations	318
datatypes/datatypes_operate_list.scad List data type operations	320
datatypes/datatypes_operate_scalar.scad	200
Scalar data type operations	322
datatypes/datatypes_table.scad Table data type operations	323
math/math_bitwise.scad	
Mathematical base-two bitwise binary functions	326
math/math_linalg.scad Linear algebra mathematical functions	328
math/math_oshapes.scad	
Other shapes mathematical functions	330
math/math_polytope.scad Polygon and polyhedron mathematical functions	331
math/math_triangle.scad Triangle solutions mathematical functions	334
math/math_utility.scad	
Miscellaneous mathematical utilities	336
math/math_vecalg.scad Vector algebra mathematical functions	337
shapes/shapes2d.scad	
Two-dimensional basic shapes	339
shapes/shapes2de.scad Linearly extruded two-dimensional basic shapes	341
,	• • • • • • • • • • • • • • • • • • • •

9 Module Documentation 57

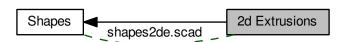
shapes/shapes3d.scad	
Three-dimensional basic shapes	342
tools/tools_align.scad	
Shape alignment tools	344
tools/tools_edge.scad	
Shape edge finishing tools	345
tools/tools_polytope.scad	
Polygon and polyhedron tools	347
tools/tools_utility.scad	
Shape transformation utility tools	348
units/units_angle.scad	
Angle units and conversions	349
units/units_coordinate.scad	
Coordinate systems and conversions	350
units/units_length.scad	
Length units and conversions	352
units/units_resolution.scad	
An abstraction for arc rendering resolution control	353

9 Module Documentation

9.1 2d Extrusions

Extruded two-dimensional geometric shapes.

Collaboration diagram for 2d Extrusions:



Files

• file shapes2de.scad

Linearly extruded two-dimensional basic shapes.

Functions

module erectangle (size, h, vr, vrm=0, center=false)

An extruded rectangle with edge, fillet, and/or chamfer corners.

module erectangle_c (size, core, h, t, co, cr=0, vr, vr1, vr2, vrm=0, vrm1, vrm2, center=false)

An extruded rectangle with a removed rectangular core.

• module erhombus (size, h, vr, center=false)

An extruded rhombus.

• module etriangle_ppp (v1, v2, v3, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded general triangle specified by three vertices.

• module etriangle_lp (v, h, vr, centroid=false, incenter=false, center=false)

An extruded general triangle specified by a list of its three vertices.

module etriangle_sss (s1, s2, s3, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded general triangle specified by its three side lengths.

module etriangle Is (v, h, vr, centroid=false, incenter=false, center=false)

An extruded general triangle specified by a list of its three side lengths.

module etriangle_ls_c (vs, vc, h, co, cr=0, vr, vr1, vr2, centroid=false, incenter=false, center=false)

A general triangle specified by its sides with a removed triangular core.

• module etriangle_sas (s1, a, s2, h, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded general triangle specified by two sides and the included angle.

• module etriangle_asa (a1, s, a2, h, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded general triangle specified by a side and two adjacent angles.

• module etriangle_aas (a1, a2, s, h, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded general triangle specified by a side, one adjacent angle and the opposite angle.

• module etriangle_ss (x, y, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded right-angled triangle specified by its opposite and adjacent side lengths.

module etriangle_sa (x, y, aa, oa, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)

An extruded right-angled triangle specified by a side length and an angle.

• module engon (n, r, h, vr, center=false)

An extruded n-sided equiangular/equilateral regular polygon.

module eellipse (size, h, center=false)

An extruded ellipse.

module eellipse_c (size, core, h, t, co, cr=0, center=false)

An extruded ellipse with a removed elliptical core.

module eellipse_s (size, h, a1=0, a2=0, center=false)

An extruded ellipse sector.

module eellipse_cs (size, core, h, t, a1=0, a2=0, co, cr=0, center=false)

An extruded sector of an ellipse with a removed elliptical core.

module estar2d (size, h, n=5, vr, center=false)

An extruded two-dimensional star.

9.1.1 Detailed Description

Extruded two-dimensional geometric shapes.

9.1.2 Function Documentation

9.1.2.1 module eellipse (size , h , center = false)

An extruded ellipse.

9.1 2d Extrusions 59

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Example

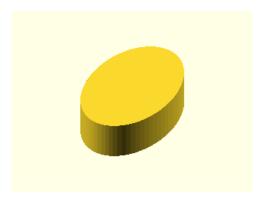


Figure 1: eellipse

eellipse(size=[25, 40], h=20, center=true);

Definition at line 777 of file shapes2de.scad.

9.1.2.2 module eellipse_c (size , core , h , t , co , cr = 0 , center = false)

An extruded ellipse with a removed elliptical core.

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
core	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
t	<pre><decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>
СО	<pre><decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2></pre>
cr	<decimal> Core z-rotation.</decimal>
center	 boolean> Center about origin.

See also

 $\label{linear_extrude_uls} \mbox{ for a description on specifying h.}$

Thickness t

• core = size - t; when t and size are given.

• size = core + t; when t and core are given.

Example



Figure 2: eellipse_c

```
eellipse_c( size=[25,40], core=[16,10], co=[0,10], cr=45, h=20, center=true );
```

Definition at line 817 of file shapes2de.scad.

```
9.1.2.3 module eellipse_cs ( size, core, h, t, a1 = 0, a2 = 0, co, cr = 0, center = false )
```

An extruded sector of an ellipse with a removed elliptical core.

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
core	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
t	<decimal-list-2 $ $ decimal $>$ A list [x, y] of decimals or a single decimal for (x=y).
a1	<decimal> The start angle in degrees.</decimal>
a2	<decimal> The stop angle in degrees.</decimal>
СО	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

Example

9.1 2d Extrusions 61

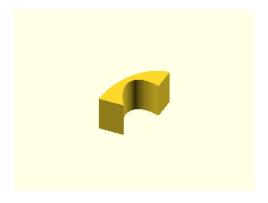


Figure 3: eellipse_cs

1 eellipse_cs(size=[25,40], t=[10,5], al=90, a2=180, co=[10,0], cr=45, h=20, center=true);

Definition at line 897 of file shapes2de.scad.

9.1.2.4 module eellipse_s (size , h , a1 = 0, a2 = 0, center = false)

An extruded ellipse sector.

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
a1	<decimal> The start angle in degrees.</decimal>
a2	<decimal> The stop angle in degrees.</decimal>
center	<boolean> Center about origin.</boolean>

See also

linear_extrude_uls for a description on specifying h.

Example

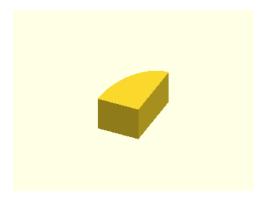


Figure 4: eellipse_s

1 eellipse_s(size=[25,40], h=20, a1=90, a2=180, center=true);

Definition at line 852 of file shapes2de.scad.

```
9.1.2.5 module engon ( n, r, h, vr, center = false )
```

An extruded n-sided equiangular/equilateral regular polygon.

Parameters

n	<decimal> The number of sides.</decimal>
r	<decimal> The ngon vertex radius.</decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal> The vertex rounding radius.</decimal>
center	<boolean> Center about origin.</boolean>

See also

linear_extrude_uls for a description on specifying h.

Example



Figure 5: engon

```
1 engon( n=6, r=25, h=20, vr=6, center=true );
```

See Wikipedia for more information.

Definition at line 747 of file shapes2de.scad.

```
9.1.2.6 module erectangle ( size , h , vr , vrm = 0 , center = false )
```

An extruded rectangle with edge, fillet, and/or chamfer corners.

size	<decimal-list-2 decimal $>$ A list [x, y] of decimals or a single decimal for (x=y).
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.

9.1 2d Extrusions 63

vr	<decimal-list-4 decimal> The corner rounding radius. A list [v1r, v2r, v3r, v4r] of decimals or a</decimal-list-4 decimal>	
	single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.	
vrm	<integer> The corner radius mode. A 4-bit encoded integer that indicates each corner finish.</integer>	
	Use bit value 0 for fillet and 1 for chamfer.	
center	<boolean> Center about origin.</boolean>	

See also

linear_extrude_uls for a description on specifying h.

Example

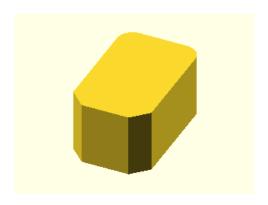


Figure 6: erectangle

1 erectangle(size=[25,40], vr=5, vrm=3, h=20, center=true);

Definition at line 114 of file shapes2de.scad.

9.1.2.7 module erectangle_c (size, core, h, t, co, cr = 0, vr, vr1, vr2, vrm = 0, vrm1, vrm2, center = false)

An extruded rectangle with a removed rectangular core.

size	<decimal-list-2 $ $ decimal $>$ A list [x, y] of decimals or a single decimal for (x=y).
core	<decimal-list-2 $ $ decimal $>$ A list [x, y] of decimals or a single decimal for (x=y).
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
t	<pre><decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>
СО	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>
vr	<decimal-list-4 decimal> The default corner rounding radius. A list [v1r, v2r, v3r, v4r] of decimals</decimal-list-4 decimal>
	or a single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.
vr1	<decimal-list-4 decimal> The outer corner rounding radius.</decimal-list-4 decimal>
vr2	<decimal-list-4 decimal> The core corner rounding radius.</decimal-list-4 decimal>
vrm	<integer> The default corner radius mode. A 4-bit encoded integer that indicates each corner</integer>
	finish. Use bit value 0 for <i>fillet</i> and 1 for <i>chamfer</i> .
vrm1	<integer> The outer corner radius mode.</integer>
vrm2	<integer> The core corner radius mode.</integer>
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

Example



Figure 7: erectangle_c

```
erectangle_c( size=[40,20], t=[10,1], co=[0,-6], cr=10, vr=5, vrm1=12, h=30, center=true );
```

Definition at line 168 of file shapes2de.scad.

```
9.1.2.8 module erhombus ( size , h , vr , center = false )
```

An extruded rhombus.

size	<decimal-list-2 decimal> A list [w, h] of decimals or a single decimal for (w=h).</decimal-list-2 decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal-list-4 decimal> The corner rounding radius. A list [v1r, v2r, v3r, v4r] of decimals or a</decimal-list-4 decimal>
	single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.
center	<boolean> Center about origin.</boolean>

9.1 2d Extrusions 65

See also

linear_extrude_uls for a description on specifying h.

Example



Figure 8: erhombus

1 erhombus(size=[40,25], h=10, vr=[3,0,3,9], center=true);

Definition at line 217 of file shapes2de.scad.

9.1.2.9 module estar2d (size, h, n = 5, vr, center = false)

An extruded two-dimensional star.

size	<pre><decimal-list-2 decimal> A list [l, w] of decimals or a single decimal for (size=l=2*w).</decimal-list-2 decimal></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
n	<decimal> The number of points.</decimal>
vr	<decimal-list-3 decimal> The vertex rounding radius. A list [v1r, v2r, v3r] of decimals or a single</decimal-list-3 decimal>
	decimal for (v1r=v2r=v3r).
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Example



Figure 9: estar2d

1 estar2d(size=[40, 15], h=15, n=5, vr=2, center=true);

Definition at line 937 of file shapes2de.scad.

9.1.2.10 module etriangle_aas (a1 , a2 , s , h , x = 1, vr , v1r , v2r , v3r , centroid = false, incenter = false, center = false)

An extruded general triangle specified by a side, one adjacent angle and the opposite angle.

a1	<decimal> The opposite angle 1 in degrees.</decimal>
a2	<decimal> The adjacent angle 2 in degrees.</decimal>
S	<decimal> The side length.</decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
X	<decimal $>$ The side to draw on the positive x-axis (x=1 for s).
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	 boolean> Center incenter at origin.
center	 <boolean> Center about origin.</boolean>

9.1 2d Extrusions 67

See also

linear_extrude_uls for a description on specifying h.

Example

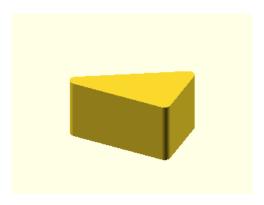


Figure 10: etriangle_aas

etriangle_aas(a1=60, a2=30, s=40, h=20, vr=2, centroid=true, center=true);

Definition at line 599 of file shapes2de.scad.

9.1.2.11 module etriangle_asa (a1 , s , a2 , h , x = 1, vr , v1r , v2r , v3r , centroid = false, incenter = false, center = false)

An extruded general triangle specified by a side and two adjacent angles.

a1	<decimal> The adjacent angle 1 in degrees.</decimal>
S	<decimal> The side length adjacent to the angles.</decimal>
a2	<decimal> The adjacent angle 2 in degrees.</decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
X	<decimal $>$ The side to draw on the positive x-axis (x=1 for s).
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Example

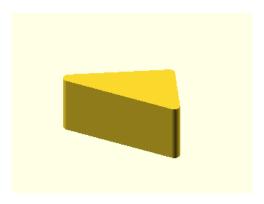


Figure 11: etriangle_asa

```
etriangle_asa( a1=30, s=50, a2=60, h=20, vr=2, centroid=true, center=true );
```

Definition at line 546 of file shapes2de.scad.

```
9.1.2.12 module etriangle_lp ( v , h , vr , centroid = false, incenter = false, center = false )
```

An extruded general triangle specified by a list of its three vertices.

Parameters

V	<pre><point-2d-list-3> A list [v1, v2, v3] of points [x, y].</point-2d-list-3></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal-list-3 decimal> The vertex rounding radius. A list [v1r, v2r, v3r] of decimals or a single</decimal-list-3 decimal>
	decimal for (v1r=v2r=v3r).
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>
center	<boolean> Center about origin.</boolean>

See also

linear_extrude_uls for a description on specifying h.

Example

```
t = triangle_sss2lp( 30, 40, 50 );
r = [2, 4, 6];
etriangle_lp( v=t, h=5, vr=r );
```

Definition at line 304 of file shapes2de.scad.

```
9.1.2.13 module etriangle_ls ( v , h , vr , centroid = false, incenter = false, center = false )
```

An extruded general triangle specified by a list of its three side lengths.

9.1 2d Extrusions 69

Parameters

V	<pre><decimal-list-3> A list [s1, s2, s3] of decimals.</decimal-list-3></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal-list-3 decimal> The vertex rounding radius. A list [v1r, v2r, v3r] of decimals or a single</decimal-list-3 decimal>
	decimal for (v1r=v2r=v3r).
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>
center	<boolean> Center about origin.</boolean>

See also

linear_extrude_uls for a description on specifying h.

Example

```
t = triangle_sss2lp( 3, 4, 5 );
s = triangle_lp2ls( t );
etriangle_ls( v=s, h=5, vr=2 );
```

Definition at line 393 of file shapes2de.scad.

```
9.1.2.14 module etriangle_ls_c ( vs , vc , h , co , cr = 0, vr , vr1 , vr2 , centroid = false, incenter = false, center = false )
```

A general triangle specified by its sides with a removed triangular core.

VS	<decimal-list-3 decimal> The size. A list [s1, s2, s3] of decimals or a single decimal for</decimal-list-3 decimal>
	(s1=s2=s3).
VC	<decimal-list-3 decimal> The core. A list [s1, s2, s3] of decimals or a single decimal for</decimal-list-3 decimal>
	(s1=s2=s3).
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
со	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>
vr	<decimal-list-3 decimal> The default vertex rounding radius. A list [v1r, v2r, v3r] of decimals or</decimal-list-3 decimal>
	a single decimal for (v1r=v2r=v3r).
vr1	<decimal-list-3 decimal> The outer vertex rounding radius.</decimal-list-3 decimal>
vr2	<decimal-list-3 decimal> The core vertex rounding radius.</decimal-list-3 decimal>
centroid	 boolean> Center centroid at origin.
incenter	 boolean> Center incenter at origin.
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Example

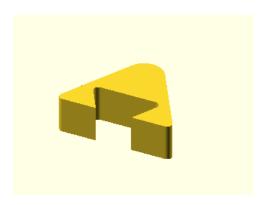


Figure 12: etriangle_ls_c

1 etriangle_ls_c(vs=50, vc=30, h=15, co=[0,-10], cr=180, vr=[2,2,8], centroid=true, center=true);

Note

The outer and inner triangles centroids are aligned prior to the core removal.

Definition at line 440 of file shapes2de.scad.

9.1.2.15 module etriangle_ppp (v1, v2, v3, h, vr, v1r, v2r, v3r, centroid = false, incenter = false, center = false)

An extruded general triangle specified by three vertices.

Parameters

v1	<pre><point-2d> A point [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A point [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A point [x, y] for vertex 3.</point-2d></pre>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	 boolean> Center centroid at origin.
incenter	<boolean> Center incenter at origin.</boolean>
center	<boolean> Center about origin.</boolean>

See also

linear_extrude_uls for a description on specifying h.

9.1 2d Extrusions 71

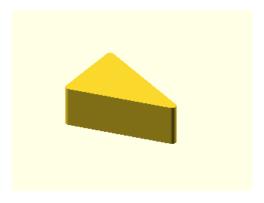


Figure 13: etriangle_ppp

1 etriangle_ppp(v1=[0,0], v2=[5,25], v3=[40,5], h=20, vr=2, centroid=true, center=true);

Definition at line 254 of file shapes2de.scad.

9.1.2.16 module etriangle_sa (x , y , aa , oa , h , vr , v1r , v2r , v3r , centroid = false, incenter = false, center = false)

An extruded right-angled triangle specified by a side length and an angle.

<decimal> The length of the side along the x-axis.</decimal>
<decimal> The length of the side along the y-axis.</decimal>
<decimal> The adjacent angle in degrees.</decimal>
<decimal> The opposite angle in degrees.</decimal>
<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
height.
<decimal> The default vertex rounding radius.</decimal>
<decimal> Vertex 1 rounding radius.</decimal>
<decimal> Vertex 2 rounding radius.</decimal>
<decimal> Vertex 3 rounding radius.</decimal>
 boolean> Center centroid at origin.
 boolean> Center incenter at origin.
 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Example

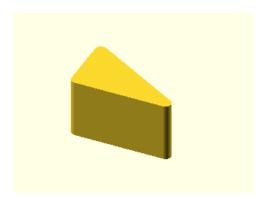


Figure 14: etriangle_sa

etriangle_sa(x=40, aa=30, h=20, vr=2, centroid=true, center=true);

Note

When both \boldsymbol{x} and \boldsymbol{y} are given, both triangles are rendered.

When both aa and oa are given, aa is used.

Definition at line 700 of file shapes2de.scad.

9.1.2.17 module etriangle_sas (s1 , a , s2 , h , x = 1, vr , v1r , v2r , v3r , centroid = false, incenter = false, center = false)

An extruded general triangle specified by two sides and the included angle.

s1	<decimal> The length of the side 1.</decimal>
а	<decimal> The included angle in degrees.</decimal>
s2	<decimal> The length of the side 2.</decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
X	<decimal $>$ The side to draw on the positive x-axis (x=1 for s1).
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>

9.1 2d Extrusions 73

incenter	<boolean> Center incenter at origin.</boolean>
center	 boolean> Center about origin.

See also

linear_extrude_uls for a description on specifying h.

Example

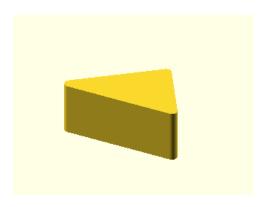


Figure 15: etriangle_sas

```
etriangle_sas( s1=50, a=60, s2=30, h=20, vr=2, centroid=true, center=true );
```

Definition at line 493 of file shapes2de.scad.

```
9.1.2.18 module etriangle_ss ( x, y, h, vr, v1r, v2r, v3r, centroid = false, incenter = false, center = false)
```

An extruded right-angled triangle specified by its opposite and adjacent side lengths.

Parameters

X	<decimal> The length of the side along the x-axis.</decimal>
у	<decimal> The length of the side along the y-axis.</decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>
center	 <boolean> Center about origin.</boolean>

See also

linear_extrude_uls for a description on specifying h.

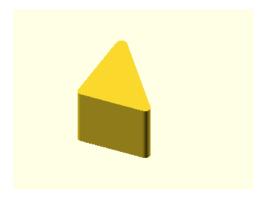


Figure 16: etriangle_ss

etriangle_ss(x=30, y=40, h=20, vr=2, centroid=true, center=true);

Definition at line 648 of file shapes2de.scad.

9.1.2.19 module etriangle_sss (s1 , s2 , s3 , h , vr , v1r , v2r , v3r , centroid = false, incenter = false, center = false)

An extruded general triangle specified by its three side lengths.

s1	<decimal> The length of the side 1 (along the x-axis).</decimal>
s2	<decimal> The length of the side 2.</decimal>
s3	<decimal> The length of the side 3.</decimal>
h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	 boolean> Center centroid at origin.
incenter	<boolean> Center incenter at origin.</boolean>
center	 boolean> Center about origin.

9.1 2d Extrusions 75

See also

linear_extrude_uls for a description on specifying h.

Example

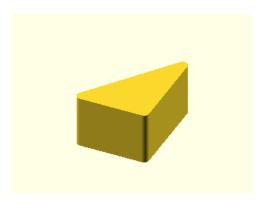


Figure 17: etriangle_sss

1 etriangle_sss(s1=30, s2=40, s3=50, h=20, vr=2, centroid=true, center=true);

Definition at line 343 of file shapes2de.scad.

9.2 2d Shapes

Two-dimensional geometric shapes.

Collaboration diagram for 2d Shapes:



Files

· file shapes2d.scad

Two-dimensional basic shapes.

Functions

module rectangle (size, vr, vrm=0, center=false)

A rectangle with edge, fillet, and/or chamfer corners.

module rectangle_c (size, core, t, co, cr=0, vr, vr1, vr2, vrm=0, vrm1, vrm2, center=false)

A rectangle with a removed rectangular core.

module rhombus (size, vr, center=false)

A rhombus.

module triangle_ppp (v1, v2, v3, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by three vertices.

module triangle_lp (v, vr, centroid=false, incenter=false)

A general triangle specified by a list of its three vertices.

module triangle_sss (s1, s2, s3, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by its three side lengths.

module triangle_ls (v, vr, centroid=false, incenter=false)

A general triangle specified by a list of its three side lengths.

• module triangle_ls_c (vs, vc, co, cr=0, vr, vr1, vr2, centroid=false, incenter=false)

A general triangle specified by its sides with a removed triangular core.

• module triangle_sas (s1, a, s2, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by two sides and the included angle.

• module triangle_asa (a1, s, a2, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by a side and two adjacent angles.

• module triangle_aas (a1, a2, s, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by a side, one adjacent angle and the opposite angle.

• module triangle_ss (x, y, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A right-angled triangle specified by its opposite and adjacent side lengths.

• module triangle_sa (x, y, aa, oa, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A right-angled triangle specified by a side length and an angle.

9.2 2d Shapes 77

• module ngon (n, r, vr)

An n-sided equiangular/equilateral regular polygon.

• module ellipse (size)

An ellipse.

• module ellipse_c (size, core, t, co, cr=0)

An ellipse with a removed elliptical core.

• module ellipse_s (size, a1=0, a2=0)

An ellipse sector.

• module ellipse cs (size, core, t, a1=0, a2=0, co, cr=0)

A sector of an ellipse with a removed elliptical core.

• module star2d (size, n=5, vr)

A two-dimensional star.

9.2.1 Detailed Description

Two-dimensional geometric shapes.

9.2.2 Function Documentation

9.2.2.1 module ellipse (size)

An ellipse.

Parameters

size <decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).

Example

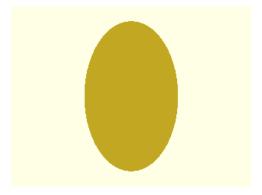


Figure 18: ellipse

1 ellipse(size=[25, 40]);

Definition at line 1102 of file shapes2d.scad.

9.2.2.2 module ellipse_c (size , core , t , co , cr = 0)

An ellipse with a removed elliptical core.

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
core	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>
t	<pre><decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>
СО	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

Example

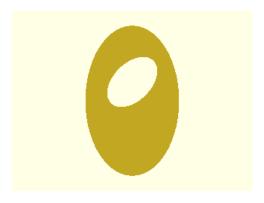


Figure 19: ellipse_c

```
ellipse_c( size=[25,40], core=[16,10], co=[0,10], cr=45 );
```

Definition at line 1143 of file shapes2d.scad.

```
9.2.2.3 module ellipse_cs ( size , core , t , a1 = 0 , a2 = 0 , co , cr = 0 )
```

A sector of an ellipse with a removed elliptical core.

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>	
core	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>	
t	(decimal-list-2 decimal) > A list [x, y] of decimals or a single decimal for (x=y).	
a1	<decimal> The start angle in degrees.</decimal>	
a2	<decimal> The stop angle in degrees.</decimal>	
со	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>	
cr	<decimal> Core z-rotation.</decimal>	

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

9.2 2d Shapes 79

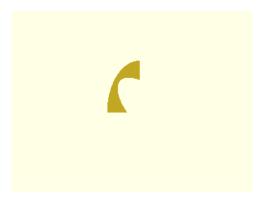


Figure 20: ellipse_cs

```
1 ellipse_cs( size=[25,40], t=[10,5], a1=90, a2=180, co=[10,0], cr=45);
```

Definition at line 1253 of file shapes2d.scad.

```
9.2.2.4 module ellipse_s ( size, a1 = 0, a2 = 0 )
```

An ellipse sector.

Parameters

size	<pre><decimal-list-2 decimal> A list [rx, ry] of decimals or a single decimal for (rx=ry).</decimal-list-2 decimal></pre>	
a1	<decimal> The start angle in degrees.</decimal>	
a2	<decimal> The stop angle in degrees.</decimal>	

Example

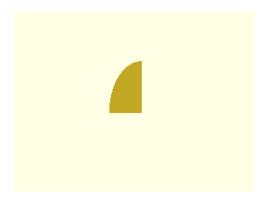


Figure 21: ellipse_s

```
1 ellipse_s( size=[25,40], a1=90, a2=180 );
```

Definition at line 1185 of file shapes2d.scad.

9.2.2.5 module ngon (n , r , vr)

An n-sided equiangular/equilateral regular polygon.

Parameters

n	<integer> The number of sides.</integer>
r	<decimal> The ngon vertex radius.</decimal>
vr	<decimal> The vertex rounding radius.</decimal>

Example

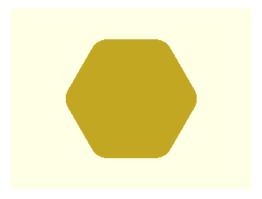


Figure 22: ngon

1 ngon(n=6, r=25, vr=6);

See Wikipedia for more information.

Definition at line 1068 of file shapes2d.scad.

9.2.2.6 module rectangle (size , vr , vrm = 0 , center = false)

A rectangle with edge, fillet, and/or chamfer corners.

Parameters

size	<pre><decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>	
vr	<decimal-list-4 decimal> The corner rounding radius. A list [v1r, v2r, v3r, v4r] of decimals or a</decimal-list-4 decimal>	
	single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.	
vrm	<integer> The corner radius mode. A 4-bit encoded integer that indicates each corner finish.</integer>	
	Use bit value 0 for fillet and 1 for chamfer.	
center	<boolean> Center about origin.</boolean>	

9.2 2d Shapes 81

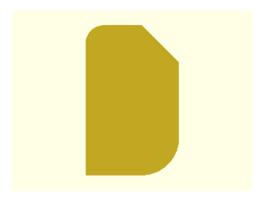


Figure 23: rectangle

1 rectangle(size=[25,40], vr=[0,10,10,5], vrm=4, center=true);

Note

A corner *fillet* replaces an edge with a quarter circle of radius vr, inset [vr, vr] from the corner vertex. A corner *chamfer* replaces an edge with an isosceles right triangle with side lengths equal to the corresponding corner rounding radius vr. Therefore the chamfer length will be vr*sqrt(2) at 45 degree angles.

Definition at line 116 of file shapes2d.scad.

9.2.2.7 module rectangle_c (size, core, t, co, cr = 0, vr, vr1, vr2, vrm = 0, vrm1, vrm2, center = false)

A rectangle with a removed rectangular core.

Parameters

size	<pre><decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>	
core	<pre><decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>	
t	<decimal-list-2 decimal> A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal>	
со	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>	
cr	<decimal> Core z-rotation.</decimal>	
vr	<decimal-list-4 decimal> The default corner rounding radius. A list [v1r, v2r, v3r, v4r] of decimals</decimal-list-4 decimal>	
	or a single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.	
vr1	<decimal-list-4 decimal> The outer corner rounding radius.</decimal-list-4 decimal>	
vr2	<decimal-list-4 decimal> The core corner rounding radius.</decimal-list-4 decimal>	
vrm	<integer> The default corner radius mode. A 4-bit encoded integer that indicates each corner</integer>	
	finish. Use bit value 0 for <i>fillet</i> and 1 for <i>chamfer</i> .	
vrm1	<integer> The outer corner radius mode.</integer>	
vrm2	<integer> The core corner radius mode.</integer>	
center	 boolean> Center about origin.	

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

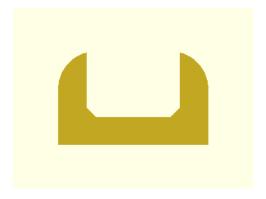


Figure 24: rectangle_c

```
1 rectangle_c( size=[40,25], t=[15,5], vr1=[0,0,10,10], vr2=2.5, vrm2=3, co=[0,5], center=true );
```

Definition at line 236 of file shapes2d.scad.

9.2.2.8 module rhombus (size , vr , center = false)

A rhombus.

Parameters

	size	<decimal-list-2 decimal> A list [w, h] of decimals or a single decimal for (w=h).</decimal-list-2 decimal>	
ĺ	vr	<decimal-list-4 decimal> The corner rounding radius. A list [v1r, v2r, v3r, v4r] of decimals or a</decimal-list-4 decimal>	
		single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.	
	center	<boolean> Center about origin.</boolean>	

Example

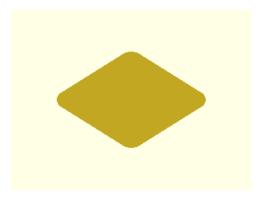


Figure 25: rhombus

```
1 rhombus( size=[40,25], vr=[2,4,2,4], center=true );
```

See Wikipedia for more information.

Definition at line 301 of file shapes2d.scad.

9.2.2.9 module star2d (size, n = 5, vr)

A two-dimensional star.

9.2 2d Shapes 83

Parameters

size	<pre><decimal-list-2 decimal> A list [I, w] of decimals or a single decimal for (size=l=2*w).</decimal-list-2 decimal></pre>	
n	decimal> The number of points.	
vr	<decimal-list-3 decimal> The vertex rounding radius. A list [v1r, v2r, v3r] of decimals or a single</decimal-list-3 decimal>	
	decimal for (v1r=v2r=v3r).	

Example



Figure 26: star2d

```
star2d( size=[40, 15], n=5, vr=2 );
```

Definition at line 1300 of file shapes2d.scad.

```
9.2.2.10 module triangle_aas ( a1 , a2 , s , x = 1, vr , v1r , v2r , v3r , centroid = false, incenter = false )
```

A general triangle specified by a side, one adjacent angle and the opposite angle.

Parameters

a1	<decimal> The opposite angle 1 in degrees.</decimal>	
a2	decimal> The adjacent angle 2 in degrees.	
S	decimal> The side length.	
X	<integer $>$ The side to draw on the positive x-axis (x=1 for s).	
vr	<decimal> The default vertex rounding radius.</decimal>	
v1r	decimal> Vertex 1 rounding radius.	
v2r	<decimal> Vertex 2 rounding radius.</decimal>	
v3r	<decimal> Vertex 3 rounding radius.</decimal>	
centroid	<boolean> Center centroid at origin.</boolean>	
incenter	<boolean> Center incenter at origin.</boolean>	

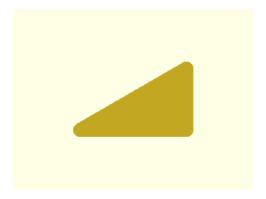


Figure 27: triangle_aas

```
triangle_aas( a1=60, a2=30, s=40, vr=2, centroid=true );
```

See Wikipedia for more information.

Definition at line 895 of file shapes2d.scad.

```
9.2.2.11 module triangle_asa ( a1 , s , a2 , x = 1, vr, v1r, v2r, v3r, centroid = false, incenter = false )
```

A general triangle specified by a side and two adjacent angles.

Parameters

a1	<decimal> The adjacent angle 1 in degrees.</decimal>	
s	<decimal> The side length adjacent to the angles.</decimal>	
a2	<decimal> The adjacent angle 2 in degrees.</decimal>	
X	cinteger $>$ The side to draw on the positive x-axis (x=1 for s).	
vr	<decimal> The default vertex rounding radius.</decimal>	
v1r	<decimal> Vertex 1 rounding radius.</decimal>	
v2r	<decimal> Vertex 2 rounding radius.</decimal>	
v3r	<decimal> Vertex 3 rounding radius.</decimal>	
centroid	<boolean> Center centroid at origin.</boolean>	
incenter	<boolean> Center incenter at origin.</boolean>	

Example

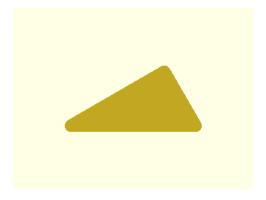


Figure 28: triangle_asa

triangle_asa(a1=30, s=50, a2=60, vr=2, centroid=true);

9.2 2d Shapes 85

See Wikipedia for more information.

Definition at line 810 of file shapes2d.scad.

```
9.2.2.12 module triangle_lp ( v , vr , centroid = false, incenter = false )
```

A general triangle specified by a list of its three vertices.

Parameters

V	<point-2d-list-3> A list [v1, v2, v3] of points [x, y].</point-2d-list-3>	
vr	<decimal-list-3 decimal> The vertex rounding radius. A list [v1r, v2r, v3r] of decimals or a single</decimal-list-3 decimal>	
	decimal for (v1r=v2r=v3r).	
centroid	<boolean> Center centroid at origin.</boolean>	
incenter	<boolean> Center incenter at origin.</boolean>	

Example

```
t = triangle_sss2lp( 30, 40, 50 );
r = [2, 4, 6];
triangle_lp( v=t, vr=r );
```

Definition at line 479 of file shapes2d.scad.

```
9.2.2.13 module triangle_ls ( v , vr , centroid = false, incenter = false )
```

A general triangle specified by a list of its three side lengths.

Parameters

V	<decimal-list-3> A list [s1, s2, s3] of decimals.</decimal-list-3>	
vr	<decimal-list-3 decimal> The vertex rounding radius. A list [v1r, v2r, v3r] of decimals or a single</decimal-list-3 decimal>	
	decimal for (v1r=v2r=v3r).	
centroid	<boolean> Center centroid at origin.</boolean>	
incenter	 <boolean> Center incenter at origin.</boolean>	

Example

```
t = triangle_sss2lp( 3, 4, 5 );
s = triangle_lp2ls( t );
triangle_ls( v=s, vr=2, centroid=true );
```

Definition at line 590 of file shapes2d.scad.

```
9.2.2.14 module triangle_ls_c ( vs , vc , co , cr = 0, vr , vr1 , vr2 , centroid = false , incenter = false )
```

A general triangle specified by its sides with a removed triangular core.

VS	<pre><decimal-list-3 decimal> The size. (s1=s2=s3).</decimal-list-3 decimal></pre>	A list [s1, s2, s3] of decimals or a single decimal for
VC	<pre><decimal-list-3 decimal> The core. (s1=s2=s3).</decimal-list-3 decimal></pre>	A list [s1, s2, s3] of decimals or a single decimal for

СО	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>
vr	<decimal-list-3 decimal> The default vertex rounding radius. A list [v1r, v2r, v3r] of decimals or</decimal-list-3 decimal>
	a single decimal for (v1r=v2r=v3r).
vr1	<decimal-list-3 decimal> The outer vertex rounding radius.</decimal-list-3 decimal>
vr2	<decimal-list-3 decimal> The core vertex rounding radius.</decimal-list-3 decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>

Example

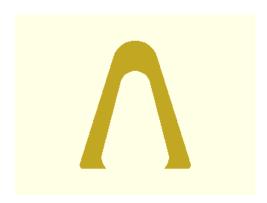


Figure 29: triangle_ls_c

```
1 \\ \\ \text{triangle\_ls\_c( vs=[30,50,50], vc=[20,40,40], co=[0,-4], vr1=[1,1,6], vr2=4, centroid=true );}
```

Note

The outer and inner triangles centroids are aligned prior to the core removal.

Definition at line 645 of file shapes2d.scad.

```
9.2.2.15 module triangle_ppp ( v1 , v2 , v3 , vr , v1r , v2r , v3r , centroid = false , incenter = false )
```

A general triangle specified by three vertices.

Parameters

V1	<pre><point-2d> A point [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A point [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A point [x, y] for vertex 3.</point-2d></pre>
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>

9.2 2d Shapes 87

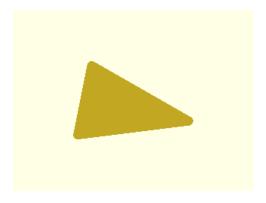


Figure 30: triangle_ppp

```
triangle_ppp( v1=[0,0], v2=[5,25], v3=[40,5], vr=2, centroid=true );
```

Warning

Currently, in order to round any vertex, all must be given a rounding radius, either via vr or individually.

Todo Replace the hull() operation with calculated tangential intersection of the rounded vertexes.

Remove the all or nothing requirement for vertex rounding.

Definition at line 402 of file shapes2d.scad.

```
9.2.2.16 module triangle_sa ( x , y , aa , oa , vr , v1r , v2r , v3r , centroid = false , incenter = false )
```

A right-angled triangle specified by a side length and an angle.

Parameters

X	<decimal> The length of the side along the x-axis.</decimal>
У	<decimal> The length of the side along the y-axis.</decimal>
aa	<decimal> The adjacent angle in degrees.</decimal>
oa	<decimal> The opposite angle in degrees.</decimal>
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	 boolean> Center centroid at origin.
incenter	<boolean> Center incenter at origin.</boolean>

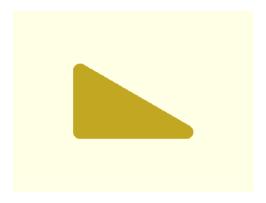


Figure 31: triangle_sa

triangle_sa(x=40, aa=30, vr=2, centroid=true);

Note

When both x and y are given, both triangles are rendered. When both aa and oa are given, aa is used.

Definition at line 1015 of file shapes2d.scad.

9.2.2.17 module triangle_sas (s1, a, s2, x = 1, vr, v1r, v2r, v3r, centroid = false, incenter = false)

A general triangle specified by two sides and the included angle.

Parameters

s1	<decimal> The length of the side 1.</decimal>
а	<decimal> The included angle in degrees.</decimal>
s2	<decimal> The length of the side 2.</decimal>
X	<pre><integer> The side to draw on the positive x-axis ($x=1$ for $s1$).</integer></pre>
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>

9.2 2d Shapes 89

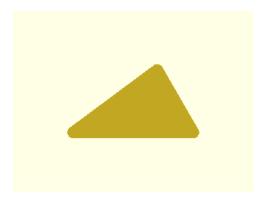


Figure 32: triangle_sas

```
triangle_sas( s1=50, a=60, s2=30, vr=2, centroid=true );
```

See Wikipedia for more information.

Definition at line 739 of file shapes2d.scad.

```
9.2.2.18 module triangle_ss ( x, y, vr, v1r, v2r, v3r, centroid = false, incenter = false )
```

A right-angled triangle specified by its opposite and adjacent side lengths.

Parameters

X	<decimal> The length of the side along the x-axis.</decimal>
У	<decimal> The length of the side along the y-axis.</decimal>
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>

Example

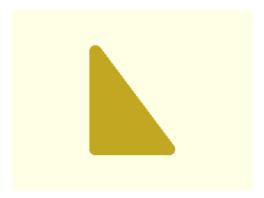


Figure 33: triangle_ss

```
1 triangle_ss( x=30, y=40, vr=2, centroid=true );
```

Definition at line 972 of file shapes2d.scad.

9.2.2.19 module triangle_sss (s1 , s2 , s3 , vr , v1r , v2r , v3r , centroid = false, incenter = false)

A general triangle specified by its three side lengths.

Parameters

s1	<decimal> The length of the side 1 (along the x-axis).</decimal>
s2	<decimal> The length of the side 2.</decimal>
s3	<decimal> The length of the side 3.</decimal>
vr	<decimal> The default vertex rounding radius.</decimal>
v1r	<decimal> Vertex 1 rounding radius.</decimal>
v2r	<decimal> Vertex 2 rounding radius.</decimal>
v3r	<decimal> Vertex 3 rounding radius.</decimal>
centroid	<boolean> Center centroid at origin.</boolean>
incenter	<boolean> Center incenter at origin.</boolean>

Example

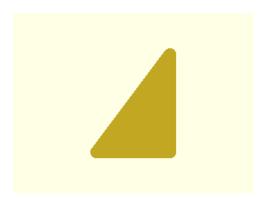


Figure 34: triangle_sss

triangle_sss(s1=30, s2=40, s3=50, vr=2, centroid=true);

See Wikipedia for more information.

Definition at line 529 of file shapes2d.scad.

9.3 3d Shapes 91

9.3 3d Shapes

Three-dimensional geometric shapes.

Collaboration diagram for 3d Shapes:



Files

· file shapes3d.scad

Three-dimensional basic shapes.

Functions

• module cone (r, h, d, vr, vr1, vr2)

A cone.

• module cuboid (size, vr, vrm=0, center=false)

A cuboid with edge, fillet, or chamfer corners.

• module ellipsoid (size)

An ellipsoid.

• module ellipsoid_s (size, a1=0, a2=0)

A sector of an ellipsoid.

module pyramid_t (size, center=false)

A pyramid with trilateral base formed by four equilateral triangles.

• module pyramid_q (size, center=false)

A pyramid with quadrilateral base.

• module star3d (size, n=5, half=false)

A three-dimensional star.

module torus_rp (size, core, r, l, t, co, cr=0, vr, vr1, vr2, vrm=0, vrm1, vrm2, pa=0, ra=360, m=255, center=false, profile=false)

A rectangular cross-sectional profile revolved about the z-axis.

• module torus_tp (size, core, r, l, co, cr=0, vr, vr1, vr2, pa=0, ra=360, m=255, centroid=false, incenter=false, profile=false,)

A triangular cross-sectional profile revolved about the z-axis.

• module torus_ep (size, core, r, l, t, a1=0, a2=0, co, cr=0, pa=0, ra=360, m=255, profile=false)

An elliptical cross-sectional profile revolved about the z-axis.

9.3.1 Detailed Description

Three-dimensional geometric shapes.

9.3.2 Function Documentation

9.3.2.1 module cone (r, h, d, vr, vr1, vr2)

A cone.

Parameters

r	<decimal> The base radius.</decimal>
h	<decimal> The height.</decimal>
d	<decimal> The base diameter.</decimal>
vr	<decimal> The default corner rounding radius.</decimal>
vr1	<decimal> The base corner rounding radius.</decimal>
vr2	<decimal> The point corner rounding radius.</decimal>

Example



Figure 35: cone

```
1 cone( h=25, r=10, vr=2 );
```

Definition at line 107 of file shapes3d.scad.

9.3.2.2 module cuboid (size, vr, vrm = 0, center = false)

A cuboid with edge, fillet, or chamfer corners.

Parameters

size	(decimal-list-3 decimal> A list [x, y, z] of decimals or a single decimal for (x=y=z).	
vr	<decimal> The corner rounding radius.</decimal>	
vrm	<integer> The radius mode. A 2-bit encoded integer that indicates edge and vertex finish. B0</integer>	
	controls edge and B1 controls vertex.	
center	 <boolean> Center about origin.</boolean>	

9.3 3d Shapes 93

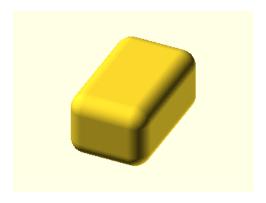


Figure 36: cuboid

cuboid(size=[25,40,20], vr=5, center=true);

vrm	B1	В0	Description
0	0	0	fillet edges with fillet
			vertexes
1	0	1	chamfer edges with
			sphere vertexes
2	1	0	fillet edges with chamfer
			vertexes
3	1	1	chamfer edges with
			chamfer vertexes

Note

Using *fillet* replaces all edges with a quarter circle of radius vr, inset [vr, vr] from the each edge. Using *chamfer* replaces all edges with isosceles right triangles with side lengths equal to the corner rounding radius vr. Therefore the chamfer length will be vr*sqrt(2) at 45 degree angles.

Definition at line 172 of file shapes3d.scad.

9.3.2.3 module ellipsoid (size)

An ellipsoid.

Parameters

size	<decimal-list-2 decimal> A list [w, h] of decimals or a single decimal for (w=h).</decimal-list-2 decimal>
0.20	

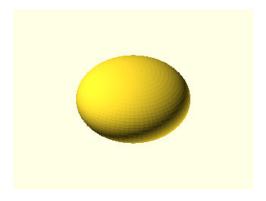


Figure 37: ellipsoid

1 ellipsoid(size=[40,25]);

Definition at line 257 of file shapes3d.scad.

9.3.2.4 module ellipsoid_s (size, a1 = 0, a2 = 0)

A sector of an ellipsoid.

Parameters

size	size <decimal-list-2 decimal> A list [w, h] of decimals or a single decimal for (w=h).</decimal-list-2 decimal>	
a1	<decimal> The start angle in degrees.</decimal>	
a2	<decimal> The stop angle in degrees.</decimal>	

Example



Figure 38: ellipsoid_s

1 ellipsoid_s(size=[60,15], a1=0, a2=270);

Definition at line 289 of file shapes3d.scad.

9.3.2.5 module pyramid_q (size , center = false)

A pyramid with quadrilateral base.

9.3 3d Shapes 95

Parameters

size	<pre><decimal-list-3 decimal> A list [x, y, z] of decimals or a single decimal for (x=y=z).</decimal-list-3 decimal></pre>
center	 boolean> Center about origin.

Example



Figure 39: pyramid_q

pyramid_q(size=[35,20,5], center=true);

Todo Support vertex rounding radius.

Definition at line 391 of file shapes3d.scad.

9.3.2.6 module pyramid_t (size , center = false)

A pyramid with trilateral base formed by four equilateral triangles.

Parameters

size	<decimal> The face radius.</decimal>
center	 <boolean> Center about origin.</boolean>

Example

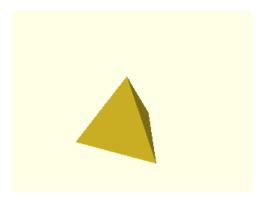


Figure 40: pyramid_t

pyramid_t(size=20, center=true);

Todo Support vertex rounding radius.

Definition at line 347 of file shapes3d.scad.

9.3.2.7 module star3d (size, n = 5, half = false)

A three-dimensional star.

Parameters

size	<pre><decimal-list-3 decimal> A list [I, w, h] of decimals or a single decimal for (size=l=2*w=4*h).</decimal-list-3 decimal></pre>
n	<decimal> The number of points.</decimal>
half	<boolean> Render upper half only.</boolean>

Example



Figure 41: star3d

star3d(size=40, n=5, half=true);

Definition at line 437 of file shapes3d.scad.

9.3.2.8 module torus_ep (size, core, r, I, t, a1 = 0, a2 = 0, co, cr = 0, pa = 0, ra = 360, m = 255, profile = false)

An elliptical cross-sectional profile revolved about the z-axis.

size	<pre><decimal-list-2 decimal> The profile size. A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>
core	<decimal-list-2 decimal $>$ The profile core. A list [x, y] of decimals or a single decimal for (x=y).
r	<decimal> The rotation radius.</decimal>
1	<pre><decimal-list-2 decimal> The elongation length. A list [x, y] of decimals or a single decimal for</decimal-list-2 decimal></pre>
	(x=y)
t	<pre><decimal-list-2 decimal> The profile thickness. A list [x, y] of decimals or a single decimal for</decimal-list-2 decimal></pre>
	(x=y).
a1	<decimal> The profile start angle in degrees.</decimal>
a2	<decimal> The profile stop angle in degrees.</decimal>

9.3 3d Shapes 97

СО	<pre><decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2></pre>
cr	<decimal> Core z-rotation.</decimal>
ра	<decimal> The profile pitch angle in degrees.</decimal>
ra	<decimal> The rotation sweep angle in degrees.</decimal>
m	<integer> The section render mode. An 8-bit encoded integer that indicates the revolution</integer>
	sections to render.
profile	<boolean> Show profile only (do not extrude).</boolean>

See also

rotate_extrude_tre for description of extrude parameters.

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

Example

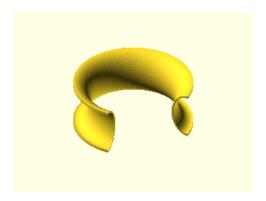


Figure 42: torus_ep

```
torus_ep( size=[20,15], t=[2,4], r=50, a1=0, a2=180, pa=90, ra=270, co=[0,2] );
```

Definition at line 659 of file shapes3d.scad.

```
9.3.2.9 module torus_rp( size, core, r, I, t, co, cr = 0, vr, vr1, vr2, vrm = 0, vrm1, vrm2, pa = 0, ra = 360, m = 255, center = false, profile = false)
```

A rectangular cross-sectional profile revolved about the z-axis.

size	<pre><decimal-list-2 decimal> The profile size. A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>
core	<pre><decimal-list-2 decimal> The profile core. A list [x, y] of decimals or a single decimal for (x=y).</decimal-list-2 decimal></pre>
r	<decimal> The rotation radius.</decimal>

1	<pre><decimal-list-2 decimal> The elongation length. A list [x, y] of decimals or a single decimal for</decimal-list-2 decimal></pre>
	(x=y)
t	<pre><decimal-list-2 decimal> The profile thickness. A list [x, y] of decimals or a single decimal for</decimal-list-2 decimal></pre>
	(x=y).
со	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>
vr	<decimal-list-4 decimal> The profile default corner rounding radius. A list [v1r, v2r, v3r, v4r] of</decimal-list-4 decimal>
	decimals or a single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.
vr1	<decimal-list-4 decimal> The profile outer corner rounding radius.</decimal-list-4 decimal>
vr2	<decimal-list-4 decimal> The profile core corner rounding radius.</decimal-list-4 decimal>
vrm	<integer> The default corner radius mode. A 4-bit encoded integer that indicates each corner</integer>
	finish. Use bit value 0 for <i>fillet</i> and 1 for <i>chamfer</i> .
vrm1	<integer> The outer corner radius mode.</integer>
vrm2	<integer> The core corner radius mode.</integer>
ра	<decimal> The profile pitch angle in degrees.</decimal>
ra	<decimal> The rotation sweep angle in degrees.</decimal>
m	<integer> The section render mode. An 8-bit encoded integer that indicates the revolution</integer>
	sections to render.
center	 <boolean> Rotate about profile center.</boolean>
profile	<boolean> Show profile only (do not extrude).</boolean>

See also

rotate_extrude_tre for description of extrude parameters.

Thickness t

- core = size t; when t and size are given.
- size = core + t; when t and core are given.

Example

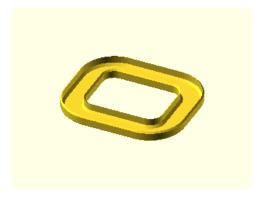


Figure 43: torus_rp

 $1 \\ \text{torus_rp(size=[40,20], core=[35,20], r=40, l=[90,60], co=[0,2.5], vr=4, vrm=15, center=true);}$

Definition at line 520 of file shapes3d.scad.

9.3 3d Shapes 99

9.3.2.10 module torus_tp (size, core, r, I, co, cr = 0, vr, vr1, vr2, pa = 0, ra = 360, m = 255, centroid = false, incenter = false, profile = false)

A triangular cross-sectional profile revolved about the z-axis.

Parameters

size	<pre><decimal-list-3 decimal> The size. A list [s1, s2, s3] of decimals or a single decimal for</decimal-list-3 decimal></pre>
	(s1=s2=s3).
core	<decimal-list-3 decimal> The core. A list [s1, s2, s3] of decimals or a single decimal for</decimal-list-3 decimal>
	(s1=s2=s3).
r	<decimal> The rotation radius.</decimal>
1	<pre><decimal-list-2 decimal> The elongation length. A list [x, y] of decimals or a single decimal for</decimal-list-2 decimal></pre>
	(x=y)
СО	<decimal-list-2> Core offset. A list [x, y] of decimals.</decimal-list-2>
cr	<decimal> Core z-rotation.</decimal>
vr	<decimal-list-3 decimal=""> The default vertex rounding radius. A list [v1r, v2r, v3r] of decimals or</decimal-list-3>
	a single decimal for (v1r=v2r=v3r).
vr1	<decimal-list-3 decimal> The outer vertex rounding radius.</decimal-list-3 decimal>
vr2	<decimal-list-3 decimal> The core vertex rounding radius.</decimal-list-3 decimal>
ра	<decimal> The profile pitch angle in degrees.</decimal>
ra	<decimal> The rotation sweep angle in degrees.</decimal>
m	<integer> The section render mode. An 8-bit encoded integer that indicates the revolution</integer>
	sections to render.
centroid	<boolean> Rotate about profile centroid.</boolean>
incenter	 <boolean> Rotate about profile incenter.</boolean>
profile	<boolean> Show profile only (do not extrude).</boolean>

See also

rotate_extrude_tre for description of extrude parameters.

Example



Figure 44: torus_tp

torus_tp(size=40, core=30, r=60, co=[0,-4], vr=4, pa=90, ra=270, centroid=true);

Note

The outer and inner triangles centroids are aligned prior to the core removal.

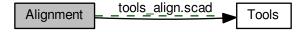
Definition at line 592 of file shapes3d.scad.

9.4 Alignment 101

9.4 Alignment

Shape alignment tools.

Collaboration diagram for Alignment:



Files

• file tools_align.scad

Shape alignment tools.

Functions

• module orient_ll (l=z_axis3d_ul, rl=z_axis3d_ul, r=0)

Orient a line or vector to a reference line or vector.

module align_ll (l=z_axis3d_ul, rl=z_axis3d_ul, ap=0, rp=0, r=0, to=origin3d, ro=zero3d)

Align a line or vector to a reference line or vector.

• module align_l (rl=z_axis3d_ul, rp=0, r=0, to=origin3d, ro=zero3d, d=z_axis_ci)

Align a shapes' x, y, or z Cartesian axis to reference line or vector.

9.4.1 Detailed Description

Shape alignment tools.

9.4.2 Function Documentation

9.4.2.1 module align_I (rI = z_axis3d_uI, rp = 0, r = 0, to = origin3d, ro = zero3d, d = z_axis_ci)

Align a shapes' x, y, or z Cartesian axis to reference line or vector.

rl	= 3d line-2d > The reference line or vector.
rp	<integer> The reference-line alignment point (see table).</integer>
r	<pre><decimal> Roll about axis rl (in degrees).</decimal></pre>
to	<pre><vector-3d vector-2d> Translation offset about rl.</vector-3d vector-2d></pre>

ro	<pre><decimal-list-1:3 decimal> Rotation offset about rl (in degrees).</decimal-list-1:3 decimal></pre>
d	<integer> The Cartesian axis index to align (0, 1, or 2).</integer>

The origin will be a translated to the specified alignment point for the reference line rl.

ap, rp	alignment point
0	none (no translation)
1	initial
2	median
3	termination
4	initial + termination

See Lines and vectors for argument specification and conventions.

Definition at line 170 of file tools_align.scad.

Align a line or vector to a reference line or vector.

Parameters

1	line-3d line-2d > The line or vector to align.
rl	line-3d line-2d > The reference line or vector.
ар	<integer> The line alignment point (see table).</integer>
rp	<integer> The reference-line alignment point (see table).</integer>
r	<pre><decimal> Roll about axis rl (in degrees).</decimal></pre>
to	<pre><vector-3d vector-2d> Translation offset about rl.</vector-3d vector-2d></pre>
ro	<pre><decimal-list-1:3 decimal> Rotation offset about rl (in degrees).</decimal-list-1:3 decimal></pre>

The specified alignment point for the line 1 will be a translated to the specified alignment point for the reference line r1.

ap, rp	alignment point
0	none (no translation)
1	initial
2	median
3	termination
4	initial + termination

See Lines and vectors for argument specification and conventions.

Definition at line 102 of file tools_align.scad.

9.4.2.3 module orient_ll (
$$I = z_axis3d_ul$$
, $rI = z_axis3d_ul$, $r = 0$)

Orient a line or vector to a reference line or vector.

Parameters

1	= 3d line-2d > The line or vector to align.
rl	= 3d line-2d > The reference line or vector.
r	<pre><decimal> Roll about axis rl (in degrees).</decimal></pre>

See Lines and vectors for argument specification and conventions.

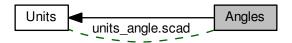
Definition at line 58 of file tools_align.scad.

9.5 Angles 103

9.5 Angles

Angle units and conversions.

Collaboration diagram for Angles:



Files

· file units_angle.scad

Angle units and conversions.

Functions

• function unit_angle_name (u=base_unit_angle)

Return the name of an angle unit identifier.

• function convert_angle (a, from=base_unit_angle, to=base_unit_angle)

Convert an angle from some units to another.

Variables

base_unit_angle = "d"

<string> The base units for angle measurements.

9.5.1 Detailed Description

Angle units and conversions.

These functions allow for angles to be specified with units. Angles specified with units are independent of (base_unit← angle). There are also unit conversion functions for converting from one unit to another.

The table below enumerates the supported units.

units id	description	type
r	radian	decimal
d	degree	decimal
dms	degree, minute, second	decimal-list-3

Example

```
include <units/units_angle.scad>;
base_unit_angle = "d";
```

```
// get base unit name
un = unit_angle_name();

// absolute angle measurements in base unit.
c1 = convert_angle(pi/6, "r");
c2 = convert_angle(pi/4, "r");
c3 = convert_angle(180, "d");
c4 = convert_angle([30, 15, 50], "dms");

// convert between units.
c5 = convert_angle([30, 15, 50], from="dms", to="r");
c6 = convert_angle(0.528205, from="r", to="dms");
```

Result (base_angle_length = **r**):

```
1 ECHO: un = "radian"

2 ECHO: c1 = 0.523599

3 ECHO: c2 = 0.785398

4 ECHO: c3 = 3.14159

5 ECHO: c4 = 0.528205

6 ECHO: c5 = 0.528205

7 ECHO: c6 = [30, 15, 50.102]
```

Result (base_angle_length = **d**):

```
1 ECHO: un = "degree"

2 ECHO: c1 = 30

3 ECHO: c2 = 45

4 ECHO: c3 = 180

5 ECHO: c4 = 30.2639

6 ECHO: c5 = 0.528205

7 ECHO: c6 = [30, 15, 50.102]
```

Result (base_angle_length = **dms**):

```
1 ECHO: un = "degree, minute, second"

2 ECHO: c1 = [29, 59, 60]

3 ECHO: c2 = [45, 0, 0]

4 ECHO: c3 = [180, 0, 0]

5 ECHO: c4 = [30, 15, 50]

6 ECHO: c5 = 0.528205

7 ECHO: c6 = [30, 15, 50.102]
```

9.5.2 Function Documentation

9.5.2.1 function convert_angle (a , from = base_unit_angle, to = base_unit_angle)

Convert an angle from some units to another.

Parameters

а	<decimal decimal-list-3> An angle to convert.</decimal decimal-list-3>
from	<string> The units of the angle to be converted.</string>
to	<string> A units to which the angle should be converted.</string>

Returns

<decimal|decimal-list-3> The conversion result. Returns undef for identifiers that are not defined.

9.5.2.2 function unit_angle_name (u = base_unit_angle)

Return the name of an angle unit identifier.

9.5 Angles 105

Parameters

и	<string> An angle unit identifier.</string>

Returns

<string> The units name for the given angle unit identifier. Returns **undef** for identifiers that are not defined.

9.6 Bitwise

Base-two bitwise binary operations.

Collaboration diagram for Bitwise:



Files

· file math_bitwise.scad

Mathematical base-two bitwise binary functions.

Functions

function bitwise_is_equal (v, b, t=1)

Test if a base-two bit position of an integer value equals a test bit.

• function bitwise_i2v (v, w=1, bv=1)

Encode an integer value as a base-two list of bits.

function bitwise v2i (v)

Decode a base-two list of bits to an integer value.

• function bitwise_i2s (v, w=1)

Encode an integer value as a base-two string of bits.

function bitwise_s2i (v)

Decode a base-two string of bits to an integer value.

• function bitwise_imi (v, w, s)

Decode the integer in a value at a shifted base-two bit mask of width-w.

function bitwise and (v1, v2, bv=1)

Base-two bitwise AND operation for integers.

• function bitwise_or (v1, v2, bv=1)

Base-two bitwise OR operation for integers.

• function bitwise_xor (v1, v2, bv=1)

Base-two bitwise XOR operation for integers.

• function bitwise_not (v, w=1, bv=1)

Base-two bitwise NOT operation for an integer.

function bitwise_lsh (v, s=1, bm=1, bv=1)

Base-two bitwise left-shift operation for an integer.

• function bitwise_rsh (v, s=1)

Base-two bitwise right-shift operation for an integer.

9.6 Bitwise 107

9.6.1 Detailed Description

Base-two bitwise binary operations.

See Wikipedia binary numbers and operations for more information.

See validation results.

9.6.2 Function Documentation

```
9.6.2.1 function bitwise_and (v1, v2, bv = 1)
```

Base-two bitwise AND operation for integers.

Parameters

v1	<integer> An integer value.</integer>
v2	<integer> An integer value.</integer>
bv	(an internal recursion loop variable).

Returns

<integer> result of the base-two bitwise AND of v1 and v2. Returns undef when v1 or v2 is not an integer.

9.6.2.2 function bitwise_i2s (v, w = 1)

Encode an integer value as a base-two string of bits.

Parameters

ſ	V	<integer> An integer value.</integer>
	W	<integer> The minimum bit width.</integer>

Returns

 <bit-string> of bits base-two encoding of the integer value. Returns **undef** when v or w is not an integer.

9.6.2.3 function bitwise_i2v (v, w = 1, bv = 1)

Encode an integer value as a base-two list of bits.

Parameters

V	<integer> An integer value.</integer>
W	<integer> The minimum bit width.</integer>
bv	(an internal recursion loop variable).

Returns

 <bit-list> of bits base-two encoding of the integer value. Returns **undef** when v or w is not an integer.

9.6.2.4 function bitwise_imi (v , w , s)

Decode the integer in a value at a shifted base-two bit mask of width-w.

Parameters

V	<integer> An integer value.</integer>
W	<integer> The bit mask width.</integer>
s	<integer> The bit mask shift offset.</integer>

Returns

<integer> value of the w bits of v starting at bit position s up to bit (w+s-1).

9.6.2.5 function bitwise_is_equal (v, b, t = 1)

Test if a base-two bit position of an integer value equals a test bit.

Parameters

V	<integer> An integer value.</integer>
b	<integer> A base-two bit position.</integer>
t	

Returns

<booklean> true when the base-two bit position of the integer value equals t, otherwise returns false.

9.6.2.6 function bitwise_lsh (v, s = 1, bm = 1, bv = 1)

Base-two bitwise left-shift operation for an integer.

Parameters

V	v <integer> An integer value.</integer>	
S	<integer> The number of bits to shift.</integer>	
bm (an internal recursion loop variable).		
bv	(an internal recursion loop variable).	

Returns

<integer> result of the base-two bitwise left-shift of v by s bits. Returns undef when v or s is not an integer.

9.6.2.7 function bitwise_not (v, w = 1, bv = 1)

Base-two bitwise NOT operation for an integer.

Parameters

V	<integer> An integer value.</integer>
W	<integer> The minimum bit width.</integer>
bv	(an internal recursion loop variable).

Returns

<integer> result of the base-two bitwise NOT of v. Returns **undef** when v is not an integer.

9.6.2.8 function bitwise_or (v1, v2, bv = 1)

Base-two bitwise OR operation for integers.

9.6 Bitwise 109

Parameters

	v1	<integer> An integer value.</integer>
	v2	<integer> An integer value.</integer>
Ī	bv	(an internal recursion loop variable).

Returns

<integer> result of the base-two bitwise OR of v1 and v2. Returns undef when v1 or v2 is not an integer.

9.6.2.9 function bitwise_rsh (v, s = 1)

Base-two bitwise right-shift operation for an integer.

Parameters

V	<integer> An integer value.</integer>
S	<integer> The number of bits to shift.</integer>

Returns

<integer> result of the base-two bitwise right-shift of v by s bits. Returns **undef** when v or s is not an integer.

9.6.2.10 function bitwise_s2i (v)

Decode a base-two string of bits to an integer value.

Parameters

ν <bit-string> A value encoded as a base-two string of bits.</bit-string>	
---	--

Returns

<integer> value encoding of the base-two string of bits. Returns undef when v is not a string of bit values.

9.6.2.11 function bitwise_v2i (v)

Decode a base-two list of bits to an integer value.

Parameters

V	 bit-list> A value encoded as a base-two list of bits.
---	---

Returns

<integer> value encoding of the base-two list of bits. Returns undef when v is not a list of bit values.

9.6.2.12 function bitwise_xor (v1, v2, bv = 1)

Base-two bitwise XOR operation for integers.

Parameters

V1	v1 <integer> An integer value.</integer>	
v2	<integer> An integer value.</integer>	
bv	(an internal recursion loop variable).	

Returns

<integer> result of the base-two bitwise XOR of v1 and v2. Returns **undef** when v1 or v2 is not an integer.

9.7 Component 111

9.7 Component

Component specifications.

Collaboration diagram for Component:

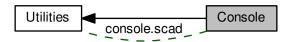


Component specifications.

9.8 Console

Console message logging.

Collaboration diagram for Console:



Files

· file console.scad

Message logging functions.

Functions

• function stack (b=0, t=0)

Format the function call stack as a string.

• module log_echo (m)

Output message to console.

• module log_debug (m)

Output diagnostic message to console.

• module log_info (m)

Output information message to console.

• module log_warn (m)

Output warning message to console.

• module log_error (m)

Output error message to console.

9.8.1 Detailed Description

Console message logging.

Example

```
use <console.scad>;
$log_debug = true;
message = "console log message";

// general
log_echo( message );

// debugging
log_debug( message );
log_debug( message, $log_debug = false );
```

9.8 Console 113

```
// information
log_info( message );

// warning
log_warn( message );

// error
log_error( message );
```

Result

```
1 ECHO: "console log message"
2 ECHO: "[ DEBUG ] root(); console log message"
3 ECHO: "[ INFO ] root(); console log message"
4 ECHO:
5 ECHO: "root()"
6 ECHO: "###################################
7 ECHO: "# [ WARNING ] console log message #"
8 ECHO: "###################################
9 ECHO:
10 ECHO: "root()"
13 ECHO: "##
14 ECHO: "## [ ERROR ] console log message ##"
15 ECHO: "##
```

9.8.2 Function Documentation

9.8.2.1 module log_debug (m)

Output diagnostic message to console.

Parameters

```
m <string> An output message.
```

Message is written if and only if \$log_debug is true.

Definition at line 105 of file console.scad.

```
9.8.2.2 module log_echo ( m )
```

Output message to console.

Parameters

```
m <string> An output message.
```

Definition at line 90 of file console.scad.

```
9.8.2.3 module log_error ( m )
```

Output error message to console.

Parameters

```
m <string> An output message.
```

Output an error message to the console. Ideally, rendering should halt and the script should exit. However, no suitable abort function exists. To alert of the critical error, the error message is also rendered graphically.

Definition at line 166 of file console.scad.

9.8.2.4 module log_info (m)

Output information message to console.

9.8 Console 115

Parameters

т	<string> An output message.</string>
---	--------------------------------------

Definition at line 122 of file console.scad.

9.8.2.5 module log_warn (m)

Output warning message to console.

Parameters

m	<string> An output message.</string>

Definition at line 137 of file console.scad.

9.8.2.6 function stack (b = 0, t = 0)

Format the function call stack as a string.

Parameters

b	<integer> The stack index bottom offset. Return function names above this offset.</integer>	
t	<integer> The stack index top offset. Return function names below this offset.</integer>	

Returns

<string> A string-formatted colon-separated list of functions names for the current function call stack.

Note

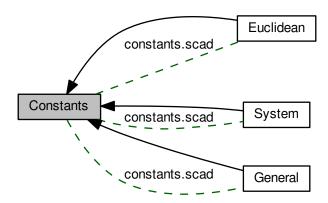
Returns undef when b is greater than the current number of function instances (ie: $b > parent_{\leftarrow} modules-1$).

Returns the string "root ()" when the function call stack is empty (ie: at the root of a script).

9.9 Constants

Design constant definitions.

Collaboration diagram for Constants:



Modules

• Euclidean

Euclidean space axis mapping.

General

General design constants.

• System

System and program limits.

Files

· file constants.scad

Design constant definitions.

9.9.1 Detailed Description

Design constant definitions.

9.10 Coordinates 117

9.10 Coordinates

Coordinate systems and conversions.

Collaboration diagram for Coordinates:



Files

· file units coordinate.scad

Coordinate systems and conversions.

Functions

function coordinates_name (s=base_coordinates)

Return the name of the given coordinate system identifier.

function convert_coordinate (c, from=base_coordinates, to=base_coordinates)

Convert point from one coordinate system to another.

• function coordinates_cpc (c, r, t=false)

Radially scale a list of 2d cartesian coordinates.

• function coordinates_pc (p, r, t=false)

Radially scale and convert a list of 2d polar coordinates to cartesian.

function coordinates csc (c, r, t=false)

Radially scale a list of 3d cartesian coordinates.

• function coordinates_sc (s, r, t=false)

Radially scale and convert a list of 3d spherical coordinates to cartesian.

Variables

```
• base_coordinates = "c"
```

<string> The base coordinate system.

· coordinates_positive_angles = true

boolean> When converting to angular measures add 360 to negative angles.

9.10.1 Detailed Description

Coordinate systems and conversions.

These functions allow for geometric points in space to be specified using multiple coordinate systems. Some geometric calculations are specified more naturally in one or another coordinate system. These conversion functions allow for the movement between the most convenient for a particular application.

For more information see Wikipedia on coordinate system.

The table below enumerates the supported coordinate systems.

system id	description	dimensions	point convention
С	cartesian	2d or 3d	[x, y] or [x, y, z]
р	polar	2d	[r, aa]
у	cylindrical	3d	[r, aa, z]
S	spherical	3d	[r, aa, pa]

The symbols used in the convention column are as follows:

symbol	description	units	reference
x, y, z	coordinate distance	any	xyz-axis
r	radial distance	any	z-axis / xyz-origin
aa	azimuthal angle	degrees	positive x-axis
ра	polar / zenith angle	degrees	positive z-axis

Note

The azimuthal angle is a measure of the radial vector orthogonal projection onto the xy-plane measured from the positive x-axis.

The polar angle is measured from the z-axis (zenith) to the radial vector.

Example

```
include <units/units_coordinate.scad>;
base_coordinates = "c";

// get the base coordinate system name
cs = coordinates_name();

// absolute coordinates in a specified coordinate system.
c1 = convert_coordinate([1, 1, 1], "c");
c2 = convert_coordinate([1, 180], "p");
c3 = convert_coordinate([1, 180], "p");
c4 = convert_coordinate([1, 90, -1], "y");
c4 = convert_coordinate([1, 5, 50], "s");

// convert between system.
c5 = convert_coordinate([10*sqrt(2), 45, 45], from="s", to="y");
c6 = convert_coordinate([sqrt(2), 45], from="p", to="c");
```

Result (base_coordinates = **c**):

```
1 ECHO: cs = "cartesian"

2 ECHO: c1 = [1, 1, 1]

3 ECHO: c2 = [-1, 0]

4 ECHO: c3 = [0, 1, -1]

5 ECHO: c4 = [0.763129, 0.0667652, 0.642788]

6 ECHO: c5 = [10, 45, 10]

7 ECHO: c6 = [1, 1]
```

Result (base_coordinates = **p**):

```
1 ECHO: cs = "polar"
2 ECHO: c1 = [1.41421, 45]
3 ECHO: c2 = [1, 180]
4 ECHO: c3 = [1, 90]
5 ECHO: c4 = [0.766044, 5]
6 ECHO: c5 = [10, 45, 10]
7 ECHO: c6 = [1, 1]
```

Result (base coordinates = y):

9.10 Coordinates 119

```
1 ECHO: cs = "cylindrical"

2 ECHO: c1 = [1.41421, 45, 1]

3 ECHO: c2 = [1, 180, 0]

4 ECHO: c3 = [1, 90, -1]

5 ECHO: c4 = [0.766044, 5, 0.642788]

6 ECHO: c5 = [10, 45, 10]

7 ECHO: c6 = [1, 1]
```

Result (base_coordinates = s):

```
1 ECHO: cs = "spherical"

2 ECHO: c1 = [1.73205, 45, 54.7356]

3 ECHO: c2 = [undef, 180, undef]

4 ECHO: c3 = [1.41421, 90, 135]

5 ECHO: c4 = [1, 5, 50]

6 ECHO: c5 = [10, 45, 10]

7 ECHO: c6 = [1, 1]
```

9.10.2 Function Documentation

9.10.2.1 function convert coordinate (c, from = base coordinates, to = base coordinates)

Convert point from one coordinate system to another.

Parameters

(c ·	<point> A point to convert.</point>
fron	7	<string> The coordinate system identifier of the point to be converted.</string>
to)	<string> The coordinate system identifier to which the point should be converted.</string>

Returns

<point> The converted result. Returns **undef** for identifiers that are not defined.

9.10.2.2 function coordinates_cpc (c , r , t = false)

Radially scale a list of 2d cartesian coordinates.

Parameters

С	<pre><coords-2d> A list of cartesian coordinates [[x, y],].</coords-2d></pre>
r	<decimal> A polar radius.</decimal>
t	<boolean> Translate or scale radius.</boolean>

Returns

<coords-2d> A list of scaled cartesian coordinates.

When t is **true**, all coordinates will terminate on a circle of radius r. When t is **false**, the radius of each coordinate is scaled by r.

```
9.10.2.3 function coordinates_csc(c, r, t = false)
```

Radially scale a list of 3d cartesian coordinates.

Parameters

С	<pre><coords-3d> A list of cartesian coordinates [[x, y, z],].</coords-3d></pre>
r	<decimal> A spherical radius.</decimal>
t	<boolean> Translate or scale radius.</boolean>

Returns

<coords-3d> A list of scaled cartesian coordinates.

When t is **true**, all coordinates will terminate on a sphere of radius r. When t is **false**, the radius of each coordinate is scaled by r.

9.10.2.4 function coordinates_name (s = base_coordinates)

Return the name of the given coordinate system identifier.

Parameters

S	<string> A coordinate system identifier.</string>
---	---

Returns

<string> The system name for the given identifier. Returns **undef** for identifiers that are not defined.

9.10.2.5 function coordinates_pc (p , r , t = false)

Radially scale and convert a list of 2d polar coordinates to cartesian.

Parameters

ſ	С	<pre><coords-2d> A list of polar coordinates [[r, aa],].</coords-2d></pre>
ſ	r	<decimal> A polar radius.</decimal>
ſ	t	<boolean> Translate or scale radius.</boolean>

Returns

<coords-2d> A list of scaled cartesian coordinates.

When t is **true**, all coordinates will terminate on a circle of radius r. When t is **false**, the radius of each coordinate is scaled by r.

9.10.2.6 function coordinates_sc(s,r,t = false)

Radially scale and convert a list of 3d spherical coordinates to cartesian.

Parameters

С	<pre><coords-3d> A list of spherical coordinates [[r, aa, pa],].</coords-3d></pre>
r	<decimal> A spherical radius.</decimal>
t	<boolean> Translate or scale radius.</boolean>

Returns

<coords-3d> A list of scaled cartesian coordinates.

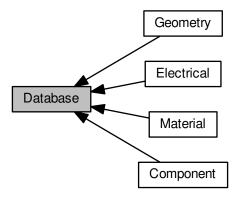
When t is **true**, all coordinates will terminate on a sphere of radius r. When t is **false**, the radius of each coordinate is scaled by r.

9.11 Database 121

9.11 Database

Design specification data.

Collaboration diagram for Database:



Modules

Component

Component specifications.

• Electrical

Electrical specifications.

Geometry

Predefined geometry.

Material

Material specifications.

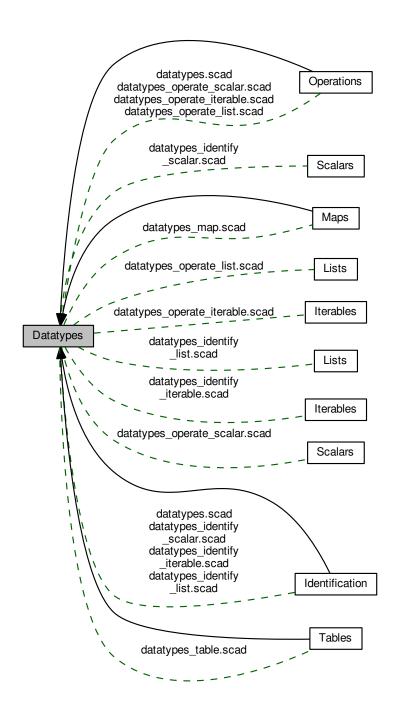
9.11.1 Detailed Description

Design specification data.

9.12 Datatypes

Data type definitions and operators.

Collaboration diagram for Datatypes:



9.12 Datatypes 123

Modules

Identification

Compile-time data type identification and tests.

Maps

Map data type operations.

Operations

Data type operation.

Tables

Table data type operations.

Files

· file datatypes.scad

Data type identification and operations.

file datatypes_identify_scalar.scad

Scalar data type identification.

file datatypes_identify_iterable.scad

Iterable data type identification.

file datatypes_identify_list.scad

List data type identification.

file datatypes_operate_scalar.scad

Scalar data type operations.

• file datatypes_operate_iterable.scad

Iterable data type operations.

• file datatypes_operate_list.scad

List data type operations.

• file datatypes_map.scad

Map data type operations.

• file datatypes_table.scad

Table data type operations.

9.12.1 Detailed Description

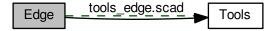
Data type definitions and operators.

See Data types for nomenclature, assumptions, and conventions used to specify values and data types throughout the library.

9.13 Edge

Shape edge finishing tools.

Collaboration diagram for Edge:



Files

• file tools_edge.scad

Shape edge finishing tools.

Functions

• module edge_profile_r (r, p=0, f=1, a=90,)

A 2d edge-finish profile specified by intersection radius.

• module edge_add_r (r, l=1, p=0, f=1, m=3, ba=45, a1=0, a2=90, center=false)

A 3d edge-finish additive shape specified by intersection radius.

9.13.1 Detailed Description

Shape edge finishing tools.

9.13.2 Function Documentation

9.13.2.1 module edge_add_r(
$$r, l = 1, p = 0, f = 1, m = 3, ba = 45, a1 = 0, a2 = 90, center = false)$$

A 3d edge-finish additive shape specified by intersection radius.

Parameters

r	<decimal> The radius length.</decimal>
1	<decimal> The edge length.</decimal>
р	<integer> The profile identifier.</integer>
f	<decimal> The mid-point offset factor.</decimal>
m	<integer> The end finish mode: (B0: bottom, B1: top).</integer>
ba	<decimal> The end bevel angle.</decimal>

9.13 Edge 125

a1	<decimal> The edge plane start angle.</decimal>
a2	<decimal> The edge plane end angle.</decimal>
center	 boolean> Center length about origin.

Example

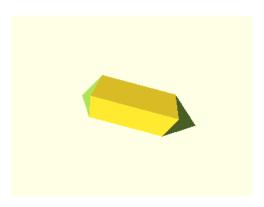


Figure 45: edge_add_r

rotate([90,-90,90]) edge_add_r(r=5, 1=20, f=5/8, center=true);

m	B1	В0	Description
0	0	0	cut bottom (-z) and top
			(+z)
1	0	1	bevel bottom and cut top
2	1	0	cut bottom and bevel top
3	1	1	bevel bottom and top

See edge_profile_r() for description of available profiles.

Definition at line 208 of file tools_edge.scad.

9.13.2.2 module edge_profile_r (r, p = 0, f = 1, a = 90)

A 2d edge-finish profile specified by intersection radius.

Parameters

r	<decimal> The radius length.</decimal>
р	<integer> The profile identifier.</integer>
f	<decimal> The mid-point offset factor.</decimal>
а	<decimal> The sweep angle.</decimal>

Example

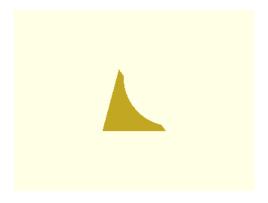


Figure 46: edge_profile_r

edge_profile_r(r=5, p=1, f=1+10/100, a=75);

Profiles:

р	Description
0	Two segment bevel with mid inflection
1	A cove with cut-out offset
2	A quarter round with offset

Note

An offset factor greater than 1 moves the mid-point away from the profile edge-vertex. A factor less than 1 move it inwards towards the edge-vertex.

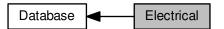
Definition at line 114 of file tools_edge.scad.

9.14 Electrical 127

9.14 Electrical

Electrical specifications.

Collaboration diagram for Electrical:

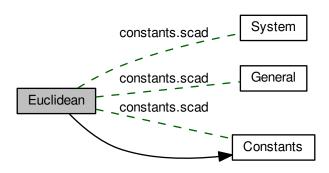


Electrical specifications.

9.15 Euclidean

Euclidean space axis mapping.

Collaboration diagram for Euclidean:



Files

· file constants.scad

Design constant definitions.

Variables

• x_axis_ci = 0

<integer> The coordinate axis index for the Euclidean space x-axis.

y_axis_ci = 1

<integer> The coordinate axis index for the Euclidean space y-axis.

• z axis ci = 2

<integer> The coordinate axis index for the Euclidean space z-axis.

• zero2d = [0, 0]

<decimal-list-2> A 2d zero vector (a list with two zeros).

• origin2d = [0, 0]

<point-2d> The origin point coordinate in 2d Euclidean space.

• x_axis2d_uv = [1, 0]

< vector-2d> The unit vector of the positive x-axis in 2d Euclidean space.

• y_axis2d_uv = [0, 1]

<vector-2d> The unit vector of the positive y-axis in 2d Euclidean space.

• x_axis2d_ul = [-x_axis2d_uv, +x_axis2d_uv]

< A positively-directed unit line centered on the x-axis in 2d Euclidean space.</p>

y_axis2d_ul = [-y_axis2d_uv, +y_axis2d_uv]

A positively-directed unit line centered on the y-axis in 2d Euclidean space.

• zero3d = [0, 0, 0]

9.15 Euclidean 129

```
• origin3d = [0, 0, 0]
      <point-3d> The origin point coordinate in 3-dimensional Euclidean space.
• x_axis3d_uv = [1, 0, 0]
      <vector-3d> The unit vector of the positive x-axis in 3d Euclidean space.
• y_axis3d_uv = [0, 1, 0]
      <vector-3d> The unit vector of the positive y-axis in 3d Euclidean space.
• z = [0, 0, 1]
      <vector-3d> The unit vector of the positive z-axis in 3d Euclidean space.
• x_axis3d_ul = [-x_axis3d_uv, +x_axis3d_uv]
      < line-3d> A positively-directed unit line centered on the x-axis in 3d Euclidean space.
y_axis3d_ul = [-y_axis3d_uv, +y_axis3d_uv]
      A positively-directed unit line centered on the y-axis in 3d Euclidean space.

    z_axis3d_ul = [-z_axis3d_uv, +z_axis3d_uv]

      < A positively-directed unit line centered on the z-axis in 3d Euclidean space.</p>
xy_plane_on = [origin3d, z_axis3d_uv]
      <plane> The right-handed xy plane centered at the origin with normal vector.
yz_plane_on = [origin3d, x_axis3d_uv]
      <plane> The right-handed yz plane centered at the origin with normal vector.
zx_plane_on = [origin3d, y_axis3d_uv]
      <plane> The right-handed zx plane centered at the origin with normal vector.
• xy_plane_os = [origin3d, [for (r=[[1,1],[1,-1],[-1,-1],[-1,1]]) [r[0],r[1],0]]]
      <plane> The right-handed xy plane centered at the origin with coplanar unit square points.

    yz_plane_os = [origin3d, [for (r=[[1,1],[1,-1],[-1,-1],[-1,1]]) [0,r[0],r[1]]]]

      <plane> The right-handed yz plane centered at the origin with coplanar unit square points.

    zx_plane_os = [origin3d, [for (r=[[1,1],[1,-1],[-1,-1],[-1,1]]) [r[1],0,r[0]]]]

      <plane> The right-handed zx plane centered at the origin with coplanar unit square points.
```

<decimal-list-2> A 3d zero vector (a list with three zeros).

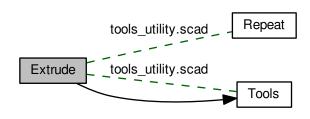
9.15.1 Detailed Description

Euclidean space axis mapping.

9.16 Extrude

Shape extrusion tools.

Collaboration diagram for Extrude:



Files

· file tools_utility.scad

Shape transformation utility tools.

Functions

• module rotate_extrude_tr (r, pa=0, ra=360, profile=false)

Translate, rotate, and revolve the 2d shape about the z-axis.

• module rotate_extrude_tre (r, l, pa=0, ra=360, m=255, profile=false)

Translate, rotate, and revolve the 2d shape about the z-axis with linear elongation.

• module linear_extrude_uls (h, center=false)

Linearly extrude 2d shape with extrusion upper and lower scaling.

9.16.1 Detailed Description

Shape extrusion tools.

9.16.2 Function Documentation

9.16.2.1 module linear_extrude_uls(h, center = false)

Linearly extrude 2d shape with extrusion upper and lower scaling.

Parameters

h	<decimal-list-3:9 decimal> A list of decimals or a single decimal to specify simple extrusion</decimal-list-3:9 decimal>
	height.

9.16 Extrude 131

center	<boolean> Center extrusion about origin.</boolean>

When h is a decimal, the shape is extruded linearly as normal. To scale the upper and lower slices of the extrusion, h must be assigned a list with a minimum of three decimal values as described in the following table.

sym	h[n]	default	description
h	0		total extrusion height
n1	1		(+z) number of scaled
			extrusion slices
h1	2		(+z) extrusion scale
			percentage
x1	3	-h1	(+z) x-dimension scale
			percentage
y1	4	x1	(+z) y-dimension scale
			percentage
n2	5	n1	(-z) number of scaled
			extrusion slices
h2	6	h1	(-z) extrusion scale
			percentage
x2	7	x1	(-z) x-dimension scale
			percentage
y2	8	y1	(-z) y-dimension scale
			percentage

Example

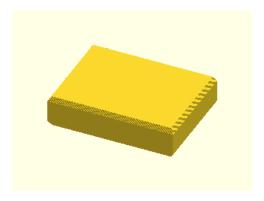


Figure 47: linear_extrude_uls

linear_extrude_uls([5,10,15,-5], center=true) square([20,15], center=true);

Note

When symmetrical scaling is desired, shape must be centered about origin.

Todo This function should be rewritten to use the built-in scaling provided by linear_extrude() in the upper and lower scaling zones.

Definition at line 233 of file tools_utility.scad.

9.16.2.2 module rotate_extrude_tr (r, pa = 0, ra = 360, profile = false)

Translate, rotate, and revolve the 2d shape about the z-axis.

Parameters

r	<decimal> The rotation radius.</decimal>
ра	<decimal> The profile pitch angle in degrees.</decimal>
ra	<decimal> The rotation sweep angle in degrees.</decimal>
profile	<boolean> Show profile only (do not extrude).</boolean>

Example



Figure 48: rotate_extrude_tr

```
1 rotate_extrude_tr( r=50, pa=45, ra=270 ) square( [10,5], center=true );
```

Definition at line 103 of file tools_utility.scad.

```
9.16.2.3 module rotate_extrude_tre (r, I, pa = 0, ra = 360, m = 255, profile = false)
```

Translate, rotate, and revolve the 2d shape about the z-axis with linear elongation.

Parameters

r	<decimal> The rotation radius.</decimal>
1	<decimal-list-2 decimal> The elongation length. A list [x, y] of decimals or a single decimal for</decimal-list-2 decimal>
	(x=y)
ра	<decimal> The profile pitch angle in degrees.</decimal>
ra	<decimal> The rotation sweep angle in degrees.</decimal>
m	<integer> The section render mode. An 8-bit encoded integer that indicates the revolution</integer>
	sections to render. Bit values 1 enables the corresponding section and bit values 0 are disabled.
	Sections are assigned to the bit position in counter-clockwise order.
profile	<boolean> Show profile only (do not extrude).</boolean>

Example

9.16 Extrude 133



Figure 49: rotate_extrude_tre

```
1 rotate_extrude_tre( r=25, l=[5, 50], pa=45, m=31 ) square( [10,5], center=true );
```

Note

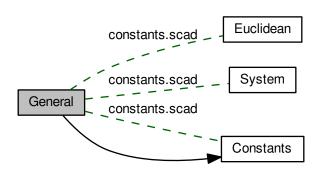
When elongating (1 > 0), ra is ignored. However, m may be used to control which complete revolution section to render.

Definition at line 139 of file tools_utility.scad.

9.17 General

General design constants.

Collaboration diagram for General:



Files

· file constants.scad

Design constant definitions.

Variables

- aeps = 0.001
 - <decimal> Epsilon, small distance to deal with overlapping shapes.
- pi = 3.14159265358979323
 - <decimal> The ratio of a circle's circumference to its diameter.
- tau = 2*pi
 - <decimal> The ratio of a circle's circumference to its radius.
- phi = (1 + sqrt(5)) / 2
 - <decimal> The golden ratio.

9.17.1 Detailed Description

General design constants.

9.18 Geometry 135

9.18 Geometry

Predefined geometry.

Collaboration diagram for Geometry:



Modules

• Polyhedra

Tables of polyhedra vertices, faces, and edges.

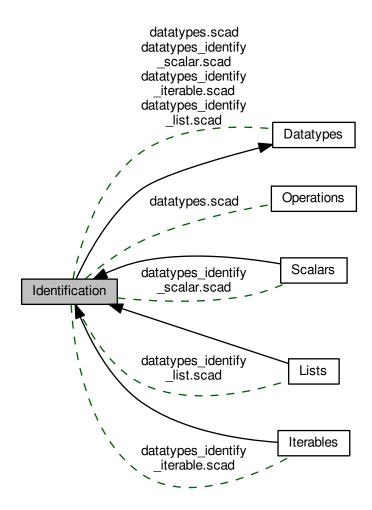
9.18.1 Detailed Description

Predefined geometry.

9.19 Identification

Compile-time data type identification and tests.

Collaboration diagram for Identification:



Modules

• Iterables

Iterable data type identification.

• Lists

List data type identification.

Scalars

Scalar data type identification.

9.19 Identification 137

Files

• file datatypes.scad

Data type identification and operations.

• file datatypes_identify_scalar.scad

Scalar data type identification.

 $\bullet \ \, {\sf file\ datatypes_identify_iterable.scad}$

Iterable data type identification.

• file datatypes_identify_list.scad List data type identification.

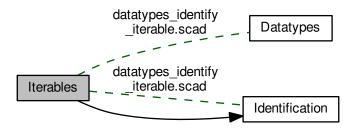
9.19.1 Detailed Description

Compile-time data type identification and tests.

9.20 Iterables

Iterable data type identification.

Collaboration diagram for Iterables:



Files

· file datatypes identify iterable.scad

Iterable data type identification.

Functions

• function all_equal (v, cv)

Test if a list of values equal a comparison value.

• function any_equal (v, cv)

Test if any element of a list of values equal a comparison value.

function all_defined (v)

Test if no element of a list of values is undefined.

• function any_undefined (v)

Test if any element of a list of values is undefined.

function all_scalars (v)

Test if all elements of a list of values are scalars.

• function all_lists (v)

Test if all elements of a list of values are lists.

• function all_strings (v)

Test if all elements of a list of values are strings.

• function all_numbers (v)

Test if all elements of a list of values are numbers.

function all_len (v, l)

Test if all elements of a list of values are lists of a specified length.

9.20 Iterables 139

9.20.1 Detailed Description

Iterable data type identification.

See validation results.

9.20.2 Function Documentation

```
9.20.2.1 function all_defined ( v )
```

Test if no element of a list of values is undefined.

Parameters

V	A list of values.

Returns

<boolean> true when no element is undefined and false otherwise.

Warning

Always returns true when the list is empty.

```
9.20.2.2 function all_equal ( v , cv )
```

Test if a list of values equal a comparison value.

Parameters

V	A list of values.
CV	<value> A comparison value.</value>

Returns

<boolean> true when all elements equal the value cv and false otherwise.

Warning

Always returns true when the list is empty.

```
9.20.2.3 function all_len ( v , I )
```

Test if all elements of a list of values are lists of a specified length.

Parameters

V	A list of values.
1	<integer> The test length.</integer>

Returns

 <boolean> true when all elements are lists of the specified length and false otherwise. Returns true when the
 list is a single list of length 1.

Warning

Always returns true when v is empty.

```
9.20.2.4 function all_lists ( v )
```

Test if all elements of a list of values are lists.

Parameters

```
v st> A list of values.
```

Returns

<booklean> true when all elements are lists and false otherwise. Returns true when the list is a single list value.

Warning

Always returns true when the list is empty.

```
9.20.2.5 function all_numbers ( v )
```

Test if all elements of a list of values are numbers.

Parameters

```
ν st> A list of values.
```

Returns

<boolean> true when all elements are numerical values and false otherwise. Returns true when the list is a single numerical value.

Warning

Always returns true when the list is empty.

```
9.20.2.6 function all_scalars ( v )
```

Test if all elements of a list of values are scalars.

Parameters

```
v st> A list of values.
```

Returns

doolean> true when all elements are scalar values and false otherwise. Returns true when the list is a single scalar value.

Warning

Always returns true when the list is empty.

9.20.2.7 function all_strings (v)

Test if all elements of a list of values are strings.

9.20 Iterables 141

Parameters

V	A list of values.

Returns

 true when all elements are string values and false otherwise. Returns true when the list is a single string value.

Warning

Always returns true when the list is empty.

9.20.2.8 function any_equal (v , cv)

Test if any element of a list of values equal a comparison value.

Parameters

V	A list of values.
CV	<value> A comparison value.</value>

Returns

<boolean> true when any element equals the value CV and false otherwise.

Warning

Always returns false when the list is empty.

9.20.2.9 function any_undefined (v)

Test if any element of a list of values is undefined.

Parameters

V	A list of values.

Returns

<boolean> true when any element is undefined and false otherwise.

Warning

Always returns false when the list is empty.

9.21 Iterables

Iterable data type operations.

Collaboration diagram for Iterables:



Files

· file datatypes_operate_iterable.scad

Iterable data type operations.

Functions

• function edefined_or (v, i, d)

Return an iterable element when it exists or a default value when it does not.

function find (mv, v, c=1, i, i1=0, i2)

Find the occurrences of a match value in an iterable value.

• function count (mv, v, s=true, i)

Count all occurrences of a match value in an iterable value.

• function exists (mv, v, s=true, i)

Check for the existence of a match value in an iterable value.

function first (v)

Return the first element of an iterable value.

function second (v)

Return the second element of an iterable value.

• function third (v)

Return the third element of an iterable value.

• function last (v)

Return the last element of an iterable value.

• function nfirst (v, n=1)

Return a list containing the first n elements of an iterable value.

• function nlast (v, n=1)

Return a list containing the last n elements of an iterable value.

function nhead (v, n=1)

Return a list containing all but the last n elements of an iterable value.

• function ntail (v, n=1)

Return a list containing all but the first n elements of an iterable value.

• function reverse (v)

Reverse the elements of an iterable value.

• function rselect (v, i)

Select a range of elements from an iterable value.

9.21 Iterables 143

• function nssequence (v, n=1, s=1, w=false)

Return a list of all n-element sequential-subsets of an iterable value.

• function eappend (nv, v, r=true, j=true, l=true)

Append a value to each element of an iterable value.

• function insert (nv, v, i=0, mv, mi=0, s=true, si)

Insert a new value into an iterable value.

• function delete (v, i, mv, mc=1, s=true, si)

Delete elements from an iterable value.

function strip (v, mv=empty lst)

Strip all matching values from an iterable value.

• function unique (v)

Return the unique elements of an iterable value.

9.21.1 Detailed Description

Iterable data type operations.

See validation results.

9.21.2 Function Documentation

Count all occurrences of a match value in an iterable value.

Parameters

mv	<value> A match value.</value>
V	<value> An iterable value.</value>
S	<pre><boolean> Use search for element matching (assign false to use find()).</boolean></pre>
i	<integer> The element index to consider for iterable elements.</integer>

Returns

<integer> The number of times my occurs in the list.

```
9.21.2.2 function delete (v, i, mv, mc = 1, s = true, si)
```

Delete elements from an iterable value.

Parameters

V	<value> An iterable value.</value>
i	<range list integer> Deletion Indexes.</range list integer>
mv	<pre>string value> Match value candidates.</pre>
тс	<integer $>$ A match count. For $(mc>=1)$, remove the first mc matches. For $(mc<=0)$, remove
	all matches.

s	<boolean> Use search for element matching (assign false to use find()).</boolean>
si	<integer> The element column index when matching.</integer>

Returns

A list with all specified elements removed. Returns **undef** when i does not map to an element of v. Returns **undef** when v is not defined or is not iterable.

The elements to delete can be specified by an index position, a list of index positions, an index range, an element match value, or a list of element match values (when using search). When mv is a list of match values, all values of mv that exists in v are candidates for deletion. For each matching candidate, mc indicates the quantity to remove. When more than one deletion criteria is specified, the order of precedence is: mv, i.

```
9.21.2.3 function eappend ( nv , v , r = true, j = true, l = true )
```

Append a value to each element of an iterable value.

Parameters

nv	<value> A new value to append.</value>
V	<value> An iterable value.</value>
r	<boolean> Reduce list element value before appending.</boolean>
j	<boolean> Join each appendage as a separate list.</boolean>
1	<boolean> Append new value to last element.</boolean>

Returns

A list with nv appended to each element of v. Returns undef when v is not defined or is not iterable.

Example

```
v1=[["a"], ["b"], ["c"], ["d"]];
v2=[1, 2, 3];
echo( eappend( v2, v1 ) );
echo( eappend( v2, v1, r=false ) );
echo( eappend( v2, v1, j=false, l=false ) );
```

Result

```
ECHO: [["a", 1, 2, 3], ["b", 1, 2, 3], ["c", 1, 2, 3], ["d", 1, 2, 3]]
ECHO: [[["a"], 1, 2, 3], [["b"], 1, 2, 3], [["c"], 1, 2, 3], [["d"], 1, 2, 3]]
ECHO: ["a", 1, 2, 3, "b", 1, 2, 3, "c", 1, 2, 3, "d"]
```

Note

Appending with reduction causes nv to be appended to the *elements* of each iterable value. Otherwise, nv is appended to the iterable value itself.

```
9.21.2.4 function edefined_or ( v , i , d )
```

Return an iterable element when it exists or a default value when it does not.

9.21 Iterables 145

Parameters

V	<value> An iterable value.</value>
i	<integer> An element index.</integer>
d	<value> A default value.</value>

Returns

<value> \lor [i] when it is defined or d otherwise.

9.21.2.5 function exists (mv , v , s = true, i)

Check for the existence of a match value in an iterable value.

Parameters

mv	<value> A match value.</value>
V	<value> An iterable value.</value>
S	<boolean> Use search for element matching (assign false to use find()).</boolean>
i	<integer> The element index to consider for iterable elements.</integer>

Returns

boolean> true when mv exists in the list and false otherwise.

9.21.2.6 function find (mv, v, c = 1, i, i1 = 0, i2)

Find the occurrences of a match value in an iterable value.

Parameters

mv	<value> A match value.</value>
V	<value> An iterable value.</value>
С	<integer> A match count. For $(c>=1)$, return the first c matches. For $(c<=0)$, return all</integer>
	matches.
i	<integer> The element index to consider for iterable elements.</integer>
i1	<integer> The element index where find begins (default: first).</integer>
i2	<integer> The element index where find ends (default: last).</integer>

Returns

A list of indexes where elements match mv. Returns **empty_lst** when no element of v matches mv or when v is not iterable.

The use-cases for find() and search () are summarized in the following tables.

Find:

mv / v	string	list of scalars	list of iterables
scalar		(a)	(b) see note 1
string	(c)		(b) see note 1
list of scalars			(b) see note 1

list of iterables		(b) see note 1

Search:

mv / v	string	list of scalars	list of iterables
scalar		(a)	(b)
string	(d)	invalid	(e) see note 2
list of scalars		(f)	(g)
list of iterables			(g)

Key:

- (a) Identify each element of v that equals mv.
- (b) Identify each element of v where mv equals the element at the specified column index, i, of each iterable value in v.
- (c) If, and only if, mv is a single character, identify each character in v that equals mv.
- (d) For each character of mv, identify where it exists in v. **empty_lst** is returned for each character of mv absent from v.
- (e) For each character of mv, identify where it exists in v either as a numeric value or as a character at the specified column index, i. **empty_lst** is returned for each character of mv absent from v.
- (f) For each scalar of mv, identify where it exists in v. **empty_lst** is returned for each scalar of mv absent from v.
- (g) For each element of mv, identify where it equals the element at the specified column index, i, of each iterable value in v. **empty_lst** is returned for each element of mv absent from v in the specified column index.

Note

- 1: When i is specified, that element column is compared. Otherwise, the entire element is compared. Functions find() and search () behave differently in this regard.
- **2**: Invalid use combination when any element of v is a string. However, an element that is a list of one or more strings is valid. In which case, only the first character of each string element is considered.

9.21.2.7 function first (v)

Return the first element of an iterable value.

Parameters

V	<value> An iterable value.</value>

Returns

<value> The first element of v. Returns undef when v is not defined, is not iterable, or is empty.

Note

Value may also be a range.

9.21.2.8 function insert (nv , v , i = 0, mv , mi = 0, s = true, si)

Insert a new value into an iterable value.

9.21 Iterables 147

Parameters

nv	<value> A new value to insert.</value>
V	<value> An iterable value.</value>
i	<integer> An insert position index.</integer>
mv	<pre>string value> Match value candidates.</pre>
mi	<integer> A match index.</integer>
S	<boolean> Use search for element matching (assign false to use find()).</boolean>
si	<integer> The element column index when matching.</integer>

Returns

A list with nv inserted into v at the specified position. Returns **undef** when no value of mv exists in v.
Returns **undef** when (mi + 1) exceeds the matched element count. Returns **undef** when i does not map to an element of v. Returns **undef** when v is not defined or is not iterable.

The insert position can be specified by an index, an element match value, or list of potential match values (when using search). When multiple matches exists, mi indicates the insert position. When more than one insert position criteria is specified, the order of precedence is: mv, i.

9.21.2.9 function last (v)

Return the last element of an iterable value.

Parameters

V	<value> An iterable value.</value>
---	------------------------------------

Returns

<value> The last element of v. Returns undef when v is not defined, is not iterable, or is empty.

9.21.2.10 function nfirst (v, n = 1)

Return a list containing the first n elements of an iterable value.

Parameters

V	<value> An iterable value.</value>
n	<integer> The element count.</integer>

Returns

A list containing the first n elements of v. Returns undef when v is not defined, is not iterable, or is empty.

9.21.2.11 function nhead (v, n = 1)

Return a list containing all but the last n elements of an iterable value.

Parameters

V	<value> An iterable value.</value>

n	<integer> The element count.</integer>

Returns

<list> A list containing all but the last n elements of v. Returns **empty_lst** when v contains fewer than n elements. Returns **undef** when v is not defined, is not iterable, or is empty.

```
9.21.2.12 function nlast (v, n = 1)
```

Return a list containing the last n elements of an iterable value.

Parameters

V	<value> An iterable value.</value>
n	<integer> The element count.</integer>

Returns

A list containing the last n elements of v. Returns undef when v is not defined, is not iterable, or is empty.

```
9.21.2.13 function nssequence (v, n = 1, s = 1, w = false)
```

Return a list of all n-element sequential-subsets of an iterable value.

Parameters

V	<value> An iterable value.</value>
n	<integer> The number of elements for each subset.</integer>
s	<integer> The iteration step size.</integer>
W	 boolean> Use wrap-at-end circular subset selection.

Returns

A list of all n-element sequential-subsets of v skipping s elements between each subset selection. Returns **empty lst** when v is empty, is not defined or is not iterable.

Example

```
v = [1, 2, 3, 4];
nssequence( v, 3, 1, false ); // [ [1,2,3], [2,3,4] ]
nssequence( v, 3, 1, true ); // [ [1,2,3], [2,3,4], [3,4,1], [4,1,2] ]
```

9.21.2.14 function ntail (v , n = 1)

Return a list containing all but the first n elements of an iterable value.

Parameters

V	<value> An iterable value.</value>
n	<integer> The element count.</integer>

Returns

A list containing all but the first n elements of v. Returns **empty_lst** when v contains fewer than n elements. Returns **undef** when v is not defined, is not iterable, or is empty.

9.21 Iterables 149

9.21.2.15 function reverse (v)

Reverse the elements of an iterable value.

Parameters

V	<value> An iterable value.</value>

Returns

A list containing the elements of v in reversed order. Returns **empty_Ist** when v is empty. Returns **undef** when v is not defined or is not iterable.

9.21.2.16 function rselect (v , i)

Select a range of elements from an iterable value.

Parameters

V	<value> An iterable value.</value>
i	<range list integer> The index selection.</range list integer>

Returns

A list containing the identified element(s). Returns **undef** when i does not map to an element of v. Returns **empty_lst** when v is empty. Returns **undef** when v is not defined or is not iterable.

9.21.2.17 function second (v)

Return the second element of an iterable value.

Parameters

V	<value> An iterable value.</value>
---	------------------------------------

Returns

<value> The second element of v. Returns undef when v is not defined, is not iterable, or is empty.

Note

Value may also be a range.

9.21.2.18 function strip (v , mv = empty_lst)

Strip all matching values from an iterable value.

Parameters

V	<value> An iterable value.</value>
mv	<value> The match value.</value>

Returns

A list with all elements equal to mv removed. Returns undef when v is not defined or is not iterable.

9.21.2.19 function third (v)

Return the third element of an iterable value.

9.21 Iterables 151

Parameters

V	<value> An iterable value.</value>
---	------------------------------------

Returns

<value> The second element of v. Returns undef when v is not defined, is not iterable, or is empty.

Note

Value may also be a range.

9.21.2.20 function unique (v)

Return the unique elements of an iterable value.

Parameters

V	<value> An iterable value.</value>

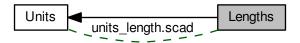
Returns

A list of unique elements with order preserved. Returns **undef** when v is not defined or is not iterable.

9.22 Lengths

Length units and conversions.

Collaboration diagram for Lengths:



Files

· file units_length.scad

Length units and conversions.

Functions

• function unit_length_name (u=base_unit_length, d=1, w=false)

Return the name for a length unit identifier with dimension.

• function convert_length (v, from=base_unit_length, to=base_unit_length, d=1)

Convert a value from from one units to another with dimensions.

Variables

• base_unit_length = "mm"

<string> The base unit for length measurements.

9.22.1 Detailed Description

Length units and conversions.

These functions allow for lengths to be specified with units. Lengths specified with units are independent of (base_ unit_length). There are also unit conversion functions for converting from one unit to another.

The table below enumerates the supported units.

units id	description
pm	picometer
nm	nanometer
um	micrometer
mm	millimeter
cm	centimeter

153 9.22 Lengths

dm	decimeter
m	meter
km	kilometer
thou, mil	thousandth of an inch
in	inch
ft	feet
yd	yard
mi	mile

Example

```
include <units/units_length.scad>;
     base_unit_length = "mm";
     // get base unit name
     un = unit_length_name();
     // absolute length measurements in base unit.
    c1 = convert_length(1/8, "in");
c2 = convert_length(3.175, "mm");
    c3 = convert_length(25, "mil");
c4 = convert_length(1, "ft", d=3);
     // convert between units.
    c5 = convert_length(10, from="mil", to="in");
c6 = convert_length(10, from="ft", to="mm");
Result (base_unit_length = mm):
1 ECHO: un = "millimeter"
```

```
2 \text{ ECHO: } c1 = 3.175
3 \text{ ECHO: } c2 = 3.175
4 ECHO: c3 = 0.635
5 \text{ ECHO: } c4 = 2.83168e+07
6 ECHO: c5 = 0.01
7 \text{ ECHO: } c6 = 3048
```

Result (base_unit_length = **cm**):

```
1 ECHO: un = "centimeter"
2 \text{ ECHO: } c1 = 0.3175
3 \text{ ECHO: } c2 = 0.3175
4 \text{ ECHO: } c3 = 0.0635
5 \text{ ECHO: } c4 = 28316.8
6 ECHO: c5 = 0.01
7 ECHO: c6 = 3048
```

Result (base_unit_length = **mil**):

```
1 ECHO: un = "thousandth"
2 \text{ ECHO: } c1 = 125
3 \text{ ECHO: } c2 = 125
4 \text{ ECHO: } c3 = 25
5 ECHO: c4 = 1.728e+12
6 ECHO: c5 = 0.01
7 ECHO: c6 = 3048
```

Result (base_unit_length = in):

```
1 ECHO: un = "inch"
2 ECHO: c1 = 0.125
3 \text{ ECHO: } c2 = 0.125
4 \text{ ECHO: } c3 = 0.025
5 \text{ ECHO: } c4 = 1728
6 ECHO: c5 = 0.01
7 \text{ ECHO: } c6 = 3048
```

Example (equivalent lengths)

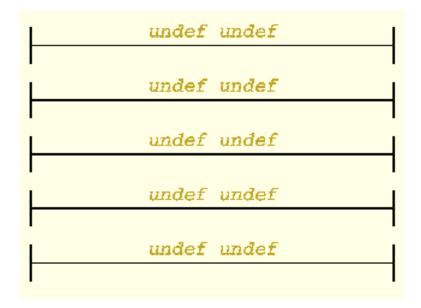


Figure 50: Unit Lengths

9.22.2 Function Documentation

9.22.2.1 function convert_length (v , from = base_unit_length, to = base_unit_length, d = 1)

Convert a value from from one units to another with dimensions.

Parameters

V	<decimal> A value to convert.</decimal>
from	<string> The units of the value to be converted.</string>
to	<string> A units to which the value should be converted.</string>
d	<integer> A dimension. One of [1 2 3].</integer>

Returns

<decimal> The conversion result. Returns undef for identifiers or dimensions that are not defined.

9.22.2.2 function unit_length_name(u = base_unit_length, d = 1, w = false)

Return the name for a length unit identifier with dimension.

Parameters

W	<boolean> true for word and false for symbol format.</boolean>
и	<string> A length unit identifier.</string>
d	<integer> A dimension. One of [1 2 3].</integer>

Returns

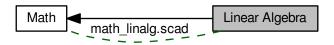
<string> The long name for a length unit identifier with dimension. Returns **undef** for identifiers or dimensions that are not defined.

9.23 Linear Algebra 155

9.23 Linear Algebra

Linear algebra transformations on Euclidean coordinates.

Collaboration diagram for Linear Algebra:



Files

· file math_linalg.scad

Linear algebra mathematical functions.

Functions

• function multmatrix_lp (c, m)

Multiply all coordinates by a 4x4 3d-transformation matrix.

• function translate_lp (c, v)

Translate all coordinates dimensions.

• function rotate_lp (c, a, v, o=origin3d)

Rotate all coordinates about one or more axes in Euclidean 2d or 3d space.

• function scale_lp (c, v)

Scale all coordinates dimensions.

function resize_lp (c, v)

Scale all coordinates dimensions proportionately to fit inside a region.

9.23.1 Detailed Description

Linear algebra transformations on Euclidean coordinates.

9.23.2 Function Documentation

9.23.2.1 function multmatrix_lp (c , m)

Multiply all coordinates by a 4x4 3d-transformation matrix.

Parameters

С	<coords-3d> A list of 3d coordinate points.</coords-3d>
m	<matrix-4x4> A 4x4 transformation matrix (decimal-list-4-list4).</matrix-4x4>

Returns

<coords-3d> A list of 3d coordinate points multiplied by the transformation matrix.

See Wikipedia and multmatrix for more information.

```
9.23.2.2 function resize_lp ( c , v )
```

Scale all coordinates dimensions proportionately to fit inside a region.

Parameters

С	<coords-nd> A list of nd coordinate points.</coords-nd>
V	<decimal-list-n> A list of bounds for each dimension.</decimal-list-n>

Returns

<coords-nd> A list of proportionately scaled coordinate points which exactly fit the region bounds v.

```
9.23.2.3 function rotate_lp ( c , a , v , o = origin3d )
```

Rotate all coordinates about one or more axes in Euclidean 2d or 3d space.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d coordinate points.</coords-3d coords-2d>
а	<decimal-list-3 decimal> The axis rotation angle. A list [ax, ay, az] or a single decimal to specify</decimal-list-3 decimal>
	az only.
V	<vector-3d> An arbitrary axis for the rotation. When specified, the rotation angle will be a or az</vector-3d>
	about the line v that passes through point o .
0	<point-3d $>$ A 3d point origin for the rotation. Ignored when v is not specified.

Returns

<coords-3d|coords-2d> A list of 3d or 2d rotated coordinates. Rotation order is rz, ry, rx.

See Wikipedia for more information on transformation matrix and axis rotation.

9.23.2.4 function scale_lp (c , v)

Scale all coordinates dimensions.

Parameters

С	<coords-nd> A list of nd coordinate points.</coords-nd>
V	<decimal-list-n> A list of scalers for each dimension.</decimal-list-n>

Returns

<coords-nd> A list of scaled coordinate points.

9.23.2.5 function translate_lp (c , v)

Translate all coordinates dimensions.

9.23 Linear Algebra 157

Parameters

С	<coords-nd> A list of nd coordinate points.</coords-nd>
V	<decimal-list-n> A list of translations for each dimension.</decimal-list-n>

Returns

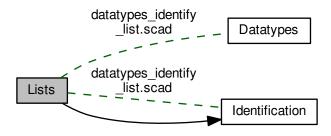
<coords-nd> A list of translated coordinate points.

See Wikipedia for more information and transformation matrix.

9.24 Lists

List data type identification.

Collaboration diagram for Lists:



Files

• file datatypes_identify_list.scad List data type identification.

Functions

• function n_almost_equal (v1, v2, p=6)

Test if all elements of two lists of numbers are sufficiently equal.

• function almost_equal (v1, v2, p=6)

Test if all numerical elements of two lists of values are sufficiently equal.

• function compare (v1, v2, s=true)

Order to lists of arbitrary values.

9.24.1 Detailed Description

List data type identification.

See validation results.

9.24.2 Function Documentation

9.24.2.1 function almost_equal (v1 , v2 , p = 6)

Test if all numerical elements of two lists of values are sufficiently equal.

9.24 Lists 159

Parameters

v1	A list of values 1.
v2	A list of values 2.
р	<number> The numerical precision.</number>

Returns

boolean> true when all elements of each lists are sufficiently equal and false otherwise.

The 'distance' between two numbers must be less than pow(10,-p) to be considered almost equal. All numerical comparisons are performed to the specified precision. All non-numeric comparisons test for equality.

Note

If the lists are scalar numbers, the function n_almost_equal() provides a more efficient test.

Warning

Always returns true when both lists are empty.

9.24.2.2 function compare (v1, v2, s = true)

Order to lists of arbitrary values.

Parameters

V1	A list of values 1.
v2	A list of values 2.
s	<boolean> Order ranges by their numerical sum.</boolean>

Returns

<integer> -1 when (v2 < v1), +1 when (v2 > v1), and 0 when (v2 == v1).

The following table summarizes how values are ordered.

order	type	s	intra-type ordering
1	undef		(singular)
2	number		numerical comparison
3	boolean		false < true
4	string		lexical comparison
5	list		lengths then element-wise
			comparison
6	range	true	compare sum of range
			elements
6	range	false	lengths then element-wise
			comparison

Note

When comparing two lists of equal length, the comparison continue element-by-element until an ordering can be determined. Two lists are equal when all elements have been compared and no ordering has been determined.

Warning

The performance of element-wise comparisons of lists degrades with list size.

The sum of a range may exceeded the intermediate variable storage capacity for long ranges.

9.24.2.3 function n_{almost_equal} (v1 , v2 , p = 6)

Test if all elements of two lists of numbers are sufficiently equal.

9.24 Lists 161

Parameters

v1	<number-list> A list of numbers 1.</number-list>
v2	<number-list> A list of numbers 2.</number-list>
р	<number> The numerical precision.</number>

Returns

<boolean> true when the distance between v1 and v2 is less than d and false otherwise. Returns false when either list contains a non-numerica values, or when the lists are not of the same length.

The 'distance' between two numbers must be less than pow(10,-p) to be considered almost equal.

Note

To compare general lists of values see almost_equal().

9.25 Lists

List data type operations.

Collaboration diagram for Lists:



Files

· file datatypes_operate_list.scad

List data type operations.

Functions

• function Istr (v)

Convert a list of values to a concatenated string.

• function Istr_html (v, b, p, a, f, d=false)

Convert a list of values to a concatenated HTML-formatted string.

• function consts (I, v, u=false)

Create a sequence of constant or incrementing elements.

• function get_index (I, s=true, rs)

Create a sequence for a list index sequence specification.

• function pad (I, w, p=0, r=true)

Pad a list to a constant width of elements.

• function dround (v, d=6)

Round all numerical values of a list to a fixed number of decimal point digits.

• function sround (v, d=6)

Round all numerical values of a list to a fixed number of significant figures.

• function limit (v, I, u)

Limit all numerical values of a list between an upper and lower bounds.

• function sum (v, i1, i2)

Compute the sum of a list of numbers.

• function mean (v)

Compute the mean/average of a list of numbers.

• function ciselect (v, i)

Case-like select a value from a list of ordered value options.

function cmvselect (v, mv)

Case-like select a value from a list of mapped key-value options.

• function eselect (v, f=true, l=false, i)

Select a specified element from each iterable value of a list.

• function smerge (v, r=false)

Serial-merge lists of iterable values.

9.25 Lists 163

• function pmerge (v, j=true)

Parallel-merge lists of iterable values.

function qsort (v, i, r=false)

Sort the numeric or string elements of a list using quick sort.

function qsort2 (v, i, d=0, r=false, s=true)

Hierarchically sort an arbitrary data list using quick sort.

9.25.1 Detailed Description

List data type operations.

See validation results.

9.25.2 Function Documentation

```
9.25.2.1 function ciselect (v, i)
```

Case-like select a value from a list of ordered value options.

Parameters

V	A list of values.
i	<integer> Element selection index.</integer>

Returns

<value> The value of the list element at the specified index. Returns the default value when i does not map to an element.

Behaves like a case statement for selecting values from a list of ordered options. The default value is: last (v).

Example

9.25.2.2 function cmvselect (v , mv)

Case-like select a value from a list of mapped key-value options.

Parameters

V	<matrix-2xn> A matrix of N key-value mapped pairs [[key, value],].</matrix-2xn>
mv	<value> Element selection key match value.</value>

Returns

<value> The value from the map that matches the key mv. Returns the default value when mv does not match any of the element identifiers of v or when mv is undefined.

Behaves like a case statement for selecting values from a list of *mapped options*. The default value is \leftarrow : second(last(v)).

Example

9.25.2.3 function consts (I, v, u = false)

Create a sequence of constant or incrementing elements.

Parameters

1	<integer> The list length.</integer>
V	<value> The element value.</value>
и	 boolean> Element values are undef .

Returns

<list> A list of 1 copies of the element. Returns **empty_lst** when 1 is not a number or if (1 < 1).

Note

When v is not specified and u is false, each element is assigned the value of its index position.

9.25.2.4 function dround (v, d = 6)

Round all numerical values of a list to a fixed number of decimal point digits.

Parameters

V	A list of values.
d	<integer> The (maximum) number of decimals.</integer>

Returns

<list> The list with all numeric values truncated to ${\tt d}$ decimal digits and rounded-up if the following digit is 5 or greater. Non-numeric values are unchanged.

```
9.25.2.5 function eselect ( v , f = true, I = false, i )
```

Select a specified element from each iterable value of a list.

Parameters

V	A list of iterable values.
f	 boolean> Select the first element.
1	 boolean> Select the last element.
i	<integer> Select a numeric element index position.</integer>

Returns

A list containing the selected element of each iterable value of v. Returns **empty_lst** when v is empty. Returns **undef** when v is not defined or is not iterable.

Note

When more than one selection criteria is specified, the order of precedence is: i, l, f.

9.25 Lists 165

```
9.25.2.6 function get_index ( I, s = true, rs )
```

Create a sequence for a list index sequence specification.

Parameters

1	The list.
s	<index> The index sequence specification.</index>
rs	<integer> An optional seed for random sequences.</integer>

Returns

<number-list> An index sequence based on the specification. Returns **empty_lst** for any v that does not fall into one of the specification forms.

See Index sequence for argument specification and conventions.

```
9.25.2.7 function limit ( v , I , u )
```

Limit all numerical values of a list between an upper and lower bounds.

Parameters

V	A list of values.
1	<number> The minimum value.</number>
и	<number> The maximum value.</number>

Returns

The list with all numeric values limited to the range [1 : u]. A value will be assigned 1 when it is less than 1 and u when it is greater than u. Non-numeric values are unchanged.

```
9.25.2.8 function lstr ( v )
```

Convert a list of values to a concatenated string.

Parameters

```
ν list> A list of values.
```

Returns

<string> Constructed by converting each element of the list to a string and concatenating together. Returns **undef** when the list is not defined.

Example

```
v1=["a", "b", "c", "d"];
v2=[1, 2, 3];
echo( lstr(concat(v1, v2)) );
```

Result

```
ECHO: "abcd123"
```

9.25.2.9 function $lstr_html(v, b, p, a, f, d = false)$

Convert a list of values to a concatenated HTML-formatted string.

9.25 Lists 167

Parameters

V	A list of values.
b	<tag-list-list> A list of tag lists. Unpaired HTML tag(s) to add before the value.</tag-list-list>
р	<tag-list-list> A list of tag lists. Paired HTML tag(s) to enclose the value.</tag-list-list>
а	<tag-list-list> A list of tag lists. Unpaired HTML tag(s) to add after the value.</tag-list-list>
f	<attr-list-list> A list of tag attribute lists for fs, where fs=["color", "size", "face"] is</attr-list-list>
	the font tag to enclose the value. Not all attributes are required, but the order is significant.
d	<boolean> Debug. When true angle brackets are replaced with curly brackets to prevent con-</boolean>
	sole decoding.

Returns

<string> Constructed by converting each element of the list to a string with specified HTML markup and concatenating. Returns **undef** when the list is not defined.

When there are fewer tag lists in b, p, a, or f, than there are value elements in v, the last specified tag list is used for all subsequent value elements.

For a list of the paired and unpaired HTML tags supported by the console see: HTML subset.

Example

```
echo( lstr_html(v="bold text", p="b", d=true) );
echo( lstr_html(v=[1,"x",3], f=[["red",6,"helvetica"],undef,["blue",10,"courier"]], d=true) );

v = ["result", "=", "mc", "2"];
b = ["hr", undef];
p = ["i", undef, ["b", "i"], ["b","sup"]];
a = concat(consts(3, u=true), "hr");
f = [undef, ["red"], undef, ["blue",4]];
echo( lstr_html(v=v, b=b, p=p, a=a, f=f, d=true) );
```

Result

9.25.2.10 function mean (v)

Compute the mean/average of a list of numbers.

Parameters

V	<number-list range> A list of numerical values or a range.</number-list range>

Returns

<number|number-list> The sum divided by the number of elements. Returns 0 when the list is empty. Returns **undef** when list non-numerical.

See Wikipedia for more information.

```
9.25.2.11 function pad ( I, w, p = 0, r = true )
```

Pad a list to a constant width of elements.

Parameters

1	The list.
W	<integer> The padded width.</integer>
р	<value> The padding value.</value>
r	 boolean> Use right padding (false for left).

Returns

A list padded to w elements.

When the list has greater than w elements, the list is returned unchanged. The empty list, **empty_lst**, has zero elements. When 1 is a string, characters are counted as individual elements. Use function lstr() to join padded values back into a single string if desired.

```
9.25.2.12 function pmerge (v, j = true)
```

Parallel-merge lists of iterable values.

Parameters

V	A list of iterable values.
j	<boolean> Join each merge as a separate list.</boolean>

Returns

A list containing the parallel-wise element concatenation of each iterable value in v. Returns **empty_lst** when any element value in v is empty. Returns **undef** when v is not defined or when any element value in v is not iterable.

Example

```
v1=["a", "b", "c", "d"];
v2=[1, 2, 3];
echo( pmerge( [v1, v2], true ) );
echo( pmerge( [v1, v2], false ) );
```

Result

```
ECHO: [["a", 1], ["b", 2], ["c", 3]]
ECHO: ["a", 1, "b", 2, "c", 3]
```

Note

The resulting list length will be limited by the iterable value with the shortest length. A single string, although iterable, is treated as a merged unit.

```
9.25.2.13 function qsort (v, i, r = false)
```

Sort the numeric or string elements of a list using quick sort.

9.25 Lists 169

Parameters

	ν	<number-list string-list> A list of values.</number-list string-list>
	i	<integer> The sort column index for iterable elements.</integer>
Ī	r	<boolean> Reverse the sort order.</boolean>

Returns

<list> A list with elements sorted in ascending order. Returns **undef** when v is not defined or is not a list.

Warning

This implementation relies on the comparison operators '<' and '>' which expect the operands to be either two scalar numbers or two strings. Therefore, this function will not correctly sort lists elements that are not numbers or strings. Elements with unknown order are placed at the end of the list.

See Wikipedia for more information.

```
9.25.2.14 function qsort2 (v, i, d = 0, r = false, s = true)
```

Hierarchically sort an arbitrary data list using quick sort.

Parameters

V	<data> A list of values.</data>
i	<integer> The sort column index for iterable elements.</integer>
d	<integer> The recursive sort depth.</integer>
r	 boolean> Reverse the sort order.
s	<boolean> Order ranges by their numerical sum.</boolean>

Returns

Vist With all elements sorted in ascending order. Returns under when v is not defined or is not a list.

Elements are ordered using compare(). See its documentation for a description of the parameter s. To recursively sort all elements, set d greater than, or equal to, the maximum level of hierarchy in v.

See Wikipedia for more information.

9.25.2.15 function smerge (v, r = false)

Serial-merge lists of iterable values.

Parameters

V	A list of iterable values.
r	<boolean> Recursively merge elements that are iterable.</boolean>

Returns

A list containing the serial-wise element concatenation of each element in v. Returns **empty_lst** when v is empty. Returns **undef** when v is not defined.

Note

A single string, although iterable, is treated as a merged unit.

9.25.2.16 function sround (v, d = 6)

Round all numerical values of a list to a fixed number of significant figures.

9.25 Lists 171

Parameters

V	A list of values.
d	<integer> The (maximum) number of significant figures.</integer>

Returns

The list with all numeric values rounded-up to d significant figures. Non-numeric values are unchanged.

See Wikipedia for more information.

9.25.2.17 function sum (v, i1, i2)

Compute the sum of a list of numbers.

Parameters

V	<number-list range> A list of numerical values or a range.</number-list range>
i1	<integer> The element index at which to begin summation (first when not specified).</integer>
i2	<integer> The element index at which to end summation (last when not specified).</integer>

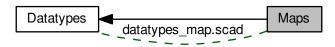
Returns

<number|number-list> The sum over the index range. Returns 0 when the list is empty. Returns **undef** when list non-numerical.

9.26 Maps

Map data type operations.

Collaboration diagram for Maps:



Files

• file datatypes_map.scad

Map data type operations.

Functions

function get_map_i (m, k)

Return the index of a map key.

• function map_exists (m, k)

Test if a key exists.

function get_map_v (m, k)

Get the map value associated with a key.

• function get_map_kl (m)

Get a list of all map keys.

• function get_map_vl (m)

Get a list of all map values.

• function get_map_size (m)

Get the number of map entries.

module map_check (m, verbose=false)

Perform some basic validation/checks on a map.

• module map_dump (m, sort=true, number=true, p=3)

Dump each map entry to the console.

9.26.1 Detailed Description

Map data type operations.

Example

```
use <datatypes/datatypes_map.scad>;
map =
[
["part1", ["screw10", [10, 11, 13]]],
```

9.26 Maps 173

```
["screw12", [20, 21, 30]]], ["screw10", [10, 10, -12]]],
   ["part3",
   ["config",
                      ["top", "front", "rear"]],
   ["version",
                      [21, 5, 0]],
   ["runid",
map_check(map, true);
echo( str("is part0 = ", map_exists(map, "part0")) );
echo( str("is part1 = ", map_exists(map, "part1")) );
p1 = get_map_v(map, "part1");
echo( c=second(p1) );
keys = get_map_kl(map);
parts = delete(keys, mv=["config", "version", "runid"]);
for (p = parts)
  echo
  (
    n=p,
     p=first(get_map_v(map, p)),
     l=second(get_map_v(map, p))
  );
map_dump(map);
```

Result

```
1 ECHO: "[ INFO ] map_check(); begin map check"
2 ECHO: "[ INFO ] map_check(); checking map format and keys."
3 ECHO: "[ INFO ] map_check(); map size: 6 entries."
4 ECHO: "[ INFO ] map_check(); end map check"
5 ECHO: "is part0 = false"
6 ECHO: "is part1 = true"
7 ECHO: c = [10, 11, 13]
8 ECHO: n = "part1", p = "screw10", l = [10, 11, 13]
9 ECHO: n = "part2", p = "screw10", l = [20, 21, 30]
10 ECHO: n = "part3", p = "screw10", l = [10, 10, -12]
11 ECHO: "003: 'config' = '["top", "front", "rear"]'"
12 ECHO: "000: 'part1' = '["screw10", [10, 11, 13]]'"
13 ECHO: "001: 'part2' = '["screw10", [20, 21, 30]]'"
14 ECHO: "002: 'part3' = '["screw10", [10, 10, -12]]'"
15 ECHO: "004: 'version' = '[21, 5, 0]'"
17 ECHO: "map size: 6 entries."
```

9.26.2 Function Documentation

9.26.2.1 function get_map_i (m , k)

Return the index of a map key.

Parameters

m	<matrix-2xn> A list of N key-value map pairs.</matrix-2xn>
k	<string> A map key.</string>

Returns

<integer> The index of the map entry if it exists. Returns undef if key is not a string or does not exists.

9.26.2.2 function get_map_kl (m)

Get a list of all map keys.

Parameters

m	<matrix-2xn> A list of N key-value map pairs.</matrix-2xn>

Returns

<string-list-N> A list of key strings for all N map entries.

9.26.2.3 function get_map_size (m)

Get the number of map entries.

Parameters

m	<matrix-2xn> A list of N key-value map pairs.</matrix-2xn>
---	--

Returns

<integer> The number of map entries.

9.26.2.4 function get_map_v (m , k)

Get the map value associated with a key.

Parameters

m	<matrix-2xn> A list of N key-value map pairs.</matrix-2xn>	
k	k <string> A map key.</string>	

Returns

<value> The value associated with key. Returns **undef** if key does not exists.

9.26.2.5 function get_map_vl (m)

Get a list of all map values.

Parameters

т	m <matrix-2xn> A list of N key-value map pairs.</matrix-2xn>
---	--

Returns

A list of values for all N map entries.

9.26.2.6 module map_check(m, verbose = false)

Perform some basic validation/checks on a map.

Parameters

m	<matrix-2xn> A list of N key-value map pairs.</matrix-2xn>

9.26 Maps 175

verbose	 <boolean> Be verbose during check.</boolean>	1
---------	--	---

Check that: (1) each entry has key-value 2-tuple, (2) each key is a string, and (3) key identifiers are unique.

Definition at line 149 of file datatypes_map.scad.

```
9.26.2.7 module map_dump ( m, sort = true, number = true, p = 3 )
```

Dump each map entry to the console.

Parameters

m	atrix-2xN> A list of N key-value map pairs.	
sort	oolean> Sort the output by key.	
number	<boolean> Output index number.</boolean>	
р	<integer> Number of places for zero-padded numbering.</integer>	

Definition at line 223 of file datatypes_map.scad.

```
9.26.2.8 function map_exists ( m , k )
```

Test if a key exists.

Parameters

m	<matrix-2xn> A list of N key-value map pairs.</matrix-2xn>	
k	k <string> A map key.</string>	

Returns

<boolean> true when the key exists and false otherwise.

9.27 Material

Material specifications.

Collaboration diagram for Material:



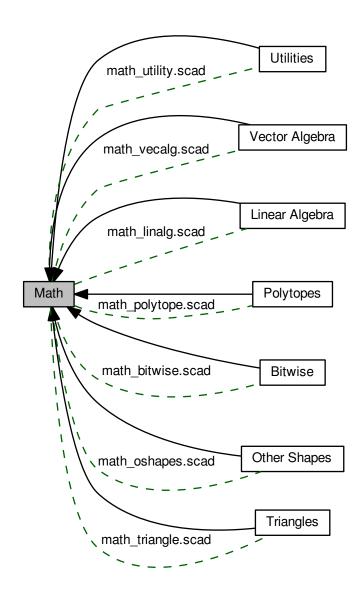
Material specifications.

9.28 Math 177

9.28 Math

Mathematical functions.

Collaboration diagram for Math:



Modules

Bitwise

Base-two bitwise binary operations.

• Linear Algebra

Linear algebra transformations on Euclidean coordinates.

· Other Shapes

Mathematical functions for other shapes.

Polytopes

Polygon and polyhedron mathematical functions.

Triangles

Triangle mathematical functions.

Utilities

Miscellaneous mathematical utilities.

Vector Algebra

Algebraic operations on Euclidean vectors.

Files

· file math.scad

Mathematical function primitives.

· file math bitwise.scad

Mathematical base-two bitwise binary functions.

• file math_linalg.scad

Linear algebra mathematical functions.

• file math_oshapes.scad

Other shapes mathematical functions.

file math_polytope.scad

Polygon and polyhedron mathematical functions.

· file math_triangle.scad

Triangle solutions mathematical functions.

· file math_utility.scad

Miscellaneous mathematical utilities.

• file math_vecalg.scad

Vector algebra mathematical functions.

9.28.1 Detailed Description

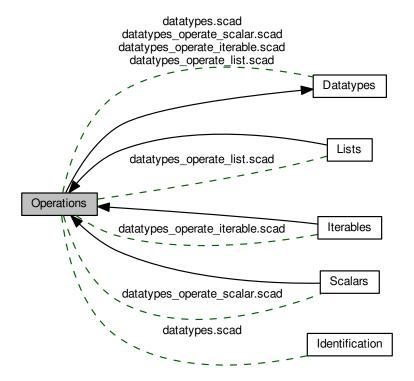
Mathematical functions.

9.29 Operations 179

9.29 Operations

Data type operation.

Collaboration diagram for Operations:



Modules

• Iterables

Iterable data type operations.

Lists

List data type operations.

Scalars

Scalar data type operations.

Files

· file datatypes.scad

Data type identification and operations.

• file datatypes_operate_scalar.scad

Scalar data type operations.

file datatypes_operate_iterable.scad

Iterable data type operations.

• file datatypes_operate_list.scad List data type operations.

9.29.1 Detailed Description

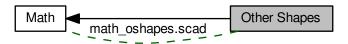
Data type operation.

9.30 Other Shapes 181

9.30 Other Shapes

Mathematical functions for other shapes.

Collaboration diagram for Other Shapes:



Files

· file math_oshapes.scad

Other shapes mathematical functions.

Functions

• function rpolygon_lp (n, r, a, vr, cw=true)

Compute the coordinates for an n-sided regular polygon.

• function rpolygon_area (n, r, a)

Compute the area of an n-sided regular polygon.

• function rpolygon_perimeter (n, r, a)

Compute the perimeter of an n-sided regular polygon.

9.30.1 Detailed Description

Mathematical functions for other shapes.

9.30.2 Function Documentation

9.30.2.1 function rpolygon_area (n , r , a)

Compute the area of an n-sided regular polygon.

Parameters

n	<integer> The number of sides.</integer>	
r	<decimal> The vertex circumradius of the circumcircle.</decimal>	
а	<decimal> The inradius of the incircle.</decimal>	

Returns

<decimal> Area of the n-sided regular polygon.

The radius can be specified by either the circumradius r or the inradius a. If both are specified, r is used.

9.30.2.2 function rpolygon_lp(n, r, a, vr, cw =true)

Compute the coordinates for an n-sided regular polygon.

9.30 Other Shapes 183

Parameters

n	<integer> The number of sides.</integer>	
r	<decimal> The vertex circumradius of the circumcircle.</decimal>	
а	<decimal> The inradius of the incircle.</decimal>	
vr	<decimal> The vertex rounding radius.</decimal>	
CW	 boolean> Use clockwise point ordering.	

Returns

```
<coords-2d> A list of coordinates points [[x, y], ...].
```

The radius can be specified by either the circumradius r or the inradius a. If both are specified, r is used.

Example

```
vr=5;
hull()
{
    for ( p = rpolygon_lp( r=20, n=5, vr=vr ) )
        translate( p )
        circle( r=vr );
}
```

9.30.2.3 function rpolygon_perimeter (n , r , a)

Compute the perimeter of an n-sided regular polygon.

Parameters

n	<integer> The number of sides.</integer>	
r	<decimal> The vertex circumradius of the circumcircle.</decimal>	
а	a <decimal> The inradius of the incircle.</decimal>	

Returns

<decimal> Perimeter length of the n-sided regular polygon.

The radius can be specified by either the circumradius r or the inradius a. If both are specified, r is used.

9.31 Parts

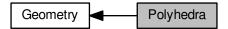
Parametric parts and assemblies.

Parametric parts and assemblies.

9.32 Polyhedra

Tables of polyhedra vertices, faces, and edges.

Collaboration diagram for Polyhedra:



Files

· file anti_prisms.scad

Table of polyhedra data group: anti_prisms.

• file archimedean_duals.scad

Table of polyhedra data group: archimedean_duals.

· file archimedean.scad

Table of polyhedra data group: archimedean.

· file cupolas.scad

Table of polyhedra data group: cupolas.

· file dipyramids.scad

Table of polyhedra data group: dipyramids.

· file johnson.scad

Table of polyhedra data group: johnson.

file platonic.scad

Table of polyhedra data group: platonic.

· file prisms.scad

Table of polyhedra data group: prisms.

· file pyramids.scad

Table of polyhedra data group: pyramids.

· file trapezohedron.scad

Table of polyhedra data group: trapezohedron.

file polyhedra_all.scad

Table of polyhedra data group: polyhedra_all.

Variables

- dtc_polyhedra_anti_prisms
- dtr_polyhedra_anti_prisms
- dtc_polyhedra_archimedean_duals
- dtr_polyhedra_archimedean_duals
- dtc_polyhedra_archimedean
- dtr_polyhedra_archimedean

- dtc_polyhedra_cupolas
- dtr_polyhedra_cupolas
- · dtc polyhedra dipyramids
- dtr_polyhedra_dipyramids
- · dtc polyhedra johnson
- dtr_polyhedra_johnson
- dtc_polyhedra_platonic
- dtr_polyhedra_platonic
- · dtc_polyhedra_prisms
- dtr_polyhedra_prisms
- dtc polyhedra pyramids
- · dtr_polyhedra_pyramids
- · dtc_polyhedra_trapezohedron
- · dtr_polyhedra_trapezohedron
- dtc_polyhedra_polyhedra_all
- dtr_polyhedra_polyhedra_all

9.32.1 Detailed Description

Tables of polyhedra vertices, faces, and edges.

Database of polyhedra vertices, faces, and edges. Classes of polyhedra are grouped into separate files. The file polyhedra_all.scad contains all polyhedra from all files. Each table uses the following column data structure.

feild	description
id	identifier
n	name
0	other name
g	group
d	data source
С	cartesian vertices
S	spherical vertices
f	faces
е	edges

Use the functions get_table_v() to retrieve feild data as show in the following example. To see a list of table identifiers consider the function get_table_ridl(), get_table_cidl(), or module table_dump(). See datatypes_table.scad for other available table functions.

Example

```
include <units/units_coordinate.scad>;
include <tools/tools_polytope.scad>;
include <datatypes/datatypes_table.scad>;
include <datatypes/datatypes_table.scad>;
include <database/geometry/polyhedra/platonic.scad>;

tc = dtc_polyhedra_platonic;
tr = dtr_polyhedra_platonic;
id = "dodecahedron";

pv = get_table_v(tr, tc, id, "c");
pf = get_table_v(tr, tc, id, "f");
sv = coordinates_csc(pv, 100);

polytope_number(sv, pf, to=[0,0,5]);
polytope_frame(sv, pf) {circle(r=2); color("grey") sphere(r=4);}
polyhedron(sv, pf);
```

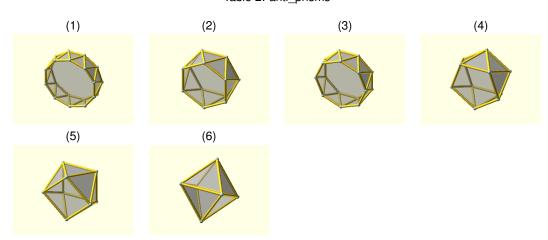
Autotests

```
1 ECHO: "checking table: "
2 ECHO: "colums = dtc_polyhedra_polyhedra_all"
3 ECHO: " rows = dtr_polyhedra_polyhedra_all"
4 ECHO: "[ INFO ] table_check(); begin table check"
5 ECHO: "[ INFO ] table_check(); row identifier found at column zero."
6 ECHO: "[ INFO ] table_check(); checking row column counts."
7 ECHO: "[ INFO ] table_check(); checking for repeat column identifiers."
8 ECHO: "[ INFO ] table_check(); checking for repeat row identifiers."
9 ECHO: "[ INFO ] table_check(); table size: 164 rows by 9 columns."
10 ECHO: "[ INFO ] table_check(); end table check"
12 ECHO: "confirming calculations stored in table:"
13 ECHO: " (1) spherical with converted cartesian coordinates:"
14 ECHO: "
            (2) edge list from faces definition:
15 ECHO: "[ INFO ] root(); edge-list not sorted: id=anti_prisms_triangular_antiprism"
16 ECHO: "[ INFO ] root(); edge-list not sorted: id=archimedean_duals_rhombic_dodecahedron"
17 ECHO: "[ INFO ] root(); edge-list not sorted: id=archimedean_cuboctahedron"
18 ECHO: "[ INFO ] root(); edge-list not sorted: id=cupolas_triangular_gyrobicupola"
19 ECHO: "[ INFO ] root(); edge-list not sorted: id=platonic_octahedron"
20 ECHO:
21 ECHO: "164 polyhedra checked."
```

For more information see Wikipedia on Polyhedron.

Group: anti_prisms

Table 2: anti prisms



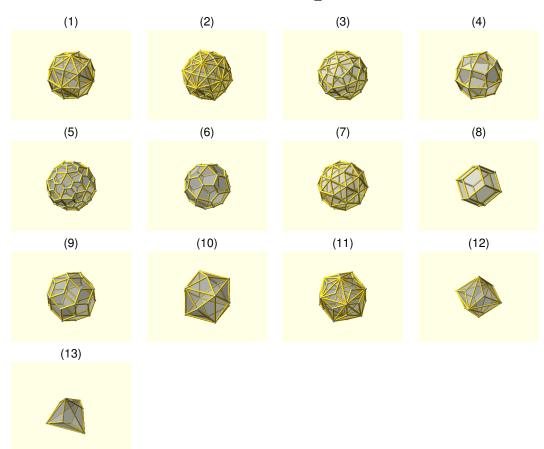
no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1		-	20	22	40	3 ^{x20} ,	20.8 ^{x40} ,	0.597 ^{x40}	36 ^{x20} ,
	decagonal	\leftarrow				10 ^{x2}	35.8 ^{x40} ,		120 ^{x60}
	_←						84.8 ^{x40} ,		
	antiprism						95.2 ^{x40}		

2	-		12	14	24	3 ^{x12} ,	34.8 ^{x24} ,	0.919 ^{x24}	60 ^{x12} ,
	hexagonal←					6 ^{x2}	59.2 ^{x24} ,		120 ^{x36}
	_←						81.1 ^{x24} ,		
	antiprism						98.9 ^{x24}		
3	-		16	18	32	3 ^{x16} ,	26 ^{x32} ,	0.727 ^{x32}	45 ^{x16} ,
	octagonal←					8 ^{x2}	44.7 ^{x32} ,		120 ^{x48}
	_←						83.4 ^{x32} ,		
	antiprism						96.6 ^{x32}		
4	-		10	12	20	3 ^{x10} ,	41.8 ^{x20} ,	1.05 ^{x20}	72^{x10} ,
	pentagonal←	د				5 ^{x2}	70.5^{x20} ,		120 ^{x30}
	_←						79.2 ^{x20} ,		
	antiprism						100.8 ^{x20}		
5	-		8	10	16	3 ^{x8} ,	52.4 ^{x16} ,	1.22 ^{x16}	90 ^{x8} ,
	square←					4 ^{x2}	76.2 ^{x16} ,		120 ^{x24}
	_←						86.7 ^{x16} ,		
	antiprism						103.8 ^{x16}		
6		Octahe-	6	8	12	3 ^{x8}	70.5 ^{x24} ,	1.41 ^{x12}	120 ^{x24}
	triangular←c	dron					109.5 ^{x24}		
	_←								
	antiprism								

Group: archimedean_duals

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1		Hexakis-	26	48	72	3 ^{x48}	6.9 ^{x48} ,	0.↩	96.7 ^{x48} ,
	disdyakis+	- ←					32.7 ^{x48} ,	606 ^{x24} ,	124.↩
	_←	Octahedro	n				38.2 ^{x96} ,	0.↩	2 ^{x48} ,
	dodecahe	dron					43.1 ^{x96} ,	765 ^{x24} ,	139.1 ^{x48}
							47.3 ^{x96} ,	0.919 ^{x24}	
							50.6 ^{x48} ,		
							54.6 ^{x48} ,		
							55.1 ^{x96}		
2		Hexakis-	62	120	180	3 ^{x120}	15.1 ^{x360} ,	0.37 ^{x60} ,	91 ^{x120} ,
	disdyakis+	- ←						0.↩	121.↩
	_←	Icosahedr	on				21.4 ^{x120} ,	581 ^{x60} ,	8 ^{x120} ,
	triacontah	edron						0.683 ^{x60}	147.←
							26.3 ^{x240} ,		2 ^{x120}
							29 ^{x240} ,		
							30.5^{x120} ,		
							40.3 ^{x240} ,		
							47.7 ^{x240} ,		
							50.4 ^{x120}		

Table 3: archimedean_duals



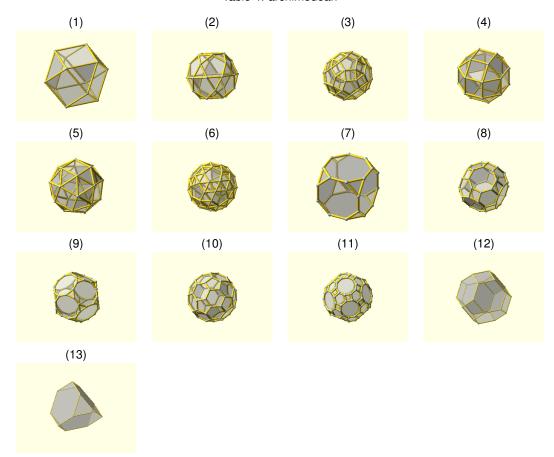
3	kite_← -	62	60	120	4 ^{x60}	25.9 ^{x240} ,	0.37 ^{x60} ,	61.7 ^{x60} ,
	hexecontahedron						0.569 ^{x60}	93 ^{x120} ,
						36.9 ^{x120} ,		112.2 ^{x60}
						42.5 ^{x120}		
4	kite_← -	26	24	48	4 ^{x24}	41.9 ^{x96} ,	0.←	64.7 ^{x24} ,
	icositetrahedron					60.7 ^{x72}	592 ^{x24} ,	98.4 ^{x72}
							0.765 ^{x24}	
5	-	92	60	150	5 ^{x60}	26.8 ^{x300} ,	0.↩	61.9 ^{x240} ,
	pentagonal←						278 ^{x90} ,	
	_←					44.1 ^{x120}	0.486 ^{x60}	112.5 ^{x60}
	hexecontahedron							
6	-	38	24	60	5 ^{x24}	43.7 ^{x120} ,	0.←	65.2 ^{x96} ,
	pentagonal←						436 ^{x36} ,	99.2 ^{x24}
	_					63.5 ^{x24}	0.619 ^{x24}	
	icositetrahedron							

7	pentakis↔	-	32	60	90	3 ^{x60}	23.3 ^{x180} , 38.1 ^{x120} ,	0. <i>←</i> 678 ^{x60} , 0.764 ^{x30}	111.← 4 ^{x60} ,
	_← dodecahed	dron						0.764	124. <i>←</i> 3 ^{<i>x</i>120}
							40.9 ^{x240} ,		
							47.6 ^{x120}		
8		-	14	12	24	4 ^{x12}	60 ^{x48} ,	0.866 ^{x24}	70.5 ^{x24} ,
	rhombic←						90 ^{x24}		109.5 ^{x24}
	_ ←								
	dodecahed	dron				20	×100	V60	ve0
9		-	32	30	60	4 ^{x30}	36 ^{x120} ,	0.691 ^{x60}	63.4 ^{x60} ,
	rhombic↩						60 ^{x120}		116.6 ^{x60}
	-←								
	triacontahe	edron				- ×24	×70		
10		-	14	24	36	3 ^{x24}	36.9 ^{x72} ,	1.06 ^{x24} ,	96.4 ^{x24} ,
	tetrakis←						53.1 ^{x24} ,	1.41 ^{x12}	131.8 ^{x48}
	_←						66.4 ^{x96} ,		
	hexahedro					- v60	78.5 ^{x48}		v60
11	triakis↔	-	32	60	90	3 ^{x60}	19.4 ^{x180} ,	0.↔	61 ^{x60} ,
	-←						×240	717 ^{x60} ,	149. <i>←</i>
	icosahedro	on					37.4 ^{x240} ,	1.24 ^{x30}	5 ^{x120}
							52.3 ^{x240} ,		
							62.4 ^{x240} ,		
							66 ^{x120}		
12	triakis↔	-	14	24	36	3 ^{x24}	32.6 ^{x72} ,	1.17 ^{x24} ,	62.8 ^{x24} ,
	_←						62.6 ^{x96} ,	2 ^{x12}	148.6 ^{x48}
	octahedror	า					85.5 ^{x96} ,		
							94.5 ^{x48}		
13	triakis↩	-	8	12	18	3 ^{x12}	50.5 ^{x36} ,	1.7 ^{x12} ,	67.1 ^{x12} ,
	_←						95.2 ^{x48} ,	2.83 ^{x6}	146.4 ^{x24}
	tetrahedro	n					117 ^{x24}		

Group: archimedean

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1	cubocta-	-	12	14	24	3 ^{x8} ,	54.7 ^{x48} ,	1.41 ^{x24}	90 ^{x24} ,
	hedron					4 ^{x6}	70.5 ^{x24} ,		120 ^{x24}
							90 ^{x24}		
2	icosido-	-	30	32	60	3 ^{x20} ,	37.4 ^{x120} ,	0.65 ^{x60}	72 ^{x60} ,
	decahe-					5 ^{x12}			120 ^{x60}
	dron						41.8 ^{x60} ,		
							63.4 ^{x60}		

Table 4: archimedean



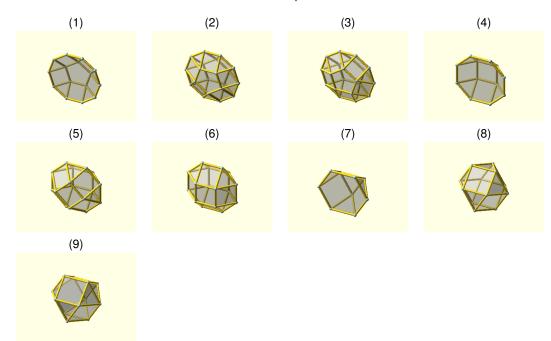
3	rhombi cosido-	60	62	120	3^{x20} , 4^{x30} , 5^{x12}	20.9 ^{x120} ,	0.46 ^{x120}	$ 72^{x60}, 90^{x120}, 120^{x60} $
	decahe- dron				5^2	31.7^{x120} , 36^{x120} , 37.4^{x120}		120^00
4	rhom- bicuboc- tahe- dron	24	26	48	3 ^{x8} , 4 ^{x18}	35.3 ^{x48} , 45 ^{x48} , 54.7 ^{x48} , 60 ^{x48}	0.765 ^{x48}	90 ^{x72} , 120 ^{x24}
5	snub_← - cuboctahedro	24 on	38	60	3 ^{x32} , 4 ^{x6}	26.8 ^{x72} , 37 ^{x48} , 45.9 ^{x96} , 50.4 ^{x48} , 54.7 ^{x96}	0.802 ^{x60}	90 ^{×24} , 120 ^{×96}

6	snub_← Snub	60	92	150	3 ^{x80} ,	15.8 ^{x180} ,	0.↩	72 ^{x60} ,
	icosidodeca Dedie rra-				5 ^{x12}		477 ^{x150}	120 ^{x240}
	hedron					27.1 ^{x120} ,		
						27.3 ^{x240} ,		
						31 ^{x120} , 37.4 ^{x240}		
7	-	24	14	36	3 ^{x8} ,	54.7 ^{x48} ,	0.586 ^{x36}	45 ^{x48} ,
	truncated←				8 ^{x6}	90 ^{x24}		120 ^{x24}
	_cube							
8	Great	48	26	72	4 ^{x12} ,	35.3 ^{x48} ,	0.442 ^{x72}	45 ^{x48} ,
	truncated←Rhom-				6 ^{x8} ,	45 ^{x48} ,		60 ^{x48} , 90 ^{x48}
	_← bicuboc-				8 ^{x6}	54.7 ^{x48}		90 ^{x48}
	cuboctahed tah e-							
	dron							
9	-	60	32	90	3 ^{x20} ,	37.4 ^{x120} ,	0.342 ^{x90}	36 ^{x120} ,
	truncated←				10 ^{x12}			120 ^{x60}
	_←					63.4 ^{x60}		
	dodecahedron							
10	Soccer	60	32	90	5 ^{x12} ,	37.4 ^{x120} ,	0.412 ^{x90}	60 ^{x120} ,
	truncated←Ball				6 ^{x20}			72 ^{x60}
	_←					41.8 ^{x60}		
	icosahedron				20			1/100
11	Great	120	62	180	4×30,	20.9 ^{x120} ,	0.←	36 ^{x120} ,
	truncated Rhombi-				6^{x20} ,	a . =v120	265 ^{x180}	60 ^{x120} ,
	_← cosido-				10 ^{x12}	31.7 ^{x120} ,		90 ^{x120}
	icosidodeca dedabre -					07 4×120		
-10	dron	0.4	4.4		4 ×6	37.4 ^{x120}	4 4 ¥36	00 Y48
12	-	24	14	36	4 ^{x6} ,	54.7 ^{x48} ,	1.41 ^{x36}	60 ^{x48} , 90 ^{x24}
	truncated←				6 ^{x8}	70.5 ^{x24}		90^24
	_ _							
10	octahedron	10		10	0.x4	70 Ex24	0.00×18	60 ^{x24} .
13		12	8	18	3 ^{x4} , 6 ^{x4}	70.5 ^{x24} , 109.5 ^{x12}	2.83 ^{x18}	120 ^{x12}
	truncated←				6^7	109.5^12		120^12
	_ 							
	tetrahedron							

Group: cupolas

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1		J5	15	12	25	3 ^{x5} ,	20.9 ^{x20} ,	0.614 ^{x25}	36 ^{x10} ,
	pentagona	l←				4 ^{x5} ,	31.7 ^{x10} ,		72 ^{x5} ,
	_cupola					5 ^{x1} ,	36 ^{x10} ,		90 ^{x20} ,
						10 ^{x1}	37.4 ^{x10} ,		120 ^{x15}
							142.↩		
							6 ^{x10} ,		
							148.3 ^{x10}		

Table 5: cupolas



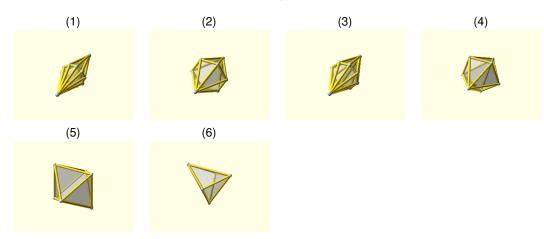
2	J31 pentagonal ← _← gyrobicupola	20	22	40	3 ^{x10} , 4 ^{x10} , 5 ^{x2}	$ \begin{array}{c} 20.9^{\times 40}, \\ 31.7^{\times 20}, \\ 36^{\times 20}, \\ 37.4^{\times 20}, \\ 109. \leftarrow \\ 5^{\times 20}, \\ 110. \leftarrow \\ 9^{\times 20}, \\ 4.005^{\times 20} \end{array} $	0.618 ^{×40}	72 ^{x10} , 90 ^{x40} , 120 ^{x30}
3	J30 pentagonal ← _← orthobicupola	20	22	40	3 ^{x10} , 4 ^{x10} , 5 ^{x2}	$ \begin{array}{c} 120^{x20} \\ 20.9^{x40}, \\ 31.7^{x20}, \\ 36^{x20}, \\ 37.4^{x20}, \\ 105. & \\ 2^{x10}, \\ 114. & \\ 7^{x40}, \\ 116.6^{x10} \end{array} $	0.618 ^{x40}	72 ^{x10} , 90 ^{x40} , 120 ^{x30}

4	square← _cupola	J4	12	10	20	3 ^{x4} , 4 ^{x5} , 8 ^{x1}	35.3 ^{x16} , 45 ^{x8} , 54.7 ^{x8} , 60 ^{x8} , 125.3 ^{x8} ,	0.753 ^{×20}	45 ^{x8} , 90 ^{x20} , 120 ^{x12}
5	square← _← gyrobicup¢		16	18	32	3 ^{x8} , 4 ^x 10	35.3 ^{x32} , 45 ^{x16} , 54.7 ^{x16} , 60 ^{x16} , 80.3 ^{x16} , 82.1 ^{x16} , 98.4 ^{x16}	0.765 ^{x32}	90 ^{x40} , 120 ^{x24}
6	square ↔ _ ↔ orthobicup	J28 ola	16	18	32	3 ^{x8} , 4 ^{x10}	35.3 ^{x32} , 45 ^{x16} , 54.7 ^{x16} , 60 ^{x16} , 70.5 ^{x8} , 90 ^{x40}	0.765 ^{x32}	90 ^{x40} , 120 ^{x24}
7	triangular∢ _cupola	J3 -	9	8	15	3 ^{x4} , 4 ^{x3} , 6 ^{x1}	54.7 ^{x18} , 70.5 ^{x6} , 90 ^{x6} , 109.5 ^{x6} ,	0.965 ^{x15}	60 ^{x6} , 90 ^{x12} , 120 ^{x12}
8	triangular _← gyrobicupe	dron	12	14	24	3 ^{x8} , 4 ^{x6}	54.7 ^{x48} , 70.5 ^{x24} , 90 ^{x24}	1 ^{×24}	90 ^{x24} , 120 ^{x24}
9	triangular _← orthobicup		12	14	24	3 ^{x8} , 4 ^{x6}	38.9 ^{x6} , 54.7 ^{x36} , 70.5 ^{x18} , 78.9 ^{x24} , 90 ^{x12}	1 ^{×24}	90 ^{x24} , 120 ^{x24}

Group: dipyramids

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1		-	12	20	30	3 ^{x20}	34.3 ^{x60} ,	0.↩	95.5 ^{x40} ,
	decagonal	\leftarrow					49.4 ^{x40} ,	201 ^{x10} ,	169 ^{x20}
	bi←						68.3 ^{x40} ,	1.05 ^{x20}	
	pyramid						101.↩		
							2^{x40} ,		
							130.↩		
							6 ^{x40} ,		
							145.7 ^{x20}		

Table 6: dipyramids



2		-	8	12	18	3 ^{x12}	53.1 ^{x36} ,	0.577 ^{x6} ,	104.↩
	hexagonal	_			_		78.5 ^{x24} ,	, ,	5 ^{x24} ,
	bi←						101.↩	1.15 ^{x12}	151 ^x 12
	pyramid						5 ^{x24} ,		
							126.9 ^{x12}		
3		-	10	16	24	3 ^{x16}	41.9 ^{x48} ,	0.317 ^{x8} ,	98.4 ^{x32} ,
	octagonal	ے					60.7 ^{x32} ,		163.2 ^{x16}
	di <i>←</i>						82.7 ^{x32} ,	1.08 ^{x16}	
	pyramid						119.←		
							3 ^{x32} ,		
							138.1 ^{x16}		
4		-	7	10	15	3 ^{x10}	60.9 ^{x30} ,	0.854 ^{x5} ,	110.↩
	pentagonal	\leftarrow					91.6 ^{x20} ,		2 ^{x20} ,
	di←						110.2 ^{x20}	1.24 ^{x10}	139.6 ^{x10}
	pyramid								
5		Octahe-	6	8	12	3 ^{x8}	70.5 ^{x24} ,	1.41 ^{x12}	120 ^{x24}
	square←	dron					109.5 ^{x24}		
	di←								
	pyramid								
6		-	5	6	9	3 ^{x6}	81.8 ^{x18} ,	1.15 ^{x6} ,	82.8 ^{x6} ,
	triangular←	٥					135.6 ^{x12}	1.73 ^{x3}	138.6 ^{x12}
	di←								
	pyramid								

Group: johnson

no.	table id	other name	vertices	faces	edges	face- verticies	face- angles	edge- lengths	edge- angles
1	augmente	J58	21	16	35	3 ^{x5} , 5 ^{x11}	26.1 ^{x10} , 41.8 ^{x10} ,	0.641 ^{x35}	72 ^{x55} ,
	dodecahe						58.5 ^{x20} , 63.4 ^{x50} , 70.5 ^{x10}		
2	augmente _← hexagonal _prism		13	11	22	3 ^{x4} , 4 ^{x5} , 6 ^{x2}	70.5 ^{x14} , 35.3 ^{x4} , 60 ^{x8} , 70.5 ^{x8} , 73.2 ^{x8} , 90 ^{x28} , 109.5 ^{x4}	0.689 ^{x22}	60 ^{x12} , 90 ^{x20} , 120 ^{x12}
3	augmente _← pentagona _prism	al⊷	11	10	19	3 ^{x4} , 4 ^{x4} , 5 ^{x2}	17.3 ^{x4} , 35.3 ^{x4} , 70.5 ^{x8} , 72 ^{x6} , 79.7 ^{x8} , 90 ^{x24} , 109.5 ^{x4}	0.788 ^{x19}	72 ^{x10} , 90 ^{x16} , 120 ^{x12}
4	augmente ← sphenocor		11	17	26	3 ^x 16, 4 ^x 1	8.2 ^{x2} , 15.7 ^{x4} , 20.1 ^{x4} , 27.8 ^{x2} , 31 ^{x4} , 36.5 ^{x8} , 44 ^{x8} , 48.6 ^{x2} , 61.1 ^{x8} , 61.7 ^{x4} , 62 ^{x4} , 68.2 ^{x8} , 69.7 ^{x4} , 70.5 ^{x12} , 70.7 ^{x4} , 72.3 ^{x8} , 74.8 ^{x4} , 74.8 ^{x4} , 74.8 ^{x4} , 78.9 ^{x8} , 81.3 ^{x4} , 82.5 ^{x2} , 82.7 ^{x8} , 87.5 ^{x8} , 104.5 ^{x4} ,	0.858 ^{×26}	90 ^{x4} , 120 ^{x48}

5	augmented _← triangular← _prism	ب	7	8	13	3 ^{x6} , 4 ^{x2}	35.3 ^{x4} , 65.3 ^{x4} , 70.5 ^{x8} , 90 ^{x16} , 106.8 ^{x8} , 109.5 ^{x4} ,	1.14 ^{x13}	90 ^{x8} , 120 ^{x18}
6	augmented _←' tridiminished _←' icosahedro	ed← on	10	10	18	3 ^{x7} , 5 ^{x3}	8.7 $^{\times 6}$, 41.8 $^{\times 6}$, 70.5 $^{\times 6}$, 79.2 $^{\times 12}$, 100.8 $^{\times 6}$, 113. \leftarrow 6 $^{\times 12}$, 116.6 $^{\times 6}$	0.976 ^{x18}	72 ^{x15} , 120 ^{x21}
7	augmented _← truncated← _cube		28	22	48	3 ^{x12} , 4 ^{x5} , 8 ^{x5}	9.7 ^{x8} , 35.3 ^{x24} , 36.4 ^{x16} , 45 ^{x8} , 54.7 ^{x48} , 60 ^{x24} , 90 ^{x16}	0.506 ^{x48}	45 ^{x40} , 90 ^{x20} , 120 ^{x36}
8	augmented _← truncated← _← dodecahed	د	65	42	105	3 ^{x25} , 4 ^{x5} , 5 ^{x1} , 10 ^{x11}	5.7 ^{x10} , 20.9 ^{x20} , 21.6 ^{x20} , 26.1 ^{x10} , 31.7 ^{x10} , 36 ^{x10} , 37.4 ^{x120} , 40.5 ^{x20} , 63.4 ^{x50}	0. <i>⇔</i> 316 ^{<i>x</i>105}	36 ^{x110} , 72 ^{x5} , 90 ^{x20} , 120 ^{x75}
9	augmented _← truncated← _← tetrahedror	د	15	14	27	3 ^{x8} , 4 ^{x3} , 6 ^{x3}	15.8 ^{x6} , 38.9 ^{x6} , 54.7 ^{x18} , 56.3 ^{x12} , 70.5 ^{x24} , 78.9 ^{x12} , 90 ^{x6} , 109.5 ^{x6}	0.7 ^{×27}	60 ^{x18} , 90 ^{x12} , 120 ^{x24}

10		J53	12	13	23	3 ^{x8} ,	17.3 ^{x8} ,	0.727 ^{x23}	72^{x10} ,
.0	biaugment			.0		4 ^{x3} ,	35.3 ^{x8} ,	0.727	90 ^{×12} ,
		.cu~				5 ^{x2}	70.5^{x16} ,		120 ^{x24}
	_← pentagona	de s					70.5 , 72 ^{x2} ,		120
	_ I	µ←					72 , 79.7 ^{x16} ,		
	_prism						90 ^{×28} ,		
							90 , 109.5 ^{x8}		
11		J50	8	11	17	3 ^{x10} ,	109.5 10.5 ^{x2} ,	1.06 ^{x17}	90 ^{x4} ,
11	higuamont		0	''	17	4 ^{x1}	35.3 ^{x8} ,	1.00	120 ^{x30}
	biaugment	.eu←				4	60 ^{x4} ,		120
	_← triongular						65.3 ^{x4} ,		
	triangular∢)					70.5^{x16} ,		
	_prism						70.5 °, 76 ^{x8} ,		
							90 ^{x20} ,		
							106.8 ^{x8} ,		
							109.5 ^{x8}		
12		J67	32	30	60	3 ^{x16} ,	9.7 ^{×16} ,	0.49 ^{x60}	45 ^{x32} ,
12	higuamont		32	30	60	4 ^{x10} ,	35.3 ^{x48} ,	0.49	45 , ΩΩΧ 40
	biaugment	.eu⇔				8 ^{x4}	36.4 ^{x32} ,		90 ^{x40} , 120 ^{x48}
	_← _—					0	45 ^{x16} ,		120
	truncated	_					54.7 ^{×48} ,		
	_cube						60^{x48} ,		
							90 ^{x8}		
13		J79	55	52	105	3 ^{x15} ,	20.9 ^{x70} ,	0.←	36 ^{x10} ,
13	bigyrate←	0/3	33	32	103	4 ^{x25} ,	26.1 ^{x20} ,	0.← 423 ^{×105}	72 ^{×55} ,
						5^{x11} ,	26.6 ^{x20} ,	420	90 ^{×100} ,
	_← diminished	l, ,				10 ^{x1}	31.7^{x80} ,		120 ^{×45}
		ı <i>←</i>				10	33.3^{x40} ,		120
	_← rhombicos	idodecahec	dron				36 ^{×50} ,		
	mombicos	luouecanec	JIOH				37.4 ^{x50} ,		
							40.5^{x40} ,		
							58.3^{x10} ,		
							63.4 ^{x10}		
14	bilun-	J91	14	14	26	3 ^{x8} ,	20.9 ^{x8} ,	0.714 ^{x26}	72×20
17	abiro-	331	' -	'-		$4^{\times 2}$,	37.4^{x16} ,	0.717	72 ^{x20} , 90 ^{x8} ,
	tunda					5 ^{x4}	41.8 ^{x4} ,		120 ^{x24}
	lunda						63.4 ^{x4} ,		120
							69.1 ^{x8} ,		
							70.5^{x16} ,		
							70.3° , $79.2^{\times 16}$,		
							90 ^{x16} ,		
							116.6 ^{x4}		
							110.0		

15		J76	55	52	105	3 ^{x15} ,	20.9 ^{x90} ,	0.↩	36 ^{x10} ,
	diminished					4 ^{x25} ,	31.7 ^{x100} ,	423 ^{x105}	72^{x55} ,
	_	Ţ,				5 ^{x11} ,	, ,	0	90 ^{x100}
		idodecahed	ron			10 ^{x1}	36 ^{x90} ,		90 ^{x100} , 120 ^{x45}
	111011101000	lacaccarica				10	37.4 ^{×90} ,		120
							58.3 ^{x10} ,		
							63.4 ^{x10}		
16	disphe-	J90	16	24	38	3 ^{x20} ,	13.2 ^{x8} ,	0.825 ^{x38}	90 ^{x16} ,
10	nocingu-	030	10		30	4 ^{x4}	25.6 ^{x8} ,	0.023	120 ^{×60}
	lum					7	31.6 ^{x16} ,		120
	luiii						39.7 ^{×16} ,		
							43.7 ^{x16} ,		
							43.7 , 46.4 ^{x16} ,		
							46.4 , 49.3 ^{×16} ,		
							49.3 ^{×16} ,		
							53.9 ^{x16} ,		
							55.3 ^{x8} ,		
							62.3 ^{x16} ,		
							64.2 ^{x16} ,		
							66.6 ^{x16} ,		
							74.3 ^{x16} ,		
							77.7 ^{x8} ,		
							79.8 ^{x4} ,		
							82.6 ^{x16}		
17		J20	25	22	45	3 ^{x5} ,	20.9 ^{x20} ,	0.567 ^{x45}	36 ^{x10} ,
	elongated	\leftarrow				4 ^{x15} ,	31.7 ^{x10} ,		72^{x5} ,
	_←					5^{x1} ,	36 ^{x30} ,		90 ^{x60} , 120 ^{x15}
	pentagona	ll←				10 ^{x1}	37.4 ^{x10} ,		120 ^{x15}
	_cupola						52.6 ^{x10} ,		
							58.3 ^{x10} ,		
							60.6 ^{x20} ,		
							64.8 ^{x20} ,		
							90 ^{x20}		
18		J16	12	15	25	3 ^{x10} ,	41.8 ^{x20} ,	0.975 ^{x25}	90 ^{x20} ,
	elongated					4 ^{x5}	52.6 ^{x20} ,		120 ^{x30}
	_←						70.5 ^{x20} ,		
	pentagona	←					72 ^{x10} ,		
	_~						79.2×40		
	dipyramid								
19	17	J39	30	32	60	3 ^{x10} ,	20.9 ^{x40} ,	0.59 ^{x60}	72 ^{x10} ,
	elongated		-		-	4 ^{x20} ,	31.7 ^{x20} ,		90 ^{x80} ,
	_~					5 ^{x2}	36 ^{x40} ,		120 ^{x30}
	pentagona	 ←				-	37.4 ^{x20} ,		
	_						52.6 ^{x20} ,		
	gyrobicup	nla					58.3 ^{x20} ,		
	gyrobioupt	,.u					60.6 ^{x40} ,		
							64.8 ^{x40}		
							04.0		

20	J43 elongated	40	42	80	3 ^{x20} , 4 ^{x10} , 5 ^{x12}	10.8 ^{x20} , 26.6 ^{x20} , 36 ^{x20} , 37.4 ^{x140} ,	0.485 ^{x80}	72 ^{×60} , 90 ^{×40} , 120 ^{×60}
	_← pentagonal← _←					36 ^{x20} ,		
	pentagonal ← _←					37.4 ^{x140} ,		120
	_←					U1. - ,		
				1				
						41.8 ^{x40} ,		
	gyroonotarida					43.6 ^{x40} ,		
						63.4 ^{×40}		
21	J41	35	37	70	3 ^{x15} ,	10.8 ^{x10} ,	0.552 ^{x70}	72 ^{x35} ,
	elongated←		0,	'	4 ^{x15} ,	20.9 ^{x20} ,	0.002	90 ^{×60} ,
					5 ^{x7} ,	26.6 ^{x10} ,		120 ^{x45}
	_← pentagonal←					31.7 ^{x10} ,		120
						36 ^{x30} ,		
	_← gyrocupolarotunda					37.4 ^{x80} ,		
	gyrocupolarolurida					41.8 ^{x20} ,		
						43.6 ^{x20} ,		
						52.6 ^{×10} ,		
						58.3 ^{x10} ,		
						60.6 ^{x20} ,		
						63.4 ^{×20} ,		
						64.8 ^{×20}		
00	100	00	00		0 <i>X</i> 10		0.59 ^{x60}	72 ^{x10} ,
22	J38	30	32	60	3 ^{x10} , 4 ^{x20} ,	20.9 ^{x40} , 31.7 ^{x20} ,	0.59***	
	elongated←				5 ^{x2}			90 ^{x80} , 120 ^{x30}
	_				5^-	36 ^{x40} ,		120/33
	pentagonal←					37.4 ^{x20} ,		
	-					52.6 ^{x20} ,		
	orthobicupola							
						60.6 ²⁴⁰ ,		
					- ×20		v00	v60
23		40	42	80	3 ^{×20} ,		0.485	
	elongated⊬				4*10,			90*40,
	_ ←				5*12	36 ^{x20} ,		120***
	pentagonal←					37.4 ^{x 140} ,		
	_ ←					40		
	orthobirotunda					41.8 ^{x40} ,		
						43.6 ^{x40} ,		
					\	63.4 ^{x40}	70	v95
24		35	37	70			0.552 ^{x70}	72 ^{x35} ,
	∣ elongated⊢				$4^{x_{15}}$,	20.9^{x20} ,		90 ^{x60} ,
	_←				$\int 5^{x7}$	26.6 ^{x10} ,		120 ^{x45}
	pentagonal←							
	_←					36 ^{x30} ,		
	orthocupo arotunda							
1								
	1							
				1	1	av10	I .	
						52.6 ^{x10} ,		
						58.3 ^{x10} ,		
						58.3^{x10} , 60.6^{x20} .		
						58.3 ^{x10} ,		
23	orthobicupola J42 elongated pentagonal orthobirotunda J40 elongated pentagonal	35	37	70	3 ^{x20} , 4 ^{x10} , 5 ^{x12}	58.3 ^{×20} , 60.6 ^{×40} , 64.8 ^{×40} 10.8 ^{×20} , 26.6 ^{×20} , 37.4 ^{×140} , 41.8 ^{×40} , 43.6 ^{×40} , 63.4 ^{×40} 10.8 ^{×10} , 20.9 ^{×20} , 26.6 ^{×10} , 31.7 ^{×10} , 36 ^{×30} , 37.4 ^{×80} , 41.8 ^{×20} , 43.6 ^{×20} ,	0.485 ^{x80} 0.552 ^{x70}	72 ^{x60} , 90 ^{x40} , 120 ^{x60} 72 ^{x35} , 90 ^{x60} , 120 ^{x45}

25		J9	11	11	20	3 ^{x5} ,	41.8 ^{x10} ,	0.964 ^{x20}	72 ^{x5} ,
25	alangatad		' '	• •	20	4 ^{x5} ,	52.6 ^{x10} ,	0.504	
	elongated-								90 ^{x20} , 120 ^{x15}
	_←					5 ^{x1}	70.5 ^{x10} ,		120^13
	pentagona	ll←					72 ^{x10} ,		
	_←						79.2 ^{x20} ,		
	pyramid						90 ^{x10}		
26		J21	30	27	55	3 ^{x10} ,	10.8 ^{x10} ,	0.52 ^{x55}	36 ^{x10} ,
	elongated					4 ^{x10} ,	26.6 ^{x10} ,		72^{x30} ,
						5 ^{x6} ,	36 ^{x20} ,		90 ^{×40} ,
	_←	.1				10 ^{x1}	37.4 ^{×70} ,		120 ^{×30}
	pentagona	u←				10			120
	_←						41.8 ^{x20} ,		
	rotunds						43.6 ^{x20} ,		
							63.4 ^{x20} ,		
							90 ^{x20}		
27		J19	20	18	36	3 ^{x4} ,	35.3 ^{x24} ,	0.666 ^{x36}	45 ^{x8} ,
	elongated					4 ^{x13} ,	45 ^{x32} ,		90^{x52} , 120^{x12}
	_←					8 ^{x1}	54.7 ^{x24} ,		120 ^{x12}
							60 ^{x24} ,		.20
	square←						90 ^{x16}		
00	_cupola	14.5	10	40	00	0.18	I	0.000×20	00×16
28		J15	10	12	20	3 ^{x8} ,	35.3 ^{x16} ,	0.828 ^{x20}	90 ^{x16} ,
	elongated-	\vdash				4 ^{x4}	70.5 ^{x16} ,		120 ^{x24}
	_←						90 ^{x40} ,		
	square←						109.5 ^{x8}		
	_←								
	dipyramid								
29	1,7	J37	24	26	48	3 ^{x8} ,	35.3 ^{x48} ,	0.715 ^{x48}	90 ^{x72} ,
	elongated					4 ^{x18}	45 ^{x48} ,		120 ^{x24}
						7	54.7 ^{x48} ,		120
	_←						60 ^{x48}		
	square←						60		
	_←								
	gyrobicupo	ola				_			
30		J8	9	9	16	3 ^{x4} ,	35.3 ^{x8} ,	0.932 ^{x16}	90 ^{x20} ,
	elongated	\leftarrow				4 ^{x5}	70.5 ^{x8} ,		120 ^{x12}
	_←						90 ^{x32} ,		
	square						109.5 ^{x4}		
							. 50.0		
	_← pyramid								
01	pyramid	11.0	15	1.1	07	2×4	10 EX6	0.70EX27	cox6
31		J18	15	14	27	3^{x4} ,	19.5 ^{x6} ,	0.795^{x27}	60 ^{x6} ,
	elongated-	\vdash				4 ^{x9} ,	35.3 ^{x6} ,		90 ^{x36} ,
	_←					6 ^{x1}	54.7 ^{x18} ,		120 ^{x12}
	triangular₄	<u>ا</u>					60 ^{x12} ,		
	_cupola						61.9 ^{x12} ,		
							65.9 ^{x12} ,		
							70.5 ^{x6} ,		
							90 ^{x18}		
							30		

32	J14	8	9	15	3 ^{x6} ,	19.5 ^{x12} ,	0.76 ^{x15}	90 ^{x12} ,
	elongated ←				4 ^{x3}	109. ← 5 ^{x12} ,		120 ^{x18}
	triangular					118.←		
	_←					1 ^{x24} ,		
	dipyramid					120 ^{x6}		
33	J36	18	20	36	3 ^{x8} ,	19.5 ^{x12} ,	0.696 ^{x36}	90 ^{x48} ,
	elongated⊬				4 ^{x12}	35.3 ^{x12} ,		120 ^{x24}
	_←					54.7 ^{x36} ,		
	triangular←					60 ^{x12} ,		
	_←					61.9 ^{x24} ,		
	gyrobicupola					65.9 ^{x24} ,		
						70.5^{x12} ,		
0.4	105	10			0.78	90 ^{x12}	0.000¥36	00 Y48
34	J35	18	20	36	3 ^{x8} , 4 ^{x12}	19.5 $^{x_{12}}$, 35.3 $^{x_{12}}$,	0.696 ^{x36}	90 ^{x48} , 120 ^{x24}
	elongated←				4^2	54.7 ^{x36} ,		120/24
	_← triangular					60 ^{x12} ,		
	triangular←					61.9 ^{x24} ,		
	_← orthobicupola					65.9 ^{x24} ,		
	oi triobicupola					70.5^{x12} ,		
						90 ^{x12}		
35	J7	7	7	12	3 ^{x4} ,	19.5 ^{x6} ,	0.886 ^{x12}	90 ^{x12} ,
	elongated←				4 ^{x3}	90 ^{x6} ,		120 ^{x12}
						109.5 ^{x6} ,		
	triangular-							
	_~					118.←		
	pyramid					1 ^{x12} ,		
						120 ^{x6}		
36	gyrate← J82	50	42	90	3 ^{x10} ,	20.9 ^{x50} ,	0.41 ^{x90}	36 ^{x20} ,
	_←				4 ^{x20} ,	26.1 ^{x10} ,		72 ^{x50} ,
	bidiminished←				5 ^{x10} ,	26.6 ^{x10} ,		90 ^{x80} ,
	_←				10 ^{x2}	31.7 ^{x70} ,		120 ^{x30}
	rhombicosidodecahed	lron				33.3 ^{x20} ,		
						36^{x40} ,		
						37.4^{x40} ,		
						40.5 ^{x20} ,		
						58.3 ^{x20} ,		
						63.4 ^{x20}		

37	gyrate ← _←	J72	60	62	120	3 ^{x20} , 4 ^{x30} ,	20.9 ^{x110} ,	0. <i>←</i> 448 ^{<i>x</i>120}	72^{x60} , 90^{x120} ,
	rhombicos	idodecahed	dron			5 ^{x12}	$26.1^{x10}, 26.6^{x10}, 31.7^{x110},$		120 ^{×60}
							33.3^{x20} , 36^{x100} , 37.4^{x100} ,		
							40.5 ^{x20}		
38	gyrobi- fastigium	J26	8	8	14	3 ^{x4} , 4 ^{x4}	30 ^{x8} , 90 ^{x24} , 104.5 ^{x8} ,	1 ^{x14}	90 ^{x16} , 120 ^{x12}
							104.5 ,		
39	gyroelong. pentagona bicupola		30	42	70	3 ^{x30} , 4 ^x 10, 5 ^{x2}	20.8 ^{x40} , 20.9 ^{x40} , 31.7 ^{x20} , 35.8 ^{x40} , 36 ^{x20} , 37.4 ^{x20} , 47.4 ^{x20} , 53 ^{x20} , 55.8 ^{x40} ,	0.597 ^{x70}	72 ^x 10, 90 ^x 40, 120 ^x 90
40		J48	40	52	90	3 ^{×40} ,	59.9 ^{x80} , 65.2 ^{x40} 5.6 ^{x20} ,	0.501 ^{x90}	72 ^{×60} ,
40	gyroelong. pentagona birotunda	ated↔	40	32	30	5 , 5 ^{x12}	20.8 ^{×40} , 21.3 ^{×20} , 24.1 ^{×40} , 35.8 ^{×40} , 36.1 ^{×40} , 37.4 ^{×100} , 40.4 ^{×40} , 41.8 ^{×40} , 63.4 ^{×40}	0.301	120 ^{×120}

41	Τ.	J24	25	32	55	3 ^{x25} ,	20.8 ^{x40} ,	0.577 ^{x55}	36 ^{x10} ,
	gyroelongate			-		4 ^{x5} ,	20.9 ^{x20} ,	0.077	72^{x5} ,
	_~	54 \				5^{x1} ,	31.7 ^{x10} ,		90 ^{x20} ,
	_` pentagonal←					10 ^{x1}	35.8 ^{x40} ,		120 ^{x75}
	_cupola	_				10	36 ^{x10} ,		120
	_capola						37.4 ^{x10} ,		
							47.4 ^{x10} ,		
							53 ^{x10} ,		
							55.8 ^{x20} ,		
							59.8 ^{x12} ,		
							59.9 ^{x28} ,		
							65.2 ^{×20} ,		
							84.8 ^{x20} ,		
							95.2 ^{x20}		
42		J47	35	47	80	3 ^{x35} ,	5.6 ^{x10} ,	0.563 ^{x80}	72 ^{x35} ,
42			33	47	00	4 ^{x5} ,	20.8 ^{x40} ,	0.565	90 ^{x20} ,
	gyroelongate	eu←				5^{x7}	20.9 ^{x20} ,		120 ^{×105}
	_← pontagonal					5	20.9 , 21.3 ^{x10} ,		120
	pentagonal←	_					24.1 ^{x20} ,		
	_←	do					31.7 ^{x10} ,		
	cupolarotuno	Ja					35.8 ^{x40} ,		
							36 ^{x10} ,		
							36.1 ^{x20} ,		
							36.1 ····································		
							36.3 ^{x20} , 37.4 ^{x60} ,		
							40.4 ^{x20} ,		
							40.4****,		
							41.8 ^{x20} ,		
							47.4 ^{x10} ,		
							53 ^{x10} ,		
							55.8 ^{x20} ,		
							59.9 ^{x40} ,		
							63.4 ^{x20} ,		
							65.2 ^{x20}		

43		J11	11	16	25	3 ^{x15} ,	41.8 ^{x40} ,	1.01 ^{x25}	72 ^{x5} ,
	gyroelonga					5 ^{x1}	70.5 ^{x70} ,		120 ^{x45}
	_ ←						79.2 ^{x10} ,		0
	pentagona	اسا					100.8 ^{x10}		
		I _					100.0		
	_← pyramid								
44	руганна	J25	30	37	65	3 ^{x30} ,	5.6 ^{x10} ,	0.591 ^{x65}	36 ^{x10} ,
77	gyroelonga		30	37	03	5^{x6} ,	20.8 ^{×40} ,	0.551	72 ^{x30} ,
	I I	iicu←				10 ^{x1}	21.3 ^{x10} ,		120 ^{×90}
	_← nontagona	L.				10	24.1 ^{x20} ,		120
	pentagona	ı—					35.8 ^{x40} ,		
	_← rotundo						36.1 ^{x20} ,		
	rotunda						36.3 ^{<i>x</i>20} ,		
							37.4 ^{x50} ,		
							40.4 ^{x20} ,		
							40.4 ^{x20} , 41.8 ^{x20} ,		
							63.4 ^{x20} ,		
							84.8 ^{x20} , 95.2 ^{x20}		
		145				0.724		o =o=v56	0.0 //10
45		J45	24	34	56	3 ^{x24} ,	26 ^{x32} ,	0.727 ^{x56}	90 ^{x40} ,
	gyroelonga	ıted←				4 ^{x10}	28.7 ^{x16} ,		120 ^{x72}
	_←						35.3 ^{x32} ,		
	square←						38.4 ^{x16} ,		
	_←						44.7 ^{x32} ,		
	bicupola						45 ^{x16} ,		
							46.9 ^{x32} ,		
							50.2 ^{x32} ,		
							54.7 ^{x48} ,		
							55.4 ^{x32} ,		
							60 ^{x16}		
46		J23	20	26	44	3 ^{x20} ,	26 ^{x32} ,	0.741 ^{x44}	45 ^{x8} ,
	gyroelonga	ıted←				4 ^{x5} ,	28.7 ^{x8} ,		90 ^{x20} ,
	_←					8 ^{x1}	35.3 ^{x16} ,		120 ^{x60}
	square←						38.4 ^{x8} ,		
	_cupola						44.7 ^{x32} ,		
							45 ^{x8} ,		
							46.9 ^{x16} ,		
							50.2 ^{x16} ,		
							54.7 ^{x24} ,		
							55.4 ^{x16} ,		
							60 ^{x8} ,		
				1		1			
							83.4 ^{x16} , 96.6 ^{x16}		

47	J17	10	16	24	3 ^{x16}	21.4 ^{x16} ,	0.887 ^{x24}	120 ^{x48}
	gyroelongated←		. •			52.4 ^{x16} ,	0.007	0
						65 ^{x32} ,		
	_← square←					70.5^{x16} ,		
						82.1 ^{x32} ,		
	_ 					86.7 ^{x16} ,		
	dipyramid					109.5 ^{x8}		
40	140		10		0 12		0.000×20	00×4
48	J10	9	13	20	3 ^{x12} ,	21.4 ^{x8} ,	0.998 ^{x20}	90 ^{x4} ,
	gyroelongated←				4 ^{x1}	52.4 ^{x16} ,		120 ^{x36}
	_←					65 ^{x16} ,		
	square←					70.5 ^{x8} ,		
	_←					76.2 ^{x8} ,		
	pyramid					82.1 ^{x16} ,		
						86.7 ^{x16} ,		
						103.8 ^{x8} ,		
						,		
						109.5 ^{x4}		
49	J44	18	26	42	3 ^{x20} ,	10.6 ^{x12} ,	0.729 ^{x42}	90 ^{x24} ,
	gyroelongated←			-	4 ^{x6}	26.4 ^{x12} ,	*** = *	120 ^{x60}
	gyrooiongatou√				'	34.8 ^{x24} ,		1.20
	_` triangular←					41 ^{x24} ,		
						52.5 ^{x24} ,		
	_← higunala					54.7 ^{x36} ,		
	bicupola					54.7 4, 58.8 ^{x24} ,		
						59.2 ^{x24} ,		
						60.5 ^{x24} ,		
						70.5 ^{x12} ,		
						90 ^{x12}		
50	J22	15	20	33	3 ^{x16} ,	10.6 ^{x6} ,	0.891 ^{x33}	60 ^{x6} ,
	gyroelongated←				4 ^{x3} ,	26.4 ^{x6} ,		90 ^{x12} ,
	_←				6 ^{x1}	34.8 ^{x24} ,		120 ^{x48}
	triangular←					41 ^{x12} ,		
	_cupola					52.5 ^{x12} ,		
						54.7 ^{x18} ,		
						58.8 ^{x12} ,		
						59.2 ^{x24} ,		
						60.5^{x12} ,		
						$70.5^{\times 6}$,		
						70.5 ^x , 81.1 ^{x12} ,		
						90 ^{x6} ,		
						98.9 ^{x12}		

51	hebe-	J89	14	21	33	3 ^{x18} ,	22.9 ^{x16} ,	0.889 ^{x33}	90 ^{x12} ,
0.	sphe-					4×3	27 ^{x4} ,	0.000	120 ^{x54}
	nomega-					•	30.4 ^{x2} ,		0
	corona						38.7 ^{x8} ,		
	Josephia						39.3 ^{x8} ,		
							46 ^{x12} ,		
							51.5^{x16} ,		
							51.8 ^{x8} ,		
							53.3 ^{x8} ,		
							56.4 ^{x8} ,		
							60^{x8} ,		
							65.1 ^{<i>x</i>16} ,		
							65.5 ^{x4} ,		
							68.3 ^{x4} ,		
							70.2 ^{x8} ,		
							76.2 , 76 ^{x8} ,		
							76 °, 77.5 ^{x4} ,		
							80.7 ^{x16} ,		
							81.3 ^{x16} ,		
							85.3 ^{x8}		
52		J60	22	20	40	3 ^{x10} ,	26.1 ^{x20} ,	0.625 ^{x40}	72 ^{×50} ,
52	motobious		22	20	40	5^{x10}	41.8 ^{x20} ,	0.023	120 ^{×30}
	metabiaug	Imenteu⇔				5	58.5 ^{x40} ,		120
	_← dadaaaha	dran					63.4 ^{x40} ,		
	dodecahe	uron					70.5 ^{x20}		
53		J56	14	14	26	3 ^{x8} ,	5.3 ^{x8} ,	0.658 ^{x26}	60 ^{x12} ,
55	metabiauc	mented⊷	14	14	20	4 ^{x4} ,	35.3 ^{x8} ,	0.030	90 ^{x16} ,
		Interiteu←				6 ^{x2}	60^{x4} ,		120 ^{×24}
	_← hovagonal					0	70.5 ^{x16} ,		120
	hexagonal	 ←					70.5, 73.2^{x16} ,		
	_prism						90^{x32} ,		
							109.5 ^{x8}		
54		J70	70	52	120	3 ^{x30} ,	5.7 ^{x20} ,	0.⇔	36 ^{x100} ,
J -1	metabiaug		'0	32	120	4 ^{x10} ,	20.9 ^{x40} ,	0. <i>←</i> 313 ^{<i>x</i>120}	72^{x10} ,
		,,,,e,,,eu←				5 ^{x2} ,	20.9 , 21.6 ^{x40} ,	010	90 ^{×40} ,
	_← truncated∢					10 ^{x10}	26.1 ^{x20} ,		120 ^{×90}
		Γ				10	31.7 ^{x20} ,		120
	_← dodecahe	dron					36 ^{x20} ,		
	uouecane	uron					37.4 ^{x120} ,		
							37. 4 ,		
							40.5 ^{x40} ,		
							63.4 ^{x40}		
							03.4"		

55	J62	10	12	20	3 ^{x10} ,	41.8 ^{x22} ,	0.964 ^{x20}	72 ^{x10} ,
50	metabidiminished <i>←</i>				5 ^{x2}	70.5 ^{x32} ,	0.00	120 ^{x30}
	_~					79.2 ^{x16} ,		0
	icosahedron					100.←		
	1000411041011					8 ^{x12} ,		
						116.6 ^{x2}		
56	J81	50	42	90	3 ^{x10} ,	20.9 ^{x60} ,	0.41 ^{x90}	36 ^{x20} ,
	metabidiminished <i>←</i>				4 ^{x20} ,	31.7 ^{x80} ,	0	72 ^{×50} ,
	_				5 ^x 10.	36 ^{×60} ,		90 ^{x80} ,
	rhombicosidodecahed	ron			10 ^{x2}	37.4 ^{x60} ,		120 ^{x30}
	membleedideedaned				'	58.3 ^{x20} ,		120
						63.4 ^{x20}		
57	J74	60	62	120	3 ^{x20} ,	20.9 ^{x100} ,	0.←	72 ^{x60} ,
0,	metabigyrate←		02	1.20	4 ^{x30} ,	20.0	448 ^{x120}	90 ^{x120} ,
	_				5 ^{x12}	26.1 ^{x20} ,	1.0	120 ^{x60}
	rhombicosidodecahed	ron				26.6 ^{x20} ,		120
	mombiocolacocarioa					31.7 ^{x100} ,		
						,		
						33.3 ^{x40} ,		
						36 ^{x80} ,		
						37.4 ^{x80} ,		
						40.5 ^{x40}		
58	J78	55	52	105	3 ^{x15} ,	20.9 ^{x80} ,	0.⇔	36 ^{x10} ,
00	metagyrate←		02	100	4 ^{x25} ,	26.1 ^{x10} ,	423 ^{x105}	72 ^{x55} ,
	_				5 ^{x11} ,	26.6 ^{x10} ,	120	90 ^{x100} ,
	_` diminished↔				10 ^{x1}	31.7 ^{x90} ,		120 ^{x45}
					10	33.3 ^{x20} ,		120
	_← rhombicosidodecahed	lron				36 ^{x70} ,		
	mombiocolacocarioa					37.4 ^{x70} ,		
						40.5 ^{x20} ,		
						58.3 ^{x10} ,		
						63.4 ^{x10}		
59	J59	22	20	40	3 ^{x10} ,	26.1 ^{x20} ,	0.61 ^{x40}	72 ^{x50} ,
33	parabiaugmented <i>←</i>			13	5 ^{x10}	41.8 ^{x20} ,	0.01	120 ^{x30}
						58.5 ^{x40} ,		120
	_← dodecahedron					63.4 ^{x40} ,		
	dodoodiiodioii					70.5 ^{x20}		
60	J55	14	14	26	3 ^{x8} ,	5.3 ^{x8} ,	0.636 ^{x26}	60 ^{x12} ,
00	parabiaugmented <i>←</i>	1-	'-	20	4 ^{x4} ,	35.3 ^{x8} ,	0.000	90 ^{x16} ,
					6 ^{x2}	60 ^{x4} ,		120 ^{x24}
	_← hexagonal←					70.5 ^{x16} ,		120
	_prism					70.3 , 73.2^{x16} ,		
	_piisiii					90 ^{x32} ,		
						90 ,		
	1					109.5 ^{x8}		

61		J69	70	52	120	3 ^{x30} ,	5.7 ^{x20} ,	0.↩	36 ^{x100} ,
	parabiaugm					4 ^{x10} ,	20.9 ^{x40} ,	319 ^{x120}	72^{x10} ,
	_←					5^{x2} ,	21.6 ^{x40} ,		90 ^{x40} ,
	truncated←					10 ^{x10}	26.1 ^{x20} ,		120 ^{x90}
	_						31.7 ^{x20} ,		
	dodecahedr	on					36 ^{x20} ,		
		• • • • • • • • • • • • • • • • • • • •					37.4 ^{x120} ,		
							, ,		
							40.5 ^{x40} ,		
							63.4 ^{x40}		
62		J80	50	42	90	3 ^{x10} ,	20.9 ^{x60} ,	0.448 ^{x90}	36 ^{x20} ,
	parabidimini					4 ^{x20} .	31.7 ^{x80} ,		72 ^{x50} ,
	_←					5 ^{x10} ,	36 ^{x60} ,		90 ^{x80} ,
	rhombicosid	odecahed	ron			10 ^{x2}	37.4 ^{x60} ,		120 ^{x30}
							58.3 ^{x20} ,		
							63.4 ^{x20}		
63		J73	60	62	120	3 ^{x20} ,	20.9 ^{x100} ,	0.↩	72 ^{x60} ,
	parabigyrate	; ←				4 ^{x30} ,		448 ^{x120}	90 ^{x120} ,
	_~					5 ^{x12}	26.1 ^{x20} ,		120 ^{x60}
	rhombicosid	odecahed	ron				26.6 ^{x20} ,		
							31.7 ^{x100} ,		
							,		
							33.3 ^{x40} ,		
							36 ^{x80} ,		
							37.4 ^{x80} ,		
							40.5 ^{x40}		
64	,	J77	55	52	105	3 ^{x15} ,	20.9 ^{x80} ,	0.↩	36 ^{x10} ,
	paragyrate←	د				4 ^{x25} ,	26.1 ^{x10} ,	423 ^{x105}	72 ^{x55} ,
	_					5 ^{x11} ,	26.6 ^{x10} ,		90^{x100} ,
	diminished←	ـ				10 ^{x1}	31.7 ^{x90} ,		120 ^{x45}
							33.3 ^{x20} ,		
	rhombicosid	odecahed	ron				36 ^{x70} ,		
							37.4 ^{x70} ,		
							40.5 ^{x20} ,		
							58.3 ^{x10} ,		
							63.4 ^{x10}		
65		J5	15	12	25	3 ^{x5} ,	20.9 ^{x20} ,	0.614 ^{x25}	36 ^{x10} ,
	pentagonal₄	_				4 ^{x5} ,	31.7 ^{x10} ,		72 ^{x5} ,
	_cupola					5^{x1} ,	36 ^{x10} ,		90 ^{x20} .
						10 ^{x1}	37.4 ^{x10} ,		120 ^{x15}
							142.↩		
							6 ^{x10} ,		
							148.3 ^{x10}		

66	J13	7	10	15	3 ^{x10}	41.8 ^{x20} ,	1.18 ^{x15}	120 ^{x30}
	pentagonal←					70.5 ^{x20} ,		0
	_ _					105.↩		
	dipyramid					2 ^{x10} ,		
						121.2 ^{x20}		
67	J31	20	22	40	3 ^{x10} ,	20.9 ^{x40} ,	0.618 ^{x40}	72 ^{x10} ,
	pentagonal←				4 ^{x10} ,	31.7 ^{x20} ,		90 ^{x40} ,
	_←				5 ^{x2}	36 ^{x20} ,		120 ^{x30}
	gyrobicupola					37.4 ^{x20} ,		
						109.↩		
						5 ^{x20} ,		
						110.←		
						9 ^{x20} ,		
						120 ^{x20}		
68	J33	25	27	50	3 ^{x15} ,	20.9 ^{x20} ,	0.605 ^{x50}	72^{x35} ,
	pentagonal←				4 ^{x5} ,	31.7 ^{x10} ,		90 ^{x20} ,
	_←				5 ^{x7}	36 ^{x10} ,		120 ^{x45}
	gyrocupolarotunda					37.4 ^{x60} ,		
						41.8 ^{x20} ,		
						63.4 ^{x30} ,		
						75 ^{x20} ,		
						84.8 ^{x10} ,		
					- v10	85.2 ^{x20}	· - ×40	v10
69	J30	20	22	40	3 ^{x10} ,	20.9 ^{x40} ,	0.618 ^{x40}	72 ^{×10} ,
	pentagonal←				4^{x10} , 5^{x2}	31.7 ^{x20} ,		90 ^{x40} , 120 ^{x30}
	_←) 5 ^{^-}	36^{x20} , 37.4^{x20} ,		120/199
	orthobicupola					37.4 , 105. <i>←</i>		
						2 ^{x10} ,		
						, 114. <i>←</i> -		
						7^{x40} ,		
						116.6 ^{x10}		
70	J34	30	32	60	3 ^{x20} ,	21.6 ^{x10} ,	0.618 ^{x60}	72 ^{x60} ,
	pentagonal←				5 ^{x12}	37.4 ^{x100} ,		120 ^{x60}
						,		
	orthobirotunda					41.8 ^{x40} ,		
						51.2 ^{x40} ,		
						53.1 ^{x10} ,		
						63.4 ^{x40}		
71	J32	25	27	50	3 ^{x15} ,	20.9 ^{x20} ,	0.605 ^{x50}	72 ^{x35} ,
	pentagonal←				4 ^{x5} ,	31.7 ^{x10} ,		90 ^{x20} ,
	_~				5 ^{x7}	36 ^{x10} ,		120 ^{x45}
	orthocupo arontunda					37.4 ^{x60} ,		
						41.8 ^{x20} ,		
						63.4 ^{x20} ,		
						69.1 ^{x10} ,		
						70.5 ^{x20} ,		
						79.2 ^{x10} ,		
						90 ^{x20}		

9.32 Polyhedra 211

72		J2	6	6	10	3 ^{x5} ,	41.8 ^{x10} ,	1.17 ^{x10}	72 ^{x5} ,
	pentagona		ŭ			5 ^{x1}		,	120 ^{x15}
							70.5 x10 , 142.6 x10		120
	_← pyromid						142.0		
70	pyramid	10	00	4-7	0.5	ox10	37.4 ^{x50} ,	0.584 ^{x35}	00×10
73		J6	20	17	35	3 ^{x10} ,		0.584	36 ^{x10} ,
	pentagona	lI←				5 ^{x6} ,	41.8 ^{x20} ,		72 ^{x30} , 120 ^{x30}
	_←					10 ^{x1}	63.4 ^{x20} ,		120 ^{x30}
	rotunda						100.↩		
							8 ^{x10} ,		
							116.6 ^{x10}		
74	snub_←	J84	8	12	18	3 ^{x12}	13.5 ^{x2} ,	1.08 ^{x18}	120 ^{x36}
	disphenoi	k					13.6 ^{x6} ,		
	.						58.2 ^{x4} ,		
							58.3 ^{x12} ,		
							65.7 ^{x16} ,		
							83.8 ^{x12} ,		
							90.6 ^{x4} ,		
							90.6 , 90.7 ^{x12} ,		
							90.7 ^x , 94.9 ^x ,		
							94.9.0,		
						- ×24	111.5 ^{x16}	· - ×40	1/9
75	snub_←	J85	16	26	40	3 ^{x24} ,	15.7 ^{x16} ,	0.815 ^{x40}	90 ^{x8} ,
	square←					4 ^{x2}	34.6 ^{x16} ,		120 ^{x72}
	_←						35.9 ^{x32} ,		
	antiprism						45.5 ^{x32} ,		
							47.3 ^{x16} ,		
							59.9 ^{x32} ,		
							61 ^{<i>x</i>16} ,		
							65.4 ^{x16} ,		
							73.9 ^{x32} ,		
							85.9 ^{x32}		
76	spheno-	J86	10	14	22	3 ^{x12} ,	20.1 ^{x4} ,	1 ^{x22}	90 ^{x8} ,
7.0	corona	000	10	• •		4 ^{x2}	36.5 ^{x8} ,	.	120 ^{x36}
	Corona					7	44 ^{x8} ,		120
							44 , 48.6 ^{x2} ,		
							61.1 ^{x8} ,		
							62 ^{x4} ,		
							63 ^{x2} ,		
							68.2 ^{x8} ,		
							70.5 ^{x8} ,		
							72 ^{x8} ,		
							72.3 ^{x8} ,		
							74 ^{x4} ,		
							81.3 ^x 8,		
							82.5 ^{x4} ,		
							82.7 ^{x8} ,		
							87.5 ^{x8} ,		
							104.5 ^{x8}		
							104.5		

77	sphe-	J88	12	18	28	3 ^{x16} ,	8.3 ^{x4} ,	0.↩	90 ^{x8} ,
	nomega-					4 ^{x2}	8.4 ^{x4} ,	761 ^{x26} ,	120 ^{x48}
	corona						18.6 ^{x2} ,	0.762 ^{x2}	
							25.3 ^{x8} ,		
							36.3 ^{x8} ,		
							41 ^{x8} ,		
							42.8 ^{x4} ,		
							48.4 ^{x8} ,		
							48.4 ^{x8} , 50.6 ^{x8} ,		
							55.1 ^{x8} ,		
							61.6 ^{x4} ,		
							62.6 ^{x4} ,		
							62.7 ^{x4} ,		
							65.8 ^{x8} ,		
							67 ^{x12} ,		
							67.1 ^{x4} ,		
							72.8 ^{x4} ,		
							72.9 ^{x4} ,		
							83.8 ^{x8} ,		
							93.3 ^{x8} ,		
							97.4 ^{x8} ,		
							100.6 ^{x4} ,		
							_		
							105.4 ^{x8} ,		
							107x2		
							107 ^{x2} , 114.9 ^{x8}		
							114.9^		

9.32 Polyhedra 213

78		J4	12	10	20	3 ^{x4} ,	35.3 ^{x16} ,	0.753 ^{x20}	45 ^{x8} ,
	square←	• •		.0		4^{x5} ,	45 ^{x8} ,	0.700	90 ^{x20} ,
						8 ^{x1}	54.7 ^{x8} ,		120 ^{x12}
	_cupola					0	60 ^{x8} ,		120
							125.3 ^{x8} ,		
							40=V8		
		100	1.0			0 V 8	135 ^{x8}	0 705 ×32	00×40
79		J29	16	18	32	3 ^{x8} ,	35.3 ^{x32} ,	0.765 ^{x32}	90 ^{x40} ,
	square←					4 ^{x10}	45 ^{x16} ,		120 ^{x24}
	_←						54.7 ^{x16} ,		
	gyrobicup	ola					60 ^{x16} ,		
							80.3 ^{x16} ,		
							82.1 ^{x16} ,		
							98.4 ^{x16}		
80		J28	16	18	32	3 ^{x8} ,	35.3 ^{x32} ,	0.765 ^{x32}	90 ^{x40} ,
	square←					4 ^{x10}	45 ^{x16} ,		120 ^{x24}
	_←						54.7 ^{x16} ,		
	orthobicup	ola					60 ^{x16} ,		
							70.5 ^{x8} ,		
							90 ^{x40}		
81		J1	5	5	8	3 ^{x4} ,	70.5 ^{x8} ,	1.41 ^{x8}	90 ^{x4} ,
01	square↩	0.		•	J	4 ^{x1}	109.5 ^{x4} ,		120 ^{x12}
						7	103.5 ,		120
	_← _—						125.3 ^{x8}		
00	pyramid	10	9	0	15	0.x4	54.7 ^{x18} ,	0.965 ^{x15}	60 ^{x6} ,
82		J3	9	8	15	3^{x4} ,		0.965***	
	triangular					4 ^{x3} ,	70.5 ^{x6} ,		90^{x12} ,
	_cupola					6 ^{x1}	90 ^{x6} ,		120 ^{x12}
							109.5 ^{x6} ,		
							125.3 ^{x6}		
83		J12	5	6	9	3 ^{x6}	38.9 ^{x6} ,	1.22 ^{x9}	120 ^{x18}
	triangular∢	<u> </u>					109.←		
	_←						5 ^{x12} ,		
	dipyramid						123.7 ^{x12}		
84		J92	18	20	36	3 ^{x13} ,	20.9 ^{x12} ,	0.661 ^{x36}	60 ^{x6} ,
	triangular	Ļ				4 ^{x3} ,	37.4 ^{x18} ,		72 ^{x15} ,
	_←					5 ^{x3} ,	41.8 ^{x24} ,		90 ^{x12} ,
	hebesphe	norotunda				6 ^{x1}	54.7 ^{x12} ,		120 ^{x39}
							63.4 ^{x6} ,		
							69.1 ^{x12} ,		
							70.5 ^{x30} ,		
							79.2 ^{x12} ,		
							η σ.Σ , αρχ12		
							90 ^{x12} , 100.8 ^{x6}		
							100.8^•		

85	J27	12	14	24	3 ^{x8} ,	38.9 ^{x6} ,	1 ^{x24}	90 ^{x24} ,
	triangular←				4×6	54.7 ^{×36} ,		120 ^{x24}
	_←					70.5^{x18} ,		
	orthobicupola					78.9 ^{x24} ,		
						90 ^{x12}		
86	J61	23	24	45	3 ^{x15} ,	26.1 ^{x30} ,	0.613 ^{x45}	72 ^{x45} ,
	triaugmented←				5 ^{x9}	41.8 ^{x30} ,		120 ^{x45}
	_~					58.5 ^{x60} ,		
	dodecahedron					63.4 ^{x30} ,		
	dodddandan					70.5 ^{x30}		
87	J57	15	17	30	3 ^{x12} ,	5.3 ^{x12} ,	0.636 ^{x30}	60 ^{x12} ,
	triaugmented←				4 ^{x3} ,	35.3 ^{x12} ,		90 ^{x12} ,
	_←				6 ^{x2}	70.5 ^{×24} ,		120 ^{x36}
	 hexagonal					73.2 ^{x24} ,		
	_prism					90 ^{x36} ,		
						109.5 ^{x12}		
88	J51	9	14	21	3 ^{x14}	10.5 ^{x6} ,	1 x21	120 ^{x42}
	triaugmented←					35.3 ^{x12} ,		
	_←					60 ^{x12} ,		
	triangular					$70.5^{x^{24}}$,		
	_prism					76 ^{x24} ,		
						90 ^{x24} ,		
						109.5 ^{x12}		
89	J71	75	62	135	3 ^{x35} ,	5.7 ^{x30} ,	0.←	36 ^{x90} ,
	triaugmented←				4 ^{x15} ,	9.8 ^{x4} ,	307 ^{x135}	72 ^{x15} ,
	_←				5 ^{x3} ,	11.3 ^{x2} ,		90 ^{x60} ,
	truncated				10 ^x 9	18 ^{x8} ,		120 ^{x105}
	_					20.9 ^{x60} ,		. = 0
	dodecahedron					$21.6^{\times 60}$,		
						26.1 ^{x26} ,		
						31.7 ^{x30} ,		
						36 ^{x30} ,		
						37.4 ^{x120} ,		
						,		
						40.5 ^{x52} ,		
						63.4 ^{x32}		

9.32 Polyhedra 215

90	J6	3 9	8	15	3 ^{x5} ,	41.8 ^{x6} ,	1 ^{x15}	72 ^{x15} ,
	tridiminished←	د			5 ^{x3}	70.5 ^{x6} ,		120 ^{x15}
	_←					79.2 ^{x18} ,		
	icosahedron					100.8 ^{x6} ,		
						116.6 ^{x6}		
91	J8	3 45	32	75	3 ^{x5} ,	20.9 ^{x30} ,	0.424 ^{x75}	36 ^{x30} ,
	tridiminished←	د			4 ^{x15} ,	31.7^{x60} ,		72 ^{x45} ,
	_~				5 ^{x9} ,	36 ^{x30} ,		90 ^{×60} ,
	rhombicosidoo	decahedron			10 ^{x3}	37.4 ^{x30} ,		120 ^{x15}
						58.3 ^{x30} ,		
						63.4 ^{x30}		
92	J7	5 60	62	120	3 ^{x20} ,	20.9 ^{x90} ,	0.↩	72 ^{x60} ,
	trigyrate←				4 ^{x30} ,	26.1 ^{x30} ,	448 ^{x120}	90 ^{x120} ,
	_~				5 ^{x12}	26.6 ^{x30} ,		120 ^{x60}
	rhombicosidoo	decahedron				31.7 ^{x90} ,		
						33.3 ^{x60} ,		
						36 ^{x60} ,		
						37.4^{x60} ,		
						40.5 ^{x60}		

Group: platonic

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1	cube	Hexahe- dron	8	6	12	4 ^{x6}	90 ^{x24}	2 ^{x12}	90 ^{x24}
2	dodeca- hedron	-	20	12	30	5 ^{x12}	63.4 ^{×60}	0.714 ^{x30}	72 ^{×60}
3	icosahe- dron	-	12	20	30	3 ^{x20}	41.8 ^{x60} , 70.5 ^{x120}	1.24 ^{x30}	120 ^{x60}
4	octahe- dron	-	6	8	12	3 ^{x8}	70.5 ^{x24} , 109.5 ^{x24}	1.41 ^{x12}	120 ^{x24}
5	tetrahe- dron	-	4	4	6	3 ^{x4}	109.5 ^{x12}	2.83 ^{x6}	120 ^{x12}

Group: prisms

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
		name				verticies	angles	lengths	angles
1		-	20	12	30	4 ^{x10} ,	36 ^{x20} ,	0.59 ^{x30}	36 ^{x20} ,
	decagonal	\leftarrow				10 ^{x2}	90 ^{x40}		90 ^{x40}
	_prism								
2		-	12	8	18	4 ^{x6} ,	60 ^{x12} ,	0.894 ^{x18}	60 ^{x12} ,
	hexagonal	\leftarrow				6 ^{x2}	90 ^{x24}		90 ^{x24}
	_prism								
3		-	16	10	24	4 ^{x8} ,	45 ^{x16} ,	0.715 ^{x24}	45 ^{x16} ,
	octagonal	Ļ				8 ^{x2}	90 ^{x32}		90 ^{x32}
	_prism								

4		-	10	7	15	4 ^{x5} ,	72 ^{x10} ,	1.01 ^{x15}	72 ^{x10} ,
	pentagona	l←				5 ^{x2}	90 ^{x20}		90 ^{x20}
	_prism								
5		Cube	8	6	12	4 ^{x6}	90 ^{x24}	1.15 ^{x12}	90 ^{x24}
	square←								
	_prism								
6		-	6	5	9	3 ^{x2} ,	90 ^{x12} ,	1.31 ^{x9}	90 ^{x12} ,
	triangular	L				4 ^{x3}	120 ^{x6}		120 ^{x6}
	_prism								

Group: pyramids

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
1		name J13	7	10	15	verticies 3 ^{x10}	angles 41.8 ^{x20} ,	1.18 ^{x15}	angles 120 ^{x30}
1	nontogone		/	10	15	3		1.10	120 33
	pentagona	u←					70.5 ^{x20} ,		
	_—						105. <i>←</i> 2 ^{x10} ,		
	dipyramid						2 , 121.2 ^{x20}		
2		J2	6	6	10	3 ^{x5} ,	41.8 ^{x10} ,	1.17 ^{x10}	72 ^{x5} ,
	nontagona	_	0	0	10	5^{x1}	70.5^{x10} ,	1.17	120 ^{x15}
	pentagona	u←				3	142.6 ^{x10}		120
	_← pyramid						142.0		
3	pyramid	Octahe-	6	8	12	3 ^x 8	70.5 ^{x24} ,	1.41 ^{x12}	120 ^{x24}
3	square/	dron	0		12	3	109.5 ^{x24}	1.71	120
	square←	uion					103.5		
	_← dipryamid								
4	dipryamid	J1	5	5	8	3 ^{x4} ,	70.5 ^{x8} ,	1.41 ^{x8}	90 ^{x4} ,
7	square←	01	3	3	0	4 ^{x1}	109.5 ^{x4} ,	1.71	120 ^{x12}
						-	103.5 ,		120
	_← nyramid						125.3 ^{x8}		
5	pyramid	J12	5	6	9	3×6	38.9 ^{x6} ,	1.22 ^{x9}	120 ^{x18}
	triangular	_					00.0 , 109. <i>←</i>	1.22	120
		_					$5^{x_{12}}$,		
	_← dipyramid						123.7 ^{x12}		
6	аругани	Tetrahe-	4	4	6	3 ^{x4}	109.5 ^{x12}	1.63 ^{x6}	120 ^{x12}
	triangular			=			100.0	1.00	120
		– uron							
	_← nyramid								
	pyramid								

9.32 Polyhedra 217

Group: trapezohedron

no.	table id	other	vertices	faces	edges	face-	face-	edge-	edge-
4		name	00	00	40	verticies 4 ^{x20}	angles 34.7 ^{x80} ,	lengths	angles 63.2 ^{x60} ,
1	doogganal	ļ -	22	20	40	4	69.2 ^{x40} ,	0. <i>←</i> 0991 ^{<i>x</i>20} ,	170.4 ^{×20}
	decagonal	\leftarrow					09.∠ , 102. <i>⇔</i>	0991 ,	170.4
	_← trapezohe	dron					8 ^{x40} ,	1.01 ^{x20}	
	liapezone						0 , 133. <i>⇔</i>	1.01	
							6^{x40} ,		
							150.2 ^{x20}		
2		-	20	18	36	4 ^{x18}	38.3 ^{x72} ,	0.↩	63.9 ^{x54} ,
	enneagon	al⊷					76.1 ^{x36} ,	122 ^{x18} ,	168.3 ^{x18}
	_←						112.↩	1.02 ^{x18}	
	trapezohe	dron					3 ^{x36} ,		
							141.7 ^{x36}		
3		-	16	14	28	4 ^{x14}	48 ^{x56} ,	0.←	66.4 ^{x42} ,
	heptagona	ll←					94.2 ^{x28} ,	203 ^{x14} ,	160.9 ^{x14}
	_←						132 ^{x28}	1.03 ^{x14}	
	trapezohe	dron							
4		-	14	12	24	4 ^{x12}	54.7 ^{x48} ,	0.←	68.5 ^{x36} ,
	hexagonal	\leftarrow					105.←	277 ^{x12} ,	154.4 ^{x12}
	- ←						5^{x24} ,	1.04 ^{x12}	
_	trapezohe	dron	10	40		4 × 16	133.7 ^{x12} 42.6 ^{x64} ,		0.4.0.748
5		-	18	16	32	4 ^{x16}	42.6 ^{x34} , 84.4 ^{x32} ,	0. <i>←</i> 155 ^{<i>x</i>16} ,	64.9 ^{x48} , 165.2 ^{x16}
	octagonal						84.4 , 122. <i>←</i>	1.02 ^{x16}	165.2***
	_← transzobor	dron					7 ^{x32} ,	1.02	
	trapezohe	uron					7 , 143.6 ^{x16}		
6		_	12	10	20	4 ^x 10	63.4 ^{x40} ,	0.↩	72 ^{x30} ,
Ü	pentagona	ا بالب	12	10	20	-	116.6 ^{x20}	402 ^{×10} ,	144 ^{x10}
	_						110.0	1.05 ^{x10}	
	trapezohe	dron							
7	2 242 2 2 2 2	-	10	8	16	4 ^{x8}	74.9 ^{x32} ,	0.634 ^{x8} ,	78 ^{x24} ,
	square←						118.5 ^{x8}		125.9 ^{x8}
	_←							1.08 ^{x8}	
	trapezohe	dron							
8		Cube	8	6	12	4 ^{x6}	90 ^{x24}	1.15 ^{x12}	90 ^{x24}
	triangular∢	Ļ							
	_←								
	trapezohe	dron							

9.32.2 Variable Documentation

9.32.2.1 dtc_polyhedra_anti_prisms

Definition at line 114 of file anti_prisms.scad.

9.32.2.2 dtc_polyhedra_archimedean

<matrix-2x9> archimedean polyhedra data table columns definition.

Definition at line 114 of file archimedean.scad.

9.32.2.3 dtc_polyhedra_archimedean_duals

<matrix-2x9> archimedean_duals polyhedra data table columns definition.

Definition at line 114 of file archimedean duals.scad.

9.32.2.4 dtc_polyhedra_cupolas

<matrix-2x9> cupolas polyhedra data table columns definition.

Definition at line 114 of file cupolas.scad.

9.32.2.5 dtc_polyhedra_dipyramids

<matrix-2x9> dipyramids polyhedra data table columns definition.

Definition at line 114 of file dipyramids.scad.

9.32.2.6 dtc_polyhedra_johnson

<matrix-2x9> johnson polyhedra data table columns definition.

Definition at line 114 of file johnson.scad.

9.32.2.7 dtc_polyhedra_platonic

<matrix-2x9> platonic polyhedra data table columns definition.

Definition at line 114 of file platonic.scad.

9.32.2.8 dtc_polyhedra_polyhedra_all

<matrix-2x9> polyhedra_all polyhedra data table columns definition.

Definition at line 125 of file polyhedra_all.scad.

9.32.2.9 dtc_polyhedra_prisms

<matrix-2x9> prisms polyhedra data table columns definition.

Definition at line 114 of file prisms.scad.

9.32.2.10 dtc_polyhedra_pyramids

<matrix-2x9> pyramids polyhedra data table columns definition.

Definition at line 114 of file pyramids.scad.

9.32.2.11 dtc_polyhedra_trapezohedron

<matrix-2x9> trapezohedron polyhedra data table columns definition.

Definition at line 114 of file trapezohedron.scad.

9.32.2.12 dtr_polyhedra_anti_prisms

<matrix-9xR> anti_prisms polyhedra data table rows.

Definition at line 129 of file anti prisms.scad.

9.32 Polyhedra 219

9.32.2.13 dtr_polyhedra_archimedean

<matrix-9xR> archimedean polyhedra data table rows.

Definition at line 129 of file archimedean.scad.

9.32.2.14 dtr_polyhedra_archimedean_duals

<matrix-9xR> archimedean_duals polyhedra data table rows.

Definition at line 129 of file archimedean_duals.scad.

9.32.2.15 dtr_polyhedra_cupolas

<matrix-9xR> cupolas polyhedra data table rows.

Definition at line 129 of file cupolas.scad.

9.32.2.16 dtr_polyhedra_dipyramids

<matrix-9xR> dipyramids polyhedra data table rows.

Definition at line 129 of file dipyramids.scad.

9.32.2.17 dtr_polyhedra_johnson

<matrix-9xR> johnson polyhedra data table rows.

Definition at line 129 of file johnson.scad.

9.32.2.18 dtr_polyhedra_platonic

<matrix-9xR> platonic polyhedra data table rows.

Definition at line 129 of file platonic.scad.

9.32.2.19 dtr_polyhedra_polyhedra_all

<matrix-9xR> polyhedra_all polyhedra data table rows.

Definition at line 140 of file polyhedra_all.scad.

9.32.2.20 dtr_polyhedra_prisms

<matrix-9xR> prisms polyhedra data table rows.

Definition at line 129 of file prisms.scad.

9.32.2.21 dtr_polyhedra_pyramids

<matrix-9xR> pyramids polyhedra data table rows.

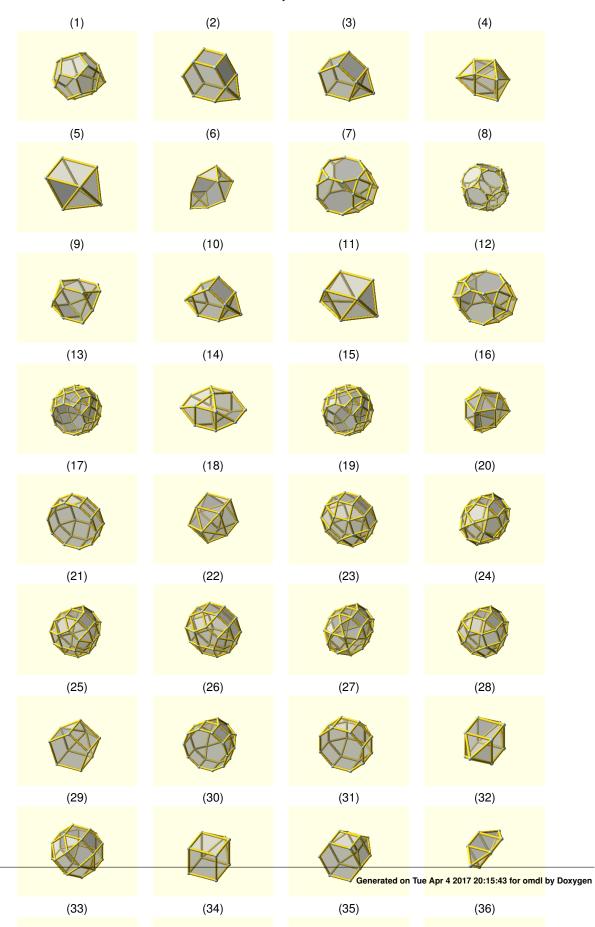
Definition at line 129 of file pyramids.scad.

9.32.2.22 dtr_polyhedra_trapezohedron

<matrix-9xR> trapezohedron polyhedra data table rows.

Definition at line 129 of file trapezohedron.scad.

Table 7: johnson



9.32 Polyhedra 221



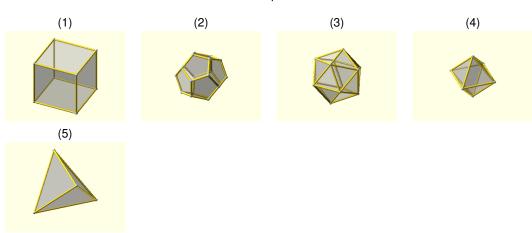


Table 9: prisms

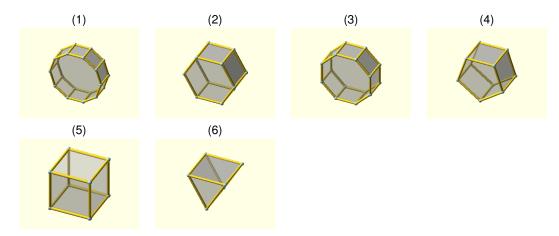


Table 10: pyramids

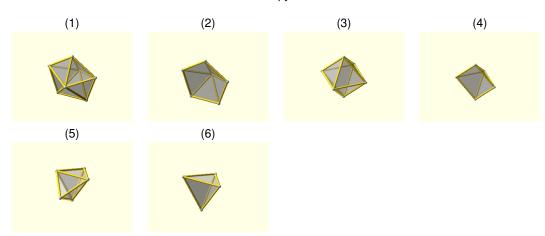
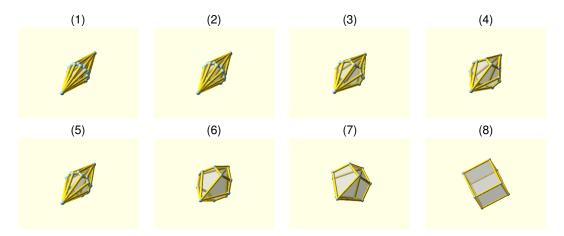


Table 11: trapezohedron

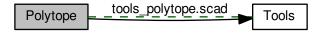


9.33 Polytope 223

9.33 Polytope

Polygon and polyhedron tools.

Collaboration diagram for Polytope:



Files

• file tools_polytope.scad

Polygon and polyhedron tools.

Functions

- module polytope_number (c, f, e, vi=true, fi=true, ei=true, sp=false, ts, th, to, tr=0)

 Label the vertices, paths, and edges of a polytope.
- module polytope_frame (c, f, e, vi=true, fi=true, ei=true, vc=1, fc=2, ec=0)

 Assemble a polytope skeletal frame using child objects.
- module polytope_bbox (c, f, a)

The 3d or 2d bounding box shape for a polytope.

9.33.1 Detailed Description

Polygon and polyhedron tools.

9.33.2 Function Documentation

9.33.2.1 module polytope_bbox (c , f , a)

The 3d or 2d bounding box shape for a polytope.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d cartesian coordinates [[x, y (, z)],].</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
а	< decimal-list-1:3 decimal> The box padding. A list of lengths to equally pad the box dimensions.

Generates: (1) the 3d box shape that completely encloses the defined 3d polyhedron with the box sides oriented parallel to the coordinate axes. Or: (2) the 2d box shape that exactly encloses the defined 2d polygon with the box sides oriented parallel to the coordinate axes.

Note

When f is not given, the listed order of the coordinates c establishes the path.

See also

polytope_limits for warning about secondary Shapes.

Definition at line 298 of file tools polytope.scad.

```
9.33.2.2 module polytope_frame ( c, f, e, vi = true, fi = true, ei = true, vc = 1, fc = 2, ec = 0 )
```

Assemble a polytope skeletal frame using child objects.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d coordinate points.</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>
vi	<index> Vertex index. An index sequence specification.</index>
fi	<index> Face index. An index sequence specification.</index>
ei	<index> Edge index. An index sequence specification.</index>
VC	<integer> Vertex child index.</integer>
fc	<integer> Face child index.</integer>
ec	<integer> Edge child index.</integer>

This function constructs a skeletal frame for a given polytope. A 2d child object is linearly extruded along specified edges of the polytope to form the frame. Additional 3d child objects can be centered on specified vertices and/or the mean coordinates of specified faces.

Example

```
include <tools/tools_polytope.scad>;
s = second(xy_plane_os) * 25;
p = linear_extrude_pp2pf(s, h=50);

polytope_frame(first(p), second(p)) {
   circle(r=2);
   color("grey") sphere(r=4);
   color("blue") cube(4);
}
```

Note

To disable a child assignment to the vertices, faces, or edges, use an index that is less than zero or greater than the number of children.

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

When e is not specified, it is computed from f using polytope_faces2edges().

Definition at line 216 of file tools_polytope.scad.

```
9.33.2.3 module polytope_number ( c, f, e, vi = true, fi = true, ei = true, sp = false, ts, th, to, tr = 0 )
```

Label the vertices, paths, and edges of a polytope.

9.33 Polytope 225

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d coordinate points.</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>
vi	<index> Vertex index. An index sequence specification.</index>
fi	<index> Face index. An index sequence specification.</index>
ei	<index> Edge index. An index sequence specification.</index>
sp	 boolean> Show polyhedron shape.
ts	<decimal> The text size override.</decimal>
th	<decimal> The text extrusion height override.</decimal>
to	<vector-3d vector-2d> The text offset override.</vector-3d vector-2d>
tr	<decimal-list-1:3 decimal> The text rotation (in degrees).</decimal-list-1:3 decimal>

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

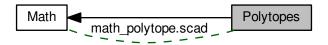
When e is not specified, it is computed from f using polytope_faces2edges().

Definition at line 78 of file tools_polytope.scad.

9.34 Polytopes

Polygon and polyhedron mathematical functions.

Collaboration diagram for Polytopes:



Files

· file math polytope.scad

Polygon and polyhedron mathematical functions.

Functions

function polytope_faces2edges (f)

List the edge coordinate index pairs of a polytope.

• function polytope limits (c, f, a, d=[0:2], s=true)

Determine the bounding limits of a polytope.

• function polytope_bbox_pf (c, f, a)

Generate a bounding box polytope for another polytope in 3d or 2d.

• function polytope_line (c, f, e, i, l, r=false)

Get a line from an edge or any two vetices of a polytope.

• function polytope_vertex_av (f, i)

List the adjacent vertices for a given polytope vertex.

• function polytope_vertex_af (f, i)

List the adjacent face indexes for a polytope vertex.

• function polytope_edge_af (f, e, i)

List the adjacent face indexes for a polytope edge.

• function polytope_vertex_n (c, f, i)

Get a normal vector for a polytope vertex.

• function polytope_edge_n (c, f, e, i)

Get a normal vector for a polytope edge.

• function polytope_face_n (c, f, i, l, cw=true)

Get the normal vector of a polytope face.

• function polytope face m (c, f, i, l)

Get the mean coordinate of all vertices of a polytope face.

function polytope_face_mn (c, f, i, l, cw=true)

Get the mean coordinate and normal vector of a polytope face.

function polytope plane (c, f, i, l, cw=true)

Get a plane for a polytope face.

• function polytope face vcounts (f)

List the vertex counts for all polytope faces.

• function polytope face angles (c, f)

List the angles between all adjacent faces of a polyhedron.

• function polytope_edge_lengths (c, e)

List the edge lengths of a polytope.

• function polytope_edge_angles (c, f)

List the adjacent edge angles for each polytope vertex.

function polytope_faces_are_regular (c, f, e, d=6)

Test if the faces of a polytope are all regular.

• function polytope_triangulate_ft (f)

Triangulate the faces of a convex polytope using fan triangulation.

function polygon2d_perimeter (c, p)

Calculate the perimeter length of a polygon in 2d.

function polygon2d_area (c, p, s=false)

Compute the signed area of a polygon in a Euclidean 2d-space.

• function polygon3d_area (c, p, n)

Compute the area of a polygon in a Euclidean 3d-space.

• function polygon2d_centroid (c, p)

Compute the center of mass of a polygon in a Euclidean 2d-space.

function polygon2d_is_cw (c, p)

Test the vertex ordering of a polygon in a Euclidean 2d-space.

function polygon2d_is_convex (c, p)

Test the convexity of a polygon in a Euclidean 2d-space.

function polygon2d_winding (c, p, t)

Compute the winding number of a polygon about a point in a Euclidean 2d-space.

• function polygon2d_is_pip_wn (c, p, t)

Test if a point is inside a polygon in a Euclidean 2d-space using winding number.

function polygon2d_is_pip_as (c, p, t)

Test if a point is inside a polygon in a Euclidean 2d-space using angle summation.

• function polyhedron_area (c, f)

Compute the surface area of a polyhedron in a Euclidean 3d-space.

function polyhedron_volume_tf (c, f)

Compute the volume of a triangulated polyhedron in a Euclidean 3d-space.

function polyhedron_centroid_tf (c, f)

Compute the center of mass of a triangulated polyhedron in a Euclidean 3d-space.

function linear_extrude_pp2pf (c, p, h=1, centroid=false, center=false)

Convert a polygon to a polyhedron by adding a height dimension.

9.34.1 Detailed Description

Polygon and polyhedron mathematical functions.

9.34.2 Function Documentation

9.34.2.1 function linear_extrude_pp2pf(c, p, h = 1, centroid = false, center = false)

Convert a polygon to a polyhedron by adding a height dimension.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.
h	<decimal> The polyhedron height.</decimal>
centroid	<boolean> Center polygon centroid at z-axis.</boolean>
center	<boolean> Center polyhedron height about xy-plane.</boolean>

Returns

<datastruct> A structure [points, faces], where points are <coords-3d> and faces are a <integer-list-list>, that define the bounding box of the given polyhedron.

Note

When p is not given, the listed order of the coordinates c establishes the path.

9.34.2.2 function polygon2d_area (c, p, s = false)

Compute the signed area of a polygon in a Euclidean 2d-space.

Parameters

С	<coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.
S	<boolean> Return the vertex ordering sign.</boolean>

Returns

<decimal> The area of the given polygon.

See Wikipedia for more information.

Note

When p is not given, the listed order of the coordinates c establishes the path.

Warning

This function does not track secondary shapes subtraction as implemented by the polygon() function.

9.34.2.3 function polygon2d_centroid (c , p)

Compute the center of mass of a polygon in a Euclidean 2d-space.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
---	--

р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.

Returns

<point-2d> The center of mass of the given polygon.

See Wikipedia for more information.

Note

When p is not given, the listed order of the coordinates c establishes the path.

Warning

This function does not track secondary shapes subtraction as implemented by the polygon() function.

```
9.34.2.4 function polygon2d_is_convex ( c , p )
```

Test the convexity of a polygon in a Euclidean 2d-space.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.

Returns

<boolean> true if the polygon is convex, false otherwise.

Note

When p is not given, the listed order of the coordinates c establishes the path.

9.34.2.5 function polygon2d_is_cw (c , p)

Test the vertex ordering of a polygon in a Euclidean 2d-space.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.

Returns

boolean> true if the vertex are ordered *clockwise*, false if the vertex are *counterclockwise* ordered, and undef if the ordering can not be determined.

Note

When p is not given, the listed order of the coordinates c establishes the path.

9.34.2.6 function polygon2d_is_pip_as (c , p , t)

Test if a point is inside a polygon in a Euclidean 2d-space using angle summation.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.
t	<pre><point-2d> A test point coordinate [x, y].</point-2d></pre>

Returns

<boolean> true when the point is *inside* the polygon and false otherwise.

See Wikipedia for more information.

Note

When p is not given, the listed order of the coordinates c establishes the path.

Warning

This function does not track secondary shapes subtraction as implemented by the polygon() function.

9.34.2.7 function polygon2d_is_pip_wn(c, p, t)

Test if a point is inside a polygon in a Euclidean 2d-space using winding number.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.
t	<pre><point-2d> A test point coordinate [x, y].</point-2d></pre>

Returns

 <boolean> true when the point is *inside* the polygon and false otherwise.

Note

When p is not given, the listed order of the coordinates c establishes the path.

See also

polygon2d_winding for warning about secondary Shapes.

9.34.2.8 function polygon2d_perimeter (c , p)

Calculate the perimeter length of a polygon in 2d.

Parameters

С	<coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.

Returns

<decimal> The sum of all polygon primary and secondary perimeter lengths.

Note

When p is not given, the listed order of the coordinates c establishes the path.

9.34.2.9 function polygon2d_winding(c,p,t)

Compute the winding number of a polygon about a point in a Euclidean 2d-space.

Parameters

С	<pre><coords-2d> A list of 2d cartesian coordinates [[x, y],].</coords-2d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.
t	<pre><point-2d> A test point coordinate [x, y].</point-2d></pre>

Returns

<integer> The winding number.

Computes the winding number, the total number of counterclockwise turns that the polygon paths makes around the test point in a Euclidean 2d-space. Will be 0 *iff* the point is outside of the polygon. Function patterned after Dan Sunday, 2012.

Copyright

Copyright 2000 softSurfer, 2012 Dan Sunday This code may be freely used and modified for any purpose providing that this copyright notice is included with it. iSurfer.org makes no warranty for this code, and cannot be held liable for any real or imagined damage resulting from its use. Users of this code must verify correctness for their application.

Note

When p is not given, the listed order of the coordinates c establishes the path.

Warning

Where there are secondary paths, the vertex ordering of each must be the same as the primary path.

9.34.2.10 function polygon3d_area (c , p , n)

Compute the area of a polygon in a Euclidean 3d-space.

Parameters

С	<pre><coords-3d> A list of 3d cartesian coordinates [[x, y, z],].</coords-3d></pre>
р	<integer-list-list> An optional list of paths that define one or more closed shapes where each is</integer-list-list>
	a list of coordinate indexes.
n	<vector-3d> An optional normal vector, [x, y, z], to the polygon plane. When not given, a normal</vector-3d>
	vector is constructed from the first three points of the primary path.

Returns

<decimal> The area of the given polygon.

Function patterned after Dan Sunday, 2012.

Note

When p is not given, the listed order of the coordinates c establishes the path.

Warning

This function does not track secondary shapes subtraction as implemented by the polygon() function.

9.34.2.11 function polyhedron_area (c , f)

Compute the surface area of a polyhedron in a Euclidean 3d-space.

Parameters

С	<pre><coords-3d> A list of 3d cartesian coordinates [[x, y, z],].</coords-3d></pre>
f	<integer-list-list> A list of faces that enclose the shape where each face is a list of coordinate</integer-list-list>
	indexes.

Returns

<decimal> The surface area of the given polyhedron.

9.34.2.12 function polyhedron_centroid_tf (c , f)

Compute the center of mass of a triangulated polyhedron in a Euclidean 3d-space.

Parameters

С	<pre><coords-3d> A list of 3d cartesian coordinates [[x, y, z],].</coords-3d></pre>
f	<integer-list-3-list> A list of triangular faces that enclose the polyhedron where each face is a</integer-list-3-list>
	list of three coordinate indexes.

Returns

<point-3d> The center of mass of the given polyhedron.

See Wikipedia for more information on centroid determined via the divergence theorem and midpoint quadrature.

Note

All faces are assumed to be a union of triangles oriented clockwise from the outside inwards.

9.34.2.13 function polyhedron_volume_tf (c , f)

Compute the volume of a triangulated polyhedron in a Euclidean 3d-space.

Parameters

С	<pre><coords-3d> A list of 3d cartesian coordinates [[x, y, z],].</coords-3d></pre>
f	<integer-list-3-list> A list of triangular faces that enclose the polyhedron where each face is a</integer-list-3-list>
	list of three coordinate indexes.

Returns

<decimal> The volume of the given polyhedron.

See Wikipedia for more information on volumes determined using the divergence theorem.

Note

All faces are assumed to be a union of triangles oriented clockwise from the outside inwards.

9.34.2.14 function polytope_bbox_pf(c,f,a)

Generate a bounding box polytope for another polytope in 3d or 2d.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d cartesian coordinates [[x, y (, z)],].</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
а	<decimal-list-1:3 decimal> The box padding. A list of lengths to equally pad the box dimensions.</decimal-list-1:3 decimal>

Returns

<datastruct> A structure: (1) [points, faces], where points are <coords-3d> and faces are a
<integer-list-list>, that define the bounding box of the given polyhedron. Or: (2) [points, path], where
points are <coords-2d> and path is a <integer-list-list>, that define the bounding box of the given polygon.

Polyhedron faces will be ordered *clockwise* when looking from outside the shape inwards. Polygon path will be ordered clockwise when looking from the top (positive z) downwards.

Note

When f is not specified, all coordinates are used to determine the geometric limits, which, simplifies the calculation. Parameter f is needed when a subset of the coordinates should be considered.

See also

polytope_limits for warning about secondary Shapes.

9.34.2.15 function polytope_edge_af (f, e, i)

List the adjacent face indexes for a polytope edge.

Parameters

Generated on Tue Apr 4 2017 20:15:43 for omdl by Doxygen

f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>
i	<integer> The edge index.</integer>

Returns

<integer-list> The list of face indexes adjacent to the given polytope edge.

Note

When e is not specified, it is computed from f using polytope_faces2edges().

9.34.2.16 function polytope_edge_angles (c , f)

List the adjacent edge angles for each polytope vertex.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d cartesian coordinates [[x, y (, z)],].</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.

Returns

<decimal-list> A list of the polytope adjacent edge angles.

9.34.2.17 function polytope_edge_lengths (c , e)

List the edge lengths of a polytope.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d cartesian coordinates [[x, y (, z)],].</coords-3d coords-2d>
е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>

Returns

<decimal-list> A list of the polytope edge lengths.

9.34.2.18 function polytope_edge_n (c , f , e , i)

Get a normal vector for a polytope edge.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d coordinate points.</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.

е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>
i	<integer> The edge index.</integer>

Returns

<vector-3d> A normal vector for the polytope edge.

The normal is computed as the mean of the adjacent faces.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

When e is not specified, it is computed from f using polytope_faces2edges() iff the line is identified by i.

9.34.2.19 function polytope_face_angles (c , f)

List the angles between all adjacent faces of a polyhedron.

Parameters

С	<pre><coords-3d> A list of 3d cartesian coordinates [[x, y, z],].</coords-3d></pre>
f	<integer-list-list> A list of faces that enclose the shape where each face is a list of coordinate</integer-list-list>
	indexes.

Returns

<decimal-list> A list of the polyhedron adjacent face angles.

See Wikipedia for more information on dihedral angles.

9.34.2.20 function polytope_face_m (c , f , i , l)

Get the mean coordinate of all vertices of a polytope face.

Parameters

С	<coords-3d coords-2d=""> A list of 3d or 2d coordinate points.</coords-3d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
i	<integer> The face specified as an face index.</integer>
1	<integer-list> The face specified as a list of all the coordinate indexes that define it.</integer-list>

Returns

<coords-3d> The mean coordinate of a polytope face.

The face can be identified using either parameter i or 1. When using 1, the parameter f is not required.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

9.34.2.21 function polytope_face_mn(c, f, i, I, cw = true)

Get the mean coordinate and normal vector of a polytope face.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d coordinate points.</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
i	<integer> The face specified as an face index.</integer>
1	<integer-list> The face specified as a list of all the coordinate indexes that define it.</integer-list>
CW	 boolean> Face vertex ordering.

Returns

<plane> [mp, nv], where mp is coords-3d, the mean coordinate, and nv is vector-3d, the normal vector, of the polytope face-plane.

The face can be identified using either parameter i or l. When using l, the parameter f is not required.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

9.34.2.22 function polytope_face_n (c, f, i, I, cw = true)

Get the normal vector of a polytope face.

Parameters

С	<coords-3d coords-2d=""> A list of 3d or 2d coordinate points.</coords-3d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
i	<integer> The face specified as an face index.</integer>
1	<integer-list> The face-plane specified as a list of three or more coordinate indexes that are a</integer-list>
	part of the face.
CW	

Returns

<vector-3d> The normal vector of a polytope face.

The face can be identified using either parameter i or 1. When using 1, the parameter f is not required.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

9.34.2.23 function polytope_face_vcounts (f)

List the vertex counts for all polytope faces.

Parameters

f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.

Returns

<integer-list> A list with a vertex count of every face.

9.34.2.24 function polytope_faces2edges (f)

List the edge coordinate index pairs of a polytope.

Parameters

f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.

Returns

<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes that form the shape.

Note

Although the edge list is not sorted, each pair is sorted with the smallest index first.

9.34.2.25 function polytope_faces_are_regular (c , f , e , d = 6)

Test if the faces of a polytope are all regular.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d cartesian coordinates [[x, y (, z)],].</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>
d	<integer> The number of significant figures used when comparing lengths and angles.</integer>

Returns

<booklean> true when there is both a single edge length and a single edge angle and false otherwise.

Note

When e is not specified, it is computed from f using polytope_faces2edges().

9.34.2.26 function polytope_limits (c, f, a, d = [0:2], s = true)

Determine the bounding limits of a polytope.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d cartesian coordinates [[x, y (, z)],].</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
а	<decimal-list-1:3 decimal> The box padding. A list of lengths to equally pad the box dimensions.</decimal-list-1:3 decimal>
d	<range integer="" list="" =""> The dimensions to consider. A range of dimensions, a list of dimensions,</range>
	or a single dimension.
S	<boolean> Return box size rather than coordinate limits.</boolean>

Returns

<datastruct> A list with the bounding-box limits (see: table).

The returned list will be of the following form:

	S	x	у	Z	datastruct form
2d	false	[min,max]	[min,max]	-	decimal-list-2-
					list-2
2d	true	max-min	max-min	-	decimal-list-list-
					2
3d	false	[min,max]	[min,max]	[min,max]	decimal-list-2-
					list-3
3d	true	max-min	max-min	max-min	decimal-list-list-
					3

Note

When f is not specified, all coordinates are used to determine the geometric limits, which, simplifies the calculation. Parameter f is needed when a subset of the coordinates should be considered.

Warning

This function does not track secondary shapes subtraction as implemented by the polygon() function.

9.34.2.27 function polytope_line(c,f,e,i,l,r = false)

Get a line from an edge or any two vetices of a polytope.

Parameters

С	<coords-3d coords-2d> A list of 3d or 2d coordinate points.</coords-3d coords-2d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
е	<integer-list-2-list> A list of edges where each edge is a list of two coordinate indexes.</integer-list-2-list>
i	<integer> A line specified as an edge index.</integer>
1	<integer-list-2> A line specified as a list of coordinate index pairs.</integer-list-2>
r	 <boolean> Reverse the line start and end points.</boolean>

Returns

line-3d line-2d> The line as a pair of coordinates.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

When e is not specified, it is computed from f using polytope_faces2edges() iff the line is identified by i.

```
9.34.2.28 function polytope_plane ( c, f, i, I, cw = true )
```

Get a plane for a polytope face.

Parameters

С	<coords-3d coords-2d=""> A list of 3d or 2d coordinate points.</coords-3d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
i	<integer> The face specified as an face index.</integer>
1	<integer-list> The face specified as a list of all the coordinate indexes that define it.</integer-list>
CW	<boolean> Face vertex ordering.</boolean>

Returns

<plane> [mp, nv], where mp is coords-3d, the mean coordinate, and nv is vector-3d, the normal vector, of the polytope face-plane.

The face can be identified using either parameter i or 1. When using 1, the parameter f is not required.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

9.34.2.29 function polytope_triangulate_ft (f)

Triangulate the faces of a convex polytope using fan triangulation.

Parameters

f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.

Returns

<integer-list-3-list> A list of triangular faces that enclose the polytope where each face is a list of three coordinate indexes with vertex ordering is maintained.

See Wikipedia for more information on fan triangulation.

Warning

This method does not support concave polytopes.

9.34.2.30 function polytope_vertex_af (f, i)

List the adjacent face indexes for a polytope vertex.

Parameters

f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
i	<integer> The vertex index.</integer>

Returns

<integer-list> The list of face indexes adjacent to the given polytope vertex.

9.34.2.31 function polytope_vertex_av (f, i)

List the adjacent vertices for a given polytope vertex.

Parameters

f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of coordinate indexes.</integer-list-list>
i	<integer> A vertex index.</integer>

Returns

<integer-list> The list of adjacent vertex indexes for the given vertex index.

The adjacent vertices are those neighboring vertices that are directly connected to the given vertex by a common edge.

Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

9.34.2.32 function polytope_vertex_n (c , f , i)

Get a normal vector for a polytope vertex.

Parameters

С	<coords-3d coords-2d="" =""> A list of 3d or 2d coordinate points.</coords-3d>
f	<integer-list-list> A list of faces (or paths) that enclose the shape where each face is a list of</integer-list-list>
	coordinate indexes.
i	<integer> The vertex index.</integer>

Returns

<vector-3d> A normal vector for the polytope vertex.

The normal is computed as the mean of the adjacent faces.

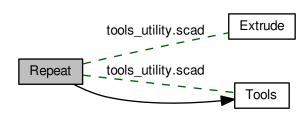
Note

Parameter f is optional for polygons. When it is not given, the listed order of the coordinates c establishes the polygon path.

9.35 Repeat

Shape repetition and distribution tools.

Collaboration diagram for Repeat:



Files

· file tools_utility.scad

Shape transformation utility tools.

Functions

• module radial_repeat (n, r=1, angle=true, move=false)

Distribute copies of a 2d or 3d shape equally about a z-axis radius.

• module grid_repeat (g, i, c=1, center=false)

Distribute copies of 2d or 3d shapes about Cartesian grid.

9.35.1 Detailed Description

Shape repetition and distribution tools.

9.35.2 Function Documentation

9.35.2.1 module grid_repeat (g , i , c = 1, center = false)

Distribute copies of 2d or 3d shapes about Cartesian grid.

Parameters

g	<integer-list-3 integer> The grid division count. A list [x, y, z] of integers or a single integer for</integer-list-3 integer>
	(x=y=z).

9.35 Repeat 243

i	<decimal-list-3 decimal="" =""> The grid increment size. A list [x, y, z] of decimals or a single decimal</decimal-list-3>
	for $(x=y=z)$.
С	<integer> The number of copies. Number of times to iterate over children.</integer>
center	 boolean> Center distribution about origin.

Example

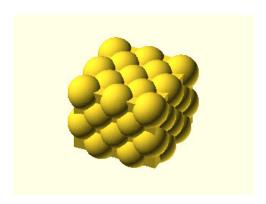


Figure 51: grid_repeat

grid_repeat(g=[5,5,4], i=10, c=50, center=true) {cube(10, center=true); sphere(10);}

Definition at line 356 of file tools_utility.scad.

9.35.2.2 module radial_repeat (n , r = 1, angle = true, move = false)

Distribute copies of a 2d or 3d shape equally about a z-axis radius.

Parameters

n	<integer> The number of equally spaced radii.</integer>
r	<decimal> The shape move radius.</decimal>
angle	<boolean> Rotate each copy about z-axis.</boolean>
move	<boolean> Move each shape copy to radii coordinate.</boolean>

Example

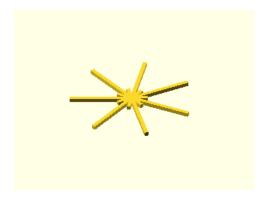


Figure 52: radial_repeat

radial_repeat(n=7, r=6, move=true) square([20,1], center=true);

244 **CONTENTS** Definition at line 325 of file tools_utility.scad.

9.36 Resolutions 245

9.36 Resolutions

Arch rendering resolution management.

Collaboration diagram for Resolutions:



Files

· file units_resolution.scad

An abstraction for arc rendering resolution control.

Functions

function resolution_fn (r)

Return facets number for the given arc radius.

• function resolution_fs ()

Return minimum facets size.

• function resolution_fa (r)

Return the minimum facets angle.

• function resolution_reduced ()

Return the radius at which arc resolution will begin to degrade.

module resolution_info (r)

Output resolution information to the console for given radius.

• function resolution_facets (r)

Return facet count used to render a radius.

• function resolution_facetsv (r)

Return facet count information list used to render a radius.

Variables

• \$resolution_mode = "fast"

<string> Global special variable that configures the arc resolution mode.

• \$resolution_value = 0

< number> Global special variable for modes that use custom resolutions.

9.36.1 Detailed Description

Arch rendering resolution management.

Return the minimum facets angle.

Functions, global variables, and configuration presets to provide a common mechanism for managing arc rendering resolution. Specifically, the number of fragments/facets with which arcs (circles, spheres, and cylinders, etc.) are rendered in OpenSCAD.

Example

```
include <units/units_resolution.scad>;
    base unit length = "in";
    // set resolution to 25 fpi
    $resolution_mode = "fpi";
    $resolution_value = 25;
    // use radius length of 1 inch
r = convert_length(1, "in");
    $fs=resolution_fs();
    $fa=resolution_fa( r );
    resolution_info( r );
    f = resolution_facets( r );
echo(str("for r = ", r, " ", unit_length_name(), ", facets = ", f));
Result (base unit length = mm):
1 ECHO: "$resolution_mode = [fpi], $resolution_value = 25, base_unit_length = millimeter"
2 ECHO: "$fn = 0, $fa = 2.29183, $fs = 1.016"
3 ECHO: "resolution reduction at radius > 25.4 millimeter"
4 ECHO: "for radius = 25.4 millimeter facets limited to 158 by $fs=1.016 millimeter"
5 ECHO: "for r = 25.4 millimeter, facets = 158"
Result (base_unit_length = cm):
1 ECHO: "$resolution_mode = [fpi], $resolution_value = 25, base_unit_length = centimeter"
2 ECHO: "$fn = 0, $fa = 2.29183, $fs = 0.1016"
3 ECHO: "resolution reduction at radius > 2.54 centimeter"
4 ECHO: "for radius = 2.54 centimeter facets limited to 158 by $fs=0.1016 centimeter"
5 ECHO: "for r = 2.54 centimeter, facets = 158"
Result (base_unit_length = mil):
1 ECHO: "$resolution_mode = [fpi], $resolution_value = 25, base_unit_length = thousandth"
2 ECHO: "$fn = 0, $fa = 2.29183, $fs = 40"
3 ECHO: "resolution reduction at radius > 1000 thousandth"
4 ECHO: "for radius = 1000 thousandth facets limited to 158 by $fs=40 thousandth"
5 ECHO: "for r = 1000 thousandth, facets = 158"
Result (base unit length = in):
1 ECHO: "$resolution_mode = [fpi], $resolution_value = 25, base_unit_length = inch"
2 ECHO: "$fn = 0, $fa = 2.29183, $fs = 0.04"
3 ECHO: "resolution reduction at radius > 1 inch"
4 ECHO: "for radius = 1 inch facets limited to 158 by $fs=0.04 inch"
5 ECHO: "for r = 1 inch, facets = 158"
9.36.2 Function Documentation
9.36.2.1 function resolution_fa(r)
```

9.36 Resolutions 247

Parameters

r	<decimal> The arc radius.</decimal>
---	-------------------------------------

Returns

<decimal> Minimum facet angle to be assigned to \$fa.

The result of this function can be assigned to the special variables \$fa to render arcs.

9.36.2.2 function resolution_facets (r)

Return facet count used to render a radius.

Parameters

r	<decimal> The arc radius.</decimal>

Returns

<integer> The number of fragments/facets that will be used to render a radius given the current values for \$fn, \$fa, and \$fs.

9.36.2.3 function resolution_facetsv (r)

Return facet count information list used to render a radius.

Parameters

```
r <decimal> The arc radius.
```

Returns

list-3> A 3-tuple list of the form: [facets <integer>,limiter <string>,value <decimal>].

Where facets is the number of fragments/facets that will be used to render the radius given the current values for \$fn, \$fa, and \$fs. limiter identifies the special variable that currently limits the facets, and value is the current value assigned to the limiter.

9.36.2.4 function resolution_fn (r)

Return facets number for the given arc radius.

Parameters

```
r <decimal> The arc radius.
```

Returns

<integer> The number of facets to be assigned to \$fn.

The result of this function can be assigned to the special variables fn to render arcs according to the resolution mode set by $resolution_mode$ and $resolution_value$.

The following table shows the modes that require \$resolution_value to be set prior to specifying the custom values used during resolution calculation.

\$resolution_mode	\$resolution_value sets	radius dependent
set	fixed value	no
upf	units per facet	yes
fpu	facets per unit	yes
fpi	facets per inch	yes

The following table has common resolution presets. Equivalent configuration can be obtained using \$resolution_mode and \$resolution_value as described in the preview table.

\$resolution_mode	preset description	radius dependent
fast	fast rendering mode	no
low	low resolution	yes
medium	medium resolution	yes
high	high resolution	yes
50um	50 micron per facets	yes
100um	100 micron per facets	yes
200um	200 micron per facets	yes
300um	300 micron per facets	yes
400um	400 micron per facets	yes
500um	500 micron per facets	yes
50mil	50 thousandth per facets	yes
100mil	100 thousandth per facets	yes
200mil	200 thousandth per facets	yes
300mil	300 thousandth per facets	yes
400mil	400 thousandth per facets	yes
500mil	500 thousandth per facets	yes

9.36.2.5 function resolution_fs ()

Return minimum facets size.

Returns

<integer> Minimum facet size to be assigned to \$fs.

The result of this function can be assigned to the special variables fs to render arcs according to the resolution mode set by $resolution_mode$ and $resolution_value$.

The following table shows the modes that require \$resolution_value to be set prior to calling this function in order to specify the custom values used during resolution calculation.

\$resolution_mode	\$resolution_value sets	radius dependent
set	fixed value	no
upf	units per facet	no
fpu	facets per unit	no
fpi	facets per inch	no

The following table has common resolution presets. Equivalent configuration can be obtained using \$resolution_mode and \$resolution_value as described in the preview table.

\$resolution_mode	preset description	radius dependent
fast	fast rendering mode	no
low	low resolution	no
medium	medium resolution	no

9.36 Resolutions 249

high	high resolution	no
50um	50 micron per facets	no
100um	100 micron per facets	no
200um	200 micron per facets	no
300um	300 micron per facets	no
400um	400 micron per facets	no
500um	500 micron per facets	no
50mil	50 thousandth per facets	no
100mil	100 thousandth per facets	no
200mil	200 thousandth per facets	no
300mil	300 thousandth per facets	no
400mil	400 thousandth per facets	no
500mil	500 thousandth per facets	no

9.36.2.6 module resolution_info (r)

Output resolution information to the console for given radius.

Parameters

r	<decimal> The arc radius.</decimal>

Definition at line 324 of file units resolution.scad.

9.36.2.7 function resolution_reduced ()

Return the radius at which arc resolution will begin to degrade.

Returns

<decimal> Transition radius where resolution reduction begins.

The special variables fs and fa work together when fn = 0. For a given fs, the fragment angle of a drawn arc gets smaller with increasing radius. In other words, the fragment angle is inversely proportional to the arc radius for a given fragment size.

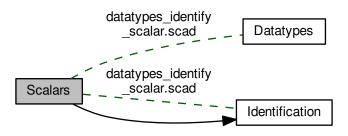
The special variable \$fa enforces a minimum fragment angle limit and at some radius, the fragment angle would becomes smaller than this limit. At this point, OpenSCAD limits further reduction in the facet angle which forces the use of increased fragment size. This in effect begins the gradual reduction of arc resolution with increasing radius.

The return result of this function indicates the radius at which this enforced limiting begins. When \$fn != 0, returns undef.

9.37 Scalars

Scalar data type identification.

Collaboration diagram for Scalars:



Files

• file datatypes_identify_scalar.scad Scalar data type identification.

Functions

function is_defined (v)

Test if a value is defined.

function not_defined (v)

Test if a value is not defined.

• function is_nan (v)

Test if a numerical value is invalid.

• function is_inf (v)

Test if a numerical value is infinite.

• function is_scalar (v)

Test if a value is a single non-iterable value.

• function is_iterable (v)

Test if a value has multiple parts and is iterable.

function is_empty (v)

Test if an iterable value is empty.

function is_number (v)

Test if a value is a number.

• function is_integer (v)

Test if a value is an integer.

function is_decimal (v)

Test if a value is a decimal.

function is_boolean (v)

9.37 Scalars 251

Test if a value is a predefined boolean constant.

function is_string (v)

Test if a value is a string.

function is list (v)

Test if a value is an iterable list of values.

function is_range (v)

Test if a value is a range definition.

function is_even (v)

Test if a numerical value is even.

• function is_odd (v)

Test if a numerical value is odd.

• function is_between (v, I, u)

Test if a numerical value is between an upper and lower bounds.

9.37.1 Detailed Description

Scalar data type identification.

See validation results.

9.37.2 Function Documentation

Test if a numerical value is between an upper and lower bounds.

Parameters

V	<number> A numerical value.</number>
1	<number> The minimum value.</number>
и	<number> The maximum value.</number>

Returns

boolean> true when the value is between the upper and lower bounds and false otherwise.

9.37.2.2 function is_boolean (v)

Test if a value is a predefined boolean constant.

Parameters

V	<value> A value.</value>

Returns

<boolean> true when the value is one of the predefined boolean constants [true|false] and false otherwise.

9.37.2.3 function is_decimal (v)

Test if a value is a decimal.

Parameters

V	<value> A value.</value>
---	--------------------------

Returns

<boolean> true when the value is a decimal and false otherwise.

9.37.2.4 function is_defined (v)

Test if a value is defined.

Parameters

```
v <value> A value.
```

Returns

<boolean> true when the value is defined and false otherwise.

9.37.2.5 function is_empty (v)

Test if an iterable value is empty.

Parameters

V	<value> An iterable value.</value>
---	------------------------------------

Returns

<boolean> true when the iterable value has zero elements and false otherwise.

9.37.2.6 function is_even (v)

Test if a numerical value is even.

Parameters

V	<value> A numerical value.</value>
---	------------------------------------

Returns

boolean> **true** when the value is determined to be *even* and **false** otherwise (The value may be positive or negative).

9.37.2.7 function is_inf (v)

Test if a numerical value is infinite.

Parameters

V	<value> A numerical value.</value>

Returns

boolean> **true** when the value is determined to be **inf** (greater than the largest representable number) and **false** otherwise.

9.37 Scalars 253

9.37.2.8 function is_integer (v)

Test if a value is an integer.

Parameters

V	<value> A value.</value>

Returns

<boolean> true when the value is an integer and false otherwise.

9.37.2.9 function is_iterable (v)

Test if a value has multiple parts and is iterable.

Parameters

Returns

<boolean> true when the value is an iterable multi-part value and false otherwise.

data type	defined
number(s)	false
boolean	false
string	true
list	true
range	not defined
undef	false
inf	false
nan	false

9.37.2.10 function is_list (v)

Test if a value is an iterable list of values.

Parameters

v <value> A value.</value>

Returns

<boolean> true when the value is a list and false otherwise.

9.37.2.11 function is_nan (v)

Test if a numerical value is invalid.

Parameters

v <value> A numerical value.</value>
--

Returns

 <boolean> true when the value is determined to be nan (Not A Number) and false otherwise.

9.37.2.12 function is_number (v)

Test if a value is a number.

9.37 Scalars 255

Parameters

V	<value> A value.</value>
---	--------------------------

Returns

<boolean> true when the value is a number and false otherwise.

Note

Returns true for inf and nan values.

9.37.2.13 function is_odd (v)

Test if a numerical value is odd.

Parameters

V	<value> A numerical value.</value>
---	------------------------------------

Returns

boolean> true when the value is determined to be
 odd and false otherwise (The value may be positive or negative).

9.37.2.14 function is_range (v)

Test if a value is a range definition.

Parameters

v <value> A value.</value>

Returns

<boolean> true when the value is a range definition and false otherwise.

9.37.2.15 function is_scalar (v)

Test if a value is a single non-iterable value.

Parameters

```
v <value> A value.
```

Returns

<boolean> true when the value is a single non-iterable value and false otherwise.

data type	defined
number(s)	true
boolean	true

string	false
list	false
range	not defined
undef	true
inf	true
nan	true

9.37.2.16 function is_string (v)

Test if a value is a string.

Parameters

V	<value> A value.</value>

Returns

<boolean> true when the value is a string and false otherwise.

9.37.2.17 function not_defined (v)

Test if a value is not defined.

Parameters

V	<value> A value.</value>

Returns

<boolean> true when the value is not defined and false otherwise.

9.38 Scalars 257

9.38 Scalars

Scalar data type operations.

Collaboration diagram for Scalars:



Files

• file datatypes_operate_scalar.scad

Scalar data type operations.

Functions

• function defined_or (v, d)

Return a value when it is defined or a default value when it is not.

• function circular_index (i, l, f=0)

Map an index position into a circularly indexed list.

9.38.1 Detailed Description

Scalar data type operations.

See validation results.

9.38.2 Function Documentation

9.38.2.1 function circular_index (i , I , f = 0)

Map an index position into a circularly indexed list.

Parameters

i	<integer> Any index, in or out of bounds.</integer>
1	<integer> The circular list length.</integer>
f	<integer> The starting index number.</integer>

Returns

<integer> A index position in the circular list within the range [f:1+f-1].

9.38.2.2 function defined_or (v , d)

Return a value when it is defined or a default value when it is not.

Parameters

V	<value> A value.</value>
d	<value> A default value.</value>

Returns

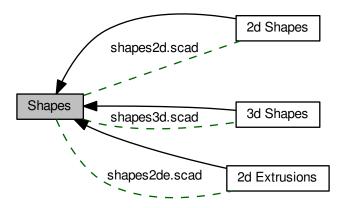
<value $> \lor$ when it is defined and d otherwise.

9.39 Shapes 259

9.39 Shapes

2d and 3d shapes.

Collaboration diagram for Shapes:



Modules

· 2d Extrusions

Extruded two-dimensional geometric shapes.

• 2d Shapes

Two-dimensional geometric shapes.

• 3d Shapes

Three-dimensional geometric shapes.

Files

· file shapes2d.scad

Two-dimensional basic shapes.

• file shapes2de.scad

Linearly extruded two-dimensional basic shapes.

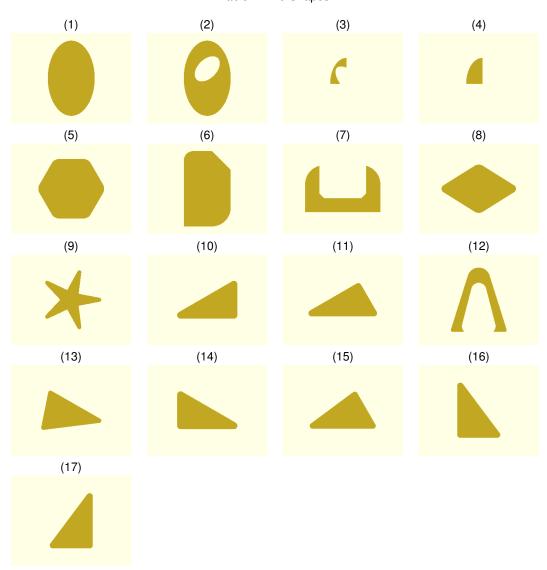
· file shapes3d.scad

Three-dimensional basic shapes.

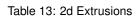
9.39.1 Detailed Description

2d and 3d shapes.

Table 12: 2d Shapes



9.39 Shapes 261



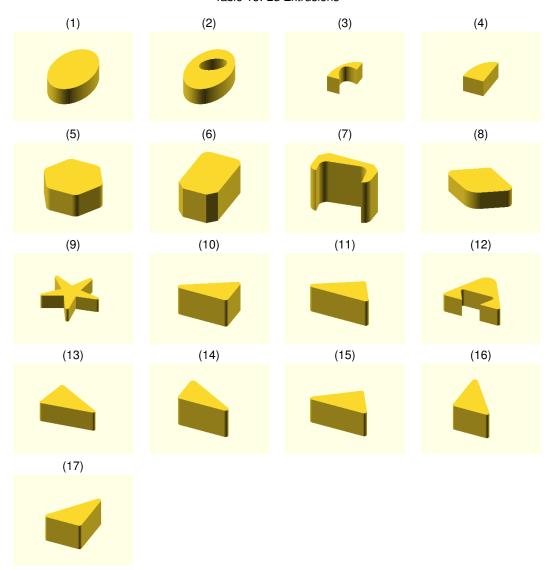
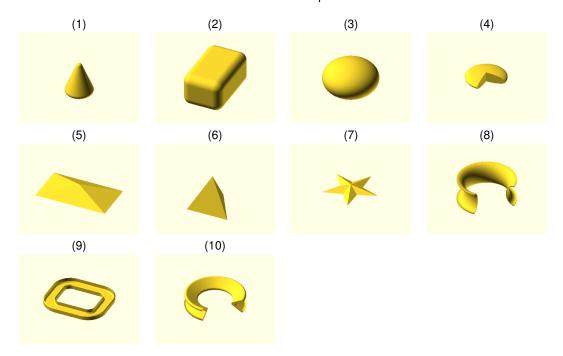


Table 14: 3d Shapes

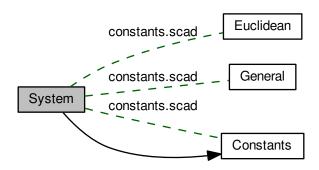


9.40 System 263

9.40 System

System and program limits.

Collaboration diagram for System:



Files

· file constants.scad

Design constant definitions.

Variables

```
• number_max = 1e308
```

<decimal> The largest representable number in OpenSCAD scripts.

• number_min = -1e308

<decimal> The smallest representable number in OpenSCAD scripts.

• empty_str = ""

<string> A string with no characters (the empty string).

• empty_lst = []

< list> A list with no values (the empty list).

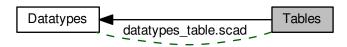
9.40.1 Detailed Description

System and program limits.

9.41 Tables

Table data type operations.

Collaboration diagram for Tables:



Files

· file datatypes_table.scad

Table data type operations.

Functions

• function get_table_ri (r, ri)

Get the table row index that matches a table row identifier.

• function get_table_r (r, ri)

Get the table row that matches a table row identifier.

• function get_table_ci (c, ci)

Get the table column index that matches a table column identifier.

• function get_table_c (c, ci)

Get the table column that matches a table column identifier.

• function get_table_v (r, c, ri, ci)

Get the table cell value for a specified row and column identifier.

• function get_table_crl (r, c, ci)

Form a list of a select column across all table rows.

function get_table_ridl (r)

Form a list of all table row identifiers.

function get_table_cidl (c)

Form a list of all table column identifiers.

• function table_exists (r, c, ri, ci)

Test the existence of a table row and column identifier.

• function get_table_size (r, c)

Get the size of a table.

• function get table copy (r, c, rl, cl)

Create a new matrix from select rows and columns of a table.

• function get_table_sum (r, c, rl, cl)

Sum select rows and columns of a table.

• module table check (r, c, verbose=false)

9.41 Tables 265

Perform some basic validation/checks on a table.

module table_dump (r, c, rl, cl, number=true)

Dump a table to the console.

9.41.1 Detailed Description

Table data type operations.

Example

```
<datatypes/datatypes_table.scad>;
base_unit_length = "mm";
table_cols =
[ // id, description
  ["id",
         "row identifier"],
  ["ht",
         "head type [r|h|s]"],
  ["td", "thread diameter"],
  ["tl", "thread length"],
  ["hd", "head diameter"],
  ["hl", "head length"],
  ["nd", "hex nut flat-to-flat width"],
  ["nl", "hex nut length"]
table rows =
 1;
table_check( table_rows, table_cols, true );
table_dump( table_rows, table_cols );
m3r16r tl = get table v( table rows, table cols, "m3r16r", "tl" );
if ( table_exists( c=table_cols, ci="nl" ) )
 echo ( "metric 'nl' available" );
table_ids = get_table_ridl( table_rows );
table_cols_tl = get_table_crl( table_rows, table_cols, "tl" );
echo ( table_ids=table_ids );
echo ( table_cols_tl=table_cols_tl );
tnew = get_table_copy( table_rows, table_cols, cl=["tl", "nl"] );
tsum = get_table_sum( table_rows, table_cols, cl=["tl", "nl"] );
echo ( m3r16r_tl=m3r16r_tl );
echo ( tnew=tnew );
echo ( tsum=tsum );
```

Result

```
1 ECHO: "[ INFO ] table_check(); begin table check"
2 ECHO: "[ INFO ] table_check(); row identifier found at column zero."
3 ECHO: "[ INFO ] table_check(); checking row column counts.
4 ECHO: "[ INFO ] table_check(); checking for repeat column identifiers."
5 ECHO: "[ INFO ] table_check(); checking for repeat row identifiers."
6 ECHO: "[ INFO ] table_check(); table size: 4 rows by 8 columns."
7 ECHO: "[ INFO ] table_check(); end table check"
8 ECHO: ""
9 ECHO: "row: 0"
10 ECHO: "[m3r08r] [id] (row identifier)
                                                         = [m3r08r]"
11 ECHO: "[m3r08r] [ht] (head type [r|h|s])
                                                        = [r]"
12 ECHO: "[m3r08r] [td] (thread diameter)
                                                        = [3]"
13 ECHO: "[m3r08r] [t1] (thread length)
14 ECHO: "[m3r08r] [hd] (head diameter)
                                                         = [8]"
                                                        = [5.5]'
15 ECHO: "[m3r08r] [h1] (head length)
                                                         = [3]"
```

```
16 ECHO: "[m3r08r] [nd] (hex nut flat-to-flat width) = [5.5]"
17 ECHO: "[m3r08r] [n1] (hex nut length)
18 ECHO: ""
19 ECHO: "row: 1"
20 ECHO: "[m3r14r] [id] (row identifier)
21 ECHO: "[m3r14r] [ht] (head type [r|h|s])
22 ECHO: "[m3r14r] [td] (thread diameter)
23 ECHO: "[m3r14r] [tl] (thread length)
                                                        = [14]"
24 ECHO: "[m3r14r] [hd] (head diameter)
25 ECHO: "[m3r14r] [h1] (head length)
26 ECHO: "[m3r14r] [nd] (hex nut flat-to-flat width) = [5.5]"
27 ECHO: "[m3r14r] [n1] (hex nut length) = [31.75]"
28 ECHO: ""
29 ECHO: "row: 2"
30 ECHO: "[m3r16r] [id] (row identifier)
                                                        = [m3r16r]"
31 ECHO: "[m3r16r] [ht] (head type [r|h|s])
                                                        = [r]"
32 ECHO: "[m3r16r] [td] (thread diameter)
                                                        = [3]"
33 ECHO: "[m3r16r] [tl] (thread length)
                                                        = [16]"
34 ECHO: "[m3r16r] [hd] (head diameter)
                                                       = [5.5]"
35 ECHO: "[m3r16r] [h1] (head length)
                                                        = [3]"
36 ECHO: "[m3r16r] [nd] (hex nut flat-to-flat width) = [5.5]'
37 ECHO: "[m3r16r] [n1] (hex nut length)
                                                       = [38.1]"
38 ECHO: ""
39 ECHO: "row: 3"
40 ECHO: "[m3r20r] [id] (row identifier)
                                                        = [m3r20r]"
41 ECHO: "[m3r20r] [ht] (head type [r|h|s])
                                                       = [r]"
                                                        = [3]"
42 ECHO: "[m3r20r] [td] (thread diameter)
43 ECHO: "[m3r20r] [tl] (thread length)
                                                        = [20]"
44 ECHO: "[m3r20r] [hd] (head diameter)
                                                        = [5.5]"
45 ECHO: "[m3r20r] [h1] (head length)
                                                        = [3]"
46 ECHO: "[m3r20r] [nd] (hex nut flat-to-flat width) = [5.5]"
47 ECHO: "[m3r20r] [nl] (hex nut length) = [44.45
                                                        = [44.45]"
48 ECHO: ""
49 ECHO: "table size: 4 rows by 8 columns."
50 ECHO: "metric 'nl' available"
51 ECHO: table_ids = ["m3r08r", "m3r14r", "m3r16r", "m3r20r"]
52 ECHO: table_cols_t1 = [8, 14, 16, 20]
53 ECHO: m3r16r_t1 = 16
54 ECHO: tnew = [[8, 25.4], [14, 31.75], [16, 38.1], [20, 44.45]]
55 \text{ ECHO: tsum} = [58, 139.7]
```

9.41.2 Function Documentation

9.41.2.1 function get_table_c (c , ci)

Get the table column that matches a table column identifier.

Parameters

```
c <matrix-2xC> The table column matrix (2 x C-columns).
ci <string> The column identifier.
```

Returns

list-2> The table column where the column identifier exists. If the identifier does not exists, returns undef.

9.41.2.2 function get_table_ci (c , ci)

Get the table column index that matches a table column identifier.

Parameters

С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
ci	<string> The column identifier.</string>

Returns

<integer> The column index where the identifier exists. If the identifier does not exists, returns empty lst.

9.41 Tables 267

9.41.2.3 function get_table_cidl (c)

Form a list of all table column identifiers.

Parameters

С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>

Returns

The list of all column identifiers.

Note

This functions assumes the first element of each table column to be the column identifier.

```
9.41.2.4 function get_table_copy ( r, c, rl, cl )
```

Create a new matrix from select rows and columns of a table.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
rl	<string-list> A list of selected row identifiers.</string-list>
cl	<string-list> A list of selected column identifiers.</string-list>

Returns

<matrix> A matrix of the selected rows and columns.

9.41.2.5 function get_table_crl (r, c, ci)

Form a list of a select column across all table rows.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
ci	<string> The column identifier.</string>

Returns

The list of a select column across all rows. If the identifier does not exists, returns undef.

9.41.2.6 function get_table_r (r , ri)

Get the table row that matches a table row identifier.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
ri	<string> The row identifier.</string>

Returns

The table row where the row identifier exists. If the identifier does not exists, returns undef.

9.41.2.7 function get_table_ri (r , ri)

Get the table row index that matches a table row identifier.

9.41 Tables 269

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
ri	<string> The row identifier.</string>

Returns

<integer> The row index where the identifier exists. If the identifier does not exists, returns empty_lst.

9.41.2.8 function get_table_ridl(r)

Form a list of all table row identifiers.

Parameters

```
r <matrix-CxR> The table data matrix (C-columns x R-rows).
```

Returns

The list of all row identifiers.

Note

This functions assumes the first element of each table row to be the row identifier, as enforced by the table_check(). As an alternative, the function get_table_crl(), of the form get_table_crl(r, c, "id"), may be used without this assumption.

9.41.2.9 function get_table_size (r , c)

Get the size of a table.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>

Returns

<decimal> The table size.

The size is reported as: (1) The number of rows when only the r parameter is specified. (2) The number of columns when only the r parameter is specified. (3) The (r * columns) when both parameters are specified.

9.41.2.10 function get_table_sum (r, c, rl, cl)

Sum select rows and columns of a table.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
rl	<string-list> A list of selected row identifiers.</string-list>

cl	<string-list> A list of selected column identifiers.</string-list>
----	--

Returns

A list with the sum of each selected rows and columns.

9.41.2.11 function get_table_v (r , c , ri , ci)

Get the table cell value for a specified row and column identifier.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
ri	<string> The row identifier.</string>
ci	<string> The column identifier.</string>

Returns

<value> The value of the matrix cell [ri, ci]. If either identifier does not exists, returns undef.

9.41.2.12 module table_check (r, c, verbose = false)

Perform some basic validation/checks on a table.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
verbose	 boolean> Be verbose during check.

Check that: (1) the first table column identifier is 'id'. (2) Make sure that each row has the same number of columns as defined in the columns vector. (3) Make sure that there are no repeating column identifiers. (4) Make sure that there are no repeating row identifiers.

Definition at line 297 of file datatypes_table.scad.

9.41.2.13 module table_dump (r, c, rl, cl, number = true)

Dump a table to the console.

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
rl	<string-list> A list of selected row identifiers.</string-list>
cl	<string-list> A list of selected column identifiers.</string-list>
number	<boolean> Number the table rows.</boolean>

Output each table row to the console. To output only select rows and columns, assign the desired identifiers to rl and cl. For example to output only the column identifiers 'c1' and 'c2', assign cl = ["c1", "c2"].

Definition at line 377 of file datatypes_table.scad.

9.41.2.14 function table_exists (r, c, ri, ci)

Test the existence of a table row and column identifier.

9.41 Tables 271

Parameters

r	<matrix-cxr> The table data matrix (C-columns x R-rows).</matrix-cxr>
С	<matrix-2xc> The table column matrix (2 x C-columns).</matrix-2xc>
ri	<string> The row identifier.</string>
ci	<string> The column identifier.</string>

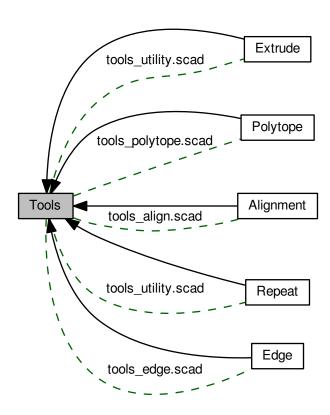
Returns

true if the row and column identifier exists, and false otherwise.

9.42 Tools

Design tools and techniques.

Collaboration diagram for Tools:



Modules

Alignment

Shape alignment tools.

• Edge

Shape edge finishing tools.

Extrude

Shape extrusion tools.

Polytope

Polygon and polyhedron tools.

Repeat

Shape repetition and distribution tools.

9.42 Tools 273

Files

• file tools_align.scad

Shape alignment tools.

file tools_edge.scad

Shape edge finishing tools.

• file tools_polytope.scad

Polygon and polyhedron tools.

file tools_utility.scad

Shape transformation utility tools.

9.42.1 Detailed Description

Design tools and techniques.

Table 15: Edge

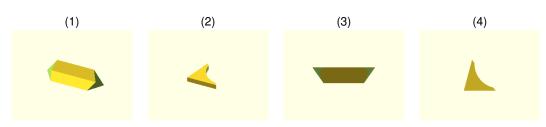
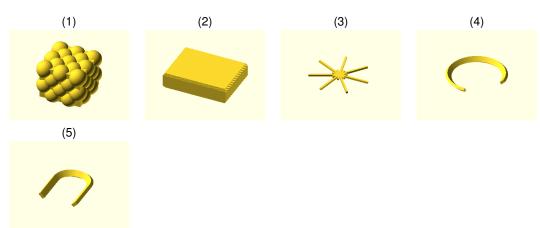


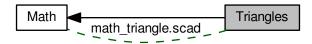
Table 16: Transformation Utilities



9.43 Triangles

Triangle mathematical functions.

Collaboration diagram for Triangles:



Files

· file math triangle.scad

Triangle solutions mathematical functions.

Functions

• function triangle_sss2lp (s1, s2, s3, cw=true)

Compute the vertex coordinates of a triangle given its side lengths.

• function triangle Is2lp (v, cw=true)

Compute the vertex coordinates of a triangle given its side lengths.

• function triangle_ppp2ls (v1, v2, v3)

Compute the side lengths of a triangle given its vertex coordinates.

• function triangle_lp2ls (v)

Compute the side lengths of a triangle given its vertex coordinates.

• function triangle_area_ppp (v1, v2, v3, s=false)

Compute the signed area of a triangle given its vertex coordinates.

function triangle_area_lp (v, s=false)

Compute the signed area of a triangle given its vertex coordinates.

• function triangle_centroid_ppp (v1, v2, v3)

Compute the centroid (geometric center) of a triangle.

function triangle_centroid_lp (v)

Compute the centroid (geometric center) of a triangle.

• function triangle_incenter_ppp (v1, v2, v3)

Compute the coordinate for the triangle's incircle.

function triangle_incenter_lp (v)

Compute the coordinate for the triangle's incircle.

function triangle inradius ppp (v1, v2, v3)

Compute the inradius of a triangle's incircle.

function triangle_inradius_lp (v)

Compute the inradius of a triangle's incircle.

function triangle is cw ppp (v1, v2, v3)

9.43 Triangles 275

Test the vertex ordering, or orientation, of a triangle.

function triangle_is_cw_lp (v)

Test the vertex ordering, or orientation, of a triangle.

function triangle_is_pit_ppp (v1, v2, v3, t)

Test if a point is inside a triangle in a Euclidean 2d-space using Barycentric.

function triangle_is_pit_lp (v, t)

Test if a point is inside a triangle in a Euclidean 2d-space using Barycentric.

9.43.1 Detailed Description

Triangle mathematical functions.

See Wikipedia for more information.

9.43.2 Function Documentation

9.43.2.1 function triangle_area_lp (v , s = false)

Compute the signed area of a triangle given its vertex coordinates.

Parameters

V	<pre><coords-2d> A list of vertex coordinates [v1, v2, v3].</coords-2d></pre>
S	 boolean> Return the vertex ordering sign.

Returns

<decimal> The area of the given triangle.

9.43.2.2 function triangle_area_ppp (v1 , v2 , v3 , s = false)

Compute the signed area of a triangle given its vertex coordinates.

Parameters

v1	<pre><point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>
S	<boolean> Return the vertex ordering sign.</boolean>

Returns

<decimal> The area of the given triangle.

9.43.2.3 function triangle_centroid_lp (v)

Compute the centroid (geometric center) of a triangle.

Parameters

V	<pre><coords-2d> A list of vertex coordinates [v1, v2, v3].</coords-2d></pre>
---	---

Returns

<point-2d> The centroid coordinate point [x, y].

9.43.2.4 function triangle_centroid_ppp (v1 , v2 , v3)

Compute the centroid (geometric center) of a triangle.

Parameters

v1	<pre><point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>

Returns

<point-2d> The centroid coordinate point [x, y].

9.43.2.5 function triangle_incenter_lp (v)

Compute the coordinate for the triangle's incircle.

Parameters

V	<coords-2d> A list of vertex coordinates [v1, v2, v3].</coords-2d>
•	Coolide Lay 7 Hot of Vortex coolidinates [V1, V2, V6].

Returns

<point-2d> The incircle coordinate point [x, y].

The interior point for which distances to the sides of the triangle are equal.

9.43.2.6 function triangle_incenter_ppp (v1 , v2 , v3)

Compute the coordinate for the triangle's incircle.

Parameters

V1	<pre><point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
<i>v3</i>	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>

Returns

<point-2d> The incircle coordinate point [x, y].

The interior point for which distances to the sides of the triangle are equal.

9.43.2.7 function triangle_inradius_lp (v)

Compute the inradius of a triangle's incircle.

9.43 Triangles 277

Parameters

	consumer Color A that of contact an auditorate a final color Color
\/	<pre><coords-2d> A list of vertex coordinates [v1, v2, v3].</coords-2d></pre>
y	Cooled Lay Tribt of Voltox Coolematos [VI, VL, VC].

Returns

<decimal> The incircle radius.

9.43.2.8 function triangle_inradius_ppp (v1, v2, v3)

Compute the inradius of a triangle's incircle.

Parameters

v1	<pre><point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>

Returns

<decimal> The incircle radius.

9.43.2.9 function triangle_is_cw_lp (v)

Test the vertex ordering, or orientation, of a triangle.

Parameters

v < coords-2d> A list of vertex coordinates [v1, v2, v3].

Returns

 <boolean> true if the vertices are ordered clockwise, false if the vertices are ordered counterclockwise, and
 undef if the ordering can not be determined.

9.43.2.10 function triangle_is_cw_ppp (v1 , v2 , v3)

Test the vertex ordering, or orientation, of a triangle.

Parameters

v1	<pre><point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
<i>v3</i>	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>

Returns

boolean> true if the vertices are ordered clockwise, false if the vertices are ordered counterclockwise, and
 undef if the ordering can not be determined.

9.43.2.11 function triangle_is_pit_lp (v , t)

Test if a point is inside a triangle in a Euclidean 2d-space using Barycentric.

Parameters

V	<coords-2d> A list of vertex coordinates [v1, v2, v3].</coords-2d>
t	<pre><point-2d> A test point coordinate [x, y].</point-2d></pre>

Returns

<boolean> true when the point is inside the polygon and false otherwise.

9.43.2.12 function triangle_is_pit_ppp (v1, v2, v3, t)

Test if a point is inside a triangle in a Euclidean 2d-space using Barycentric.

Parameters

v1	<pre><point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d></pre>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>
t	<pre><point-2d> A test point coordinate [x, y].</point-2d></pre>

Returns

<boolean> true when the point is inside the polygon and false otherwise.

See Wikipedia for more information.

9.43.2.13 function triangle_lp2ls (v)

Compute the side lengths of a triangle given its vertex coordinates.

Parameters

V	<pre><coords-2d> A list of vertex coordinates [v1, v2, v3].</coords-2d></pre>

Returns

<decimal-list-3> A list of side lengths [s1, s2, s3].

Note

Side lengths ordered according to vertex ordering.

9.43.2.14 function triangle_ls2lp (v, cw = true)

Compute the vertex coordinates of a triangle given its side lengths.

Parameters

V	<decimal-list-3> The list of side lengths [s1, s2, s3].</decimal-list-3>
CW	 boolean> Order vertices clockwise.

Returns

<coords-2d> A list of vertex coordinates [v1, v2, v3].

Geometry requires that s1 + s2 is greater then s3. A coordinates will be 'nan' when specified triangle does not exists.

9.43 Triangles 279

Note

Vertex v1 at the origin. Side length s1 is measured along the positive x-axis.

9.43.2.15 function triangle_ppp2ls (v1, v2, v3)

Compute the side lengths of a triangle given its vertex coordinates.

Parameters

V1	<point-2d> A vertex coordinate [x, y] for vertex 1.</point-2d>
v2	<pre><point-2d> A vertex coordinate [x, y] for vertex 2.</point-2d></pre>
v3	<pre><point-2d> A vertex coordinate [x, y] for vertex 3.</point-2d></pre>

Returns

<decimal-list-3> A list of side lengths [s1, s2, s3].

Note

Side lengths ordered according to vertex ordering.

9.43.2.16 function triangle_sss2lp (s1 , s2 , s3 , cw = true)

Compute the vertex coordinates of a triangle given its side lengths.

Parameters

s1	<decimal> The length of the side 1.</decimal>
s2	<decimal> The length of the side 2.</decimal>
s3	<pre><decimal> The length of the side 3.</decimal></pre>
CW	 boolean> Order vertices clockwise.

Returns

<coords-2d> A list of vertex coordinates [v1, v2, v3].

Geometry requires that s1 + s2 is greater then s3. A coordinates will be 'nan' when specified triangle does not exists.

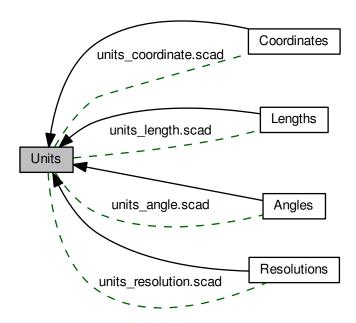
Note

Vertex v1 at the origin. Side length s1 is measured along the positive x-axis.

9.44 Units

Units and unit conversions.

Collaboration diagram for Units:



Modules

Angles

Angle units and conversions.

Coordinates

Coordinate systems and conversions.

Lengths

Length units and conversions.

· Resolutions

Arch rendering resolution management.

Files

• file units_angle.scad

Angle units and conversions.

• file units_coordinate.scad

Coordinate systems and conversions.

• file units_length.scad

Length units and conversions.

9.44 Units 281

• file units_resolution.scad

An abstraction for arc rendering resolution control.

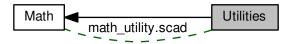
9.44.1 Detailed Description

Units and unit conversions.

9.45 Utilities

Miscellaneous mathematical utilities.

Collaboration diagram for Utilities:



Files

· file math_utility.scad

Miscellaneous mathematical utilities.

Functions

• function hist (v, m=0, cs, cb, cp, ca, cf, d=false)

Generate a histogram for the elements of a list of values.

9.45.1 Detailed Description

Miscellaneous mathematical utilities.

9.45.2 Function Documentation

9.45.2.1 function hist (
$$v$$
, $m = 0$, cs, cb, cp, ca, cf, $d = false$)

Generate a histogram for the elements of a list of values.

Parameters

V	<data> A list of values.</data>
т	<integer> The output mode (a 5-bit encoded integer).</integer>
CS	<string-list-4> A list of strings [s1, s2, s3, fs] (for custom field formatting).</string-list-4>

Returns

<list|string> with the occurrence frequency of the elements of v.

The custom formatting strings are inserted in the output stream as follows:

```
s1, value, s2, value-frequency, s3, fs
```

See lstr_html() for description of the html formatting parameters cb, cp, ca, cf, and d.

Output mode selection:

9.45 Utilities 283

bit	Description	0	1
0	output mode	numerical	string
1-3	string mode format	see table	see table
4	field separator mode	not at end	all

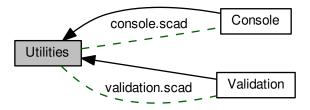
String output modes:

	B3	B2	B1	B0	Description
1	0	0	0	1	list of strings
3	0	0	1	1	text format 1
9	1	0	0	1	html format 1
15	1	1	1	1	custom
					formating

9.46 Utilities

General utilities.

Collaboration diagram for Utilities:



Modules

• Console

Console message logging.

· Validation

Function validation methods.

Files

• file console.scad

Message logging functions.

· file validation.scad

Function validation methods.

9.46.1 Detailed Description

General utilities.

9.47 Validation 285

9.47 Validation

Function validation methods.

Collaboration diagram for Validation:



Files

· file validation.scad

Function validation methods.

Functions

• function validate (d, cv, t, ev, p=4, pf=false)

Compare a computed test value with an known good result.

9.47.1 Detailed Description

Function validation methods.

Example

```
use <validation.scad>;
use <console.scad>;
// function to validate
function f1(x) = (x == undef) ? 1 : 2;
farg = undef;
                  // function test argument
                 // correct expected function result
// incorrect expected function result
erv1 = 1;
erv2 = 3;
// pass test example
pass_result = validate("test-a f1(farg)", f1(farg), "equals", erv1);
if ( !validate(cv=f1(farg), t="equals", ev=erv1, pf=true) )
  log_warn( pass_result );
else
 log_info( pass_result );
// fail test example
fail_result = validate("test-b f1(farg)", f1(farg), "equals", erv2);
if ( !validate(cv=f1(farg), t="equals", ev=erv2, pf=true) )
  log_warn( fail_result );
```

```
else
    log_info( fail_result );

//

// almost equal test example

//
tvael = [[90.001], [[45.009], true]];
tvae2 = [[90.002], [[45.010], true]];

log_info( validate("test-c", tvae1, "almost", tvae2, 3) );
log_warn( validate("test-d", tvae1, "almost", tvae2, 4) );
```

Result

9.47.2 Function Documentation

9.47.2.1 function validate (d, cv, t, ev, p = 4, pf = false)

Compare a computed test value with an known good result.

Parameters

d	<string> A description.</string>
CV	<value> A computed value to validate.</value>
t	<string boolean> The validation type.</string boolean>
ev	<value> The expected good value.</value>
р	<number> A numerical precision for approximate comparisons.</number>
pf	<boolean> Report result as pass or fail boolean value.</boolean>

Returns

<string|boolean> Validation result indicating if the test passed or failed.

validation types	pass if (else fail)	
"almost"	cv almost equals ev	
"equals"	cv equals ev	
"not"	cv not equal to ev	
"true" true	cv is true	
"false" false	cv is false	

Note

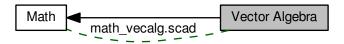
When performing an "almost" equal validation, the comparison precision is controlled by p. This specifies the number of digits of precision for each numerical comparison. A passing result indicates that cv equals ev to the number of decimal digits specified by p. The comparison is performed by the function almost_equal().

9.48 Vector Algebra 287

9.48 Vector Algebra

Algebraic operations on Euclidean vectors.

Collaboration diagram for Vector Algebra:



Files

· file math_vecalg.scad

Vector algebra mathematical functions.

Functions

function distance_pp (p1, p2)

Compute the distance between two Euclidean points.

function is_left_ppp (p1, p2, p3)

Test if a point is left, on, or right of an infinite line in a Euclidean 2d-space.

function dimension_2to3_v (v)

Return 3d vector unchanged or add a zeroed third dimension to 2d vector.

function get_line_dim (I)

Return the number of dimensions of a Euclidean line (or vector).

• function get_line_tp (l)

Return the terminal point of a Euclidean line (or vector).

• function get_line_ip (I)

Return the initial point of a Euclidean line (or vector).

• function get_line2origin (I)

Shift a Euclidean line (or vector) to the origin.

• function dot_II (I1, I2)

Compute the dot product of two lines (or vectors).

function cross_II (I1, I2)

Compute the cross product of two lines (or vectors) in a Euclidean 3d or 2d-space.

• function striple_III (I1, I2, I3)

Compute the scalar triple product of three lines (or vectors) in a Euclidean 3d or 2d-space.

function angle II (I1, I2)

Compute the angle between two lines (or vectors) in a Euclidean 3d or 2d-space.

function angle_III (I1, I2, n)

Compute the angle between two lines (or vectors) in a Euclidean 3d-space.

• function unit I (I)

Compute the normalized unit vector of a Euclidean line (or vector).

function are coplanar III (I1, I2, I3, d=6)

Test if three lines (or vectors) are coplanar in Euclidean 3d-space.

• function get_pnorm2nv (pn, cw=true)

Convert a planes' normal specification into a normal vector.

9.48.1 Detailed Description

Algebraic operations on Euclidean vectors.

See validation results.

9.48.2 Function Documentation

```
9.48.2.1 function angle_II ( I1 , I2 )
```

Compute the angle between two lines (or vectors) in a Euclidean 3d or 2d-space.

Parameters

/1	line-3d line-2d> A 3d or 2d line (or vector) 1.
12	line-3d line-2d> A 3d or 2d line (or vector) 2.

Returns

<decimal> The angle between the two lines (or vectors) in degrees. Returns **undef** when lines (or vectors) have different dimensions or when they do not intersect.

See Lines and vectors for argument specification and conventions.

Note

For 3d lines (or vectors), a normal is required to uniquely identify the perpendicular plane and axis of rotation. This function calculates the positive angle, and the plane and axis of rotation will be that which fits this assumed positive angle.

See also

angle_III().

9.48.2.2 function angle_III (I1 , I2 , n)

Compute the angle between two lines (or vectors) in a Euclidean 3d-space.

Parameters

<i>l</i> 1	A 3d line (or vector) 1.
12	A 3d line (or vector) 2.
n	A 3d normal line (or vector).

9.48 Vector Algebra 289

Returns

<decimal> The angle between the two lines (or vectors) in degrees. Returns **undef** when lines (or vectors) have different dimensions or when they do not intersect.

See Lines and vectors for argument specification and conventions.

See also

```
angle II().
```

```
9.48.2.3 function are_coplanar_III ( I1 , I2 , I3 , d = 6 )
```

Test if three lines (or vectors) are coplanar in Euclidean 3d-space.

Parameters

l1	A 3d line (or vector) 1.
12	A 3d line (or vector) 2.
13	A 3d line (or vector) 3.
d	<integer> The number of decimal places to consider.</integer>

Returns

<boolean> true when all three lines (or vectors) are coplanar, and false otherwise.

See Lines and vectors for argument specification and conventions. See Wikipedia for more information.

Note

When lines (or vectors) are specified with start and end points, this function tests if they are in a planes parallel to the coplanar.

```
9.48.2.4 function cross_II ( I1 , I2 )
```

Compute the cross product of two lines (or vectors) in a Euclidean 3d or 2d-space.

Parameters

11	line-3d line-2d> A 3d or 2d line (or vector) 1.
12	line-3d line-2d> A 3d or 2d line (or vector) 2.

Returns

<decimal|vector-2d> The cross product of 11 with 12. Returns undef when lines (or vectors) have different dimensions.

This function supports the abstraction outlined in Lines and vectors. The built-in operation will be more efficient in situations that do not make use of the aforementioned abstraction.

See Lines and vectors for argument specification and conventions. See Wikipedia cross and determinant for more information.

Note

This function returns the 2x2 determinant for 2d vectors.

9.48.2.5 function dimension_2to3_v (v)

Return 3d vector unchanged or add a zeroed third dimension to 2d vector.

Parameters

V	<vector-3d vector-2d> A vector.</vector-3d vector-2d>

Returns

<vector-3d> The 3d vector or the 2d vector converted to 3d with its third dimension assigned zero.

Warning

To reduce overhead, this function assumes any vector that is not 3d to be 2d.

9.48.2.6 function distance_pp (p1 , p2)

Compute the distance between two Euclidean points.

Parameters

p1	oint> A point coordinate 1.
p2	<pre><point> A point coordinate 2.</point></pre>

Returns

<decimal> The distance between the two points. Returns **undef** when points do not have equal dimensions.

When p2 is not given, it is assumed to be at the origin. This function is similar to norm.

9.48.2.7 function dot_II (I1 , I2)

Compute the dot product of two lines (or vectors).

Parameters

11	A n-dimensional line (or vector) 1.
12	A n-dimensional line (or vector) 2.

Returns

<decimal> The dot product of 11 with 12. Returns undef when lines (or vectors) have different dimensions.

This function supports the abstraction outlined in Lines and vectors. The built-in operation will be more efficient in situations that do not make use of the aforementioned abstraction.

See Lines and vectors for argument specification and conventions. See Wikipedia for more information.

9.48.2.8 function get_line2origin (I)

Shift a Euclidean line (or vector) to the origin.

Parameters

1	A line (or vector).

Returns

<vector> The line (or vector) shifted to the origin.

See Lines and vectors for argument specification and conventions.

9.48 Vector Algebra 291

9.48.2.9 function get_line_dim (I)

Return the number of dimensions of a Euclidean line (or vector).

Parameters

1	A line (or vector).

Returns

<integer> The number of dimensions for the line (or vector).

See Lines and vectors for argument specification and conventions.

```
9.48.2.10 function get_line_ip(I)
```

Return the initial point of a Euclidean line (or vector).

Parameters

```
/ line> A line (or vector).
```

Returns

<point> The initial point of the line (or vector).

See Lines and vectors for argument specification and conventions.

```
9.48.2.11 function get_line_tp(I)
```

Return the terminal point of a Euclidean line (or vector).

Parameters

```
/ line> A line (or vector).
```

Returns

<point> The terminal point of the line (or vector).

See Lines and vectors for argument specification and conventions.

```
9.48.2.12 function get_pnorm2nv ( pn , cw = true )
```

Convert a planes' normal specification into a normal vector.

Parameters

pn	<pre><pnorm> A plane normal specification.</pnorm></pre>
CW	 boolean> Point ordering. When the plane specified as non-collinear points, this indicates
	ordering.

Returns

<normal> A vector-3d normal to the plane.

See Planes' normal for argument specification and conventions.

```
9.48.2.13 function is_left_ppp ( p1 , p2 , p3 )
```

Test if a point is left, on, or right of an infinite line in a Euclidean 2d-space.

9.48 Vector Algebra 293

Parameters

p1	<point-2d> A 2d point coordinate 1.</point-2d>
p2	<pre><point-2d> A 2d point coordinate 2.</point-2d></pre>
рЗ	<point-2d> A 2d point coordinate 3.</point-2d>

Returns

<decimal> (> 0) for p3 *left* of the line through p1 and p2, (= 0) for p3 *on* the line, and (< 0) for p3 right of the line

Function patterned after Dan Sunday, 2012.

9.48.2.14 function striple_III (I1 , I2 , I3)

Compute the scalar triple product of three lines (or vectors) in a Euclidean 3d or 2d-space.

Parameters

11	line-3d line-2d> A 3d or 2d line (or vector) 1.
12	line-3d line-2d> A 3d or 2d line (or vector) 2.
13	line-3d line-2d> A 3d or 2d line (or vector) 3.

Returns

<decimal|vector-2d> The scalar triple product. Returns undef when lines (or vectors) have different dimensions.

$$[11, 12, 13] = 11 * (12 x 13)$$

See Lines and vectors for argument specification and conventions. See Wikipedia for more information.

Warning

Returns a 2d vector result for 2d vectors. The cross product computes the 2x2 determinant of the vectors (12 \times 13), a scalar value, which is then *multiplied* by 11.

9.48.2.15 function unit_I (I)

Compute the normalized unit vector of a Euclidean line (or vector).

Parameters

1	< A line (or vector).

Returns

<vector> The normalized unit vector.

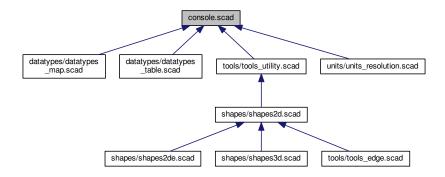
See Lines and vectors for argument specification and conventions.

10 File Documentation

10.1 console.scad File Reference

Message logging functions.

This graph shows which files directly or indirectly include this file:



Functions

• function stack (b=0, t=0)

Format the function call stack as a string.

• module log_echo (m)

Output message to console.

• module log_debug (m)

Output diagnostic message to console.

• module log_info (m)

Output information message to console.

• module log_warn (m)

Output warning message to console.

• module log_error (m)

Output error message to console.

10.1.1 Detailed Description

Message logging functions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

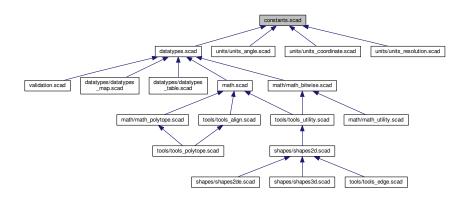
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

10.2 constants.scad File Reference

Design constant definitions.

This graph shows which files directly or indirectly include this file:



Variables

- aeps = 0.001
 - <decimal> Epsilon, small distance to deal with overlapping shapes.
- pi = 3.14159265358979323
 - <decimal> The ratio of a circle's circumference to its diameter.
- tau = 2*pi
 - <decimal> The ratio of a circle's circumference to its radius.
- phi = (1 + sqrt(5)) / 2
 - <decimal> The golden ratio.
- number max = 1e308
 - <decimal> The largest representable number in OpenSCAD scripts.
- number_min = -1e308
 - <decimal> The smallest representable number in OpenSCAD scripts.

```
• empty_str = ""
      <string> A string with no characters (the empty string).

    empty_lst = []

      < list> A list with no values (the empty list).
• x axis ci = 0
      <integer> The coordinate axis index for the Euclidean space x-axis.

    y axis ci = 1

      <integer> The coordinate axis index for the Euclidean space y-axis.

    z axis ci = 2

      <integer> The coordinate axis index for the Euclidean space z-axis.
• zero2d = [0, 0]
      <decimal-list-2> A 2d zero vector (a list with two zeros).
• origin2d = [0, 0]
      <point-2d> The origin point coordinate in 2d Euclidean space.
• x = [1, 0]
      <vector-2d> The unit vector of the positive x-axis in 2d Euclidean space.
y_axis2d_uv = [0, 1]
      <vector-2d> The unit vector of the positive y-axis in 2d Euclidean space.

    x_axis2d_ul = [-x_axis2d_uv, +x_axis2d_uv]

      A positively-directed unit line centered on the x-axis in 2d Euclidean space.

    y_axis2d_ul = [-y_axis2d_uv, +y_axis2d_uv]

      A positively-directed unit line centered on the y-axis in 2d Euclidean space.
• zero3d = [0, 0, 0]
      <decimal-list-2> A 3d zero vector (a list with three zeros).
• origin3d = [0, 0, 0]
      <point-3d> The origin point coordinate in 3-dimensional Euclidean space.
x_axis3d_uv = [1, 0, 0]
      <vector-3d> The unit vector of the positive x-axis in 3d Euclidean space.

    y axis3d uv = [0, 1, 0]

      <vector-3d> The unit vector of the positive y-axis in 3d Euclidean space.
• z axis3d uv = [0, 0, 1]
      <vector-3d> The unit vector of the positive z-axis in 3d Euclidean space.

    x_axis3d_ul = [-x_axis3d_uv, +x_axis3d_uv]

      A positively-directed unit line centered on the x-axis in 3d Euclidean space.

    y_axis3d_ul = [-y_axis3d_uv, +y_axis3d_uv]

      A positively-directed unit line centered on the y-axis in 3d Euclidean space.

    z_axis3d_ul = [-z_axis3d_uv, +z_axis3d_uv]

      < A positively-directed unit line centered on the z-axis in 3d Euclidean space.</p>
xy_plane_on = [origin3d, z_axis3d_uv]
      <plane> The right-handed xy plane centered at the origin with normal vector.
yz_plane_on = [origin3d, x_axis3d_uv]
      <plane> The right-handed yz plane centered at the origin with normal vector.
zx_plane_on = [origin3d, y_axis3d_uv]
      <plane> The right-handed zx plane centered at the origin with normal vector.

    xy_plane_os = [origin3d, [for (r=[[1,1],[1,-1],[-1,-1],[-1,1]]) [r[0],r[1],0]]]

      <plane> The right-handed xy plane centered at the origin with coplanar unit square points.

    yz_plane_os = [origin3d, [for (r=[[1,1],[1,-1],[-1,-1],[-1,1]]) [0,r[0],r[1]]]]

      <plane> The right-handed yz plane centered at the origin with coplanar unit square points.

    zx_plane_os = [origin3d, [for (r=[[1,1],[1,-1],[-1,-1],[-1,1]]) [r[1],0,r[0]]]]
```

<plane> The right-handed zx plane centered at the origin with coplanar unit square points.

10.2.1 Detailed Description

Design constant definitions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the omdl ; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

Note

Include this library file using the include statement.

10.3 database/geometry/polyhedra/anti_prisms.scad File Reference

Table of polyhedra data group: anti_prisms.

Variables

- dtc_polyhedra_anti_prisms
- dtr_polyhedra_anti_prisms

10.3.1 Detailed Description

Table of polyhedra data group: anti_prisms.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.4 database/geometry/polyhedra/archimedean.scad File Reference

Table of polyhedra data group: archimedean.

Variables

- dtc_polyhedra_archimedean
- dtr_polyhedra_archimedean

10.4.1 Detailed Description

Table of polyhedra data group: archimedean.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.5 database/geometry/polyhedra/archimedean_duals.scad File Reference

Table of polyhedra data group: archimedean_duals.

Variables

- · dtc_polyhedra_archimedean_duals
- dtr_polyhedra_archimedean_duals

10.5.1 Detailed Description

Table of polyhedra data group: archimedean_duals.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.6 database/geometry/polyhedra/cupolas.scad File Reference

Table of polyhedra data group: cupolas.

Variables

- · dtc_polyhedra_cupolas
- · dtr_polyhedra_cupolas

10.6.1 Detailed Description

Table of polyhedra data group: cupolas.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.7 database/geometry/polyhedra/dipyramids.scad File Reference

Table of polyhedra data group: dipyramids.

Variables

- dtc_polyhedra_dipyramids
- dtr_polyhedra_dipyramids

10.7.1 Detailed Description

Table of polyhedra data group: dipyramids.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.8 database/geometry/polyhedra/johnson.scad File Reference

Table of polyhedra data group: johnson.

Variables

- dtc_polyhedra_johnson
- dtr_polyhedra_johnson

10.8.1 Detailed Description

Table of polyhedra data group: johnson.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the **include** statement.

10.9 database/geometry/polyhedra/platonic.scad File Reference

Table of polyhedra data group: platonic.

Variables

- dtc_polyhedra_platonic
- dtr_polyhedra_platonic

10.9.1 Detailed Description

Table of polyhedra data group: platonic.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.10 database/geometry/polyhedra/polyhedra_all.scad File Reference

Table of polyhedra data group: polyhedra_all.

Variables

- dtc_polyhedra_polyhedra_all
- dtr_polyhedra_polyhedra_all

10.10.1 Detailed Description

Table of polyhedra data group: polyhedra_all.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the **include** statement.

10.11 database/geometry/polyhedra/prisms.scad File Reference

Table of polyhedra data group: prisms.

Variables

- dtc_polyhedra_prisms
- dtr_polyhedra_prisms

10.11.1 Detailed Description

Table of polyhedra data group: prisms.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.12 database/geometry/polyhedra/pyramids.scad File Reference

Table of polyhedra data group: pyramids.

Variables

- dtc_polyhedra_pyramids
- dtr_polyhedra_pyramids

10.12.1 Detailed Description

Table of polyhedra data group: pyramids.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the include statement.

10.13 database/geometry/polyhedra/trapezohedron.scad File Reference

Table of polyhedra data group: trapezohedron.

Variables

- · dtc_polyhedra_trapezohedron
- dtr_polyhedra_trapezohedron

10.13.1 Detailed Description

Table of polyhedra data group: trapezohedron.

Author

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

This omd1 formatted data table has been assembled using a script that converts the polyhedra data obtained from Anthony Thyssen's Studies into Polyhedra. The vertices are tabulated in both their original Cartesian as well as their converted spherical coordinate form, which is convenient when scaling. The data originates from one of three sources:

- Exact Mathematics as presented by Anthony Thyssen,
- the Polyhedron Database maintained by Netlib, and
- an Encyclopedia of Polyhedra by George W. Hart.

Note

Include this library file using the **include** statement.

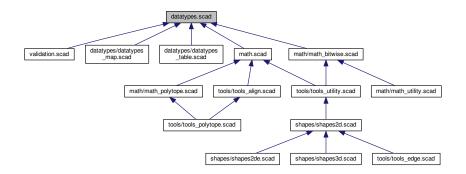
10.14 datatypes.scad File Reference

Data type identification and operations.

```
#include <constants.scad>
#include <datatypes/datatypes_identify_scalar.scad>
#include <datatypes/datatypes_identify_iterable.scad>
#include <datatypes/datatypes_identify_list.scad>
#include <datatypes/datatypes_operate_scalar.scad>
#include <datatypes/datatypes_operate_iterable.scad>
#include <datatypes/datatypes_operate_list.scad>
Include dependency graph for datatypes.scad:
```



This graph shows which files directly or indirectly include this file:



10.14.1 Detailed Description

Data type identification and operations.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

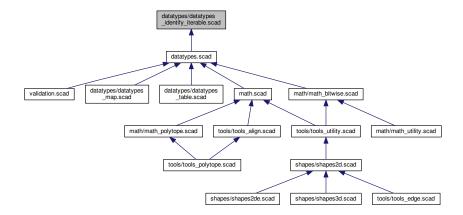
Note

Include this library file using the include statement.

10.15 datatypes/datatypes_identify_iterable.scad File Reference

Iterable data type identification.

This graph shows which files directly or indirectly include this file:



Functions

• function all_equal (v, cv)

Test if a list of values equal a comparison value.

function any_equal (v, cv)

Test if any element of a list of values equal a comparison value.

• function all_defined (v)

Test if no element of a list of values is undefined.

function any undefined (v)

Test if any element of a list of values is undefined.

• function all_scalars (v)

Test if all elements of a list of values are scalars.

function all_lists (v)

Test if all elements of a list of values are lists.

• function all_strings (v)

Test if all elements of a list of values are strings.

• function all numbers (v)

Test if all elements of a list of values are numbers.

• function all_len (v, l)

Test if all elements of a list of values are lists of a specified length.

10.15.1 Detailed Description

Iterable data type identification.

Author

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

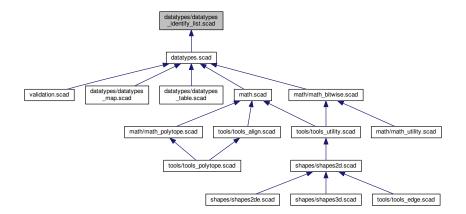
Note

Include this library file using the include statement.

10.16 datatypes/datatypes_identify_list.scad File Reference

List data type identification.

This graph shows which files directly or indirectly include this file:



Functions

function n_almost_equal (v1, v2, p=6)

Test if all elements of two lists of numbers are sufficiently equal.

function almost_equal (v1, v2, p=6)

Test if all numerical elements of two lists of values are sufficiently equal.

function compare (v1, v2, s=true)
 Order to lists of arbitrary values.

 10.16.1 Detailed Description
 List data type identification.
 Author
 Roy Allen Sutton

 Date
 2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

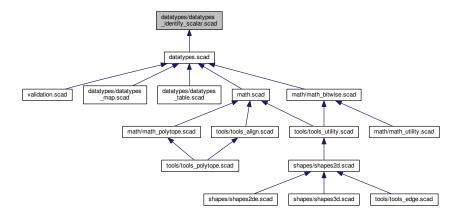
Note

Include this library file using the include statement.

10.17 datatypes/datatypes_identify_scalar.scad File Reference

Scalar data type identification.

This graph shows which files directly or indirectly include this file:



Functions

• function is_defined (v)

Test if a value is defined.

function not_defined (v)

Test if a value is not defined.

function is_nan (v)

Test if a numerical value is invalid.

function is_inf (v)

Test if a numerical value is infinite.

function is_scalar (v)

Test if a value is a single non-iterable value.

• function is_iterable (v)

Test if a value has multiple parts and is iterable.

• function is_empty (v)

Test if an iterable value is empty.

• function is_number (v)

Test if a value is a number.

• function is_integer (v)

Test if a value is an integer.

function is_decimal (v)

Test if a value is a decimal.

function is_boolean (v)

Test if a value is a predefined boolean constant.

• function is_string (v)

Test if a value is a string.

• function is list (v)

Test if a value is an iterable list of values.

function is range (v)

Test if a value is a range definition.

• function is even (v)

Test if a numerical value is even.

• function is_odd (v)

Test if a numerical value is odd.

• function is between (v, l, u)

Test if a numerical value is between an upper and lower bounds.

10.17.1 Detailed Description

Scalar data type identification.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

Note

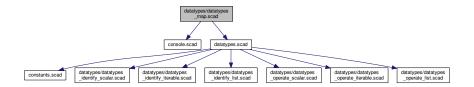
Include this library file using the include statement.

10.18 datatypes/datatypes_map.scad File Reference

Map data type operations.

```
#include <console.scad>
#include <datatypes.scad>
```

Include dependency graph for datatypes_map.scad:



Functions

function get_map_i (m, k)

Return the index of a map key.

• function map_exists (m, k)

Test if a key exists.

• function get_map_v (m, k)

Get the map value associated with a key.

function get_map_kl (m)

Get a list of all map keys.

• function get_map_vl (m)

Get a list of all map values.

• function get_map_size (m)

Get the number of map entries.

• module map_check (m, verbose=false)

Perform some basic validation/checks on a map.

module map_dump (m, sort=true, number=true, p=3)

Dump each map entry to the console.

10.18.1 Detailed Description

Map data type operations.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

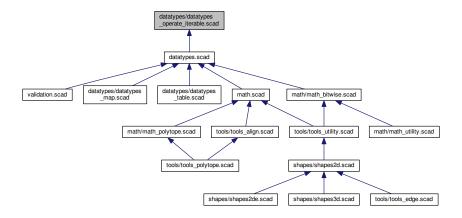
You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

Manage a collection of key-value pairs where keys are unique.

10.19 datatypes/datatypes_operate_iterable.scad File Reference

Iterable data type operations.

This graph shows which files directly or indirectly include this file:



Functions

• function edefined_or (v, i, d)

Return an iterable element when it exists or a default value when it does not.

• function find (mv, v, c=1, i, i1=0, i2)

Find the occurrences of a match value in an iterable value.

• function count (mv, v, s=true, i)

Count all occurrences of a match value in an iterable value.

function exists (mv, v, s=true, i)

Check for the existence of a match value in an iterable value.

function first (v)

Return the first element of an iterable value.

function second (v)

Return the second element of an iterable value.

• function third (v)

Return the third element of an iterable value.

function last (v)

Return the last element of an iterable value.

function nfirst (v, n=1)

Return a list containing the first n elements of an iterable value.

function nlast (v, n=1)

Return a list containing the last n elements of an iterable value.

function nhead (v, n=1)

Return a list containing all but the last n elements of an iterable value.

function ntail (v, n=1)

Return a list containing all but the first n elements of an iterable value.

function reverse (v)

Reverse the elements of an iterable value.

• function rselect (v, i)

Select a range of elements from an iterable value.

• function nssequence (v, n=1, s=1, w=false)

Return a list of all n-element sequential-subsets of an iterable value.

• function eappend (nv, v, r=true, j=true, l=true)

Append a value to each element of an iterable value.

• function insert (nv, v, i=0, mv, mi=0, s=true, si)

Insert a new value into an iterable value.

function delete (v, i, mv, mc=1, s=true, si)

Delete elements from an iterable value.

• function strip (v, mv=empty_lst)

Strip all matching values from an iterable value.

• function unique (v)

Return the unique elements of an iterable value.

10.19.1 Detailed Description

Iterable data type operations.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

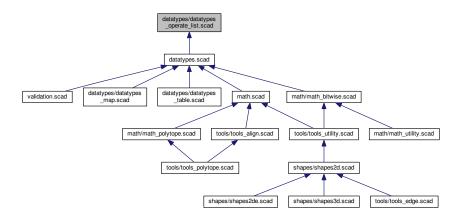
Note

Include this library file using the include statement.

10.20 datatypes/datatypes_operate_list.scad File Reference

List data type operations.

This graph shows which files directly or indirectly include this file:



Functions

• function lstr (v)

Convert a list of values to a concatenated string.

• function Istr_html (v, b, p, a, f, d=false)

Convert a list of values to a concatenated HTML-formatted string.

function consts (I, v, u=false)

Create a sequence of constant or incrementing elements.

function get index (I, s=true, rs)

Create a sequence for a list index sequence specification.

• function pad (I, w, p=0, r=true)

Pad a list to a constant width of elements.

• function dround (v, d=6)

Round all numerical values of a list to a fixed number of decimal point digits.

• function sround (v, d=6)

Round all numerical values of a list to a fixed number of significant figures.

• function limit (v, I, u)

Limit all numerical values of a list between an upper and lower bounds.

• function sum (v, i1, i2)

Compute the sum of a list of numbers.

• function mean (v)

Compute the mean/average of a list of numbers.

• function ciselect (v, i)

Case-like select a value from a list of ordered value options.

• function cmvselect (v, mv)

Case-like select a value from a list of mapped key-value options.

• function eselect (v, f=true, l=false, i)

Select a specified element from each iterable value of a list.

• function smerge (v, r=false)

Serial-merge lists of iterable values.

• function pmerge (v, j=true)

Parallel-merge lists of iterable values.

• function qsort (v, i, r=false)

Sort the numeric or string elements of a list using quick sort.

• function qsort2 (v, i, d=0, r=false, s=true)

Hierarchically sort an arbitrary data list using quick sort.

10.20.1 Detailed Description

List data type operations.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

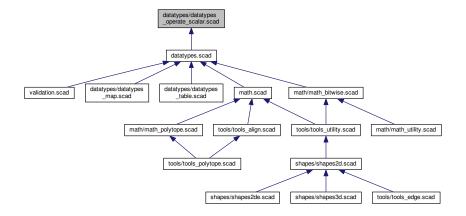
Note

Include this library file using the include statement.

10.21 datatypes/datatypes_operate_scalar.scad File Reference

Scalar data type operations.

This graph shows which files directly or indirectly include this file:



Functions

• function defined_or (v, d)

Return a value when it is defined or a default value when it is not.

• function circular_index (i, l, f=0)

Map an index position into a circularly indexed list.

10.21.1 Detailed Description

Scalar data type operations.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

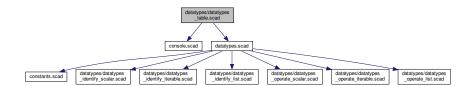
Note

Include this library file using the **include** statement.

10.22 datatypes/datatypes_table.scad File Reference

Table data type operations.

#include <console.scad>
#include <datatypes.scad>
Include dependency graph for datatypes_table.scad:



Functions

• function get table ri (r, ri)

Get the table row index that matches a table row identifier.

function get_table_r (r, ri)

Get the table row that matches a table row identifier.

function get table ci (c, ci)

Get the table column index that matches a table column identifier.

function get_table_c (c, ci)

Get the table column that matches a table column identifier.

• function get table v (r, c, ri, ci)

Get the table cell value for a specified row and column identifier.

function get_table_crl (r, c, ci)

Form a list of a select column across all table rows.

function get_table_ridl (r)

Form a list of all table row identifiers.

function get_table_cidl (c)

Form a list of all table column identifiers.

• function table_exists (r, c, ri, ci)

Test the existence of a table row and column identifier.

• function get_table_size (r, c)

Get the size of a table.

• function get_table_copy (r, c, rl, cl)

Create a new matrix from select rows and columns of a table.

function get table sum (r, c, rl, cl)

Sum select rows and columns of a table.

module table check (r, c, verbose=false)

Perform some basic validation/checks on a table.

module table_dump (r, c, rl, cl, number=true)

Dump a table to the console.

10.22.1 Detailed Description

Table data type operations.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

10.23 mainpage.scad File Reference

Documentation main page.

10.23.1 Detailed Description

Documentation main page.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

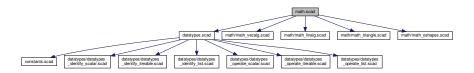
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

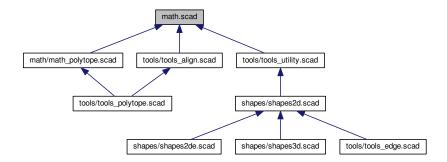
10.24 math.scad File Reference

Mathematical function primitives.

```
#include <datatypes.scad>
#include <math/math_vecalg.scad>
#include <math/math_linalg.scad>
#include <math/math_triangle.scad>
#include <math/math_oshapes.scad>
Include dependency graph for math.scad:
```



This graph shows which files directly or indirectly include this file:



10.24.1 Detailed Description

Mathematical function primitives.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

Note

Include this library file using the include statement.

10.25 math/math_bitwise.scad File Reference

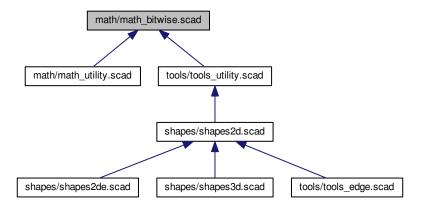
Mathematical base-two bitwise binary functions.

#include <datatypes.scad>

Include dependency graph for math_bitwise.scad:



This graph shows which files directly or indirectly include this file:



Functions

function bitwise_is_equal (v, b, t=1)

Test if a base-two bit position of an integer value equals a test bit.

function bitwise i2v (v, w=1, bv=1)

Encode an integer value as a base-two list of bits.

function bitwise_v2i (v)

Decode a base-two list of bits to an integer value.

function bitwise_i2s (v, w=1)

Encode an integer value as a base-two string of bits.

function bitwise_s2i (v)

Decode a base-two string of bits to an integer value.

function bitwise_imi (v, w, s)

Decode the integer in a value at a shifted base-two bit mask of width-w.

function bitwise_and (v1, v2, bv=1)

Base-two bitwise AND operation for integers.

function bitwise_or (v1, v2, bv=1)

Base-two bitwise OR operation for integers.

• function bitwise_xor (v1, v2, bv=1)

Base-two bitwise XOR operation for integers.

function bitwise not (v, w=1, bv=1)

Base-two bitwise NOT operation for an integer.

• function bitwise_lsh (v, s=1, bm=1, bv=1)

Base-two bitwise left-shift operation for an integer.

function bitwise rsh (v, s=1)

Base-two bitwise right-shift operation for an integer.

10.25.1 Detailed Description

Mathematical base-two bitwise binary functions.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the omdl ; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

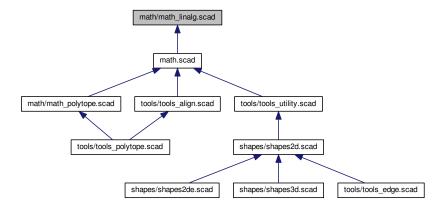
Note

Include this library file using the include statement.

10.26 math/math_linalg.scad File Reference

Linear algebra mathematical functions.

This graph shows which files directly or indirectly include this file:



Functions

• function multmatrix_lp (c, m)

Multiply all coordinates by a 4x4 3d-transformation matrix.

• function translate_lp (c, v)

Translate all coordinates dimensions.

• function rotate_lp (c, a, v, o=origin3d)

Rotate all coordinates about one or more axes in Euclidean 2d or 3d space.

• function scale_lp (c, v)

Scale all coordinates dimensions.

• function resize_lp (c, v)

Scale all coordinates dimensions proportionately to fit inside a region.

10.26.1 Detailed Description

Linear algebra mathematical functions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

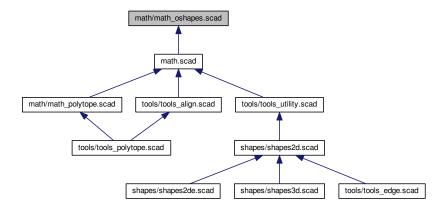
Note

Include this library file using the include statement.

10.27 math/math_oshapes.scad File Reference

Other shapes mathematical functions.

This graph shows which files directly or indirectly include this file:



Functions

function rpolygon_lp (n, r, a, vr, cw=true)

Compute the coordinates for an n-sided regular polygon.

function rpolygon_area (n, r, a)

Compute the area of an n-sided regular polygon.

function rpolygon_perimeter (n, r, a)

Compute the perimeter of an n-sided regular polygon.

10.27.1 Detailed Description

Other shapes mathematical functions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

Note

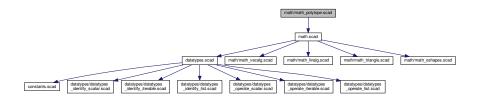
Include this library file using the **include** statement.

10.28 math/math_polytope.scad File Reference

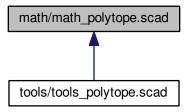
Polygon and polyhedron mathematical functions.

#include <math.scad>

Include dependency graph for math_polytope.scad:



This graph shows which files directly or indirectly include this file:



Functions

• function polytope_faces2edges (f)

List the edge coordinate index pairs of a polytope.

• function polytope_limits (c, f, a, d=[0:2], s=true)

Determine the bounding limits of a polytope.

• function polytope_bbox_pf (c, f, a)

Generate a bounding box polytope for another polytope in 3d or 2d.

• function polytope_line (c, f, e, i, l, r=false)

Get a line from an edge or any two vetices of a polytope.

function polytope_vertex_av (f, i)

List the adjacent vertices for a given polytope vertex.

function polytope_vertex_af (f, i)

List the adjacent face indexes for a polytope vertex.

• function polytope_edge_af (f, e, i)

List the adjacent face indexes for a polytope edge.

• function polytope_vertex_n (c, f, i)

Get a normal vector for a polytope vertex.

• function polytope edge n (c, f, e, i)

Get a normal vector for a polytope edge.

• function polytope_face_n (c, f, i, l, cw=true)

Get the normal vector of a polytope face.

• function polytope_face_m (c, f, i, l)

Get the mean coordinate of all vertices of a polytope face.

function polytope_face_mn (c, f, i, l, cw=true)

Get the mean coordinate and normal vector of a polytope face.

function polytope_plane (c, f, i, l, cw=true)

Get a plane for a polytope face.

function polytope_face_vcounts (f)

List the vertex counts for all polytope faces.

function polytope_face_angles (c, f)

List the angles between all adjacent faces of a polyhedron.

• function polytope_edge_lengths (c, e)

List the edge lengths of a polytope.

function polytope_edge_angles (c, f)

List the adjacent edge angles for each polytope vertex.

function polytope_faces_are_regular (c, f, e, d=6)

Test if the faces of a polytope are all regular.

function polytope_triangulate_ft (f)

Triangulate the faces of a convex polytope using fan triangulation.

• function polygon2d_perimeter (c, p)

Calculate the perimeter length of a polygon in 2d.

• function polygon2d_area (c, p, s=false)

Compute the signed area of a polygon in a Euclidean 2d-space.

• function polygon3d_area (c, p, n)

Compute the area of a polygon in a Euclidean 3d-space.

function polygon2d centroid (c, p)

Compute the center of mass of a polygon in a Euclidean 2d-space.

function polygon2d is cw (c, p)

Test the vertex ordering of a polygon in a Euclidean 2d-space.

function polygon2d_is_convex (c, p)

Test the convexity of a polygon in a Euclidean 2d-space.

• function polygon2d_winding (c, p, t)

Compute the winding number of a polygon about a point in a Euclidean 2d-space.

• function polygon2d_is_pip_wn (c, p, t)

Test if a point is inside a polygon in a Euclidean 2d-space using winding number.

function polygon2d_is_pip_as (c, p, t)

Test if a point is inside a polygon in a Euclidean 2d-space using angle summation.

• function polyhedron area (c, f)

Compute the surface area of a polyhedron in a Euclidean 3d-space.

• function polyhedron volume tf (c, f)

Compute the volume of a triangulated polyhedron in a Euclidean 3d-space.

function polyhedron_centroid_tf (c, f)

Compute the center of mass of a triangulated polyhedron in a Euclidean 3d-space.

function linear_extrude_pp2pf (c, p, h=1, centroid=false, center=false)

Convert a polygon to a polyhedron by adding a height dimension.

10.28.1 Detailed Description

Polygon and polyhedron mathematical functions.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

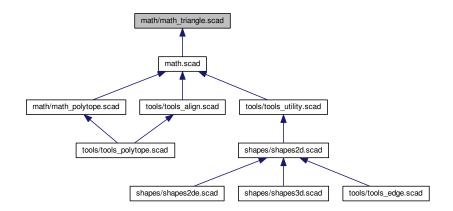
Note

Include this library file using the include statement.

10.29 math/math_triangle.scad File Reference

Triangle solutions mathematical functions.

This graph shows which files directly or indirectly include this file:



Functions

- function triangle_sss2lp (s1, s2, s3, cw=true)
 - Compute the vertex coordinates of a triangle given its side lengths.
- function triangle Is2lp (v, cw=true)
 - Compute the vertex coordinates of a triangle given its side lengths.
- function triangle_ppp2ls (v1, v2, v3)
 - Compute the side lengths of a triangle given its vertex coordinates.
- function triangle lp2ls (v)

Compute the side lengths of a triangle given its vertex coordinates.

• function triangle_area_ppp (v1, v2, v3, s=false)

Compute the signed area of a triangle given its vertex coordinates.

function triangle_area_lp (v, s=false)

Compute the signed area of a triangle given its vertex coordinates.

function triangle_centroid_ppp (v1, v2, v3)

Compute the centroid (geometric center) of a triangle.

function triangle centroid lp (v)

Compute the centroid (geometric center) of a triangle.

function triangle_incenter_ppp (v1, v2, v3)

Compute the coordinate for the triangle's incircle.

function triangle_incenter_lp (v)

Compute the coordinate for the triangle's incircle.

function triangle inradius ppp (v1, v2, v3)

Compute the inradius of a triangle's incircle.

function triangle_inradius_lp (v)

Compute the inradius of a triangle's incircle.

function triangle_is_cw_ppp (v1, v2, v3)

Test the vertex ordering, or orientation, of a triangle.

• function triangle is cw lp (v)

Test the vertex ordering, or orientation, of a triangle.

function triangle_is_pit_ppp (v1, v2, v3, t)

Test if a point is inside a triangle in a Euclidean 2d-space using Barycentric.

function triangle_is_pit_lp (v, t)

Test if a point is inside a triangle in a Euclidean 2d-space using Barycentric.

10.29.1 Detailed Description

Triangle solutions mathematical functions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omd1, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

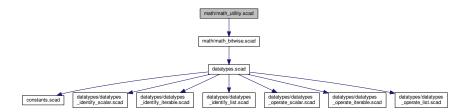
Note

Include this library file using the include statement.

10.30 math/math_utility.scad File Reference

Miscellaneous mathematical utilities.

#include <math/math_bitwise.scad>
Include dependency graph for math_utility.scad:



Functions

• function hist (v, m=0, cs, cb, cp, ca, cf, d=false)

Generate a histogram for the elements of a list of values.

10.30.1 Detailed Description

Miscellaneous mathematical utilities.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

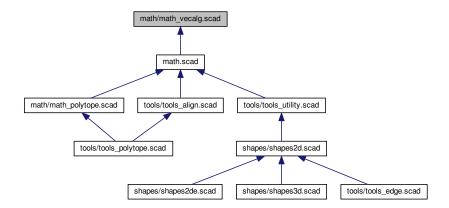
Note

Include this library file using the include statement.

10.31 math/math vecalg.scad File Reference

Vector algebra mathematical functions.

This graph shows which files directly or indirectly include this file:



Functions

function distance_pp (p1, p2)

Compute the distance between two Euclidean points.

• function is_left_ppp (p1, p2, p3)

Test if a point is left, on, or right of an infinite line in a Euclidean 2d-space.

function dimension_2to3_v (v)

Return 3d vector unchanged or add a zeroed third dimension to 2d vector.

function get_line_dim (I)

Return the number of dimensions of a Euclidean line (or vector).

function get_line_tp (l)

Return the terminal point of a Euclidean line (or vector).

function get_line_ip (l)

Return the initial point of a Euclidean line (or vector).

• function get_line2origin (I)

Shift a Euclidean line (or vector) to the origin.

function dot_II (I1, I2)

Compute the dot product of two lines (or vectors).

function cross_II (I1, I2)

Compute the cross product of two lines (or vectors) in a Euclidean 3d or 2d-space.

function striple_III (I1, I2, I3)

Compute the scalar triple product of three lines (or vectors) in a Euclidean 3d or 2d-space.

function angle II (I1, I2)

Compute the angle between two lines (or vectors) in a Euclidean 3d or 2d-space.

function angle_III (I1, I2, n)

Compute the angle between two lines (or vectors) in a Euclidean 3d-space.

function unit I (I)

Compute the normalized unit vector of a Euclidean line (or vector).

• function are_coplanar_III (I1, I2, I3, d=6)

Test if three lines (or vectors) are coplanar in Euclidean 3d-space.

function get pnorm2nv (pn, cw=true)

Convert a planes' normal specification into a normal vector.

10.31.1 Detailed Description

Vector algebra mathematical functions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

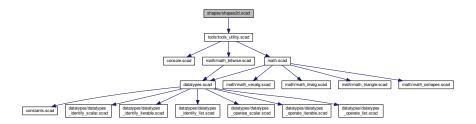
Note

Include this library file using the include statement.

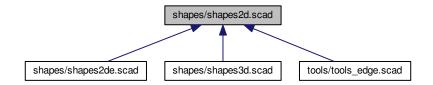
10.32 shapes/shapes2d.scad File Reference

Two-dimensional basic shapes.

#include <tools/tools_utility.scad>
Include dependency graph for shapes2d.scad:



This graph shows which files directly or indirectly include this file:



Functions

- module rectangle (size, vr, vrm=0, center=false)
 - A rectangle with edge, fillet, and/or chamfer corners.
- module rectangle_c (size, core, t, co, cr=0, vr, vr1, vr2, vrm=0, vrm1, vrm2, center=false)

A rectangle with a removed rectangular core.

• module rhombus (size, vr, center=false)

A rhombus.

• module triangle_ppp (v1, v2, v3, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by three vertices.

• module triangle_lp (v, vr, centroid=false, incenter=false)

A general triangle specified by a list of its three vertices.

• module triangle_sss (s1, s2, s3, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by its three side lengths.

• module triangle_ls (v, vr, centroid=false, incenter=false)

A general triangle specified by a list of its three side lengths.

• module triangle_ls_c (vs, vc, co, cr=0, vr, vr1, vr2, centroid=false, incenter=false)

A general triangle specified by its sides with a removed triangular core.

module triangle sas (s1, a, s2, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by two sides and the included angle.

module triangle_asa (a1, s, a2, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by a side and two adjacent angles.

• module triangle_aas (a1, a2, s, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by a side, one adjacent angle and the opposite angle.

• module triangle_ss (x, y, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A right-angled triangle specified by its opposite and adjacent side lengths.

• module triangle_sa (x, y, aa, oa, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A right-angled triangle specified by a side length and an angle.

• module ngon (n, r, vr)

An n-sided equiangular/equilateral regular polygon.

• module ellipse (size)

An ellipse.

module ellipse_c (size, core, t, co, cr=0)

An ellipse with a removed elliptical core.

module ellipse s (size, a1=0, a2=0)

An ellipse sector.

module ellipse cs (size, core, t, a1=0, a2=0, co, cr=0)

A sector of an ellipse with a removed elliptical core.

module star2d (size, n=5, vr)

A two-dimensional star.

10.32.1 Detailed Description

Two-dimensional basic shapes.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

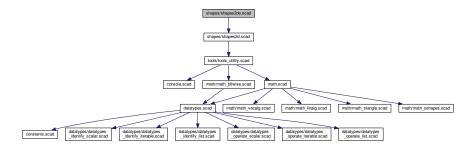
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

10.33 shapes/shapes2de.scad File Reference

Linearly extruded two-dimensional basic shapes.

#include <shapes/shapes2d.scad>
Include dependency graph for shapes2de.scad:



Functions

- module erectangle (size, h, vr, vrm=0, center=false)
 - An extruded rectangle with edge, fillet, and/or chamfer corners.
- module erectangle_c (size, core, h, t, co, cr=0, vr, vr1, vr2, vrm=0, vrm1, vrm2, center=false)
 - An extruded rectangle with a removed rectangular core.
- module erhombus (size, h, vr, center=false)
 - An extruded rhombus.
- module etriangle_ppp (v1, v2, v3, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by three vertices.
- module etriangle_lp (v, h, vr, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by a list of its three vertices.
- module etriangle_sss (s1, s2, s3, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by its three side lengths.
- module etriangle_ls (v, h, vr, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by a list of its three side lengths.
- module etriangle_ls_c (vs, vc, h, co, cr=0, vr, vr1, vr2, centroid=false, incenter=false, center=false)
 - A general triangle specified by its sides with a removed triangular core.
- module etriangle_sas (s1, a, s2, h, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by two sides and the included angle.
- module etriangle_asa (a1, s, a2, h, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by a side and two adjacent angles.
- module etriangle_aas (a1, a2, s, h, x=1, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by a side, one adjacent angle and the opposite angle.
- module etriangle ss (x, y, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded right-angled triangle specified by its opposite and adjacent side lengths.
- module etriangle_sa (x, y, aa, oa, h, vr, v1r, v2r, v3r, centroid=false, incenter=false, center=false)
 - An extruded right-angled triangle specified by a side length and an angle.
- module engon (n, r, h, vr, center=false)
 - An extruded n-sided equiangular/equilateral regular polygon.

• module eellipse (size, h, center=false)

An extruded ellipse.

module eellipse c (size, core, h, t, co, cr=0, center=false)

An extruded ellipse with a removed elliptical core.

module eellipse s (size, h, a1=0, a2=0, center=false)

An extruded ellipse sector.

• module eellipse_cs (size, core, h, t, a1=0, a2=0, co, cr=0, center=false)

An extruded sector of an ellipse with a removed elliptical core.

module estar2d (size, h, n=5, vr, center=false)

An extruded two-dimensional star.

10.33.1 Detailed Description

Linearly extruded two-dimensional basic shapes.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

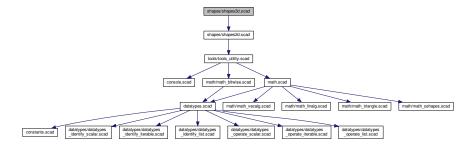
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

10.34 shapes/shapes3d.scad File Reference

Three-dimensional basic shapes.

#include <shapes/shapes2d.scad>
Include dependency graph for shapes3d.scad:



Functions

• module cone (r, h, d, vr, vr1, vr2)

A cone.

• module cuboid (size, vr, vrm=0, center=false)

A cuboid with edge, fillet, or chamfer corners.

• module ellipsoid (size)

An ellipsoid.

module ellipsoid_s (size, a1=0, a2=0)

A sector of an ellipsoid.

module pyramid_t (size, center=false)

A pyramid with trilateral base formed by four equilateral triangles.

module pyramid_q (size, center=false)

A pyramid with quadrilateral base.

• module star3d (size, n=5, half=false)

A three-dimensional star.

module torus_rp (size, core, r, l, t, co, cr=0, vr, vr1, vr2, vrm=0, vrm1, vrm2, pa=0, ra=360, m=255, center=false, profile=false)

A rectangular cross-sectional profile revolved about the z-axis.

• module torus_tp (size, core, r, l, co, cr=0, vr, vr1, vr2, pa=0, ra=360, m=255, centroid=false, incenter=false, profile=false,)

A triangular cross-sectional profile revolved about the z-axis.

module torus_ep (size, core, r, l, t, a1=0, a2=0, co, cr=0, pa=0, ra=360, m=255, profile=false)

An elliptical cross-sectional profile revolved about the z-axis.

10.34.1 Detailed Description

Three-dimensional basic shapes.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

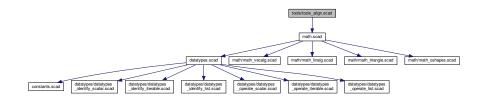
You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

Todo Complete rounded cylinder.

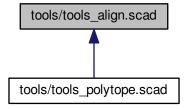
10.35 tools/tools_align.scad File Reference

Shape alignment tools.

#include <math.scad>
Include dependency graph for tools_align.scad:



This graph shows which files directly or indirectly include this file:



Functions

module orient_ll (l=z_axis3d_ul, rl=z_axis3d_ul, r=0)

Orient a line or vector to a reference line or vector.

module align II (I=z axis3d uI, rI=z axis3d uI, ap=0, rp=0, r=0, to=origin3d, ro=zero3d)

Align a line or vector to a reference line or vector.

module align I (rl=z axis3d ul, rp=0, r=0, to=origin3d, ro=zero3d, d=z axis ci)

Align a shapes' x, y, or z Cartesian axis to reference line or vector.

10.35.1 Detailed Description

Shape alignment tools.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

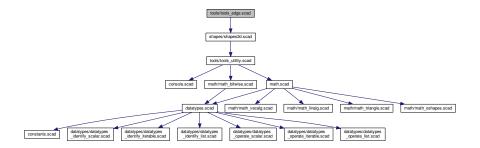
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

10.36 tools/tools_edge.scad File Reference

Shape edge finishing tools.

#include <shapes/shapes2d.scad> Include dependency graph for tools edge.scad:



Functions

module edge_profile_r (r, p=0, f=1, a=90,)

A 2d edge-finish profile specified by intersection radius.

• module edge_add_r (r, l=1, p=0, f=1, m=3, ba=45, a1=0, a2=90, center=false)

A 3d edge-finish additive shape specified by intersection radius.

10.36.1 Detailed Description

Shape edge finishing tools.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

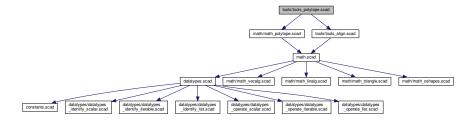
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

10.37 tools/tools_polytope.scad File Reference

Polygon and polyhedron tools.

#include <math/math_polytope.scad>
#include <tools/tools_align.scad>
Include dependency graph for tools_polytope.scad:



Functions

- module polytope_number (c, f, e, vi=true, fi=true, ei=true, sp=false, ts, th, to, tr=0)

 Label the vertices, paths, and edges of a polytope.
- module polytope_frame (c, f, e, vi=true, fi=true, ei=true, vc=1, fc=2, ec=0)

 Assemble a polytope skeletal frame using child objects.
- module polytope_bbox (c, f, a)

The 3d or 2d bounding box shape for a polytope.

10.37.1 Detailed Description

Polygon and polyhedron tools.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

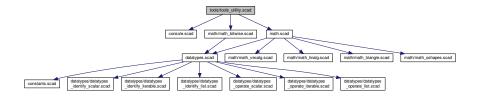
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

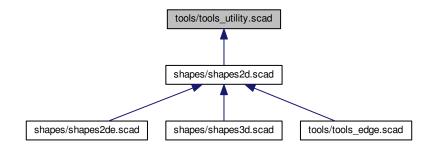
10.38 tools/tools_utility.scad File Reference

Shape transformation utility tools.

```
#include <console.scad>
#include <math.scad>
#include <math/math_bitwise.scad>
Include dependency graph for tools_utility.scad:
```



This graph shows which files directly or indirectly include this file:



Functions

• module rotate_extrude_tr (r, pa=0, ra=360, profile=false)

Translate, rotate, and revolve the 2d shape about the z-axis.

• module rotate_extrude_tre (r, l, pa=0, ra=360, m=255, profile=false)

Translate, rotate, and revolve the 2d shape about the z-axis with linear elongation.

• module linear_extrude_uls (h, center=false)

Linearly extrude 2d shape with extrusion upper and lower scaling.

• module radial_repeat (n, r=1, angle=true, move=false)

Distribute copies of a 2d or 3d shape equally about a z-axis radius.

• module grid_repeat (g, i, c=1, center=false)

Distribute copies of 2d or 3d shapes about Cartesian grid.

10.38.1 Detailed Description

Shape transformation utility tools.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

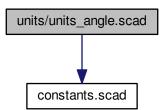
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

10.39 units/units_angle.scad File Reference

Angle units and conversions.

#include <constants.scad>
Include dependency graph for units_angle.scad:



Functions

function unit_angle_name (u=base_unit_angle)

Return the name of an angle unit identifier.

• function convert_angle (a, from=base_unit_angle, to=base_unit_angle)

Convert an angle from some units to another.

Variables

base unit angle = "d"

<string> The base units for angle measurements.

10.39.1 Detailed Description

Angle units and conversions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

Note

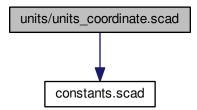
Include this library file using the include statement.

10.40 units/units_coordinate.scad File Reference

Coordinate systems and conversions.

#include <constants.scad>

Include dependency graph for units coordinate.scad:



Functions

function coordinates_name (s=base_coordinates)

Return the name of the given coordinate system identifier.

• function convert_coordinate (c, from=base_coordinates, to=base_coordinates)

Convert point from one coordinate system to another.

• function coordinates_cpc (c, r, t=false)

Radially scale a list of 2d cartesian coordinates.

function coordinates_pc (p, r, t=false)

Radially scale and convert a list of 2d polar coordinates to cartesian.

• function coordinates_csc (c, r, t=false)

Radially scale a list of 3d cartesian coordinates.

• function coordinates_sc (s, r, t=false)

Radially scale and convert a list of 3d spherical coordinates to cartesian.

Variables

• base_coordinates = "c"

<string> The base coordinate system.

coordinates_positive_angles = true

boolean> When converting to angular measures add 360 to negative angles.

10.40.1 Detailed Description

Coordinate systems and conversions.

Author

Roy Allen Sutton

Date

2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cong/licenses/.

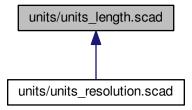
Note

Include this library file using the **include** statement.

10.41 units/units_length.scad File Reference

Length units and conversions.

This graph shows which files directly or indirectly include this file:



Functions

• function unit_length_name (u=base_unit_length, d=1, w=false)

Return the name for a length unit identifier with dimension.

function convert_length (v, from=base_unit_length, to=base_unit_length, d=1)

Convert a value from from one units to another with dimensions.

Variables

```
    base_unit_length = "mm"
    <string> The base unit for length measurements.
```

10.41.1 Detailed Description

Length units and conversions.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

Note

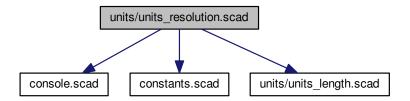
Include this library file using the include statement.

10.42 units/units_resolution.scad File Reference

An abstraction for arc rendering resolution control.

```
#include <console.scad>
#include <constants.scad>
#include <units/units_length.scad>
```

Include dependency graph for units_resolution.scad:



Functions

• function resolution_fn (r)

Return facets number for the given arc radius.

• function resolution_fs ()

Return minimum facets size.

• function resolution_fa (r)

Return the minimum facets angle.

• function resolution reduced ()

Return the radius at which arc resolution will begin to degrade.

• module resolution_info (r)

Output resolution information to the console for given radius.

• function resolution_facets (r)

Return facet count used to render a radius.

• function resolution_facetsv (r)

Return facet count information list used to render a radius.

Variables

• \$resolution mode = "fast"

<string> Global special variable that configures the arc resolution mode.

\$resolution_value = 0

< number> Global special variable for modes that use custom resolutions.

10.42.1 Detailed Description

An abstraction for arc rendering resolution control.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.egnu.org/licenses/.

Note

Include this library file using the include statement.

Test Review model for accuracy.

10.43 validation.scad File Reference

Function validation methods.

#include <datatypes.scad>
Include dependency graph for validation.scad:



Functions

function validate (d, cv, t, ev, p=4, pf=false)
 Compare a computed test value with an known good result.

10.43.1 Detailed Description

Function validation methods.

Author

Roy Allen Sutton

Date

2015-2017

Copyright

This file is part of omdl, an OpenSCAD mechanical design library.

The *omdl* is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

Index

2d Extrusions, 58	align_l
eellipse, 59	Alignment, 102
eellipse_c, 60	align_ll
eellipse_cs, 61	Alignment, 103
eellipse_s, 62	Alignment, 102
engon, 63	align_I, 102
erectangle, 63	align_II, 103
erectangle_c, 64	orient_II, 103
erhombus, 65	all defined
estar2d, 66	Iterables, 140
etriangle_aas, 67	all_equal
etriangle_asa, 68	Iterables, 140
etriangle_lp, 69	•
	all_len
etriangle_ls, 69	Iterables, 140
etriangle_ls_c, 70	all_lists
etriangle_ppp, 71	Iterables, 140
etriangle_sa, 72	all_numbers
etriangle_sas, 73	Iterables, 141
etriangle_ss, 74	all_scalars
etriangle_sss, 75	Iterables, 141
2d Shapes, 77	all_strings
ellipse, 78	Iterables, 141
ellipse_c, 78	almost_equal
ellipse cs, 79	Lists, 159
ellipse_s, 80	angle II
ngon, 80	Vector Algebra, 290
rectangle, 81	angle_III
rectangle_c, 82	Vector Algebra, 290
rhombus, 83	Angles, 104
star2d, 83	convert_angle, 105
triangle_aas, 84	unit_angle_name, 105
triangle_asa, 85	any_equal
triangle_lp, 86	Iterables, 142
triangle_ls, 86	any_undefined
triangle_ls_c, 86	Iterables, 142
triangle_ppp, 87	are_coplanar_III
triangle_sa, 88	Vector Algebra, 291
triangle_sas, 89	
triangle_ss, 90	Bitwise, 107
triangle_sss, 90	bitwise_and, 108
3d Shapes, 92	bitwise_i2s, 108
cone, 93	bitwise_i2v, 108
cuboid, 93	bitwise_imi, 108
ellipsoid, 94	bitwise_is_equal, 109
ellipsoid_s, 95	bitwise Ish, 109
pyramid_q, 95	bitwise not, 109
pyramid_t, 96	bitwise or, 109
star3d, 97	bitwise_rsh, 110
	bitwise s2i, 110
torus_ep, 97	- '
torus_rp, 98	bitwise_v2i, 110
torus_tp, 99	bitwise_xor, 110

bitwise_and	Coordinates, 118
Bitwise, 108	convert_coordinate, 120
bitwise_i2s	coordinates_cpc, 120
Bitwise, 108	coordinates_csc, 120
bitwise_i2v	coordinates_name, 121
Bitwise, 108	coordinates_pc, 121
bitwise_imi	coordinates_sc, 121
Bitwise, 108	coordinates_cpc
bitwise_is_equal	Coordinates, 120
Bitwise, 109	coordinates_csc
bitwise_lsh	Coordinates, 120
Bitwise, 109	coordinates_name
bitwise_not	Coordinates, 121
Bitwise, 109	coordinates_pc
bitwise_or	Coordinates, 121
Bitwise, 109	coordinates_sc
bitwise_rsh	Coordinates, 121
Bitwise, 110	count
bitwise_s2i	Iterables, 144
Bitwise, 110	cross_ll
bitwise_v2i	Vector Algebra, 291
Bitwise, 110	cuboid
bitwise_xor	3d Shapes, 93
Bitwise, 110	
	Database, 122
circular_index	database/geometry/polyhedra/anti_prisms.scad, 299
Scalars, 259	database/geometry/polyhedra/archimedean.scad, 300
ciselect	database/geometry/polyhedra/archimedean_duals.scad,
Lists, 164	301
cmvselect	database/geometry/polyhedra/cupolas.scad, 302
Lists, 164	database/geometry/polyhedra/dipyramids.scad, 303
compare	database/geometry/polyhedra/johnson.scad, 304
Lists, 160	database/geometry/polyhedra/platonic.scad, 305
Component, 112	database/geometry/polyhedra/polyhedra_all.scad, 306
cone	database/geometry/polyhedra/prisms.scad, 307
3d Shapes, 93	database/geometry/polyhedra/pyramids.scad, 308
Console, 113	database/geometry/polyhedra/trapezohedron.scad, 309
log_debug, 114	Datatypes, 123
log echo, 114	datatypes.scad, 310
log_error, 114	datatypes/datatypes_identify_iterable.scad, 311
log_info, 114	datatypes/datatypes_identify_list.scad, 313
log_warn, 116	datatypes/datatypes_identify_scalar.scad, 315
stack, 116	datatypes/datatypes_map.scad, 317
console.scad, 296	datatypes/datatypes_operate_iterable.scad, 318
Constants, 117	datatypes/datatypes_operate_list.scad, 320
constants.scad, 297	datatypes/datatypes_operate_scalar.scad, 322
consts	datatypes/datatypes_table.scad, 323
Lists, 165	defined_or
convert_angle	Scalars, 259
Angles, 105	delete
convert_coordinate	Iterables, 144
Coordinates, 120	dimension_2to3_v
convert_length	Vector Algebra, 291
Lengths, 155	distance_pp
•	→ ·

Vector Algebra, 292	Edge, 125
dot_ll	edge_add_r, 125
Vector Algebra, 292	edge_profile_r, 126
dround	edge_add_r
Lists, 165	Edge, 125
dtc_polyhedra_anti_prisms	edge_profile_r
Polyhedra, 218	Edge, 126
dtc_polyhedra_archimedean	eellipse
Polyhedra, 218	2d Extrusions, 59
dtc_polyhedra_archimedean_duals	eellipse_c
Polyhedra, 218	2d Extrusions, 60
dtc_polyhedra_cupolas	eellipse_cs
Polyhedra, 219	2d Extrusions, 61
dtc_polyhedra_dipyramids	eellipse_s
Polyhedra, 219	2d Extrusions, 62
dtc_polyhedra_johnson	Electrical, 128
Polyhedra, 219	ellipse
dtc_polyhedra_platonic	2d Shapes, 78
Polyhedra, 219	ellipse_c
dtc_polyhedra_polyhedra_all	2d Shapes, 78
Polyhedra, 219	ellipse_cs
dtc_polyhedra_prisms	2d Shapes, 79
Polyhedra, 219	ellipse_s
dtc_polyhedra_pyramids	2d Shapes, 80
Polyhedra, 219	ellipsoid
dtc_polyhedra_trapezohedron	3d Shapes, 94
Polyhedra, 219	ellipsoid_s
dtr_polyhedra_anti_prisms	3d Shapes, 95
Polyhedra, 219	engon
dtr_polyhedra_archimedean	2d Extrusions, 63
Polyhedra, 219	erectangle
dtr_polyhedra_archimedean_duals Polyhedra, 220	2d Extrusions, 63
•	erectangle_c
dtr_polyhedra_cupolas	2d Extrusions, 64
Polyhedra, 220	erhombus
dtr_polyhedra_dipyramids Polyhedra, 220	2d Extrusions, 65 eselect
•	
dtr_polyhedra_johnson Polyhedra, 220	Lists, 165 estar2d
dtr_polyhedra_platonic	2d Extrusions, 66
Polyhedra, 220	etriangle aas
dtr_polyhedra_polyhedra_all	2d Extrusions, 67
Polyhedra_polyhedra_alli Polyhedra, 220	etriangle_asa
dtr_polyhedra_prisms	2d Extrusions, 68
Polyhedra, 220	etriangle_lp
dtr_polyhedra_pyramids	2d Extrusions, 69
Polyhedra, 220	etriangle_ls
dtr_polyhedra_trapezohedron	2d Extrusions, 69
Polyhedra, 220	etriangle_ls_c
. Signodia, LES	2d Extrusions, 70
eappend	etriangle_ppp
Iterables, 145	2d Extrusions, 71
edefined or	etriangle_sa
Iterables, 145	2d Extrusions, 72
	24 274 4010110, 72

etriangle_sas	get_table_ri
2d Extrusions, 73	Tables, 270
etriangle_ss	get_table_ridl
2d Extrusions, 74	Tables, 271
etriangle_sss	get_table_size
2d Extrusions, 75	Tables, 271
Euclidean, 129	get_table_sum
exists	Tables, 271
Iterables, 146	get_table_v
Extrude, 131	Tables, 271
linear_extrude_uls, 131	grid_repeat
rotate_extrude_tr, 132	Repeat, 243
rotate_extrude_tre, 133	hist
find	
find	Utilities, 284
Iterables, 146	Identification, 137
first	insert
Iterables, 147	Iterables, 147
General, 135	is between
Geometry, 136	Scalars, 252
get index	is boolean
Lists, 165	Scalars, 252
get_line2origin	is decimal
Vector Algebra, 292	Scalars, 252
get_line_dim	is_defined
Vector Algebra, 292	Scalars, 253
get_line_ip	is empty
Vector Algebra, 294	Scalars, 253
get_line_tp	is even
Vector Algebra, 294	Scalars, 253
get map i	is inf
Maps, 174	Scalars, 253
get_map_kl	is integer
Maps, 174	Scalars, 253
get_map_size	is_iterable
Maps, 175	Scalars, 255
get_map_v	is_left_ppp
Maps, 175	Vector Algebra, 294
get_map_vl	is_list
Maps, 175	Scalars, 255
get_pnorm2nv	is_nan
Vector Algebra, 294	Scalars, 255
get_table_c	is number
Tables, 268	Scalars, 255
get_table_ci	is_odd
Tables, 268	Scalars, 256
get_table_cidl	is_range
Tables, 268	Scalars, 256
get_table_copy	is scalar
Tables, 270	Scalars, 256
get_table_crl	is_string
Tables, 270	Scalars, 256
get_table_r	Iterables, 139, 143
Tables, 270	all defined, 140
iables, 270	an_ueiiiieu, 140

all_equal, 140	get_index, 165
all_len, 140	limit, 167
all_lists, 140	lstr, 167
all_numbers, 141	lstr_html, 167
all_scalars, 141	mean, 168
all_strings, 141	n_almost_equal, 160
any_equal, 142	pad, 168
any_undefined, 142	pmerge, 169
count, 144	qsort, 169
delete, 144	qsort2, 170
eappend, 145	smerge, 170
edefined_or, 145	sround, 170
exists, 146	sum, 172
find, 146	log_debug
first, 147	Console, 114
insert, 147	log_echo
last, 148	Console, 114
nfirst, 148	log_error
nhead, 148	Console, 114
nlast, 149	log_info
nssequence, 149	Console, 114
ntail, 149	· · · · · · · · · · · · · · · · · · ·
	log_warn
reverse, 149	Console, 116
rselect, 151	Istr
second, 151	Lists, 167
strip, 151	lstr_html
third, 151	Lists, 167
unique, 152	mainness and OOF
la at	mainpage.scad, 325
last	map_check
Iterables, 148	Maps, 175
Lengths, 153	map_dump
convert_length, 155	Maps, 176
unit_length_name, 155	map_exists
limit	Maps, 176
Lists, 167	Maps, 173
Linear Algebra, 156	get_map_i, 174
multmatrix_lp, 156	get_map_kl, 174
resize_lp, 157	get_map_size, 175
rotate_lp, 157	get_map_v, 175
scale_lp, 157	get_map_vl, 175
translate_lp, 157	map_check, 175
linear_extrude_pp2pf	map_dump, 176
Polytopes, 228	map_exists, 176
linear_extrude_uls	Material, 177
Extrude, 131	Math, 178
Lists, 159, 163	math.scad, 325
almost_equal, 159	math/math_bitwise.scad, 326
ciselect, 164	math/math_linalg.scad, 328
cmvselect, 164	math/math_oshapes.scad, 330
compare, 160	math/math_polytope.scad, 331
consts, 165	math/math_triangle.scad, 334
dround, 165	math/math_utility.scad, 336
eselect, 165	math/math_vecalg.scad, 337
	-

mean	dtc_polyhedra_archimedean, 218
Lists, 168	dtc_polyhedra_archimedean_duals, 218
multmatrix_lp	dtc_polyhedra_cupolas, 219
Linear Algebra, 156	dtc_polyhedra_dipyramids, 219
	dtc_polyhedra_johnson, 219
n_almost_equal	dtc_polyhedra_platonic, 219
Lists, 160	dtc_polyhedra_polyhedra_all, 219
nfirst	dtc_polyhedra_prisms, 219
Iterables, 148	dtc_polyhedra_pyramids, 219
ngon	dtc_polyhedra_trapezohedron, 219
2d Shapes, 80	dtr_polyhedra_anti_prisms, 219
nhead	dtr_polyhedra_archimedean, 219
Iterables, 148	dtr_polyhedra_archimedean_duals, 220
nlast	dtr_polyhedra_cupolas, 220
Iterables, 149	dtr_polyhedra_dipyramids, 220
not_defined	dtr_polyhedra_johnson, 220
Scalars, 258	dtr_polyhedra_platonic, 220
nssequence	dtr_polyhedra_polyhedra_all, 220
Iterables, 149	dtr_polyhedra_prisms, 220
ntail	dtr polyhedra pyramids, 220
Iterables, 149	dtr polyhedra trapezohedron, 220
	polyhedron_area
Operations, 180	Polytopes, 233
orient_ll	polyhedron_centroid_tf
Alignment, 103	Polytopes, 233
Other Shapes, 182	polyhedron_volume_tf
rpolygon_area, 182	Polytopes, 233
rpolygon_lp, 182	
rpolygon_perimeter, 184	Polytope, 224
	polytope_bbox, 224
pad	polytope_frame, 225
Lists, 168	polytope_number, 225
Parts, 185	polytope_bbox
pmerge	Polytope, 224
Lists, 169	polytope_bbox_pf
polygon2d_area	Polytopes, 235
Polytopes, 229	polytope_edge_af
polygon2d_centroid	Polytopes, 235
Polytopes, 229	polytope_edge_angles
polygon2d_is_convex	Polytopes, 236
Polytopes, 230	polytope_edge_lengths
polygon2d_is_cw	Polytopes, 236
Polytopes, 230	polytope_edge_n
polygon2d_is_pip_as	Polytopes, 236
Polytopes, 230	polytope_face_angles
polygon2d_is_pip_wn	Polytopes, 237
Polytopes, 231	polytope_face_m
polygon2d_perimeter	Polytopes, 237
Polytopes, 231	polytope_face_mn
polygon2d_winding	Polytopes, 237
Polytopes, 232	polytope_face_n
polygon3d_area	Polytopes, 238
Polytopes, 232	polytope_face_vcounts
Polyhedra, 186	Polytopes, 238
dtc_polyhedra_anti_prisms, 218	polytope_faces2edges
 	· · · -

Polytopes, 239	pyramid_q
polytope_faces_are_regular	3d Shapes, 95
Polytopes, 239	pyramid_t
polytope_frame	3d Shapes, 96
Polytope, 225	geort
polytope_limits	qsort Lists, 169
Polytopes, 239	qsort2
polytope_line	Lists, 170
Polytopes, 240	Lists, 170
polytope_number	radial_repeat
Polytope, 225	Repeat, 244
polytope_plane	rectangle
Polytopes, 241	2d Shapes, 81
polytope_triangulate_ft	rectangle_c
Polytopes, 241	2d Shapes, 82
polytope_vertex_af	Repeat, 243
Polytopes, 241	grid_repeat, 243
polytope_vertex_av Polytopes, 242	radial_repeat, 244
	resize_lp
polytope_vertex_n Polytopes, 242	Linear Algebra, 157
Polytopes, 227	resolution_fa
linear_extrude_pp2pf, 228	Resolutions, 247
polygon2d_area, 229	resolution_facets
polygon2d_centroid, 229	Resolutions, 248
polygon2d_is_convex, 230	resolution_facetsv
polygon2d_is_cw, 230	Resolutions, 248
polygon2d_is_pip_as, 230	resolution_fn
polygon2d_is_pip_wn, 231	Resolutions, 248
polygon2d_perimeter, 231	resolution_fs
polygon2d_winding, 232	Resolutions, 249
polygon3d_area, 232	resolution_info
polyhedron_area, 233	Resolutions, 250 resolution reduced
polyhedron_centroid_tf, 233	Resolutions, 250
polyhedron volume tf, 233	Resolutions, 246
polytope_bbox_pf, 235	resolution fa, 247
polytope_edge_af, 235	resolution_facets, 248
polytope edge angles, 236	resolution facetsv, 248
polytope_edge_lengths, 236	resolution_fn, 248
polytope_edge_n, 236	resolution_fs, 249
polytope_face_angles, 237	resolution info, 250
polytope_face_m, 237	resolution_reduced, 250
polytope face mn, 237	reverse
polytope_face_n, 238	Iterables, 149
polytope_face_vcounts, 238	rhombus
polytope_faces2edges, 239	2d Shapes, 83
polytope_faces_are_regular, 239	rotate_extrude_tr
polytope_limits, 239	Extrude, 132
polytope_line, 240	rotate_extrude_tre
polytope_plane, 241	Extrude, 133
polytope_triangulate_ft, 241	rotate_lp
polytope_vertex_af, 241	Linear Algebra, 157
polytope_vertex_av, 242	rpolygon_area
polytope_vertex_n, 242	Other Shapes, 182

rpolygon_lp	table_dump
Other Shapes, 182	Tables, 273
rpolygon_perimeter	table_exists
Other Shapes, 184	Tables, 273
rselect	Tables, 266
Iterables, 151	get_table_c, 268
	get_table_ci, 268
Scalars, 251, 259	get_table_cidl, 268
circular_index, 259	get_table_copy, 270
defined_or, 259	get_table_crl, 270
is_between, 252	get_table_r, 270
is_boolean, 252	get_table_ri, 270
is_decimal, 252	get_table_ridl, 271
is_defined, 253	- — —
is_empty, 253	get_table_size, 271
is even, 253	get_table_sum, 271
is inf, 253	get_table_v, 271
is_integer, 253	table_check, 273
is_iterable, 255	table_dump, 273
is_list, 255	table_exists, 273
is_nan, 255	third
is_number, 255	Iterables, 151
is odd, 256	Tools, 274
is_range, 256	tools/tools_align.scad, 344
is_scalar, 256	tools/tools_edge.scad, 345
	tools/tools_polytope.scad, 347
is_string, 256	tools/tools_utility.scad, 348
not_defined, 258	torus_ep
scale_lp	3d Shapes, 97
Linear Algebra, 157	torus_rp
second	3d Shapes, 98
Iterables, 151	torus tp
Shapes, 261	3d Shapes, 99
shapes/shapes2d.scad, 339	translate_lp
shapes/shapes2de.scad, 341	Linear Algebra, 157
shapes/shapes3d.scad, 342	triangle_aas
smerge	2d Shapes, 84
Lists, 170	triangle_area_lp
sround	Triangles, 277
Lists, 170	triangle area ppp
stack	Triangles, 277
Console, 116	-
star2d	triangle_asa
2d Shapes, 83	2d Shapes, 85
star3d	triangle_centroid_lp
3d Shapes, 97	Triangles, 277
strip	triangle_centroid_ppp
Iterables, 151	Triangles, 278
striple_III	triangle_incenter_lp
Vector Algebra, 295	Triangles, 278
sum	triangle_incenter_ppp
Lists, 172	Triangles, 278
System, 265	triangle_inradius_lp
	Triangles, 278
table_check	triangle_inradius_ppp
Tables, 273	Triangles, 279

triangle_is_cw_lp Triangles, 279	unit_I Vector Algebra, 295
triangle_is_cw_ppp Triangles, 279	unit_length_name Lengths, 155
triangle_is_pit_lp	Units, 282
Triangles, 279	units/units angle.scad, 349
triangle_is_pit_ppp	units/units_coordinate.scad, 350
Triangles, 280	units/units_length.scad, 352
triangle_lp	units/units resolution.scad, 353
2d Shapes, 86	Utilities, 284, 286
triangle_lp2ls	hist, 284
Triangles, 280	
triangle_ls	validate
2d Shapes, 86	Validation, 288
triangle_ls2lp	Validation, 287
Triangles, 280	validate, 288
triangle_ls_c	validation.scad, 355
2d Shapes, 86	Vector Algebra, 289
triangle_ppp	angle_II, 290
2d Shapes, 87	angle_III, 290
triangle_ppp2ls	are_coplanar_lll, 291
Triangles, 281	cross_II, 291
triangle_sa	dimension_2to3_v, 291
2d Shapes, 88	distance_pp, 292
triangle_sas	dot_II, 292 get_line2origin, 292
2d Shapes, 89	get_line_dim, 292
triangle_ss	get_line_ip, 294
2d Shapes, 90	get_line_tp, 294
triangle_sss	get_pnorm2nv, 294
2d Shapes, 90	is_left_ppp, 294
triangle_sss2lp	striple_III, 295
Triangles, 281	unit_l, 295
Triangles, 276	<u>.</u> ., 200
triangle_area_lp, 277	
triangle_area_ppp, 277	
triangle_centroid_lp, 277	
triangle_centroid_ppp, 278	
triangle_incenter_lp, 278	
triangle_incenter_ppp, 278	
triangle_inradius_lp, 278	
triangle_inradius_ppp, 279	
triangle_is_cw_lp, 279 triangle_is_cw_ppp, 279	
triangle_is_cw_ppp, 279 triangle_is_pit_lp, 279	
triangle_is_pit_ppp, 280	
triangle_ls_pit_ppp, 200 triangle_lp2ls, 280	
triangle_lp2is, 200 triangle_ls2lp, 280	
triangle_ppp2ls, 281	
triangle_sss2lp, 281	
ag.o0002.p, =01	
unique	
Iterables, 152	
unit_angle_name	
Angles, 105	