omdl

v0.5

Generated by Doxygen 1.8.9.1

Sat Feb 4 2017 00:28:43

ii CONTENTS

Contents

1	omd	ıdl		1
	1.1	What is omdl?	 	2
	1.2	Quick Start	 	2
	1.3	Using omdl	 	2
		1.3.1 Recommended install method	 	3
	1.4	Contributing	 	3
	1.5	Contact and Support	 	4
2	Valid	idation		4
	2.1	Primitives Validation	 	4
		2.1.1 Variable Tests	 	4
		2.1.2 Vector Operations	 	18
	2.2	Computations Validation	 	29
		2.2.1 Point, Vector and Plane	 	29
		2.2.2 Bitwise Operations	 	33
3	Todo	do List		39
4	Mod	dule Index		39
	4.1	Modules	 	39
5	File	e Index		40
	5.1	File List	 	40
6	Mod	dule Documentation		41
	6.1	2D Extrusions	 	42
		6.1.1 Detailed Description	 	43
		6.1.2 Function Documentation	 	43
	6.2	2D Shapes	 	60
		6.2.1 Detailed Description	 	61
		6.2.2 Function Documentation	 	61
	6.3	3D Shapes	 	76
		6.3.1 Detailed Description	 	76
		6.3.2 Function Documentation	 	77
	6.4	Angle	 	85
		6.4.1 Detailed Description	 	85
		6.4.2 Function Documentation	 	86

6.5	Bitwise Operations	88
	6.5.1 Detailed Description	89
	6.5.2 Function Documentation	89
6.6	Console	92
	6.6.1 Detailed Description	92
	6.6.2 Function Documentation	93
6.7	Constants	95
	6.7.1 Detailed Description	95
6.8	Data	96
	6.8.1 Detailed Description	96
6.9	Datatable	97
	6.9.1 Detailed Description	98
	6.9.2 Function Documentation	99
6.10	Edge Finishing	05
	6.10.1 Detailed Description	05
	6.10.2 Function Documentation	05
6.11	Euclidean	80
	6.11.1 Detailed Description	09
6.12	Extrusions	10
	6.12.1 Detailed Description	10
	6.12.2 Function Documentation	10
6.13	General	14
	6.13.1 Detailed Description	14
6.14	Length	15
	6.14.1 Detailed Description	15
	6.14.2 Function Documentation	17
6.15	Map	18
	6.15.1 Detailed Description	18
	6.15.2 Function Documentation	19
6.16	Math	22
	6.16.1 Detailed Description	23
6.17	Parts	24
6.18	Point, Vector and Plane	25
	6.18.1 Detailed Description	25
	6.18.2 Function Documentation	25
6.19	Replications	30
	6.19.1 Detailed Description	30

iv CONTENTS

	6.19.2 Function Documentation
6.20	Resolution
	6.20.1 Detailed Description
	6.20.2 Function Documentation
6.21	Shapes
	6.21.1 Detailed Description
6.22	System
	6.22.1 Detailed Description
6.23	Tools
	6.23.1 Detailed Description
6.24	Transformations
	6.24.1 Detailed Description
6.25	Triangle Solutions
	6.25.1 Detailed Description
	6.25.2 Function Documentation
6.26	Units
	6.26.1 Detailed Description
6.27	Utilities
	6.27.1 Detailed Description
	6.27.2 Function Documentation
6.28	Variable Tests
	6.28.1 Detailed Description
	6.28.2 Function Documentation
6.29	Vector Operations
	6.29.1 Detailed Description
	6.29.2 Function Documentation
6.30	n-gon Solutions
	6.30.1 Detailed Description
	6.30.2 Function Documentation
Eilo	Documentation 180
7.1	console.scad File Reference
7.1	7.1.1 Detailed Description
7.2	constants.scad File Reference
1.4	7.2.1 Detailed Description
7.3	mainpage.scad File Reference
1.3	
	7.3.1 Detailed Description

7

1 omdl

7.4	map.scad File Reference
	7.4.1 Detailed Description
7.5	math.scad File Reference
	7.5.1 Detailed Description
7.6	math_bitwise.scad File Reference
	7.6.1 Detailed Description
7.7	primitives.scad File Reference
	7.7.1 Detailed Description
7.8	resolution.scad File Reference
	7.8.1 Detailed Description
7.9	shapes2d.scad File Reference
	7.9.1 Detailed Description
7.10	shapes2de.scad File Reference
	7.10.1 Detailed Description
7.11	shapes3d.scad File Reference
	7.11.1 Detailed Description
7.12	table.scad File Reference
	7.12.1 Detailed Description
7.13	tool_edge.scad File Reference
	7.13.1 Detailed Description
7.14	transform.scad File Reference
	7.14.1 Detailed Description
7.15	units_angle.scad File Reference
	7.15.1 Detailed Description
7.16	units_length.scad File Reference
	7.16.1 Detailed Description
7.17	utilities.scad File Reference
	7.17.1 Detailed Description
7.18	validation.scad File Reference
	7.18.1 Detailed Description

215

1 omdl

Index

A collection of documented design primitives for ${\tt OpenSCAD}$ inspired by ${\tt MCAD}$.

1.1 What is omdl?

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

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 script documentation, compilable to multiple common output formats.

With omd1, all library primitives are *parametric* with minimal, mostly zero, global variable dependencies and all library API's include <code>Doxygen markups</code> that describe its parameters, behavior, and use. Validation scripts are used to check that the core building blocks work as expected across evolving <code>OpenSCAD</code> versions. These validation are performed automatically when rebuilding the documentation.

See the validation section for the results generated with this documentation.

1.2 Quick Start

The library components may be used or included as expected.

Example:

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

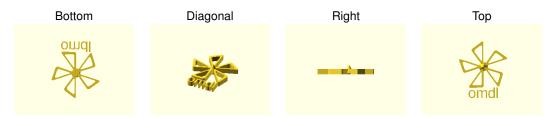
$fn = 36;

frame = triangle_vp2vl( [ [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])
st_radial_copy( n=5, angle=true )
   etriangle_vl_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



1.3 Using omdl

To use omd1 the library files must be copied to one of the OpenSCAD Library Locations. This can be done manually, as described in the OpenSCAD documentation, or can be automated using openscad—amu.

The ladder is recommended and has several advantages. When using openscad-amu, the library reference documentation is installed together with the library source code. This reference documentation is also added to a browsable

1.4 Contributing 3

data-table of installed libraries, which facilitates design reference searches. Moreover, with openscad-amu installed, one can rebuild the omdl reference manual or develop documentation for new OpenSCAD designs.

Library release files are made available in the source repository in a sub-directory called snapshots.

1.3.1 Recommended install method

More information on installing openscad—amu can be found published on Thingiverse and in the GitHib openscad—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 build version, then the install most likely completed successfully and the temporary directory created in the first step may be removed when desired.

Now the documentation for omd1 can be compiled and installed with a single command. First download the source from Thingiverse or clone the source 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 reference manual that can be views with a web browser.

Have a look in:

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

Now you may include the desired library primitives in your project as follows, replacing the version number as needed (multiple version of a library may coexists):

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

1.4 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.

1.5 Contact and 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 Validation

- · Primitives Validation
- · Computations Validation

2.1 Primitives Validation

- · Variable Tests
- Vector Operations

2.1.1 Variable Tests

- Validation Script (group1)
- Validation Results (group1)
- Validation Script (group2)
- · Validation Results (group2)

2.1.1.1 Validation Script (group1)

```
include <primitives.scad>;
    use <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", 1],
["t03", "An small even integer", 10],
["t04", "A large integer", 100000
["t05", "A small decimal (epsilon)", eps],
["t06", "The max number", number
["t07", "The min number", number
["t08", "The max number^2", number
                                                                            undef],
                                                                             100000000],
                                                                            number max],
                                                                            number_min],
                                                                           number_max *
number_max],

["t09", "The invalid number nan",

["t10", "The boolean true",

["t11", "The boolean false",

["t12", "A character string",

["t13", "A string",
                                                                            0 / 0],
                                                                            truel,
                                                                            false],
                                                                              "a"],
                                                                             "This is a longer string"],
```

```
["t14", "The empty string",
                                           empty_str],
    ["t15", "The empty vector",
                                           empty vl.
    ["t16", "A 1-tuple vector of undef",
                                           [undef]],
    ["t17", "A 1-tuple vector",
                                           [10]],
                                           [1, 2, 3]],
    ["t18", "A 3-tuple vector",
    ["t19", "A vector of vectors",
                                           [[1,2,3], [4,5,6], [7,8,9]]],
    ["t20", "A shorthand range", ["t21", "A range",
                                           [0:911,
                                           [0:0.5:9]]
  test_ids = table_get_row_ids( 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 =
                   01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21
  [ // function
     ["is_empty",
["is_scalar",
                    f, t, t, t, f, f, f, f, f],
                    t, f, f, f, f, f, f, s, s],
    ["is_iterable", f, t, t, t, t, t, t, t, t, s, s],
    ["is_string",
                    f, f, f, f, f, f, f, f, f, t, t, t, f, f, f, f, f, f, f],
    ["is vector",
                    f, t, t, t, t, t, s, s],
    ["is_boolean",
                    f, f, f, f, f, f, f, f, t, t, f, f],
    ["is_number",
                    ["is_range",
                   ["is_nan",
                   f, f, f, f, f, f, f, t, f, f],
                 ["is_inf",
    ["is_even",
                   ["is_odd",
                   // 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 ) = table_get(test_r, test_c, vid, "tv");
  module run_test( fname, fresult, vid )
    value_text = table_get(test_r, test_c, vid, "td");
    pass_value = table_get(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_empty", is_empty(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_string", is_string(get_value(vid)), vid );
for (vid=test_ids) run_test( "is_vector", is_vector(get_value(vid)), vid );
for (vid=test_ids) run_test( "is_boolean", is_boolean(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_integer", is_integer(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_number", is_number(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_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_even", is_even(get_value(vid)), vid );
for (vid=test_ids) run_test( "is_odd", is_odd(get_value(vid)), vid );
// end-of-tests
```

2.1.1.2 Validation Results (group1)

```
1 ECHO: "OpenSCAD Version [2016, 12, 21]"
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'"
6 ECHO: "[ INFO ] run_test(); t05 passed: 'is_defined(0.01)=true'"
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'
14 ECHO: "[ INFO ] run_test(); t13 passed: 'is_defined(This is a longer string)=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(0.01)=false''
28 ECHO: "[ INFO ] run_test(); t06 passed: 'not_defined(1e+308)=false'"
29 ECHO: "[ INFO ] run_test(); t07 passed: 'not_defined(-1e+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(); tll 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_empty(undef)=false'"
45 ECHO: "[ INFO ] run_test(); t02 passed: 'is_empty(1)=false'"
46 ECHO: "[ INFO ] run_test(); t03 passed: 'is_empty(10)=false'"
47 ECHO: "[ INFO ] run_test(); t04 passed: 'is_empty(1e+08)=false'"
48 ECHO: "[ INFO ] run_test(); t05 passed: 'is_empty(0.01)=false'"
49 ECHO: "[ INFO ] run_test(); t06 passed: 'is_empty(1e+308)=false''
50 ECHO: "[ INFO ] run_test(); t07 passed: 'is_empty(-1e+308)=false'"
51 ECHO: "[ INFO ] run_test(); t08 passed: 'is_empty(inf)=false'"
52 ECHO: "[ INFO ] run_test(); t09 passed: 'is_empty(nan)=false'"
53 ECHO: "[ INFO ] run_test(); t10 passed: 'is_empty(true)=false'"
54 ECHO: "[ INFO ] run_test(); tl1 passed: 'is_empty(false)=false'"
55 ECHO: "[ INFO ] run_test(); t12 passed: 'is_empty(a)=false'"
56 ECHO: "[ INFO ] run_test(); t13 passed: 'is_empty(This is a longer string)=false'"
57 ECHO: "[ INFO ] run_test(); t14 passed: 'is_empty()=true'"
58 ECHO: "[ INFO ] run_test(); t15 passed: 'is_empty([])=true'"
59 ECHO: "[ INFO ] run_test(); t16 passed: 'is_empty([undef])=false'"
60 ECHO: "[ INFO ] run_test(); t17 passed: 'is_empty([10])=false'
61 ECHO: "[ INFO ] run_test(); t18 passed: 'is_empty([1, 2, 3])=false'"
62 ECHO: "[ INFO ] run_test(); t19 passed: 'is_empty([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
63 ECHO: "[ INFO ] run_test(); t20 passed: 'is_empty([0 : 1 : 9])=false'"
64 ECHO: "[ INFO ] run_test(); t21 passed: 'is_empty([0 : 0.5 : 9])=false'"
65 ECHO: "[ INFO ] run_test(); t01 passed: 'is_scalar(undef)=true'"
66 ECHO: "[ INFO ] run_test(); t02 passed: 'is_scalar(1)=true'
67 ECHO: "[ INFO ] run_test(); t03 passed: 'is_scalar(10)=true'"
68 ECHO: "[ INFO ] run_test(); t04 passed: 'is_scalar(1e+08)=true'
69 ECHO: "[ INFO ] run_test(); t05 passed: 'is_scalar(0.01)=true'
70 ECHO: "[ INFO ] run_test(); t06 passed: 'is_scalar(1e+308)=true'
```

```
71 ECHO: "[ INFO ] run_test(); t07 passed: 'is_scalar(-1e+308)=true'"
72 ECHO: "[ INFO ] run_test(); t08 passed: 'is_scalar(inf)=true'
73 ECHO: "[ INFO ] run_test(); t09 passed: 'is_scalar(nan)=true'"
74 ECHO: "[ INFO ] run_test(); t10 passed: 'is_scalar(true)=true'"
75 ECHO: "[
            INFO ] run_test(); t11 passed: 'is_scalar(false)=true'"
76 ECHO: "[ INFO ] run_test(); t12 passed: 'is_scalar(a)=false'"
77 ECHO: "[
            INFO ] run_test(); t13 passed: 'is_scalar(This is a longer string)=false'"
78 ECHO: "[ INFO ] run_test(); t14 passed: 'is_scalar()=false'"
79 ECHO: "[ INFO ] run_test(); t15 passed: 'is_scalar([])=false'"
80 ECHO: "[ INFO ] run_test(); t16 passed: 'is_scalar([undef])=false'"
81 ECHO: "[ INFO ] run_test(); t17 passed: 'is_scalar([10])=false'
82 ECHO: "[ INFO ] run_test(); t18 passed: 'is_scalar([1, 2, 3])=false'"
83 ECHO: "[ INFO ] run_test(); t19 passed: 'is_scalar([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'" 84 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_scalar(A shorthand range)'"
85 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_scalar(A range)'"
86 ECHO: "[ INFO ] run_test(); t01 passed: 'is_iterable(undef)=false'"
87 ECHO: "[ INFO ] run_test(); t02 passed: 'is_iterable(1)=false'
88 ECHO: "[ INFO ] run_test(); t03 passed: 'is_iterable(10)=false'"
89 ECHO: "[ INFO ] run_test(); t04 passed: 'is_iterable(1e+08)=false'"
90 ECHO: "[ INFO ] run_test(); t05 passed: 'is_iterable(0.01)=false'"
91 ECHO: "[ INFO ] run_test(); t06 passed: 'is_iterable(1e+308)=false'
92 ECHO: "[ INFO ] run_test(); t07 passed: 'is_iterable(-1e+308)=false'"
93 ECHO: "[ INFO ] run_test(); t08 passed: 'is_iterable(inf)=false'
94 ECHO: "[ INFO ] run_test(); t09 passed: 'is_iterable(nan)=false'"
95 ECHO: "[ INFO ] run_test(); t10 passed: 'is_iterable(true)=false'"
96 ECHO: "[ INFO ] run_test(); t11 passed: 'is_iterable(false)=false'"
97 ECHO: "[ INFO ] run_test(); t12 passed: 'is_iterable(a)=true'
98 ECHO: "[ INFO ] run_test(); t13 passed: 'is_iterable(This is a longer string)=true'"
99 ECHO: "[ INFO ] run_test(); t14 passed: 'is_iterable()=true'"
100 ECHO: "[ INFO ] run_test(); t15 passed: 'is_iterable([])=true'"
101 ECHO: "[ INFO ] run_test(); t16 passed: 'is_iterable([undef])=true'"
102 ECHO: "[ INFO ] run_test(); t17 passed: 'is_iterable([10])=true'
103 ECHO: "[ INFO ] run_test(); t18 passed: 'is_iterable([1, 2, 3])=true'"
104 ECHO: "[ INFO ] run_test(); t19 passed: 'is_iterable([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=true'"
105 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_iterable(A shorthand range)'
106 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_iterable(A range)'"
107 ECHO: "[ INFO ] run_test(); t01 passed: 'is_string(undef)=false'
108 ECHO: "[ INFO ] run_test(); t02 passed: 'is_string(1)=false'
109 ECHO: "[ INFO ] run_test(); t03 passed: 'is_string(10)=false'
110 ECHO: "[ INFO ] run_test(); t04 passed: 'is_string(le+08)=false'"
111 ECHO: "[ INFO ] run_test(); t05 passed: 'is_string(0.01)=false'
112 ECHO: "[ INFO ] run_test(); t06 passed: 'is_string(1e+308)=false'
113 ECHO: "[ INFO ] run_test(); t07 passed: 'is_string(-1e+308)=false'"
114 ECHO: "[ INFO ] run_test(); t08 passed: 'is_string(inf)=false'
115 ECHO: "[ INFO ] run_test(); t09 passed: 'is_string(nan)=false'"
116 ECHO: "[ INFO ] run_test(); t10 passed: 'is_string(true)=false'
117 ECHO: "[ INFO ] run_test(); t11 passed: 'is_string(false)=false'"
118 ECHO: "[ INFO ] run_test(); t12 passed: 'is_string(a)=true'"
119 ECHO: "[ INFO ] run_test(); t13 passed: 'is_string(This is a longer string)=true'"
120 ECHO: "[ INFO ]
                     run_test(); t14 passed: 'is_string()=true'"
121 ECHO: "[ INFO ] run_test(); t15 passed: 'is_string([])=false'"
122 ECHO: "[ INFO ]
                     run_test(); t16 passed: 'is_string([undef])=false'"
123 ECHO: "[ INFO ] run_test(); t17 passed: 'is_string([10])=false'"
124 ECHO: "[ INFO ]
                     run_test(); t18 passed: 'is_string([1, 2, 3])=false'"
125 ECHO: "[ INFO ] run_test(); t19 passed: 'is_string([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
126 ECHO: "[ INFO ] run_test(); t20 passed: 'is_string([0 : 1 : 9])=false'"
127 ECHO: "[ INFO ] run_test(); t21 passed: 'is_string([0 : 0.5 : 9]) = false'"
128 ECHO: "[ INFO ] run_test(); t01 passed: 'is_vector(undef)=false'"
129 ECHO: "[ INFO ] run_test(); t02 passed: 'is_vector(1)=false'"
130 ECHO: "[ INFO ] run_test(); t03 passed: 'is_vector(10)=false'"
131 ECHO: "[ INFO ] run_test(); t04 passed: 'is_vector(1e+08)=false'"
132 ECHO: "[ INFO ] run_test(); t05 passed: 'is_vector(0.01)=false''
133 ECHO: "[ INFO ] run_test(); t06 passed: 'is_vector(1e+308)=false'"
134 ECHO: "[ INFO ] run_test(); t07 passed: 'is_vector(-1e+308)=false'"
135 ECHO: "[ INFO ] run_test(); t08 passed: 'is_vector(inf)=false'"
136 ECHO: "[ INFO ] run_test(); t09 passed: 'is_vector(nan)=false'"
137 ECHO: "[ INFO ] run_test(); t10 passed: 'is_vector(true)=false'"
138 ECHO: "[ INFO ] run_test(); tl1 passed: 'is_vector(false)=false'"
139 ECHO: "[ INFO ] run_test(); t12 passed: 'is_vector(a)=false'"
140 ECHO: "[ INFO ] run_test(); t13 passed: 'is_vector(This is a longer string)=false'"
141 ECHO: "[ INFO ] run_test(); t14 passed: 'is_vector()=false'"
142 ECHO: "[ INFO ] run_test(); t15 passed: 'is_vector([])=true'"
143 ECHO: "[ INFO ] run_test(); t16 passed: 'is_vector([undef])=true'"
144 ECHO: "[ INFO ] run_test(); t17 passed: 'is_vector([10]) = true'
145 ECHO: "[ INFO ] run_test(); t18 passed: 'is_vector([1, 2, 3])=true'"
146 ECHO: "[ INFO ] run_test(); t19 passed: 'is_vector([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=true'"
147 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_vector(A shorthand range)'"
148 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_vector(A range)'
149 ECHO: "[ INFO ] run_test(); t01 passed: 'is_boolean(undef)=false'
150 ECHO: "[ INFO ] run_test(); t02 passed: 'is_boolean(1)=false'"
151 ECHO: "[ INFO ] run_test(); t03 passed: 'is_boolean(10)=false'"
```

```
152 ECHO: "[ INFO ] run_test(); t04 passed: 'is_boolean(1e+08)=false'"
153 ECHO: "[ INFO ] run_test(); t05 passed: 'is_boolean(0.01)=false'"
                     run_test(); t06 passed: 'is_boolean(1e+308)=false''
154 ECHO: "[ INFO ]
155 ECHO: "[ INFO ] run_test(); t07 passed: 'is_boolean(-1e+308)=false'"
156 ECHO: "[ INFO
                     run_test(); t08 passed: 'is_boolean(inf)=false'
                     run_test(); t09 passed: 'is_boolean(nan)=false'"
157 ECHO: "[ INFO ]
158 ECHO: "[ INFO
                     run_test(); t10 passed: 'is_boolean(true)=true'"
159 ECHO: "[ INFO ] run_test(); t11 passed: 'is_boolean(false)=true'"
160 ECHO: "[ INFO ]
                     run_test(); t12 passed: 'is_boolean(a)=false'
161 ECHO: "[ INFO ] run_test(); t13 passed: 'is_boolean(This is a longer string)=false'"
162 ECHO: "[ INFO ]
                     run_test(); t14 passed: 'is_boolean()=false'"
163 ECHO: "[ INFO ] run_test(); t15 passed: 'is_boolean([])=false'"
164 ECHO: "[ INFO ]
                     run_test(); t16 passed: 'is_boolean([undef])=false'"
165 ECHO: "[ INFO ] run_test(); t17 passed: 'is_boolean([10])=false'
166 ECHO: "[ INFO ]
                     run_test(); t18 passed: 'is_boolean([1, 2, 3])=false'"
167 ECHO: "[ INFO ] run_test(); t19 passed: 'is_boolean([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
                     run_test(); t20 passed: 'is_boolean([0 : 1 : 9])=false'"
168 ECHO: "[ INFO ]
169 ECHO: "[ INFO ] run_test(); t21 passed: 'is_boolean([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(0.01)=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'"
                     run_test(); t11 passed: 'is_integer(false)=false'"
180 ECHO: "[ INFO ]
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'"
189 ECHO: "[ INFO ] run_test(); t20 passed. is_integer([0 : 0.5 : 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(0.01)=true'
196 ECHO: "[ INFO ] run_test(); t06 passed: 'is_decimal(le+308)=false'"
197 ECHO: "[ INFO ] run_test(); t07 passed: 'is_decimal(-le+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'"
                     run_test(); t15 passed: 'is_decimal([])=false'"
205 ECHO: "[ INFO ]
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_number(undef)=false'"
213 ECHO: "[ INFO ]
                     run_test(); t02 passed: 'is_number(1)=true'"
214 ECHO: "[ INFO ] run_test(); t03 passed: 'is_number(10)=true'"
215 ECHO: "[ INFO 1
                     run_test(); t04 passed: 'is_number(1e+08)=true'"
216 ECHO: "[ INFO ] run_test(); t05 passed: 'is_number(0.01)=true'"
217 ECHO: "[ INFO ]
                     run_test(); t06 passed: 'is_number(1e+308)=true'"
218 ECHO: "[ INFO ] run_test(); t07 passed: 'is_number(-1e+308)=true'"
219 ECHO: "[ INFO ]
                     run_test(); t08 passed: 'is_number(inf)=true'
220 ECHO: "[ INFO ] run_test(); t09 passed: 'is_number(nan)=true'"
221 ECHO: "[ INFO ]
                     run_test(); t10 passed: 'is_number(true)=false''
222 ECHO: "[ INFO ] run_test(); t11 passed: 'is_number(false)=false'"
223 ECHO: "[ INFO ]
                     run test(); t12 passed: 'is number(a)=false'
224 ECHO: "[ INFO ] run_test(); t13 passed: 'is_number(This is a longer string) = false'"
225 ECHO: "[ INFO ]
                     run_test(); t14 passed: 'is_number()=false'"
226 ECHO: "[ INFO ] run_test(); t15 passed: 'is_number([])=false'"
                     run_test(); t16 passed: 'is_number([undef])=false''
227 ECHO: "[ INFO ]
228 ECHO: "[ INFO ] run_test(); t17 passed: 'is_number([10])=false'
229 ECHO: "[ INFO ]
                     run_test(); t18 passed: 'is_number([1, 2, 3])=false'"
230 ECHO: "[ INFO ] run_test(); t19 passed: 'is_number([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
231 ECHO: "[ INFO ] run_test(); t20 passed: 'is_number([0 : 1 : 9])=false'"
232 ECHO: "[ INFO ] run_test(); t21 passed: 'is_number([0 : 0.5 : 9])=false'"
```

```
233 ECHO: "[ INFO ] run_test(); t01 passed: 'is_range(undef)=false'"
234 ECHO: "[ INFO ] run_test(); t02 passed: 'is_range(1)=false'"
                     run_test(); t03 passed: 'is_range(10)=false'"
235 ECHO: "[ INFO ]
236 ECHO: "[ INFO ] run_test(); t04 passed: 'is_range(1e+08)=false'"
237 ECHO: "[ INFO ] run_test(); t05 passed: 'is_range(0.01)=false'"
238 ECHO: "[ INFO ] run_test(); t06 passed: 'is_range(1e+308)=false'"
239 ECHO: "[ INFO ] run_test(); t07 passed: 'is_range(-1e+308)=false'"
240 ECHO: "[ INFO ] run_test(); t08 passed: 'is_range(inf)=false'"
241 ECHO: "[ INFO ] run_test(); t09 passed: 'is_range(nan)=false'"
242 ECHO: "[ INFO ] run_test(); t10 passed: 'is_range(true)=false'"
243 ECHO: "[ INFO ] run_test(); t11 passed: 'is_range(false)=false'"
244 ECHO: "[ INFO ] run_test(); t12 passed: 'is_range(a)=false'"
245 ECHO: "[ INFO ] run_test(); t13 passed: 'is_range(This is a longer string)=false'"
246 ECHO: "[ INFO ] run_test(); t14 passed: 'is_range()=false'"
247 ECHO: "[ INFO ] run_test(); t15 passed: 'is_range([])=false'"
248 ECHO: "[ INFO ] run_test(); t16 passed: 'is_range([undef])=false'"
249 ECHO: "[ INFO ] run_test(); t17 passed: 'is_range([10])=false'
250 ECHO: "[ INFO ] run_test(); t18 passed: 'is_range([1, 2, 3])=false'"
251 ECHO: "[ INFO ] run_test(); t19 passed: 'is_range([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
252 ECHO: "[ INFO ] run_test(); t20 passed: 'is_range([0 : 1 : 9])=true'"
253 ECHO: "[ INFO ] run_test(); t21 passed: 'is_range([0 : 0.5 : 9])=true'"
254 ECHO: "[ INFO ] run_test(); t01 passed: 'is_nan(undef)=false'"
255 ECHO: "[ INFO ] run_test(); t02 passed: 'is_nan(1)=false'
256 ECHO: "[ INFO ] run_test(); t03 passed: 'is_nan(10)=false'"
257 ECHO: "[ INFO ] run_test(); t04 passed: 'is_nan(1e+08)=false'
258 ECHO: "[ INFO ] run_test(); t05 passed: 'is_nan(0.01)=false'
259 ECHO: "[ INFO ] run_test(); t06 passed: 'is_nan(1e+308)=false'"
260 ECHO: "[ INFO ] run_test(); t07 passed: 'is_nan(-1e+308)=false'"
261 ECHO: "[ INFO ] run_test(); t08 passed: 'is_nan(inf)=false'
262 ECHO: "[ INFO ] run_test(); t09 passed: 'is_nan(nan)=true'"
263 ECHO: "[ INFO ] run_test(); t10 passed: 'is_nan(true)=false'
264 ECHO: "[ INFO ] run_test(); tll passed: 'is_nan(false)=false'"
265 ECHO: "[ INFO ] run_test(); t12 passed: 'is_nan(a)=false'
266 ECHO: "[ INFO ] run_test(); t13 passed: 'is_nan(This is a longer string)=false'"
267 ECHO: "[ INFO ] run_test(); t14 passed: 'is_nan()=false''
268 ECHO: "[ INFO ] run_test(); t15 passed: 'is_nan([])=false'"
269 ECHO: "[ INFO ] run_test(); t16 passed: 'is_nan([undef])=false'"
270 ECHO: "[ INFO ] run_test(); t17 passed: 'is_nan([10])=false'
271 ECHO: "[ INFO ] run_test(); t18 passed: 'is_nan([1, 2, 3])=false'"
272 ECHO: "[ INFO ] run_test(); t19 passed: 'is_nan([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
273 ECHO: "[ INFO ] run_test(); t20 passed: 'is_nan([0 : 1 : 9])=false'"
274 ECHO: "[ INFO ] run_test(); t21 passed: 'is_nan([0 : 0.5 : 9])=false'"
275 ECHO: "[ INFO ] run_test(); t01 passed: 'is_inf(undef)=false'"
276 ECHO: "[ INFO ] run_test(); t02 passed: 'is_inf(1)=false'
277 ECHO: "[ INFO ] run_test(); t03 passed: 'is_inf(10)=false'"
278 ECHO: "[ INFO ] run_test(); t04 passed: 'is_inf(1e+08)=false'"
279 ECHO: "[ INFO ] run_test(); t05 passed: 'is_inf(0.01)=false'"
280 ECHO: "[ INFO ] run_test(); t06 passed: 'is_inf(1e+308)=false'"
281 ECHO: "[ INFO ] run_test(); t07 passed: 'is_inf(-1e+308)=false'"
282 ECHO: "[ INFO ] run_test(); t08 passed: 'is_inf(inf)=true''
283 ECHO: "[ INFO ] run_test(); t09 passed: 'is_inf(nan)=false'"
284 ECHO: "[ INFO ] run_test(); t10 passed: 'is_inf(true)=false'"
285 ECHO: "[ INFO ] run_test(); t11 passed: 'is_inf(false)=false'"
286 ECHO: "[ INFO ] run_test(); t12 passed: 'is_inf(a)=false'"
287 ECHO: "[ INFO ] run_test(); t13 passed: 'is_inf(This is a longer string)=false'"
288 ECHO: "[ INFO ] run_test(); t14 passed: 'is_inf()=false'"
289 ECHO: "[ INFO ] run_test(); t15 passed: 'is_inf([])=false'"
290 ECHO: "[ INFO ] run_test(); t16 passed: 'is_inf([undef])=false'"
291 ECHO: "[ INFO ] run_test(); t17 passed: 'is_inf([10])=false'"
292 ECHO: "[ INFO ] run_test(); t18 passed: 'is_inf([1, 2, 3])=false'"
293 ECHO: "[ INFO ] run_test(); t19 passed: 'is_inf([[1, 2, 3], [4, 5, 6], [7, 8, 9]])=false'"
294 ECHO: "[ INFO ] run_test(); t20 passed: 'is_inf([0 : 1 : 9])=false'"
295 ECHO: "[ INFO ] run_test(); t21 passed: 'is_inf([0 : 0.5 : 9])=false'"
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'"
299 ECHO: "[ INFO ] run_test(); t04 passed: 'is_even(1e+08)=true'"
300 ECHO: "[ INFO ] run_test(); t05 passed: 'is_even(0.01)=false'"
301 ECHO: "[ INFO ] run_test(); t06 passed: 'is_even(1e+308)=true'"
302 ECHO: "[ INFO ] run_test(); t07 passed: 'is_even(-1e+308)=true'"
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)'"
305 ECHO: "[ INFO ] run_test(); t10 *skip*: 'is_even(The boolean true)'"
306 ECHO: "[ INFO ] run_test(); t11 *skip*: 'is_even(The boolean false)'"
307 ECHO: "[ INFO ] run_test(); t12 *skip*: 'is_even(A character string)'"
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 vector)'"
311 ECHO: "[ INFO ] run_test(); t16 *skip*: 'is_even(A 1-tuple vector of undef)'"
312 ECHO: "[ INFO ] run_test(); t17 *skip*: 'is_even(A 1-tuple vector)'"
313 ECHO: "[ INFO ] run_test(); t18 *skip*: 'is_even(A 3-tuple vector)'"
```

```
314 ECHO: "[ INFO ] run_test(); t19 *skip*: 'is_even(A vector of vectors)'"
315 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_even(A shorthand range)'"
                     run_test(); t21 *skip*: 'is_even(A range)'"
316 ECHO: "[ INFO ]
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(0.01)=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)''
331 ECHO: "[ INFO ] run_test(); t15 *skip*: 'is_odd(The empty vector)'"
332 ECHO: "[ INFO ] run_test(); t16 *skip*: 'is_odd(A 1-tuple vector of undef)'"
333 ECHO: "[ INFO ] run_test(); t17 *skip*: 'is_odd(A 1-tuple vector)'"
334 ECHO: "[ INFO ] run_test(); t18 *skip*: 'is_odd(A 3-tuple vector)'"
335 ECHO: "[ INFO ] run_test(); t19 *skip*: 'is_odd(A vector of vectors)'"
336 ECHO: "[ INFO ] run_test(); t20 *skip*: 'is_odd(A shorthand range)'
337 ECHO: "[ INFO ] run_test(); t21 *skip*: 'is_odd(A range)'"
```

2.1.1.3 Validation Script (group2)

```
include <primitives.scad>;
use <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", "An odd integer",
  ["t03", "The boolean true", ["t04", "The boolean false",
                                            false],
  ["t05", "A character string", ["t06", "A string",
                                            "a"],
                                           "This is a longer string"],
  ["t07", "The empty string",
                                            empty_str],
   ["t08", "The empty vector",
                                            empty_v],
  ["t09", "A shorthand range",
                                            [0:9]],
  ["t10", "A range",
                                            [0:0.5:9]],
  ["t11", "Test vector 01",
                                            [undef]],
   "t12", "Test vector 02",
                                            [1]],
                                           [1, 2, 3]],
[[1], [2], [3], [4], [5]]],
  ["t13", "Test vector 03",
   ["t14", "Test vector 04",
                                           [[1,2], [2,3]]],
  ["t15", "Test vector 05",
  ["t16", "Test vector 06",
                                            [[1,2], [2,3], [4,5], "ab"]],
  ["t17", "Test vector 07",
                                            [[1,2,3], [4,5,6], [7,8,9], ["a", "b", "c"]]],
  ["t18", "Test vector 08",
                                            [1, 2, 3, undef]],
  ["t19", "Test vector 09",
                                            [undef, undef, undef, undef]],
  ["t20", "Test vector 10",
                                            [[undef], [undef], [undef]]],
  ["t21", "Test vector 11", ["t22", "Test vector 12",
                                            [true, true, true, true, false]],
                                            [true, false, false, false, false]],
  ["t23", "Test vector 13",
                                           [true, true, true, true]]
test_ids = table_get_row_ids( 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 =
 [ // 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,
   ["all_vectors",
                    u, f, f, f, f, t, t, f, f, f, f, f, t, t, f, t, f, f, f, f, f],
   ["all_strings",
                    ["all_numbers",
                    ["all_len_1",
                    ["all_len_2",
                    u, f, f, f, f, s, s, f, f, f, f, f, f, t, t, f, f, f, f, f, f, f]
   ["all_len_3",
                    ["compare AA",
                    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 ) = table_get(test_r, test_c, vid, "tv");
 module run_test ( fname, fresult, vid )
   value_text = table_get(test_r, test_c, vid, "td");
   pass_value = table_get(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( "all_equal_U", 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_vectors", all_vectors(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);
 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
```

2.1.1.4 Validation Results (group2)

```
1 ECHO: "OpenSCAD Version [2016, 12, 21]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'all_equal_T(undef)=false'"
                   run_test(); t02 passed: 'all_equal_T(1)=false'"
3 ECHO: "[ INFO ]
                   run_test(); t03 passed: 'all_equal_T(true)=true'"
4 ECHO: "[ INFO ]
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(); tll 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], [undef]))=false'"
22 ECHO: "[ INFO ] run_test(); t21 passed: 'all_equal_T([true, true, true, true, false])=false'"
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(); t11 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])=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'"
46 ECHO: "[ INFO ] run_test(); t22 passed: 'all_equal_F([true, false, false, false, false])=false'"
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])=true'"
67 ECHO: "[ INFO ] run_test(); t20 passed: 'all_equal_U([[undef], [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(); t11 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]])=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, undef])=false'"
113 ECHO: "[ INFO ] run_test(); t20 passed: 'any_equal_F([[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'"
122 ECHO: "[ INFO ] run_test(); t06 passed: 'any_equal_U(This is a longer string)=false'"
123 ECHO: "[ INFO ] run_test(); t07 passed: 'any_equal_U()=false'"
124 ECHO: "[ INFO ] run_test(); t08 passed: 'any_equal_U([])=false'"
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])=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(); tll 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'"
```

```
155 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'"
                   ] run_test(); t19 passed: 'all_defined([undef, undef, undef])=false'"
158 ECHO: "[ INFO
                     run_test(); t20 passed: 'all_defined([[undef], [undef], [undef]])=true'"
159 ECHO: "[ INFO ]
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'"
                     run_test(); t23 passed: 'all_defined([true, true, true, true])=true'"
162 ECHO: "[ INFO ]
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'"
                     run_test(); t10 passed: 'any_undefined([0 : 0.5 : 9])=false'"
172 ECHO: "[ INFO ]
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'"
182 ECHO: "[ INFO ] run_test(); t20 passed: 'any_undefined([[undef], [undef], [undef]])=false'"
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'"
                     run_test(); t23 passed: 'any_undefined([true, true, true, true])=false''
185 ECHO: "[ INFO ]
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'"
192 ECHO: "[ INFO ] run_test(); t07 *skip*: 'all_scalars(The empty string)''
193 ECHO: "[ INFO ]
                     run_test(); t08 *skip*: 'all_scalars(The empty vector)'"
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'"
                     run_test(); t21 passed: 'all_scalars([true, true, true, true, false])=true'"
206 ECHO: "[ INFO ]
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_vectors(undef)=undef'"
210 ECHO: "[ INFO ]
                     run_test(); t02 passed: 'all_vectors(1)=false'"
                     run_test(); t03 passed: 'all_vectors(true)=false'"
211 ECHO: "[ INFO ]
212 ECHO: "[ INFO ] run_test(); t04 passed: 'all_vectors(false)=false'"
213 ECHO: "[ INFO ]
                     run_test(); t05 passed: 'all_vectors(a)=false'"
214 ECHO: "[ INFO ] run_test(); t06 passed: 'all_vectors(This is a longer string) = false'"
                     run_test(); t07 passed: 'all_vectors()=true'"
215 ECHO: "[ INFO 1
216 ECHO: "[ INFO ]
                     run_test(); t08 passed: 'all_vectors([])=true'"
217 ECHO: "[ INFO
                     run_test(); t09 passed: 'all_vectors([0 : 1 : 9])=false'"
218 ECHO: "[ INFO ] run_test(); t10 passed: 'all_vectors([0 : 0.5 : 9])=false'"
219 ECHO: "[ INFO ]
                     run_test(); t11 passed: 'all_vectors([undef])=false'"
220 ECHO: "[ INFO ] run_test(); t12 passed: 'all_vectors([1])=false'"
221 ECHO: "[ INFO ]
                     run_test(); t13 passed: 'all_vectors([1, 2, 3])=false'"
222 ECHO: "[ INFO ] run_test(); t14 passed: 'all_vectors([[1], [2], [3], [4], [5]])=true'"
223 ECHO: "[ INFO ] run_test(); t15 passed: 'all_vectors([[1, 2], [2, 3])]=true'"
224 ECHO: "[ INFO ] run_test(); t16 passed: 'all_vectors([[1, 2], [2, 3], [4, 5], "ab"])=false'"
225 ECHO: "[ INFO ] run_test(); t17 passed: 'all_vectors([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
       "c"]])=true'"
226 ECHO: "[ INFO ] run_test(); t18 passed: 'all_vectors([1, 2, 3, undef])=false'
227 ECHO: "[ INFO ] run_test(); t19 passed: 'all_vectors([undef, undef, undef, undef])=false'
228 ECHO: "[ INFO ]
                     run_test(); t20 passed: 'all_vectors([[undef], [undef], [undef]])=true'
229 ECHO: "[ INFO ] run_test(); t21 passed: 'all_vectors([true, true, true, true, false])=false'"
230 ECHO: "[ INFO ] run_test(); t22 passed: 'all_vectors([true, false, false, false])=false'"
231 ECHO: "[ INFO ] run_test(); t23 passed: 'all_vectors([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'"
                      run_test(); t03 passed: 'all_strings(true)=false'
234 ECHO: "[ INFO ]
235 ECHO: "[ INFO ] run_test(); t04 passed: 'all_strings(false)=false'"
                      run_test(); t05 passed: 'all_strings(a)=true'"
236 ECHO: "[ INFO ]
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 vector)'"
                      run_test(); t09 passed: 'all_strings([0 : 1 : 9])=false'"
240 ECHO: "[ INFO ]
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 vector)'"
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'"
279 ECHO: "[ INFO ]
                      run_test(); t02 passed: 'all_len_1(1)=false'"
280 ECHO: "[ INFO ] run_test(); t03 passed: 'all_len_1(true)=false'"
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)'"
285 ECHO: "[ INFO ]
                      run_test(); t08 *skip*: 'all_len_1(The empty vector)'"
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 ]
                      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, 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'"
302 ECHO: "[ INFO ]
                      run_test(); t02 passed: 'all_len_2(1)=false'"
303 ECHO: "[ INFO ] run_test(); t03 passed: 'all_len_2(true)=false'"
304 ECHO: "[ INFO ]
                      run_test(); t04 passed: 'all_len_2(false)=false'"
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 vector)'"
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(); t16 passed: 'all_len_2([[1, 2], [2, 3], [4, 5], "ab"])=true'"
317 ECHO: "[ INFO ] run_test(); t17 passed: 'all_len_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'"
                        run_test(); t19 passed: 'all_len_2([undef, undef, undef, undef])=false'"
319 ECHO: "[ INFO ]
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 vector)'"
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"11) = 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])=false'"
343 ECHO: "[ INFO ] run_test(); t20 passed: 'all_len_3([[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(); t21 passed: 'all_len_3([true, true, true, true])=false''
347 ECHO: "[ INFO ] run_test(); t01 passed: 'almost_equal_AA(undef)=true'"
348 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_AA(1)=true'"
349 ECHO: "[ INFO ] run_test(); t03 passed: 'almost_equal_AA(true)=true'
350 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_AA(false)=true'"
351 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_AA(a)=true'
352 ECHO: "[INFO] run_test(); t06 passed: 'almost_equal_AA(This is a longer string)=true'"
353 ECHO: "[INFO] run_test(); t07 passed: 'almost_equal_AA()=true'"
354 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_AA([])=true'"
355 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_AA([0 : 1 : 9])=true'"
356 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_AA([0 : 0.5 : 9])=true'"
357 ECHO: "[ INFO ]
                        run_test(); t11 passed: 'almost_equal_AA([undef])=true'
358 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_AA([1])=true'"
359 ECHO: "[ INFO ]
                        run_test(); t13 passed: 'almost_equal_AA([1, 2, 3])=true'"
360 ECHO: "[INFO] run_test(); t14 passed: 'almost_equal_AA([[1, 2], [3], [4], [5]])=true'"
361 ECHO: "[INFO] run_test(); t15 passed: 'almost_equal_AA([[1, 2], [2, 3]])=true'"
362 ECHO: "[INFO] run_test(); t16 passed: 'almost_equal_AA([[1, 2], [2, 3], [4, 5], "ab"])=true'"
363 ECHO: "[INFO] run_test(); t17 passed: 'almost_equal_AA([[1, 2], [2, 3], [4, 5], "ab"])=true'"
        "c"]])=true'"
364 ECHO: "[INFO] run_test(); t18 passed: 'almost_equal_AA([1, 2, 3, undef])=true'"
365 ECHO: "[INFO] run_test(); t19 passed: 'almost_equal_AA([undef, undef, undef, undef])=true'"
                        run_test(); t20 passed: 'almost_equal_AA([[undef], [undef]]) = true'"
366 ECHO: "[ INFO ]
367 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_AA([true, true, true, true, false])=true'"
368 ECHO: "[ INFO ]
                        run_test(); t22 passed: 'almost_equal_AA([true, false, false, false, false])=true'"
369 ECHO: "[ INFO ] run_test(); t23 passed: 'almost_equal_AA([true, true, true, true])=true'"
370 ECHO: "[ INFO 1
                        run_test(); t01 passed: 'almost_equal_T(undef)=false'"
371 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_T(1)=false'"
372 ECHO: "[ INFO ]
                        run_test(); t03 passed: 'almost_equal_T(true)=true'
373 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_T(false)=false'"
374 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_T(a)=false'
375 ECHO: "[ INFO ] run_test(); t06 passed: 'almost_equal_T(This is a longer string)=false'"
376 ECHO: "[ INFO ] run_test(); t07 passed: 'almost_equal_T()=false'"
377 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_T([])=false'"
378 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_T([0 : 1 : 9]) = false'"
379 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_T([0 : 0.5 : 9]) = false'"
380 ECHO: "[ INFO ] run_test(); t11 passed: 'almost_equal_T([undef])=false'"
381 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_T([1])=false'"
381 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_T([1, 2, 3])=false'"

382 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_T([1], [2], [3], [4], [5]])=false'"

383 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_T([[1, 2], [2, 3]])=false'"

384 ECHO: "[ INFO ] run_test(); t15 passed: 'almost_equal_T([[1, 2], [2, 3]])=false'"

385 ECHO: "[ INFO ] run_test(); t16 passed: 'almost_equal_T([[1, 2], [2, 3], [4, 5], "ab"])=false'"
386 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_T([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=false'"
```

```
387 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_T([1, 2, 3, undef])=false'"
388 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_T([undef, undef, undef, undef])=false'"
389 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_T([[undef], [undef]], [undef]])=false'"
390 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_T([true, true, true, true, false])=false'"
                      run_test(); t22 passed: 'almost_equal_T([true, false, false, false, false])=false'"
391 ECHO: "[ INFO ]
392 ECHO: "[ INFO ] run_test(); t23 passed: 'almost_equal_T([true, true, true, true])=false'"
393 ECHO: "[ INFO ] run_test(); t01 passed: 'almost_equal_f (undef)=false'"
394 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_f(1)=false'"
                      run_test(); t03 passed: 'almost_equal_F(true)=false''
395 ECHO: "[ INFO ]
396 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_F(false) = true'"
397 ECHO: "[ INFO ]
                      run_test(); t05 passed: 'almost_equal_F(a)=false'
398 ECHO: "[ INFO ] run_test(); t06 passed: 'almost_equal_F(This is a longer string)=false'"
399 ECHO: "[ INFO ]
                      run_test(); t07 passed: 'almost_equal_F()=false'"
400 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_F([])=false'"
401 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_F([0 : 1 : 9])=false'"
402 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_F([0 : 0.5 : 9]) = false'"
403 ECHO: "[ INFO ] run_test(); t11 passed: 'almost_equal_F([undef]) = false'"
404 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_F([1])=false'"
405 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_F([1, 2, 3])=false'"
406 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_F([[1], [2], [3], [4], [5]])=false'"
400 ECHO: "[INFO] run_test(); t15 passed: 'almost_equal_F([[1, 2], [2, 3]])=false'"
408 ECHO: "[INFO] run_test(); t16 passed: 'almost_equal_F([[1, 2], [2, 3], [4, 5], "ab"])=false'"
409 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_F([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=false'"
410 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_F([1, 2, 3, undef])=false'"
411 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_F([undef, undef, undef, undef])=false'"
412 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_F([[undef], [undef], [undef]])=false'"
413 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_F([true, true, true, true, false])=false'"
414 ECHO: "[ INFO ] run_test(); t22 passed: 'almost_equal_F([true, false, false, false, false])=false'"
414 ECHO: [INFO] run_test(); t23 passed: 'almost_equal_F([true, true, true])=false'"
416 ECHO: "[INFO] run_test(); t01 passed: 'almost_equal_U(undef)=true'"
417 ECHO: "[ INFO ] run_test(); t02 passed: 'almost_equal_U(1)=false'
418 ECHO: "[ INFO ] run_test(); t03 passed: 'almost_equal_U(true)=false'
419 ECHO: "[ INFO ] run_test(); t04 passed: 'almost_equal_U(false)=false'
420 ECHO: "[ INFO ] run_test(); t05 passed: 'almost_equal_U(a)=false'
421 ECHO: "[ INFO ] run_test(); t06 passed: 'almost_equal_U(This is a longer string)=false'"
422 ECHO: "[ INFO ] run_test(); t07 passed: 'almost_equal_U()=false'"
423 ECHO: "[ INFO ] run_test(); t08 passed: 'almost_equal_U([])=false'"
424 ECHO: "[ INFO ] run_test(); t09 passed: 'almost_equal_U([0 : 1 : 9]) = false'"
425 ECHO: "[ INFO ] run_test(); t10 passed: 'almost_equal_U([0 : 0.5 : 9])=false'"
426 ECHO: "[ INFO ] run_test(); t11 passed: 'almost_equal_U([undef])=false'"
427 ECHO: "[ INFO ] run_test(); t12 passed: 'almost_equal_U([1])=false'
428 ECHO: "[ INFO ] run_test(); t13 passed: 'almost_equal_U([1, 2, 3])=false'"
429 ECHO: "[ INFO ] run_test(); t14 passed: 'almost_equal_U([[1], [2], [3], [4], [5]])=false'"
430 ECHO: "[ INFO ] run_test(); t15 passed: 'almost_equal_U([[1, 2], [2, 3]])=false'"
431 ECHO: "[ INFO ] run_test(); t16 passed: 'almost_equal_U([[1, 2], [2, 3], [4, 5], "ab"])=false'"
432 ECHO: "[ INFO ] run_test(); t17 passed: 'almost_equal_U([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=false'"
433 ECHO: "[ INFO ] run_test(); t18 passed: 'almost_equal_U([1, 2, 3, undef])=false'"
434 ECHO: "[ INFO ] run_test(); t19 passed: 'almost_equal_U([undef, undef, undef, undef])=false'"
435 ECHO: "[ INFO ] run_test(); t20 passed: 'almost_equal_U([[undef], [undef], [undef]])=false'"
436 ECHO: "[ INFO ] run_test(); t21 passed: 'almost_equal_U([true, true, true, true, false])=false'"
437 ECHO: "[ INFO ] run_test(); t22 passed: 'almost_equal_U([true, false, false, false, false])=false'"
438 ECHO: "[ INFO ]
                      run_test(); t23 passed: 'almost_equal_U([true, true, true, true])=false'
439 ECHO: "[ INFO ] run_test(); t01 passed: 'compare_AA(undef)=true'"
440 ECHO: "[ INFO ]
                      run_test(); t02 passed: 'compare_AA(1)=true'"
441 ECHO: "[ INFO ] run_test(); t03 passed: 'compare_AA(true)=true'"
442 ECHO: "[ INFO ] run_test(); t04 passed: 'compare_AA(false)=true'"
443 ECHO: "[ INFO ] run_test(); t05 passed: 'compare_AA(a)=true'"
444 ECHO: "[ INFO ] run_test(); t06 passed: 'compare_AA(This is a longer string)=true'"
445 ECHO: "[ INFO ] run_test(); t07 passed: 'compare_AA()=true'"
446 ECHO: "[ INFO ] run_test(); t08 passed: 'compare_AA([])=true'"
447 ECHO: "[ INFO ] run_test(); t09 passed: 'compare_AA([0 : 1 : 9])=true'"
448 ECHO: "[ INFO ] run_test(); t10 passed: 'compare_AA([0 : 0.5 : 9])=true'"
449 ECHO: "[ INFO ] run_test(); t11 passed: 'compare_AA([undef])=true'"
450 ECHO: "[ INFO ] run_test(); t12 passed: 'compare_AA([1])=true'"
451 ECHO: "[ INFO ] run_test(); t13 passed: 'compare_AA([1, 2, 3])=true'"
452 ECHO: "[ INFO ] run_test(); t14 passed: 'compare_AA([[1], [2], [3], [4], [5]])=true'"
453 ECHO: "[INFO] run_test(); t15 passed: 'compare_AA([[1, 2], [2, 3]])=true'"
454 ECHO: "[INFO] run_test(); t16 passed: 'compare_AA([[1, 2], [2, 3], [4, 5], "ab"])=true'"
455 ECHO: "[ INFO ] run_test(); t17 passed: 'compare_AA([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
        "c"]])=true'
456 ECHO: "[ INFO ] run_test(); t18 passed: 'compare_AA([1, 2, 3, undef])=true'"
457 ECHO: "[ INFO ] run_test(); t19 passed: 'compare_AA([undef, undef, undef])=true'"
458 ECHO: "[ INFO ] run_test(); t20 passed: 'compare_AA([[undef], [undef], [undef]])=true'"
459 ECHO: "[ INFO ] run_test(); t21 passed: 'compare_AA([true, true, true, true, false])=true'"
460 ECHO: "[ INFO ] run_test(); t22 passed: 'compare_AA([true, false, false, false, false])=true'"
461 ECHO: "[ INFO ] run_test(); t23 passed: 'compare_AA([true, true, true, true])=true'
```

2.1.2 Vector Operations

- Validation Script
- · Validation Results

2.1.2.1 Validation Script

```
include <primitives.scad>;
use <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", "The empty vector", empty_v], [0:0.5:9]], ["t04", "A string", ["t05", "Test vector 01", ["t06", "Test vector 02", ["t07", "Test vector 03", [undef]], ["t08", "Test vector 04", [1,2], [2,3]], [4,5,6], [7,111", "Vector of integers 0 to 15", [for (i=[0:15]) i]]
                                                       "A string"],
["orange","apple","grape","banana"]],
["b","a","n","a","n","a","s"]],
                                                       ["ab",[1,2],[2,3],[4,5]]],
                                                       [[1,2,3],[4,5,6],[7,8,9],["a","b","c"]]],
];
test_ids = table_get_row_ids( 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
[ // function
  ["consts",
     empty_v,
     empty_v,
                                                                             // t02
     empty_v,
                                                                             // t03
                                                                             // t04
     empty_v,
                                                                             // t05
     empty_v,
                                                                             // t06
     empty_v,
     empty_v,
                                                                             // t07
                                                                             // t08
     empty_v,
     empty_v,
                                                                             // t09
     empty_v,
                                                                             // t10
                                                                             // t11
     empty_v
  ],
["vstr",
     undef,
                                                                             // t01
     empty_str,
"[0 : 0.5 : 9]",
                                                                             // t02
                                                                             // t03
     "A string",
                                                                             // t04
     "orangeapplegrapebanana",
                                                                             // t05
     "bananas",
                                                                             // t06
     "undef",
                                                                             // t07
     "[1, 2][2, 3]",
                                                                             // t08
     "ab[1, 2][2, 3][4, 5]",
"[1, 2, 3][4, 5, 6][7, 8, 9][\"a\", \"b\", \"c\"]",
"0123456789101112131415"
                                                                             // t09
                                                                            // t10
                                                                             // t11
  1,
```

```
["sum",
  undef,
                                                                  // t01
                                                                  // t02
  85.5,
                                                                  // t03
  undef,
                                                                  // t04
  undef,
                                                                  // t05
  undef,
                                                                  // t06
  undef,
                                                                  // t07
  [3,5],
                                                                  // t08
                                                                  // t09
  undef,
  [undef,undef,undef],
                                                                  // t10
                                                                  // t11
],
["find_12",
  empty_v,
                                                                  // t01
  empty_v,
                                                                  // t02
  empty_v,
                                                                  // t03
                                                                  // t04
  empty_v,
                                                                  // t05
  empty_v,
                                                                  // t06
  empty_v,
  empty_v,
                                                                  // t07
  [0],
                                                                  // t08
                                                                  // t09
  [1],
  empty_v,
                                                                  // t10
  empty_v
                                                                  // t11
["count_S1",
                                                                  // t01
  Ο,
                                                                  // t02
// t03
  0,
  0,
                                                                  // t04
  0,
                                                                  // t05
  Ο,
                                                                  // t06
// t07
  0.
  0,
                                                                  // t08
  1,
                                                                  // t09
  1,
                                                                  // t10
// t11
  1,
],
["exists_S1",
                                                                  // t01
  false,
                                                                  // t02
// t03
// t04
  false,
  false,
  false,
                                                                  // t05
// t06
  false,
  false,
                                                                  // t07
// t08
  false,
  true,
  true,
                                                                  // t09
                                                                  // t10
  true,
                                                                  // t11
  true
["defined_or_D",
                                                                  // t01
  "default",
  empty_v,
                                                                  // t02
  [0:0.5:9],
                                                                  // t03
  "A string",
["orange", "apple", "grape", "banana"],
["b", "a", "n", "a", "n", "a", "s"],
                                                                  // 04
                                                                  // t05
                                                                  // 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
                                                                  // t11
["edefined_or_DE3",
  "default",
                                                                  // t01
  "default",
                                                                  // t02
                                                                  // t03
  "default",
  "t",
                                                                  // t04
  "banana",
                                                                  // t05
  "a",
"default",
                                                                  // t06
                                                                  // t07
                                                                  // t08
  "default",
                                                                  // t09
  [4,5],
  ["a", "b", "c"],
                                                                  // t10
                                                                  // t11
],
["first",
  undef,
                                                                  // t01
                                                                  // t02
  undef.
```

```
undef,
                                                                       // t03
  "A",
                                                                        // t04
                                                                        // t05
   "orange",
                                                                        // t06
  "b",
  undef,
                                                                       // t07
  [1,2],
                                                                       // t08
   "ab",
                                                                       // t09
  [1,2,3],
                                                                        // t10
                                                                       // t11
],
["second",
  undef,
                                                                       // t01
  undef,
                                                                       // t02
  undef,
                                                                       // t03
                                                                       // t04
  "apple",
                                                                       // t05
  "a",
                                                                       // t06
  undef,
                                                                       // t07
                                                                       // t08
  [2,3],
  [1,2],
                                                                       // t09
                                                                        // t10
  [4,5,6],
                                                                       // t11
],
["last",
  undef,
                                                                       // t01
                                                                       // t02
  undef,
                                                                       // t03
  undef,
                                                                       // t04
   "g",
  "banana",
                                                                       // t05
                                                                       // t06
   "s",
                                                                       // t07
  undef.
                                                                       // t08
  [2,3],
                                                                       // t09
// t10
  [4,5],
["a","b","c"],
  15
                                                                       // t11
],
["nfirst_1",
                                                                       // t01
  undef,
                                                                       // t01
// t02
// t03
  undef,
  undef,
                                                                       // t04
// t05
// t06
// t07
  ["A"],
   ["orange"],
   ["b"],
  [undef],
                                                                       // t08
// t09
   [[1,2]],
   ["ab"],
                                                                       // t10
// t11
   [[1,2,3]],
  [0]
],
["nlast_1",
  undef,
                                                                       // t01
                                                                       // t02
  undef,
  undef,
                                                                       // t03
                                                                       // t04
   ["g"],
                                                                       // t05
// t06
   ["banana"],
   ["s"],
   [undef],
                                                                       // t07
   [[2,3]],
                                                                       // t08
   [[4,5]],
                                                                       // t09
   [["a","b","c"]],
                                                                        // t10
                                                                        // t11
  [15]
],
["nhead_1",
  undef,
                                                                       // t01
                                                                       // t02
  undef,
["A"," ","s","t","r","i","n"],
["orange","apple","grape"],
["b","a","n","a","n","a"],
                                                                       // t03
                                                                       // t04
                                                                       // t05
                                                                       // t06
  empty_v,
                                                                       // t07
  [[1,2]],
                                                                       // t08
  ["ab",[1,2],[2,3]],
[[1,2,3],[4,5,6],[7,8,9]],
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]
                                                                       // t09
                                                                       // t10
                                                                       // t11
["ntail_1",
  undef,
                                                                       // t01
                                                                       // t02
// t03
  undef,
  undef,
[" ","s","t","r","i","n","g"],
["apple","grape","banana"],
                                                                       // t04
// t05
```

```
["a", "n", "a", "n", "a", "s"],
                                                                                                                                                          // t06
      empty_v,
                                                                                                                                                          // 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
],
["rselect_02",
      undef,
                                                                                                                                                         // t01
      empty_v,
                                                                                                                                                          // t02
      undef,
["A"," ","s"],
                                                                                                                                                          // t03
                                                                                                                                                          // t04
      ["orange", "apple", "grape"], ["b", "a", "n"],
                                                                                                                                                          // t05
                                                                                                                                                          // t06
      undef,
                                                                                                                                                          // t07
      undef,
                                                                                                                                                          // t08
      ["ab",[1,2],[2,3]],
                                                                                                                                                          // t09
      [[1,2,3],[4,5,6],[7,8,9]],
                                                                                                                                                          // t10
                                                                                                                                                          // t11
      [0,1,2]
 ["eselect_F",
                                                                                                                                                          // t01
      undef.
                                                                                                                                                          // t02
      empty_v,
     undef,
["A"," ","s","t","r","i","n","g"],
                                                                                                                                                          // t03
                                                                                                                                                          // t04
      ["o", "a", "g", "b"],
["b", "a", "n", "a", "n", "a", "s"],
                                                                                                                                                         // t05
                                                                                                                                                          // t06
                                                                                                                                                          // t07
      [undef],
                                                                                                                                                          // t08
      [1,2],
      ["a",1,2,4],
                                                                                                                                                          // t09
      [1,4,7,"a"],
                                                                                                                                                          // t10
      skip
                                                                                                                                                          // t11
],
["eselect_L",
                                                                                                                                                          // t01
      undef.
                                                                                                                                                          // t02
      empty_v,
     "notate of the content of the c
                                                                                                                                                          // t03
                                                                                                                                                         // t04
                                                                                                                                                          // t05
                                                                                                                                                          // t06
                                                                                                                                                         // t07
// t08
       [2,3],
                                                                                                                                                         // t09
// t10
      ["b",2,3,5],
      [3,6,9,"c"],
      skip
                                                                                                                                                          // t11
],
["eselect_1",
                                                                                                                                                         // t01
     undef,
      empty_v,
                                                                                                                                                          // t02
                                                                                                                                                          // t03
      undef,
      skip,
                                                                                                                                                          // t04
                                                                                                                                                          // t05
      ["r", "p", "r", "a"],
      skip,
                                                                                                                                                          // t06
      [undef],
                                                                                                                                                          // t07
      [2,3],
                                                                                                                                                          // t08
      ["b",2,3,5],
                                                                                                                                                          // t09
      [2,5,8,"b"],
                                                                                                                                                          // t10
     skip
                                                                                                                                                          // t11
["smerge",
     undef,
                                                                                                                                                          // t01
      empty_v,
                                                                                                                                                          // t02
      [[0:0.5:9]],
                                                                                                                                                          // t03
      ["A string"],
                                                                                                                                                         // t04
      ["orange", "apple", "grape", "banana"],
["b", "a", "n", "a", "n", "a", "s"],
                                                                                                                                                         // t05
                                                                                                                                                         // t06
      [undef],
                                                                                                                                                         // t07
      [1,2,2,3],
                                                                                                                                                         // t08
      ["ab",1,2,2,3,4,5],
[1,2,3,4,5,6,7,8,9,"a","b","c"],
                                                                                                                                                         // t09
                                                                                                                                                         // t10
      [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                                                                                                         // t11
],
["pmerge",
                                                                                                                                                          // t01
      undef,
                                                                                                                                                          // t02
      empty_v,
                                                                                                                                                          // t03
      undef,
                                                                                                                                                          // t04
      ["A string"],
           ["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
                                                                      // t07
   undef,
  [[1,2],[2,3]],
[["a",1,2,4],["b",2,3,5]],
                                                                      // t08
                                                                      // t09
   [[1,4,7,"a"],[2,5,8,"b"],[3,6,9,"c"]],
                                                                      // t10
                                                                      // t11
  undef
  undef,
  empty_v,
                                                                      // t02
                                                                      // t03
  ["g", "n", "i", "r", "t", "s", " ", "A"],
["banana", "grape", "apple", "orange"],
["s", "a", "n", "a", "n", "a", "b"],
                                                                      // t04
                                                                      // t05
                                                                      // t06
   [undef],
                                                                      // t07
   [[2,3],[1,2]],
                                                                      // t08
  [[4,5],[2,3],[1,2],"ab"],
[["a","b","c"],[7,8,9],[4,5,6],[1,2,3]],
                                                                      // t09
                                                                      // t10
  [15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
                                                                      // t11
],
["gsort",
  undef,
                                                                      // t01
  empty_v,
                                                                      // t02
  undef,
                                                                      // t03
                                                                      // t04
  undef,
  ["apple", "banana", "grape", "orange"],
["a", "a", "a", "b", "n", "n", "s"],
                                                                      // t05
                                                                      // t06
  undef,
                                                                      // t07
                                                                      // t08
  undef,
  undef.
                                                                      // t09
  undef,
                                                                      // t10
  [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
                                                                      // t11
],
["qsort2_HR",
                                                                      // t01
  undef,
                                                                      // t02
// t03
  empty_v,
  undef.
                                                                      // t04
  undef.
  under,

["orange", "grape", "banana", "apple"],

["s", "n", "n", "b", "a", "a", "a"],
                                                                      // t05
                                                                      // t06
// t07
   [undef],
   [[3,2],[2,1]],
                                                                      // t08
// t09
   [[5,4],[3,2],[2,1],"ab"],
   [["c","b","a"],[9,8,7],[6,5,4],[3,2,1]],
                                                                      // t10
  [15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]
                                                                      // t11
],
["strip",
  undef,
                                                                      // t01
  empty_v,
                                                                      // t02
  undef,
["A"," ","s","t","r","i","n","g"],
                                                                      // t03
                                                                      // t04
   ["orange", "apple", "grape", "banana"],
["b", "a", "n", "a", "n", "a", "s"],
                                                                      // t05
                                                                      // t06
   [undef],
                                                                      // t07
                                                                      // t08
   [[1,2],[2,3]],
  ["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
["eappend_T0",
  undef,
                                                                      // t01
  [[0]],
                                                                      // t02
  undef,
                                                                      // t03
     ["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
                                                                      // t09
   [["ab",0],[1,2,0],[2,3,0],[4,5,0]],
   [[1,2,3,0],[4,5,6,0],[7,8,9,0],["a","b","c",0]],
                                                                      // 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",
    undef,
                                                                 // t01
    undef,
                                                                 // t02
    undef,
                                                                 // t03
                                                                 // t04
    ["orange",0,"apple","grape","banana"],
["b","a","n","a","n","a",0,"s"],
                                                                 // t05
                                                                 // t06
    undef,
                                                                 // t07
    [[1,2],0,[2,3]],
                                                                 // t08
     ["ab", [1,2],0,[2,3],[4,5]],
                                                                 // t09
                                                                 // t10
    [0,1,2,3,4,0,5,6,7,8,9,10,11,12,13,14,15]
                                                                 // t11
  ["delete_T0",
    undef,
                                                                 // t01
                                                                 // t02
    empty v.
    undef,
["A"," ","s","t","r","i","n","g"],
                                                                 // t03
                                                                 // t04
     ["orange", "grape", "banana"],
                                                                 // t05
     ["b", "a", "n", "a", "n", "a"],
                                                                 // t06
     [undef],
                                                                 // t07
     [[1,2]],
                                                                 // t08
                                                                 // t09
     ["ab",[1,2],[4,5]],
     [[1,2,3],[4,5,6],[7,8,9],["a","b","c"]],
                                                                 // t.10
    [0,1,2,3,4,6,7,8,9,10,11,12,13,14,15]
                                                                 // t11
  ["unique",
    undef.
                                                                 // t01
                                                                 // t.02
    empty_v,
    undef,
["A"," ","s","t","r","i","n","g"],
                                                                 // t03
                                                                 // t04
    ["orange", "apple", "grape", "banana"], ["b", "a", "n", "s"],
                                                                 // t05
                                                                 // t06
     [undef],
                                                                 // ±.07
     [[1,2],[2,3]],
                                                                 // ±08
     ["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
];
// 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 ) = table_get(test_r, test_c, vid, "tv");
module run_test( fname, fresult, vid )
  value_text = table_get(test_r, test_c, vid, "td");
  pass_value = table_get(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!!!
// create / convert
for (vid=test_ids) run_test( "consts", consts(get_value(vid)), vid );
for (vid=test_ids) run_test( "vstr", vstr(get_value(vid)), vid );
for (vid=test_ids) run_test( "sum", sum(get_value(vid)), 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( "defined_or_D", defined_or(get_value(vid), "default"), vid );
for (vid=test_ids) run_test( "edefined_or_DE3", edefined_or(get_value(vid),3,"default"),
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( "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( "rselect_02", rselect(get_value(vid), i=[0:2]), vid );
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),f=true), vid );
for (vid=test_ids) run_test( "eselect_L", eselect(get_value(vid),i=true), vid );
for (vid=test_ids) run_test( "eselect_1", eselect(get_value(vid),i=1), vid );
// not tested: ciselect()
// not tested: cmvselect()
// reorder
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( "reverse", reverse(get_value(vid)), vid);
for (vid=test ids) run test( "gsort", gsort(get value(vid)), vid );
for (vid=test_ids) run_test( "qsort2_HR", qsort2(get_value(vid), d=5, r=true), vid);
// grow / reduce
for (vid=test_ids) run_test( "strip", strip(get_value(vid)), vid );
for (vid=test_ids) run_test( "eappend_TO", eappend(0,get_value(vid)), vid );
for (vid=test_ids) run_test( "insert_TO", insert(0,get_value(vid),mv=["x","r","apple","s",[2,3]
,5]), vid);
for (vid=test_ids) run_test( "delete_T0", delete(get_value(vid), mv=["x", "r", "apple", "s", [2,3],5]),
vid );
for (vid=test_ids) run_test( "unique", unique(get_value(vid)), vid );
// end-of-tests
```

2.1.2.2 Validation Results

```
1 ECHO: "OpenSCAD Version [2016, 12, 21]"
2 ECHO: "[ INFO ] run_test(); t01 passed: 'consts(undef)=[]'"
### SECHO: "[ INFO ] run_test(); t02 passed: 'consts([])=[]'"

### ECHO: "[ INFO ] run_test(); t03 passed: 'consts([0: 0.5: 9])=[]'"
5 ECHO: "[ INFO ] run_test(); t04 passed: 'consts(A string)=[]'"
6 ECHO: "[ INFO ] run_test(); t05 passed: 'consts(["orange", "apple", "grape", "banana"])=[]'"
7 ECHO: "[ INFO ] run_test(); t06 passed: 'consts(["b", "a", "n", "a", "n", "a", "s"])=[]'"
8 ECHO: "[ INFO ] run_test(); t07 passed: 'consts([undef])=[]'"
8 ECHO: "[ INFO ] run_test(); t07 passed: 'consts([undef])=[]'"
9 ECHO: "[ INFO ] run_test(); t08 passed: 'consts([[1, 2], [2, 3]])=[]'"
10 ECHO: "[ INFO ] run_test(); t09 passed: 'consts(["ab", [1, 2], [2, 3], [4, 5]])=[]'"
11 ECHO: "[ INFO ] run_test(); t10 passed: 'consts([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[]'"
12 ECHO: "[ INFO ] run_test(); t11 passed: 'consts([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
         15])=[]'"
13 ECHO: "[ INFO ] run_test(); t01 passed: 'vstr(undef) = undef'"
14 ECHO: "[ INFO ] run_test(); t02 passed: 'vstr([])='"
15 ECHO: "[ INFO ] run_test(); t03 passed: 'vstr([0 : 0.5 : 9])=[0 : 0.5 : 9]'"
16 ECHO: "[ INFO ] run_test(); t04 passed: 'vstr(A string) = A string'"
17 ECHO: "[ INFO ] run_test(); t05 passed: 'vstr(["orange", "apple", "grape",
          "banana"])=orangeapplegrapebanana'"
18 ECHO: "[ INFO ] run_test(); t06 passed: 'vstr(["b", "a", "n", "a", "n", "a", "s"])=bananas'"
19 ECHO: "[ INFO ] run_test(); t07 passed: 'vstr([undef])=undef'"
20 ECHO: "[ INFO ] run_test(); t08 passed: 'vstr([[1, 2], [2, 3])]=[1, 2][2, 3]'"
21 ECHO: "[ INFO ] run_test(); t09 passed: 'vstr(["ab", [1, 2], [2, 3], [4, 5])]=ab[1, 2][2, 3][4, 5]'"
22 ECHO: "[ INFO ] run_test(); t10 passed: 'vstr([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1, 2,
          3][4, 5, 6][7, 8, 9]["a", "b", "c"]'"
23 ECHO: "[ INFO ] run_test(); t11 passed: 'vstr([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
         15])=0123456789101112131415'"
24 ECHO: "[ INFO ] run_test(); t01 passed: 'sum(undef)=undef'"
25 ECHO: "[ INFO ] run_test(); t02 passed: 'sum([])=0'"
26 ECHO: "[ INFO ] run_test(); t03 passed: 'sum([0:0.5:9])=85.5'"
27 ECHO: "[ INFO ] run_test(); t04 passed: 'sum(A string) = undef'"
28 ECHO: "[ INFO ] run_test(); t05 passed: 'sum(("orange", "apple", "grape", "banana"])=undef'"
29 ECHO: "[ INFO ] run_test(); t06 passed: 'sum(("b", "a", "n", "a", "n", "a", "s"])=undef'"
30 ECHO: "[ INFO ] run_test(); t07 passed: 'sum([undef])=undef'"
31 ECHO: "[ INFO ] run_test(); t08 passed: 'sum([[1, 2], [2, 3]])=[3, 5]'"
32 ECHO: "[ INFO ] run_test(); t09 passed: 'sum(["ab", [1, 2], [2, 3], [4, 5]])=undef'"
33 ECHO: "[ INFO ] run_test(); t10 passed: 'sum([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[undef,
          undef, undef]'"
```

```
34 ECHO: "[ INFO ] run_test(); t11 passed: 'sum([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=120'"
35 ECHO: "[ INFO ] run_test(); t01 passed: 'find_12(undef)=[]'"
36 ECHO: "[ INFO ] run_test(); t02 passed: 'find_12([])=[]'"
37 ECHO: "[ INFO ] run_test(); t03 passed: 'find_12([0 : 0.5 : 9])=[]'"
38 ECHO: "[ INFO ] run_test(); t04 passed: 'find_12(A string)=[]'"
39 ECHO: "[ INFO ] run_test(); t05 passed: 'find_12(["orange", "apple", "grape", "banana"])=[]'"
40 ECHO: "[ INFO ] run_test(); t06 passed: 'find_12(["b", "a", "n", "a", "n", "a", "s"])=[]'"
41 ECHO: "[ INFO ] run_test(); t07 passed: 'find_12([undef])=[]'"
42 ECHO: "[ INFO ] run_test(); t08 passed: 'find_12([[1, 2], [2, 3]])=[0]'"
43 ECHO: "[ INFO ] run_test(); t09 passed: 'find_12(["ab", [1, 2], [2, 3], [4, 5]])=[1]'"
44 ECHO: "[ INFO ] run_test(); t10 passed: 'find_12([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[]'"
45 ECHO: "[ INFO ] run_test(); t11 passed: 'find_12([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          15])=[]'"
46 ECHO: "[ INFO ] run_test(); t01 passed: 'count_S1(undef)=0'"
47 ECHO: "[ INFO ] run_test(); t02 passed: 'count_S1([])=0'"
48 ECHO: "[ INFO ] run_test(); t03 passed: 'count_S1([0 : 0.5 : 9])=0'"
49 ECHO: "[ INFO ] run_test(); t04 passed: 'count_S1(A string)=0'"
50 ECHO: "[ INFO ] run_test(); t05 passed: 'count_S1(["orange", "apple", "grape", "banana"])=0'"
51 ECHO: "[ INFO ] run_test(); t06 passed: 'count_S1(["b", "a", "n", "a", "n", "a", "s"])=0'"
52 ECHO: "[ INFO ] run_test(); t07 passed: 'count_S1([undef])=0'"
53 ECHO: "[ INFO ] run_test(); t08 passed: 'count_S1([1, 2], [2, 3]])=1'"
54 ECHO: "[ INFO ] run_test(); t09 passed: 'count_S1([[1, 2], [2, 3]])=1'"
55 ECHO: "[ INFO ] run_test(); t10 passed: 'count_S1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=1'"
56 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'"
57 ECHO: "[ INFO ] run_test(); t01 passed: 'exists_S1(undef)=false'
58 ECHO: "[ INFO ] run_test(); t02 passed: 'exists_S1([])=false'
59 ECHO: "[ INFO ] run_test(); t03 passed: 'exists_S1([0:0.5:9])=false'"
60 ECHO: "[ INFO ] run_test(); t04 passed: 'exists_S1(A string)=false'"
61 ECHO: "[ INFO ] run_test(); t04 passed: 'exists_S1(A string)=ralse'"
62 ECHO: "[ INFO ] run_test(); t05 passed: 'exists_S1(["orange", "apple", "grape", "banana"])=false'"
62 ECHO: "[ INFO ] run_test(); t06 passed: 'exists_S1(["b", "a", "n", "a", "n", "a", "s"])=false'"
63 ECHO: "[ INFO ] run_test(); t07 passed: 'exists_S1([undef])=false'"
64 ECHO: "[INFO] run_test(); t08 passed: 'exists_S1([[1, 2], [2, 3]])=true'"
65 ECHO: "[INFO] run_test(); t09 passed: 'exists_S1(["ab", [1, 2], [2, 3], [4, 5]])=true'"
66 ECHO: "[ INFO ] run_test(); t10 passed: 'exists_S1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
          "c"]])=true'"
67 ECHO: "[ INFO ] run_test(); tll passed: 'exists_S1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]) = true'"
68 ECHO: "[ INFO ] run_test(); t01 passed: 'defined_or_D(undef)=default'"
69 ECHO: "[ INFO ] run_test(); t02 passed: 'defined_or_D([])=[]'"
07 ECHO: [[INFO] run_test(); t02 passed: 'defined_or_D([]) = []'"
70 ECHO: "[INFO] run_test(); t03 passed: 'defined_or_D([0 : 0.5 : 9]) = [0 : 0.5 : 9]'"
71 ECHO: "[INFO] run_test(); t04 passed: 'defined_or_D(A string) = A string'"
72 ECHO: "[INFO] run_test(); t05 passed: 'defined_or_D(["orange", "apple", "grape", "banana"]) = ["orange", "apple", "grape", "banana"]'"
73 ECHO: "[ INFO ] run_test(); t06 passed: 'defined_or_D(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
           "a", "n", "a", "s"]'"
74 ECHO: "[ INFO ] run_test(); t07 passed: 'defined_or_D([undef])=[undef]'"
75 ECHO: "[ INFO ] run_test(); t08 passed: 'defined_or_D([[1, 2], [2, 3]])=[[1, 2], [2, 3]]'"
76 ECHO: "[ INFO ] run_test(); t09 passed: 'defined_or_D(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3], [4, 5]]'"
77 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"]]'"
78 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]'"
79 ECHO: "[ INFO ] run_test(); t01 passed: 'edefined_or_DE3(undef)=default'"
80 ECHO: "[ INFO ] run_test(); t02 passed: 'edefined_or_DE3([])=default'"
81 ECHO: "[ INFO ] run_test(); t03 passed: 'edefined_or_DE3([0 : 0.5 : 9]) = default'"
82 ECHO: "[ INFO ] run_test(); t04 passed: 'edefined_or_DE3(A string)=t'"
83 ECHO: "[ INFO ] run_test(); t05 passed: 'edefined_or_DE3(["orange", "apple", "grape", "banana"])=banana'"
84 ECHO: "[ INFO ] run_test(); t06 passed: 'edefined_or_DE3(["b", "a", "n", "a", "n", "a", "s"])=a'"
85 ECHO: "[ INFO ] run_test(); t07 passed: 'edefined_or_DE3([undef])=default'"
86 ECHO: "[ INFO ] run_test(); t08 passed: 'edefined_or_DE3([[1, 2], [2, 3]]) = default''
87 ECHO: "[ INFO ] run_test(); t09 passed: 'edefined_or_DE3(["ab", [1, 2], [2, 3], [4, 5]])=[4, 5]'"
88 ECHO: "[ INFO ] run_test(); t10 passed: 'edefined_or_DE3([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=["a", "b", "c"]'"
89 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'"
90 ECHO: "[ INFO ] run_test(); t01 passed: 'first(undef)=undef'"
91 ECHO: "[ INFO ] run_test(); t02 passed: 'first([])=undef'"
92 ECHO: "[ INFO ] run_test(); t03 passed: 'first([0:0.5:9])=undef'"
93 ECHO: "[ INFO ] run_test(); t04 passed: 'first(A string)=A'"
94 ECHO: "[INFO] run_test(); t05 passed: 'first(["orange", "apple", "grape", "banana"])=orange'"
95 ECHO: "[INFO] run_test(); t06 passed: 'first(["b", "a", "n", "a", "n", "a", "s"])=b'"
96 ECHO: "[INFO] run_test(); t07 passed: 'first([undef])=undef'"
97 ECHO: "[INFO] run_test(); t08 passed: 'first([[1, 2], [2, 3]])=[1, 2]'"
98 ECHO: "[ INFO ] run_test(); t09 passed: 'first(["ab", [1, 2], [2, 3], [4, 5]))=ab'"
99 ECHO: "[ INFO ] run_test(); t10 passed: 'first([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1, 2,
           3]'"
100 ECHO: "[ INFO ] run_test(); t11 passed: 'first([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=0'"
101 ECHO: "[ INFO ] run_test(); t01 passed: 'second(undef)=undef''
102 ECHO: "[ INFO ] run_test(); t02 passed: 'second([])=undef'"
```

```
103 ECHO: "[ INFO ] run_test(); t03 passed: 'second([0 : 0.5 : 9])=undef'"
104 ECHO: "[ INFO ] run_test(); t04 passed: 'second(A string) = '"
104 ECHO: "[INFO] run_test(); t04 passed: 'second(A string)= "
105 ECHO: "[INFO] run_test(); t05 passed: 'second(["orange", "apple", "grape", "banana"])=apple'"
106 ECHO: "[INFO] run_test(); t06 passed: 'second(["b", "a", "n", "a", "n", "a", "s"])=a'"
107 ECHO: "[INFO] run_test(); t07 passed: 'second([undef])=undef'"
108 ECHO: "[INFO] run_test(); t08 passed: 'second([[1, 2], [2, 3]])=[2, 3]'"
109 ECHO: "[INFO] run_test(); t09 passed: 'second(['ab", [1, 2], [2, 3], [4, 5]])=[1, 2]'"
110 ECHO: "[INFO] run_test(); t10 passed: 'second([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[4, 5,
            6]′"
111 ECHO: "[ INFO ] run_test(); t11 passed: 'second([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=1'"
112 ECHO: "[ INFO ] run_test(); t01 passed: 'last(undef)=undef'"
113 ECHO: "[ INFO ] run_test(); t02 passed: 'last([])=undef'"
                                  run_test(); t03 passed: 'last([0 : 0.5 : 9]) = undef'"
run_test(); t04 passed: 'last(A string) = g'"
run_test(); t05 passed: 'last(["orange", "apple", "grape", "banana"]) = banana'"
run_test(); t06 passed: 'last(["b", "a", "n", "a", "n", "a", "s"]) = s'"
run_test(); t07 passed: 'last([undef]) = undef'"
114 ECHO: "[ INFO ]
115 ECHO: "[ INFO ]
116 ECHO: "[ INFO ]
117 ECHO: "[ INFO ]
118 ECHO: "[ INFO ]
119 ECHO: "[ INFO ]
                                  run_test(); t08 passed: 'last([[1, 2], [2, 3]])=[2, 3]'"
120 ECHO: "[ INFO ] run_test(); t09 passed: 'last([1, 2], [2, 3], [4, 5])=[4, 5]'"

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

122 ECHO: "[ INFO ] run_test(); t11 passed: 'last([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])=15'"
123 ECHO: "[ INFO ] run_test(); t01 passed: 'nfirst_1(undef)=undef'
124 ECHO: "[ INFO ] run_test(); t02 passed: 'nfirst_1([])=undef'"
125 ECHO: "[ INFO ]
                                   run_test(); t03 passed: 'nfirst_1([0 : 0.5 : 9])=undef'"
126 ECHO: "[ INFO ] run_test(); t04 passed: 'nfirst_1(A string)=["A"]'
127 ECHO: "[ INFO ] run_test(); t05 passed: 'nfirst_1(["orange", "apple", "grape", "banana"])=["orange"]'"
128 ECHO: "[ INFO ] run_test(); t06 passed: 'nfirst_1(["b", "a", "n", "a", "n", "a", "s"])=["b"]'"
129 ECHO: "[ INFO ] run_test(); t07 passed: 'nfirst_1([undef])=[undef]'"
130 ECHO: "[ INFO ] run_test(); t08 passed: 'nfirst_1([[1, 2], [2, 3]])=[[1, 2]]'"
131 ECHO: "[ INFO ] run_test(); t09 passed: 'nfirst_1(["ab", [1, 2], [2, 3], [4, 5]])=["ab"]'"
132 ECHO: "[ INFO ] run_test(); t10 passed: 'nfirst_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1,
2, 3]]'"
133 ECHO: "[ INFO ] run_test(); tll passed: 'nfirst_1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            15])=[01'"
134 ECHO: "[ INFO ] run_test(); t01 passed: 'nlast_1(undef) = undef'"
135 ECHO: "[ INFO ] run_test(); t02 passed: 'nlast_1([])=undef'"
136 ECHO: "[ INFO ] run_test(); t03 passed: 'nlast_1([0 : 0.5 : 9])=undef'"
136 ECHO: [INFO] run_test(); t03 passed: 'nlast_1([0 : 0.3 : 5])-under
137 ECHO: "[INFO] run_test(); t04 passed: 'nlast_1([A string) = ["g"]'"
138 ECHO: "[INFO] run_test(); t05 passed: 'nlast_1(["orange", "apple", "grape", "banana"]) = ["banana"]'"
139 ECHO: "[INFO] run_test(); t06 passed: 'nlast_1(["b", "a", "n", "a", "n", "a", "s"]) = ["s"]'"
140 ECHO: "[ TNFO ]
                                   run_test(); t07 passed: 'nlast_1([undef]) = [undef]'"
140 ECHO: [ INFO ] fun_test(); to passed: infast_1([Info]; [ander]
141 ECHO: "[ INFO ] run_test(); to 8 passed: 'nlast_1([[1, 2], [2, 3]])=[[2, 3]]'"
142 ECHO: "[ INFO ] run_test(); to 9 passed: 'nlast_1(["ab", [1, 2], [2, 3], [4, 5]])=[[4, 5]]'"
143 ECHO: "[INFO] run_test(); t10 passed: 'nlast_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[["a", "b", "c"]]'"

144 ECHO: "[INFO] run_test(); t10 passed: 'nlast_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[["a", "b", "c"]]'"

144 ECHO: "[INFO] run_test(); t11 passed: 'nlast_1([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
            15])=[15]'"
145 ECHO: "[ INFO ] run_test(); t01 passed: 'nhead_1(undef)=undef'"
146 ECHO: "[ INFO ] run_test(); t02 passed: 'nhead_1([])=undef'"
147 ECHO: "[ INFO ] run_test(); t03 passed: 'nhead_1([0 : 0.5 : 9])=undef'"
148 ECHO: "[ INFO ] run_test(); t04 passed: 'nhead_1(A string)=["A", " ", "s", "t", "r", "i", "n"]'"
149 ECHO: "[ INFO ] run_test(); t05 passed: 'nhead_1(["orange", "apple", "grape", "banana"])=["orange",
            "apple", "grape"]'"
150 ECHO: "[INFO] run_test(); t06 passed: 'nhead_1(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a", "n", "a"]'"

151 ECHO: "[INFO] run_test(); t07 passed: 'nhead_1([undef])=[]'"
152 ECHO: "[INFO] run_test(); t08 passed: 'nhead_1([[1, 2], [2, 3]])=[[1, 2]]'"
153 ECHO: "[INFO] run_test(); t09 passed: 'nhead_1(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3]]'"
154 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]]'
155 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]''
156 ECHO: "[ INFO ] run_test(); t01 passed: 'ntail_1(undef)=undef'"
157 ECHO: "[ INFO ] run_test(); t02 passed: 'ntail_1([])=undef'"
158 ECHO: "[ INFO ] run_test(); t03 passed: 'ntail_1([0 : 0.5 : 9]) = undef'"

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

160 ECHO: "[ INFO ] run_test(); t05 passed: 'ntail_1(["orange", "apple", "grape", "banana"]) = ["apple", "grape",
            "banana"]'"
161 ECHO: "[ INFO ] run_test(); t06 passed: 'ntail_1(["b", "a", "n", "a", "n", "a", "s"])=["a", "n", "a", "n",
                     ..<sub>s"]</sub>/"
            "a",
162 ECHO: "[ INFO ] run_test(); t07 passed: 'ntail_1([undef])=[]'
163 ECHO: "[ INFO ] run_test(); t08 passed: 'ntail_1([[1, 2], [2, 3])]=[[2, 3]]'"

164 ECHO: "[ INFO ] run_test(); t09 passed: 'ntail_1([[ab", [1, 2], [2, 3], [4, 5]])=[[1, 2], [2, 3], [4, 5]]'"

165 ECHO: "[ INFO ] run_test(); t10 passed: 'ntail_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[4,
165 ECHO: [ [NEO ] full_test(); t10 passed: html:[1([1, 2, 3], [4, 3, 6], [7, 8, 9], [ a , b , c ]]

166 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]'"

167 ECHO: "[ INFO ] run_test(); t01 passed: 'rselect_02(undef)=undef'"
168 ECHO: "[ INFO ] run_test(); t02 passed: 'rselect_02([])=[]'"
169 ECHO: "[ INFO ] run_test(); t03 passed: 'rselect_02([0:0.5:9])=undef'"
```

```
170 ECHO: "[ INFO ] run_test(); t04 passed: 'rselect_02(A string)=["A", " ", "s"]'"
171 ECHO: "[ INFO ] run_test(); t05 passed: 'rselect_02(["orange", "apple", "grape", "banana"])=["orange",
          "apple", "grape"]'"
172 ECHO: "[INFO] run_test(); t06 passed: 'rselect_02(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n"]'"
173 ECHO: "[INFO] run_test(); t07 passed: 'rselect_02([undef])=undef'"
174 ECHO: "[INFO] run_test(); t08 passed: 'rselect_02([[1, 2], [2, 3]])=undef'"
175 ECHO: "[ INFO ] run_test(); t09 passed: 'rselect_02(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2,
          3]]'"
176 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]]'"
177 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]'"
178 ECHO: "[ INFO ] run_test(); t01 passed: 'eselect_F(undef)=undef'"
179 ECHO: "[ INFO ] run_test(); t02 passed: 'eselect_F([])=[]'"
180 ECHO: "[ INFO ] run_test(); t03 passed: 'eselect_F([0 : 0.5 : 9]) = undef'"

181 ECHO: "[ INFO ] run_test(); t04 passed: 'eselect_F(A string) = ["A", " ", "s", "t", "r", "i", "n", "g"]'"
182 ECHO: "[ INFO ] run_test(); t05 passed: 'eselect_F(["orange", "apple", "grape", "banana"])=["o", "a", "g",
          "b"]/"
183 ECHO: "[ INFO ] run_test(); t06 passed: 'eselect_F(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
          "a", "n", "a", "s"]'"
184 ECHO: "[ INFO ] run_test(); t07 passed: 'eselect_F([undef])=[undef]'"
185 ECHO: "[ INFO ] run_test(); t08 passed: 'eselect_F([[1, 2], [2, 3]])=[1, 2]'"
186 ECHO: "[ INFO ] run_test(); t09 passed: 'eselect_F(["ab", [1, 2], [2, 3], [4, 5]])=["a", 1, 2, 4]'"

187 ECHO: "[ INFO ] run_test(); t10 passed: 'eselect_F([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[1,
         4, 7, "a"]'
188 ECHO: "[ INFO ] run_test(); t11 *skip*: 'eselect_F(Vector of integers 0 to 15)'"
189 ECHO: "[ INFO ] run_test(); t01 passed: 'eselect_L(undef)=undef'
190 ECHO: "[ INFO ] run_test(); t02 passed: 'eselect_L([])=[]'"
191 ECHO: "[ INFO ] run_test(); t03 passed: 'eselect_L([0 : 0.5 : 9]) = undef'"

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

193 ECHO: "[ INFO ] run_test(); t05 passed: 'eselect_L(["orange", "apple", "grape", "banana"]) = ["e", "e", "e",
          "a"]'"
194 ECHO: "[ INFO ] run_test(); t06 passed: 'eselect_L(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
194 ECHO: "[ INFO ] run_test(); t06 passed: 'eselect_L(["D", "a", "n", "a", "1, a, s], -[ b, a, ..., "", "a", "n", "a", "s"]'"

195 ECHO: "[ INFO ] run_test(); t07 passed: 'eselect_L([undef])=[undef]'"

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

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

198 ECHO: "[ INFO ] run_test(); t10 passed: 'eselect_L([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[3, ""]'"
199 ECHO: "[ INFO ] run_test(); t11 *skip*: 'eselect_L(Vector of integers 0 to 15)'"
200 ECHO: "[ INFO ] run_test(); t01 passed: 'eselect_1(undef)=undef'"
201 ECHO: "[ INFO ] run_test(); t02 passed: 'eselect_1([])=[]'"
201 ECHO: "[ INFO ] run_test(); t03 passed: essetect_1([0 : 0.5 : 9]) = undef'"
203 ECHO: "[ INFO ] run_test(); t04 *skip*: 'eselect_1(A string)'"
205 ECHO: "[ INFO ] run_test(); t06 *skip*: 'eselect_1(Test vector 02)'"
206 ECHO: "[ INFO ] run_test(); t07 passed: 'eselect_1([undef])=[undef]'"
207 ECHO: "[ INFO ] run_test(); t08 passed: 'eselect_1([[1, 2], [2, 3]])=[2, 3]'"
208 ECHO: "[ INFO ] run_test(); t09 passed: 'eselect_1(["ab", [1, 2], [2, 3], [4, 5]])=["b", 2, 3, 5]'"
209 ECHO: "[ INFO ] run_test(); t10 passed: 'eselect_1([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[2, 5, 8, "b"]'"
210 ECHO: "[ INFO ] run_test(); t11 *skip*: 'eselect_1(Vector of integers 0 to 15)'"
211 ECHO: "[ INFO ] run_test(); t01 passed: 'smerge(undef)=undef'"
212 ECHO: "[ INFO ] run_test(); t02 passed: 'smerge([])=[]'"
213 ECHO: "[ INFO ] run_test(); t03 passed: 'smerge([0 : 0.5 : 9])=[[0 : 0.5 : 9]]'"
214 ECHO: "[ INFO ] run_test(); t04 passed: 'smerge(A string)=["A string"]'"
215 ECHO: "[ INFO ] run_test(); t05 passed: 'smerge(["orange", "apple", "grape", "banana"])=["orange", "apple",
          "grape", "banana"]'"
216 ECHO: "[ INFO ] run_test(); t06 passed: 'smerge(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
"n", "a", "s"]'"

217 ECHO: "[ INFO ] run_test(); t07 passed: 'smerge([undef])=[undef]'"

218 ECHO: "[ INFO ] run_test(); t08 passed: 'smerge([[1, 2], [2, 3]])=[1, 2, 2, 3]'"
219 ECHO: "[ INFO ] run_test(); t09 passed: 'smerge(("ab", [1, 2], [2, 3], [4, 5]])=("ab", 1, 2, 2, 3, 4, 5]'"
220 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"]'"
221 ECHO: "[ INFO ] run_test(); tll 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]"
222 ECHO: "[ INFO ] run_test(); t01 passed: 'pmerge(undef)=undef'"
223 ECHO: "[ INFO ] run_test(); t02 passed: 'pmerge([])=[]'"
224 ECHO: "[ INFO ] run_test(); t03 passed: 'pmerge([0:0.5:9])=undef'"
"n", "a", "s"]]'"

228 ECHO: "[INFO] run_test(); t07 passed: 'pmerge([undef])=undef'
229 ECHO: "[ INFO ] run_test(); t08 passed: 'pmerge([[1, 2], [2, 3]])=[[1, 2], [2, 3]]'"
230 ECHO: "[ INFO ] run_test(); t09 passed: 'pmerge(["ab", [1, 2], [2, 3], [4, 5]])=[["a", 1, 2, 4], ["b", 2,
          3, 5]]'"
231 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"]]'"
232 ECHO: "[ INFO ] run_test(); t11 passed: 'pmerge([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
          15])=undef'
233 ECHO: "[ INFO ] run_test(); t01 passed: 'reverse(undef)=undef'"
234 ECHO: "[ INFO ] run_test(); t02 passed: 'reverse([])=[]'"
235 ECHO: "[ INFO ] run_test(); t03 passed: 'reverse([0 : 0.5 : 9])=undef'"
236 ECHO: "[ INFO ] run_test(); t04 passed: 'reverse(A string)=["g", "n", "i", "r", "t", "s", " ", "A"]'"
237 ECHO: "[ INFO ] run_test(); t05 passed: 'reverse(["orange", "apple", "grape", "banana"])=["banana",
          "grape", "apple", "orange"]'"
238 ECHO: "[INFO] run_test(); t06 passed: 'reverse(["b", "a", "n", "a", "n", "a", "s"])=["s", "a", "n", "a", "b"]'"
"n", "a", "b"]'"

239 ECHO: "[ INFO ] run_test(); t07 passed: 'reverse([undef])=[undef]'"
240 ECHO: "[ INFO ] run_test(); t08 passed: 'reverse([[1, 2], [2, 3]])=[[2, 3], [1, 2]]'"
241 ECHO: "[ INFO ] run_test(); t09 passed: 'reverse(["ab", [1, 2], [2, 3], [4, 5]])=[[4, 5], [2, 3], [1, 2],
          "ab"]'"
242 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]]'"
243 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]'"
244 ECHO: "[ INFO ] run_test(); t01 passed: 'qsort(undef)=undef'"
245 ECHO: "[ INFO ] run_test(); t02 passed: 'qsort([])=[]''
246 ECHO: "[ INFO ] run_test(); t03 passed: 'qsort([0 : 0.5 : 9])=undef'"
247 ECHO: "[ INFO ] run_test(); t04 passed: 'qsort([ "orange", "apple", "grape", "banana"])=["apple", "banana",
          "grape", "orange"]'"
249 ECHO: "[ INFO ] run_test(); t06 passed: 'qsort(["b", "a", "n", "a", "n", "a", "s"])=["a", "a", "a", "b", "n", "n", "s"]'"
250 ECHO: "[ INFO ] run_test(); t07 passed: 'qsort([undef])=undef'"
256 ECHO: "[ INFO ] run_test(); t02 passed: 'qsort2_HR([])=[]
257 ECHO: "[ INFO ] run_test(); t03 passed: 'qsort2_HR([0 : 0.5 : 9])=undef'"
258 ECHO: "[ INFO ] run_test(); t04 passed: 'qsort2_HR(A string) = undef'"
259 ECHO: "[ INFO ] run_test(); t05 passed: 'qsort2_HR(["orange", "apple", "grape", "banana"]) = ["orange",
          "grape", "banana", "apple"]'"
260 ECHO: "[ INFO ] run_test(); t06 passed: 'qsort2_HR(["b", "a", "n", "a", "n", "a", "s"])=["s", "n", "n",
"b", "a", "a", "a", "a"]'"

261 ECHO: "[ INFO ] run_test(); t07 passed: 'qsort2_HR([undef])=[undef]'"

262 ECHO: "[ INFO ] run_test(); t08 passed: 'qsort2_HR([[1, 2], [2, 3]])=[[3, 2], [2, 1]]'"

263 ECHO: "[ INFO ] run_test(); t09 passed: 'qsort2_HR(["ab", [1, 2], [2, 3], [4, 5]])=[[5, 4], [3, 2], [2, 1],
          "ab"]'"
264 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]]'"
265 ECHO: "[ INFO ] run_test(); tll 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]'"
266 ECHO: "[ INFO ] run_test(); t01 passed: 'strip(undef)=undef'
267 ECHO: "[ INFO ] run_test(); t02 passed: 'strip([])=[]'"
268 ECHO: "[ INFO ] run_test(); t03 passed: 'strip([0:0.5:9])=undef'"
269 ECHO: "[ INFO ] run_test(); t04 passed: 'strip(A string)=["A", " ", "s", "t", "r", "i", "n", "g"]'"
270 ECHO: "[ INFO ] run_test(); t05 passed: 'strip(["orange", "apple", "grape", "banana"])=["orange", "apple",
          "grape", "banana"]'"
271 ECHO: "[ INFO ] run_test(); t06 passed: 'strip(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "a",
"n", "a", "s"]'"

272 ECHO: "[ INFO ] run_test(); t07 passed: 'strip([undef])=[undef]'"

273 ECHO: "[ INFO ] run_test(); t08 passed: 'strip([[1, 2], [2, 3]])=[[1, 2], [2, 3]]'"
274 ECHO: "[ INFO ] run_test(); t09 passed: 'strip(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [2, 3], [4,
          5]]'"
275 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"]]'"
276 ECHO: "[ INFO ] run_test(); tll 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]'"
277 ECHO: "[ INFO ] run_test(); t01 passed: 'eappend_T0(undef)=undef'
278 ECHO: "[ INFO ] run_test(); t02 passed: 'eappend_T0([])=[[0]]'"
279 ECHO: "[ INFO ] run_test(); t03 passed: 'eappend_T0([0 : 0.5 : 9]) = undef'"
280 ECHO: "[ INFO ] run_test(); t04 passed: 'eappend_T0(A string) = [["A", 0], [" ", 0], ["s", 0], ["t", 0],
["r", 0], ["i", 0], ["g", 0]]/"
281 ECHO: "[ INFO ] run_test(); t05 passed: 'eappend_T0(["orange", "apple", "grape", "banana"])=[["orange", 0],
     ["apple", 0], ["grape", 0], ["banana", 0]]'"

ECHO: "[INFO] run_test(); t06 passed: 'eappend_T0(["b", "a", "n", "a", "n", "a", "s"])=[["b", 0], ["a",
0], ["n", 0], ["a", 0], ["n", 0], ["a", 0], ["s", 0]]"

283 ECHO: "[INFO] run_test(); t07 passed: 'eappend_T0([undef])=[[undef, 0]]'"
284 ECHO: "[ INFO ] run_test(); t08 passed: 'eappend_T0([[1, 2], [2, 3]])=[[1, 2, 0], [2, 3, 0]]'"
285 ECHO: "[ INFO ] run_test(); t09 passed: 'eappend_T0(["ab", [1, 2], [2, 3], [4, 5]])=[["ab", 0], [1, 2, 0],
[2, 3, 0], [4, 5, 0]]'"

286 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]]'"

287 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]]'"
                         "[ INFO ] run_test(); t01 passed: 'insert_T0(undef)=undef'"
289 ECHO: "[ INFO ] run_test(); t02 passed: 'insert_T0([]) = undef'"
290 ECHO: "[ INFO ] run_test(); t03 passed: 'insert_T0([0 : 0.5 : 9])=undef'"
291 ECHO: "[ INFO ] run_test(); t04 passed: 'insert_T0(A string)=undef'"
292 ECHO: "[ INFO ] run_test(); t05 passed: 'insert_T0(["orange", "apple", "grape", "banana"])=["orange", 0,
                 "apple", "grape", "banana"]'"
 293 ECHO: "[ INFO ] run_test(); t06 passed: 'insert_T0(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
"a", "n", "a", 0, "s"]'"
294 ECHO: "[ INFO ] run_test(); t07 passed: 'insert_T0([undef]) = undef'"
 295 ECHO: "[ INFO ] run_test(); t08 passed: 'insert_T0([[1, 2], [2, 3]])=[[1, 2], 0, [2, 3]]'"
296 ECHO: "[ INFO ] run_test(); t09 passed: 'insert_T0(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], 0, [2, 3], [4, 5]]'"
 297 ECHO: "[ INFO ] run_test(); t10 passed: 'insert_T0([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b",
                 "c"]]) = undef'"
298 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]
299 ECHO:
                         "[ INFO ] run_test(); t01 passed: 'delete_T0(undef)=undef'"
 300 ECHO: "[ INFO ] run_test(); t02 passed: 'delete_T0([])=[]''
301 ECHO: "[ INFO ] run_test(); t03 passed: 'delete_T0([0 : 0.5 : 9])=undef'"
302 ECHO: "[ INFO ] run_test(); t04 passed: 'delete_T0(A string)=["A", " ", "s", "t", "r", "i", "n", "g"]'"
303 ECHO: "[ INFO ] run_test(); t05 passed: 'delete_T0(["orange", "apple", "grape", "banana"])=["orange",
                 "grape", "banana"]'"
 304 ECHO: "[ INFO ] run_test(); t06 passed: 'delete_T0(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n",
                            "n", "a"]/"
                 "a",
305 ECHO: "[ INFO ] run_test(); t07 passed: 'delete_T0([undef])=[undef]'"
306 ECHO: "[ INFO ] run_test(); t08 passed: 'delete_T0([[1, 2], [2, 3]])=[[1, 2]]'"
307 ECHO: "[ INFO ] run_test(); t09 passed: 'delete_T0(["ab", [1, 2], [2, 3], [4, 5]])=["ab", [1, 2], [4, 5]])"
308 ECHO: "[ INFO ] run_test(); t10 passed: 'delete_T0([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]])=[[1,
DUO LCHO: "[ INFO ] run_test(); t1U passed: 'delete_TU([[1, 2, 3], [4, 5, 6], [7, 8, 9], ["a", "b", "c"]]'

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

309 ECHO: "[ INFO ] run_test(); t11 passed: 'delete_T0([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])'=

310 ECHO: "[ INFO ] run_test(); t01 passed: 'unique(undef)=undef'"

311 ECHO: "[ INFO ] run_test(); t02 passed: 'unique([]) = []'"

312 ECHO: "[ INFO ] run_test(); t02 passed: 'unique([]) = []'"
 312 ECHO: "[ INFO ] run_test(); t03 passed: 'unique([0 : 0.5 : 9]) = undef'"
313 ECHO: "[ INFO ] run_test(); t04 passed: 'unique([a string)=["A", " ", "s", "t", "r", "i", "n", "g"]'"
314 ECHO: "[ INFO ] run_test(); t05 passed: 'unique(["orange", "apple", "grape", "banana"])=["orange", "apple",
                 "grape", "banana"]'"
 315 ECHO: "[ INFO ] run_test(); t06 passed: 'unique(["b", "a", "n", "a", "n", "a", "s"])=["b", "a", "n", "s"]'"
316 ECHO: [INFO] run_test(); t00 passed: unique([B, a, n, a, n, a, s])-[B, a, n, s]
316 ECHO: "[INFO] run_test(); t07 passed: 'unique([undef])=[undef]'"
317 ECHO: "[INFO] run_test(); t08 passed: 'unique([[1, 2], [2, 3]])-[[1, 2], [2, 3]]'"
318 ECHO: "[INFO] run_test(); t09 passed: 'unique(["ab", [1, 2], [2, 3], [4, 5]])-["ab", [2, 3], [4, 5]])-["ab
                5]]/"
319 ECHO: "[INFO] run_test(); tl0 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"]]'"

320 ECHO: "[INFO] run_test(); tl1 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]'"
```

2.2 Computations Validation

- · Point, Vector and Plane
- Bitwise Operations

2.2.1 Point, Vector and Plane

- · Validation Script
- · Validation Results

2.2.1.1 Validation Script

```
include <math.scad>;
use <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",
["crp", "Result precision",
["t01", "All undefined",
["t02", "All empty vector",
                                                   undef],
                                                   undef],
                                                    [undef, undef, undef, undef, undef]],
                                                    [empty_v,empty_v,
empty_v, empty_v, empty_v, empty_v]],
  ["t03", "All scalars",
["t04", "All 1D vectors",
["t05", "All 2D vectors",
                                                   [60, 50, 40, 30, 20, 10]],
[[99], [58], [12], [42], [15], [1]]],
                                                      [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]
                                                   ]],
  ["t08", "Orthogonal vectors",
                                                      +x_axis3d_uv, +
y_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,
                                                   ]]
1;
test_ids = table_get_row_ids( 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
  ["distance_pp",
     2,
                                                                        // fac
                                                                        // crp
     undef,
                                                                        // t01
     undef,
                                                                        // t02
                                                                        // t03
     undef,
     41,
                                                                        // t04
     43.3244,
                                                                        // t05
     106.2873,
                                                                        // t06
                                                                        // t07
     undef,
     1.4142,
                                                                        // t08
     1.4142
                                                                        // t09
   ["dot_vv",
                                                                        // fac
     4,
     4,
                                                                        // crp
                                                                        // t01
     undef,
                                                                        // t02
     undef,
     400,
                                                                        // t03
                                                                        // t04
     1392,
     1269,
                                                                        // t05
     17888,
                                                                        // t06
     22599,
                                                                        // t07
                                                                        // t08
     1,
                                                                        // t09
     -2
  ],
```

```
["cross_vv",
     4,
                                                                     // fac
                                                                     // crp
     4,
                                                                     // t01
     skip,
     skip,
                                                                     // t02
     skip,
                                                                     // t03
                                                                     // t04
     skip,
     917,
                                                                     // t05
     [2662,-11727,21929],
                                                                     // t06
    skip, [1,-1,1],
                                                                     // t07
                                                                     // t08
                                                                     // t09
     [0,0,-1]
  ],
["striple_vvv",
     6,
                                                                     // fac
     4,
                                                                     // crp
                                                                     // t01
// t02
     skip,
     skip,
                                                                     // t03
     skip,
     skip,
                                                                     // t04
     [-75981,14663],
                                                                     // t05
     199188,
                                                                     // t06
                                                                     // t07
// t08
     skip,
     8,
    0
                                                                     // t09
  ],
["angle_vv",
    4,
                                                                     // fac
                                                                    // crp
// t01
// t02
// t03
     4,
     undef,
     undef,
     undef.
                                                                    // t04
// t05
     undef,
     35.8525,
                                                                    // t06
// t07
     54.4261,
     undef,
                                                                     // t08
// t09
    60,
153.4350
  ],
["angle_vvn",
                                                                    // fac
// crp
// t01
// t02
     6,
     4,
     skip,
     skip,
                                                                     // t03
// t04
     skip,
     skip,
                                                                     // t05
// t06 (verify)
     skip,
     83.2771,
                                                                     // t07
// t08
     skip,
     90,
                                                                     // t09
     0
  ["unit_v",
     2,
                                                                     // fac
                                                                     // crp
// t01
     undef,
                                                                     // t02
// t03
     undef,
     undef,
                                                                     // t04
     [1],
     [0.9464,-0.3231],
                                                                     // t05
     [0.3857,-0.9032,-0.1882],
                                                                     // t06
                                                                     // t07
     [0.7071,-0.7071,0],
                                                                     // t08
     [0.7071, -0.7071, 0]
                                                                     // t09
  ["are_coplanar_vvv",
     6,
                                                                     // fac
     4,
                                                                     // crp
                                                                     // t01
     skip,
                                                                     // t02
// t03
     skip,
     skip,
     skip,
                                                                     // t04
     skip,
                                                                     // t05
                                                                     // t06
     false,
     skip,
                                                                     // t07
                                                                     // t08
// t09
     false.
     true
];
// 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 ) = table_get(test_r, test_c, vid, "tv");
function gv( vid, e ) = get_value( vid )[e];
module run( fname, vid )
  value_text = table_get(test_r, test_c, vid, "td");
  if ( table_get(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 = table_get(test_r, test_c, vid, "td");
  fname_argc = table_get(good_r, good_c, fname, "fac");
  comp_prcsn = table_get(good_r, good_c, fname, "crp");
 pass_value = table_get(good_r, good_c, fname, vid);
  test_pass = validate(cv=fresult, t="almost", ev=pass_value, p=comp_prcsn, pf=true);
  farg_text = vstr(eappend(", ", rselect(get_value(vid), [0:fname_argc-1]), r=false
, 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"] );
for (vid=run_ids) run("distance_pp",vid) test( "distance_pp", distance_pp(gv(vid,0),gv(vid
,1),gv(vid,2),gv(vid,3),gv(vid,4),gv(vid,5)), vid);
for (vid=run_ids) run("dot_vv",vid) test( "dot_vv", dot_vv(gv(vid,0),gv(vid,1),gv(vid,2),gv(vid
,3),gv(vid,4),gv(vid,5)), vid );
for (vid=run_ids) run("cross_vv", vid) test( "cross_vv", cross_vv(gv(vid,0),gv(vid,1),gv(vid,2
),gv(vid,3),gv(vid,4),gv(vid,5)), vid );
for (vid=run_ids) run("striple_vvv", vid) test( "striple_vvv", striple_vvv(gv(vid,0),gv(vid
,1),gv(vid,2),gv(vid,3),gv(vid,4),gv(vid,5)), vid);
for (vid=run_ids) run("angle_vv",vid) test( "angle_vv", angle_vv(gv(vid,0),gv(vid,1),gv(vid,2)
),gv(vid,3),gv(vid,4),gv(vid,5)), vid );
for (vid=run_ids) run("angle_vvn", vid) test( "angle_vvn", angle_vvn(gv(vid,0),gv(vid,1),gv(
vid, 2), gv(vid, 3), gv(vid, 4), gv(vid, 5)), vid);
for (vid=run_ids) run("unit_v",vid) test( "unit_v", unit_v(gv(vid,0),gv(vid,1),gv(vid,2),gv(vid
,3),gv(vid,4),gv(vid,5)), vid);
for (vid=run_ids) run("are_coplanar_vvv", vid) test( "are_coplanar_vvv",
are_coplanar_vvv(gv(vid,0),gv(vid,1),gv(vid,2),gv(vid,3),gv(vid,4),gv(vid,5)), vid );
// end-of-tests
```

2.2.1.2 Validation Results

```
1 ECHO: "OpenSCAD Version [2016, 12, 21]"
2 ECHO: "[INFO] run(): test(); t01 passed: 'distance_pp(undef, undef) = ~undef'"
3 ECHO: "[ INFO ] run(): test(); t02 passed: 'distance_pp([], [])=~undef'"
4 ECHO: "[ INFO ] run(): test(); t03 passed: 'distance_pp(60, 50)=~undef'"
5 ECHO: "[ INFO ]
                      run(): test(); t04 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])=~undef'"
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: 'dot_vv(undef, undef, undef, undef)=~undef'"
12 ECHO: "[ INFO ] run(): test(); t02 passed: 'dot_vv([], [], [], [])=~undef'
12 ECHO: "[ INFO ] run(): test(); t02 passed: 'dot_vv(60, 50, 40, 30)=~400'"

14 ECHO: "[ INFO ] run(): test(); t04 passed: 'dot_vv(199], [58], [12], [42])=~1392'"

15 ECHO: "[ INFO ] run(): test(); t05 passed: 'dot_vv(199, 2], [58, 16], [12, 43], [42, 13])=~1269'"

16 ECHO: "[ INFO ] run(): test(); t06 passed: 'dot_vv(199, 20, 55], [158, 116, 75], [12, 43, 90], [42, 13,
        34])=~17888'"
```

```
17 ECHO: "[ INFO ] run(): test(); t07 passed: 'dot_vv([169, 27, 35, 10], [178, 16, 25, 20], [12, 43, 90, 30],
[42, 13, 34, 60])=-22599'"

18 ECHO: "[ INFO ] run(): test(); t08 passed: 'dot_vv([1, 0, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0])=~1'"
19 ECHO: "[ INFO ] run(): test(); t09 passed: 'dot_vv([1, 0, 0], [0, 1, 0], [2, 2, 0], [0, 0, 0])=~-2'"
20 ECHO: "[ INFO ] run(); *skip*: t01 'cross_vv(All undefined)''
21 ECHO: "[ INFO ] run(); *skip*: t02 'cross_vv(All empty vector)'"
22 ECHO: "[ INFO ] run(); *skip*: t03 'cross_vv(All scalars)'"
23 ECHO: "[ INFO ] run(); *skip*: t04 'cross_vv(All 1D vectors)'"
24 ECHO: "[ INFO ] run(): test(); t05 passed: 'cross_vv([99, 2], [58, 16], [12, 43], [42, 13])=~917'"
25 ECHO: "[ INFO ] run(): test(); t06 passed: 'cross_vv([199, 20, 55], [158, 116, 75], [12, 43, 90], [42, 13,
           34])=~[2662, -11727, 21929]'"
26 ECHO: "[ INFO ] run(); *skip*: t07 'cross_vv(All 4D vectors)'"
27 ECHO: "[ INFO ] run(): test(); t08 passed: 'cross_vv([1, 0, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0])=~[1, -1,
          1]/"
28 ECHO: "[ INFO ] run(): test(); t09 passed: 'cross_vv([1, 0, 0], [0, 1, 0], [2, 2, 0], [0, 0, 0])=~[0, 0,
           -1]'"
29 ECHO: "[ INFO ] run(); *skip*: t01 'striple_vvv(All undefined)'"
30 ECHO: "[ INFO ] run(); *skip*: t02 'striple_vvv(All empty vector)'"
31 ECHO: "[ INFO ] run(); *skip*: t03 'striple_vvv(All scalars)'
32 ECHO: "[ INFO ] run(); *skip*: t04 'striple_vvv(All 1D vectors)'"
33 ECHO: "[ INFO ] run(): test(); t05 passed: 'striple_vvv([99, 2], [58, 16], [12, 43], [42, 13], [15, 59], [1, 85])=~[-75981, 14663]'"
34 ECHO: "[ INFO ] run(): test(); t06 passed: 'striple_vvv([199, 20, 55], [158, 116, 75], [12, 43, 90], [42,
          13, 34], [15, 59, 45], [62, 33, 69])=~199188'"
35 ECHO: "[ INFO ] run(); *skip*: t07 'striple_vvv(All 4D vectors)'"
36 ECHO: "[ INFO ] run(): test(); t08 passed: 'striple_vvv([1, 0, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0], [0, 0, 1], [-1, 0, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [0, 1, 0], [
          -1, 0], [0, 0, -1])=~8'"
37 ECHO: "[ INFO ] run(): test(); t09 passed: 'striple_vvv([1, 0, 0], [0, 1, 0], [2, 2, 0], [0, 0, 0], [0, 0,
0], [0, 0, 0])=~0'"

38 ECHO: "[ INFO ] run(): test(); t01 passed: 'angle_vv(undef, undef, undef, undef)=~undef'"
39 ECHO: "[ INFO ] run(): test(); t02 passed: 'angle_vv([], [], [], [])=~undef'"
40 ECHO: "[ INFO ] run(): test(); t03 passed: 'angle_vv(60, 50, 40, 30)=~undef'"
41 ECHO: "[ INFO ] run(): test(); t04 passed: 'angle_vv([99], [58], [12], [42])=~undef'"

42 ECHO: "[ INFO ] run(): test(); t05 passed: 'angle_vv([99, 2], [58, 16], [12, 43], [42, 13])=~35.8525'"
43 ECHO: "[ INFO ] run(): test(); t06 passed: 'angle_vv([199, 20, 55], [158, 116, 75], [12, 43, 90], [42, 13,
          34])=~54.4261'"
44 ECHO: "[ INFO ] run(): test(); t07 passed: 'angle_vv([169, 27, 35, 10], [178, 16, 25, 20], [12, 43, 90, 30], [42, 13, 34, 60])=~undef'"
45 ECHO: "[ INFO ] run(): test(); t08 passed: 'angle_vv([1, 0, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0])=~60'"
46 ECHO: "[ INFO ] run(): test(); t09 passed: 'angle_vv([1, 0, 0], [0, 1, 0], [2, 2, 0], [0, 0, 0])=~153.435'"
47 ECHO: "[ INFO ] run(); *skip*: t01 'angle_vvn(All undefined)'"
48 ECHO: "[ INFO ] run(); *skip*: t02 'angle_vvn(All empty vector)'"
49 ECHO: "[ INFO ] run(); *skip*: t03 'angle_vvn(All scalars)'"
50 ECHO: "[ INFO ] run(); *skip*: t04 'angle_vvn(All 1D vectors)'
51 ECHO: "[ INFO ] run(); *skip*: t05 'angle_vvn(All 2D vectors)'"
52 ECHO: "[ INFO ] run(): test(); t06 passed: 'angle_vvn([199, 20, 55], [158, 116, 75], [12, 43, 90], [42, 13,
34], [15, 59, 45], [62, 33, 69])=~83.2771'"

53 ECHO: "[ INFO ] run(); *skip*: t07 'angle_vvn(All 4D vectors)'"
54 ECHO: "[ INFO ] run(): test(); t08 passed: 'angle_vvn([1, 0, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0], [0, -1,
0], [0, 0, -1])=~90'"

55 ECHO: "[INFO] run(): test(); t09 passed: 'angle_vvn([1, 0, 0], [0, 1, 0], [2, 2, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0])=~0'"
56 ECHO: "[ INFO ] run(): test(); t01 passed: 'unit_v(undef, undef) =~undef'"
57 ECHO: "[ INFO ] run(): test(); t02 passed: 'unit_v([], [])=~undef'"
58 ECHO: "[ INFO ] run(): test(); t03 passed: 'unit_v(60, 50)=~undef'"
59 ECHO: "[ INFO ] run(): test(); t04 passed: 'unit_v([99], [58])=~[1]'"
60 ECHO: "[ INFO ] run(): test(); t05 passed: 'unit_v([99, 2], [58, 16])=~[0.9464, -0.3231]'"
61 ECHO: "[ INFO ] run(): test(); t06 passed: 'unit_v([199, 20, 55], [158, 116, 75])=~[0.3857, -0.9032,
           -0.1882]'"
62 ECHO: "[ INFO ] run(): test(); t07 passed: 'unit_v([169, 27, 35, 10], [178, 16, 25, 20])=~undef'"
63 ECHO: "[ INFO ] run(): test(); t08 passed: 'unit_v([1, 0, 0], [0, 1, 0])=~[0.7071, -0.7071, 0]'"
64 ECHO: "[ INFO ] run(): test(); t09 passed: 'unit_v([1, 0, 0], [0, 1, 0])=~[0.7071, -0.7071, 0]'"
65 ECHO: "[ INFO ] run(); *skip*: t01 'are_coplanar_vvv(All undefined)'"
66 ECHO: "[ INFO ] run(); *skip*: t02 'are_coplanar_vvv(All empty vector)'
67 ECHO: "[ INFO ] run(); *skip*: t03 'are_coplanar_vvv(All scalars)'"
68 ECHO: "[ INFO ] run(); *skip*: t04 'are_coplanar_vvv(All 1D vectors)'
69 ECHO: "[ INFO ] run(); *skip*: t05 'are_coplanar_vvv(All 2D vectors)'"
70 ECHO: "[ INFO ] run(): test(); t06 passed: 'are_coplanar_vvv([199, 20, 55], [158, 116, 75], [12, 43, 90],
[42, 13, 34], [15, 59, 45], [62, 33, 69])=~false'"
71 ECHO: "[ INFO ] run(); *skip*: t07 'are_coplanar_vvv(All 4D vectors)'"
72 ECHO: "[ INFO ] run(): test(); t08 passed: 'are_coplanar_vvv([1, 0, 0], [0, 1, 0], [0, 0, 1], [-1, 0, 0],
[0, -1, 0], [0, 0, -1])=~false'"

73 ECHO: "[ INFO ] run(): test(); t09 passed: 'are_coplanar_vvv([1, 0, 0], [0, 1, 0], [2, 2, 0], [0, 0, 0],
           [0, 0, 0], [0, 0, 0]) =~true'"
```

2.2.2 Bitwise Operations

Validation Script

· Validation Results

2.2.2.1 Validation Script

```
include <math_bitwise.scad>;
use <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 =
  ["fac", "Function argument count", undef], ["t01", "All undefined", [undef, u"], "102", "All empty vector", [empty_t"], [254, 0], ["t03", "test value 1", [254, 1], ["t05", "test value 3", [255, 0], ["t06", "test value 4", [0, 255], ["t07", "test value 5", [126, 63, 12], ["t09", "test value 6", [25, 10], ["t109", "test value 7", [1024, 5], ["t11", "test value 8", [4253, 3], ["t11", "test value 9", [835, 76], ["t12", "test value 10", [856, 62]]
                                                           [undef,undef]],
                                                            [empty_v,empty_v]],
                                                           [254, 0]],
[254, 1]],
                                                           [255, 0]],
                                                           [0, 255]],
                                                           [126, 63]],
[25, 10]],
                                                          [1024, 512]],
[4253, 315]],
[835, 769]],
[856, 625]]
];
test_ids = table_get_row_ids( 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,
                                                      // t01
      false,
                                                       // t02
                                                      // t03
      true,
      false,
                                                      // t04
      false,
                                                      // t05
      true,
                                                      // t06
      true,
                                                      // t07
                                                      // t08
      true,
                                                      // t09
      true,
                                                      // t10
      true,
      true,
                                                      // t11
                                                      // t12
      true
   ["bitwise_is_equal_1", 2,
      false,
                                                      // t01
                                                      // t02
      false,
                                                      // t03
      false,
      true,
                                                      // t04
                                                      // t05
      true,
                                                      // t06
      false,
                                                      // t07
      false,
      false,
                                                      // t08
                                                      // t09
      false.
                                                      // t10
      false,
                                                      // t11
      false.
                                                      // t12
      false
   ["bitwise_i2v", 1,
      undef,
                                                      // t01
```

```
// t02
  undef,
                                       // t03
// t04
  [1,1,1,1,1,1,1,0],
  [1,1,1,1,1,1,1,0],
                                       // t05
  [1,1,1,1,1,1,1,1],
  [1,0,0,0,0,1,0,0,1,1,1,0,1], // t10
                                       // t11
  [1,1,0,1,0,0,0,0,1,1],
  [1,1,0,1,0,1,1,0,0,0]
                                       // t12
["bitwise_i2v_v2i", 1,
                                       // t01
 undef,
                                       // t02
// t03
  undef,
  254,
                                       // t04
// t05
  254,
  255,
                                       // t06
// t07
  Ο,
  126,
                                       // t08
// t09
  25,
  1024,
                                       // t10
  4253,
  835,
                                       // t11
  856
                                       // t12
["bitwise_i2s", 1,
                                       // t01
 undef,
 undef, "11111110",
                                       // t02
// t03
 "11111110",
"111111111",
                                       // t04
                                       // t05
                                     // t06
// t07
 "0",
"1111110",
                                      // t08
// t09
  "11001",
  "10000000000",
  "1000010011101",
"1101000011",
"1101011000"
                                    // t10
// t11
// t12
["bitwise_i2s_s2i", 1,
                                       // t01
// t02
// t03
 undef,
  undef,
  254,
                                       // t03
// t04
// t05
// t06
// t07
  254,
  255,
  Ο,
  126,
                                       // t08
  25,
                                       // t09
// t10
// t11
  1024,
  4253,
  835,
  856
                                       // t12
["bitwise_and", 2,
 undef,
                                       // t01
  undef,
                                       // t02
                                       // t03
// t04
  Ο,
                                       // t05
// t06
  Ο,
  Ο,
                                       // t07
// t08
  62,
  8,
                                       // t09
  Ο,
                                       // t10
  769,
                                       // t11
                                       // t12
["bitwise_or", 2,
                                       // t01
 undef,
  undef,
                                       // t02
                                       // t03
// t04
  254,
  255,
                                       // t05
// t06
  255,
  255,
  127,
                                       // t07
                                       // t08
  27,
  1536,
                                       // t09
                                       // t10
  4543,
                                        // t11
  835,
                                        // t12
  889
```

```
],
["bitwise_xor", 2,
                                          // t01
    undef,
    undef,
                                          // t02
    254,
                                          // t03
                                          // t04
    255,
    255,
                                          // t05
    255,
                                          // t06
    65,
                                          // t07
                                          // t08
    19,
    1536,
                                          // t09
    4518,
                                          // t10
    66,
                                          // t11
                                          // t12
  ["bitwise_not", 1,
                                          // t01
    undef,
                                          // t02
    undef,
                                          // t03
    1,
                                          // t04
    1,
                                          // t05
// t06
    0,
    1.
                                          // t07
    1,
    6.
                                          // t08
    1023,
                                          // t09
    3938,
                                          // t10
                                          // t11
// t12
    188.
    167
  ],
["bitwise_lsh", 1,
    undef,
                                          // t01
                                          // t02
    undef.
                                          // t03
// t04
    508.
    508.
                                          // t05
    510,
                                          // t06
    0,
                                          // t07
// t08
    252,
    50.
                                          // t09
    2048.
                                          // t10
    8506,
                                          // t11
// t12
    1670,
    1712
  ["bitwise_rsh", 1,
                                         // t01
// t02
// t03
// t04
    undef,
    undef.
    127,
    127,
                                          // t05
// t06
// t07
// t08
    127,
    Ο,
    63,
    12,
    512,
                                          // t09
                                          // t10
    2126,
    417,
                                          // t11
    428
                                          // t12
// 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 ) = table_get(test_r, test_c, vid, "tv");
function gv( vid, e ) = get_value( vid )[e];
module run( fname, vid )
  value_text = table_get(test_r, test_c, vid, "td");
 if ( table_get(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 = table_get(test_r, test_c, vid, "td");
fname_argc = table_get(good_r, good_c, fname, "fac");
pass_value = table_get(good_r, good_c, fname, vid);
```

```
test_pass = validate(cv=fresult, t="equals", ev=pass_value, pf=true);
  farg_text = vstr(eappend(", ", rselect(get_value(vid), [0:fname_argc-1]), r=false,
  j=false, l=false));
  test_text = validate(str(fname, "(", farg_text, ")=", 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!!!
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(qv(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_and",vid) test( "bitwise_and", bitwise_and(gv(vid,0),gv(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(gv(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
```

2.2.2.2 Validation Results

```
1 ECHO: "OpenSCAD Version [2016, 12, 21]"
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'"
4 ECHO: "[ INFO ] 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'"
6 ECHO: "[ INFO ] 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'"
                     run(): test(); t07 passed: 'bitwise_is_equal_0(126, 63)=true'"
8 ECHO: "[ INFO ]
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(); tll 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, 1, 0]'"
29 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_i2v(254)=[1, 1, 1, 1, 1, 1, 1, 1, 1]'"
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, 1, 0]'"
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'
                   run(): test(); t02 passed: 'bitwise_i2v_v2i([]) = undef'"
39 ECHO: "[ INFO
40 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_i2v_v2i(254)=254'"
                   run(): test(); t04 passed: 'bitwise_i2v_v2i(254)=254'"
41 ECHO: "[
            INFO
                 ] run(): test(); t05 passed: 'bitwise_i2v_v2i(255)=255'"
42 ECHO: "[ INFO
43 ECHO: "[
            INFO
                 ] run(): test(); t06 passed: 'bitwise_i2v_v2i(0)=0'"
44 ECHO: "[ INFO
                 ] run(): test(); t07 passed: 'bitwise_i2v_v2i(126)=126'"
                 ] run(): test(); t08 passed: 'bitwise_i2v_v2i(25)=25'"
45 ECHO: "[
            INFO
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: "[ INFO
                 ] run(): test(); t01 passed: 'bitwise_i2s(undef)=undef'"
51 ECHO: "[
            TNFO
                 ] 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'"
                 ] run(): test(); t05 passed: 'bitwise_i2s_s2i(255)=255'"
66 ECHO: "[ INFO
67 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_i2s_s2i(0)=0'"
68 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_i2s_s2i(126)=126'
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'"
73 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_i2s_s2i(856)=856'"
74 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_and(undef, undef)=undef'"
75 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_and([], [])=undef'
76 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_and(254, 0)=0'
77 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_and(254, 1)=0'"
78 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_and(255, 0)=0'"
79 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_and(0, 255)=0'"
80 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_and(126, 63)=62'"
81 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_and(25, 10)=8'"
82 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_and(1024, 512)=0'"
83 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_and(4253, 315)=25'"
84 ECHO: "[ INFO ] run(): test(); tl1 passed: 'bitwise_and(835, 769)=769'"
85 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_and(856, 625)=592'"
86 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_or(undef, undef)=undef'"
87 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_or([], [])=undef'
88 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_or(254, 0)=254'"
89 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_or(254, 1)=255'"
90 ECHO: "[ INFO
                   run(): test(); t05 passed: 'bitwise_or(255, 0)=255'"
91 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_or(0, 255)=255'"
92 ECHO: "[ INFO
                   run(): test(); t07 passed: 'bitwise_or(126, 63)=127'"
93 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_or(25, 10)=27'"
94 ECHO: "[
                 run(): test(); t09 passed: 'bitwise_or(1024, 512)=1536'"
            INFO
95 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_or(4253, 315)=4543'"
                   run(): test(); t11 passed: 'bitwise_or(835, 769)=835'"
96 ECHO: "[ INFO
97 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_or(856, 625)=889'"
98 ECHO: "[ INFO
                   run(): test(); t01 passed: 'bitwise_xor(undef, undef) = undef'"
99 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_xor([], [])=undef'"
100 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_xor(254, 0)=254'"
101 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_xor(254, 1)=255'"
102 ECHO: "[ INFO
                    run(): test(); t05 passed: 'bitwise_xor(255,
103 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_xor(0, 255)=255'"
104 ECHO: "[ INFO ]
                    run(): test(); t07 passed: 'bitwise_xor(126, 63)=65'"
105 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_xor(25, 10)=19'"
106 ECHO: "[ INFO ]
                    run(): test(); t09 passed: 'bitwise_xor(1024, 512)=1536'"
107 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_xor(4253, 315)=4518'"
108 ECHO: "[ INFO ]
                    run(): test(); t11 passed: 'bitwise_xor(835, 769)=66'"
                    run(): test(); t12 passed: 'bitwise_xor(856, 625)=297'"
109 ECHO: "[ INFO
110 ECHO: "[ INFO ]
                    run(): test(); t01 passed: 'bitwise_not(undef) = undef'"
111 ECHO: "[ INFO ]
                    run(): test(); t02 passed: 'bitwise_not([]) = undef'
112 ECHO: "[ INFO ]
                    run(): test(); t03 passed: 'bitwise_not(254)=1'"
                    run(): test(); t04 passed: 'bitwise not(254)=1'"
          "[ INFO ]
113 ECHO:
114 ECHO: "[ INFO ]
                    run(): test(); t05 passed: 'bitwise_not(255)=0'"
115 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_not(0)=1'"
116 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_not(126)=1'"
117 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_not(25)=6'"
```

3 Todo List 39

```
118 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_not(1024)=1023'"
119 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_not(4253)=3938'"
120 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_not(835)=188'"
121 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_not(856)=167'"
122 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_lsh(undef)=undef'"
123 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_lsh([])=undef'"
124 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_lsh(254)=508'"
125 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_1sh(254)=508'"
126 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_lsh(255)=510'"
127 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_lsh(0)=0'"
128 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_lsh(126)=252'"
129 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_lsh(25)=50'"
130 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_lsh(1024)=2048'"
131 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_lsh(4253)=8506'"
132 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_lsh(835)=1670'"
133 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_lsh(856)=1712'"
134 ECHO: "[ INFO ] run(): test(); t01 passed: 'bitwise_rsh(undef)=undef'"
135 ECHO: "[ INFO ] run(): test(); t02 passed: 'bitwise_rsh([])=undef'"
136 ECHO: "[ INFO ] run(): test(); t03 passed: 'bitwise_rsh(254)=127'"
137 ECHO: "[ INFO ] run(): test(); t04 passed: 'bitwise_rsh(254)=127'"
138 ECHO: "[ INFO ] run(): test(); t05 passed: 'bitwise_rsh(255)=127'"
139 ECHO: "[ INFO ] run(): test(); t06 passed: 'bitwise_rsh(0)=0'"
140 ECHO: "[ INFO ] run(): test(); t07 passed: 'bitwise_rsh(126)=63'
141 ECHO: "[ INFO ] run(): test(); t08 passed: 'bitwise_rsh(25)=12'"
142 ECHO: "[ INFO ] run(): test(); t09 passed: 'bitwise_rsh(1024)=512'
143 ECHO: "[ INFO ] run(): test(); t10 passed: 'bitwise_rsh(4253)=2126'"
144 ECHO: "[ INFO ] run(): test(); t11 passed: 'bitwise_rsh(835)=417'"
145 ECHO: "[ INFO ] run(): test(); t12 passed: 'bitwise_rsh(856)=428'"
```

3 Todo List

globalScope> Global pyramid_q (size, center=false)

Support vertex rounding radius.

File resolution.scad

Review model for accuracy.

File shapes3d.scad

Complete rounded cylinder.

globalScope> Global tetrahedron (size, center=false)

Support vertex rounding radius.

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.

4 Module Index

4.1 Modules

Here is a list of all modules:

Constants	98
Euclidean	108
General	114
System	141
Data	96

	Console	92
	Datatable	97
	Мар	118
M	lath	122
	Bitwise Operations	88
	Point, Vector and Plane	125
	Triangle Solutions	145
	Variable Tests	154
	Vector Operations	165
	n-gon Solutions	178
Pa	arts	124
SI	hapes	137
	2D Extrusions	42
	2D Shapes	60
	3D Shapes	76
To	ools	142
	Edge Finishing	105
Tr	ransformations	143
	Extrusions	110
	Replications	130
Uı	nits	150
	Angle	85
	Length	115
	Resolution	132
Ut	tilities	151

5 File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

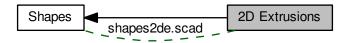
6 Module Documentation 41

console.scad Message logging functions	180
constants.scad Mechanical design constants	181
	101
mainpage.scad Documentation main page	183
map.scad	404
Mapped key-value pair data access	184
math.scad Mathematical functions	186
math_bitwise.scad Mathematical bitwise binary (base-two) functions	188
primitives.scad Mathematical primitive functions	190
resolution.scad Arc rendering resolution abstraction	194
shapes2d.scad Two-dimensional basic shapes	196
shapes2de.scad Linearly extruded two-dimensional basic shapes	198
shapes3d.scad	
Three-dimensional basic shapes	201
Data table encoding and lookup	202
tool_edge.scad Shape edge finishing tools	204
transform.scad	204
Shape transformation functions	206
units_angle.scad Angle units and conversions	208
units_length.scad Length units and conversions	209
utilities.scad Miscellaneous utilities	210
validation.scad Result validation functions	212
Module Documentation	

6.1 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 vp (v, h, vr, centroid=false, incenter=false, center=false)

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

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

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

• module etriangle_vl_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_lal (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_ala (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 aal (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_II (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 la (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.

6.1.1 Detailed Description

Extruded two dimensional geometric shapes.

6.1.2 Function Documentation

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

An extruded ellipse.

Parameters

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

See also

st_linear_extrude_scale for a description on specifying h.

Example

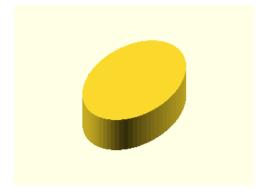


Figure 1: eellipse

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

Definition at line 767 of file shapes2de.scad.

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

An extruded ellipse with a removed elliptical core.

Parameters

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

See also

st_linear_extrude_scale 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

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

Definition at line 807 of file shapes2de.scad.

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

See also

st_linear_extrude_scale 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 3: eellipse_cs

```
1 eellipse_cs( size=[25,40], t=[10,5], a1=90, a2=180, co=[10,0], cr=45, h=20, center=true );
```

Definition at line 887 of file shapes2de.scad.

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

An extruded ellipse sector.

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

h	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>
a1	<decimal> The start angle in degrees.</decimal>
a2	<decimal> The stop angle in degrees.</decimal>
center	 boolean> Center about origin.

See also

st_linear_extrude_scale for a description on specifying h.

Example



Figure 4: eellipse_s

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

Definition at line 842 of file shapes2de.scad.

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

An extruded n-sided equiangular/equilateral regular polygon.

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

See also

st_linear_extrude_scale for a description on specifying h.

Example

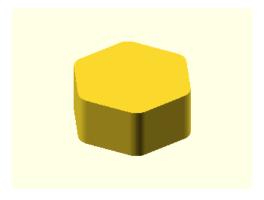


Figure 5: engon

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

See $\ensuremath{\mathtt{Wikipedia}}$ for more information.

Definition at line 737 of file shapes2de.scad.

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

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

size	<pre><vector decimal> A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal></pre>
h	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>
vr	<vector decimal="" =""> The corner rounding radius. A vector [v1r, v2r, v3r, v4r] of decimals or a single</vector>
	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 <i>fillet</i> and 1 for <i>chamfer</i> .
center	 boolean> Center about origin.

See also

st_linear_extrude_scale 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 110 of file shapes2de.scad.

6.1.2.7 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.

Parameters

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

See also

st_linear_extrude_scale 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

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

Definition at line 164 of file shapes2de.scad.

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

An extruded rhombus.

Parameters

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

See also

st_linear_extrude_scale for a description on specifying h.

Example

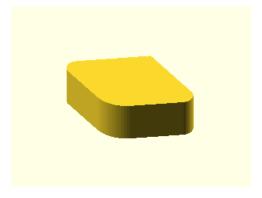


Figure 8: erhombus

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

Definition at line 213 of file shapes2de.scad.

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

An extruded two dimensional star.

Parameters

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

See also

st_linear_extrude_scale 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 926 of file shapes2de.scad.

```
6.1.2.10 module etriangle_aal ( 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	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>
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.
incenter	 boolean> Center incenter at origin.
center	 boolean> Center about origin.

See also

st_linear_extrude_scale for a description on specifying h.

Example



Figure 10: etriangle_aal

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

Definition at line 589 of file shapes2de.scad.

6.1.2.11 module etriangle_ala (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	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>

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

See also

st_linear_extrude_scale for a description on specifying h.

Example

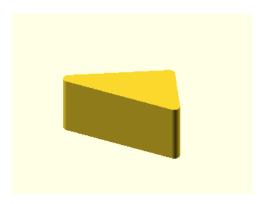


Figure 11: etriangle_ala

 $1 \\ \\ \text{etriangle_ala(a1=30, s=50, a2=60, h=20, vr=2, centroid=true, center=true);}$

Definition at line 537 of file shapes2de.scad.

6.1.2.12 module etriangle_la (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.

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>
h	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector 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>
center	<boolean> Center about origin.</boolean>

See also

st_linear_extrude_scale for a description on specifying h.

Example

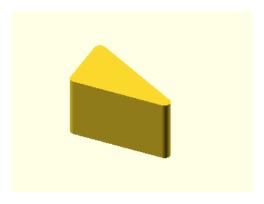


Figure 12: etriangle_la

etriangle_la(x=40, aa=30, h=20, vr=2, centroid=true, center=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 690 of file shapes2de.scad.

6.1.2.13 module etriangle_lal (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	<pre><decimal> The length of the side 1.</decimal></pre>
а	<decimal> The included angle in degrees.</decimal>
s2	<decimal> The length of the side 2.</decimal>
h	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>
Х	<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>
incenter	<boolean> Center incenter at origin.</boolean>
center	<boolean> Center about origin.</boolean>

See also

st_linear_extrude_scale for a description on specifying h.

Example

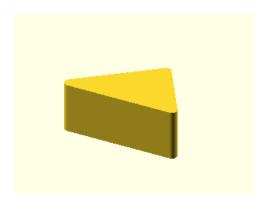


Figure 13: etriangle_lal

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

Definition at line 485 of file shapes2de.scad.

6.1.2.14 module etriangle_ll(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.

<decimal> The length of the side along the x-axis.</decimal>
<decimal> The length of the side along the y-axis.</decimal>
<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>
<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.</boolean>

See also

st_linear_extrude_scale for a description on specifying h.

Example

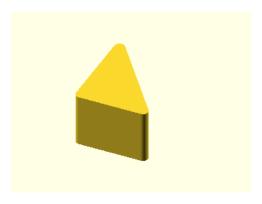


Figure 14: etriangle_ll

```
etriangle_11( x=30, y=40, h=20, vr=2, centroid=true, center=true );
```

Definition at line 638 of file shapes2de.scad.

```
6.1.2.15 module etriangle_III ( 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	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector 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>
center	<boolean> Center about origin.</boolean>

See also

st_linear_extrude_scale for a description on specifying h.

Example

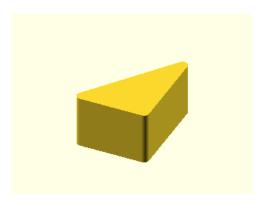


Figure 15: etriangle_III

1 etriangle_ll1(s1=30, s2=40, s3=50, h=20, vr=2, centroid=true, center=true);

Definition at line 338 of file shapes2de.scad.

6.1.2.16 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.

V1	<vector> A vector [x, y] for vertex 1.</vector>
v2	<vector> A vector [x, y] for vertex 2.</vector>
v3	<vector> A vector [x, y] for vertex 3.</vector>
h	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector 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.
center	<boolean> Center about origin.</boolean>

See also

st_linear_extrude_scale for a description on specifying h.

Example



Figure 16: etriangle_ppp

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

Definition at line 250 of file shapes2de.scad.

```
6.1.2.17 module etriangle_vI(v, h, vr, centroid = false, incenter = false, center = false)
```

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

Parameters

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

See also

st_linear_extrude_scale for a description on specifying h.

Example

```
t = triangle_lll2vp( 3, 4, 5 );
s = triangle_vp2vl( t );
etriangle_vl( v=s, h=5, vr=2 );
```

Definition at line 387 of file shapes2de.scad.

```
6.1.2.18 module etriangle_vl_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.

Parameters

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

See also

st_linear_extrude_scale for a description on specifying h.

Example



Figure 17: etriangle_vl_c

etriangle_vl_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 433 of file shapes2de.scad.

6.1.2.19 module etriangle_vp (v , h , vr , centroid = false, incenter = false, center = false)

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

V	<vector> A vector [v1, v2, v3] of vectors [x, y].</vector>
h	<vector decimal> A vector of decimals or a single decimal to specify simple extrusion height.</vector decimal>
vr	<vector decimal> The vertex rounding radius. A vector [v1r, v2r, v3r] of decimals or a single</vector 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

st_linear_extrude_scale for a description on specifying h.

Example

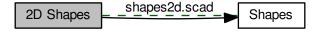
```
t = triangle_1112vp( 30, 40, 50 );
r = [2, 4, 6];
etriangle_vp( v=t, h=5, vr=r );
```

Definition at line 299 of file shapes2de.scad.

6.2 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_vp (v, vr, centroid=false, incenter=false)

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

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

A general triangle specified by its three side lengths.

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

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

• module triangle_vl_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_lal (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_ala (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_aal (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_II (x, y, vr, v1r, v2r, v3r, centroid=false, incenter=false)

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

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

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

6.2 2D Shapes 61

• 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.

6.2.1 Detailed Description

Two dimensional geometric shapes.

6.2.2 Function Documentation

6.2.2.1 module ellipse (size)

An ellipse.

Parameters

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

Example

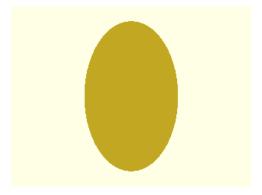


Figure 18: ellipse

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

Definition at line 1088 of file shapes2d.scad.

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

An ellipse with a removed elliptical core.

Parameters

size	<vector decimal> A vector [rx, ry] of decimals or a single decimal for (rx=ry).</vector decimal>
core	<pre><vector decimal> A vector [rx, ry] of decimals or a single decimal for (rx=ry).</vector decimal></pre>
t	<pre><vector decimal> A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal></pre>
со	<vector> Core offset. A vector [x, y] of decimals.</vector>
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

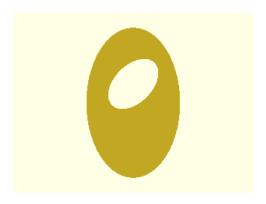


Figure 19: ellipse_c

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

Definition at line 1129 of file shapes2d.scad.

```
6.2.2.3 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	<vector decimal> A vector [rx, ry] of decimals or a single decimal for (rx=ry).</vector decimal>
core	<vector decimal> A vector [rx, ry] of decimals or a single decimal for (rx=ry).</vector decimal>
t	<vector decimal=""> A vector [x, y] of decimals or a single decimal for (x=y).</vector>
a1	<decimal> The start angle in degrees.</decimal>
a2	<decimal> The stop angle in degrees.</decimal>
со	<vector> Core offset. A vector [x, y] of decimals.</vector>
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

6.2 2D Shapes 63

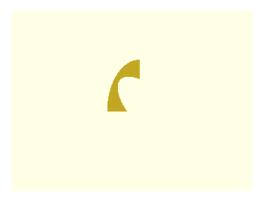


Figure 20: ellipse_cs

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

Definition at line 1239 of file shapes2d.scad.

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

An ellipse sector.

Parameters

size	<vector decimal> A vector [rx, ry] of decimals or a single decimal for (rx=ry).</vector decimal>
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 1171 of file shapes2d.scad.

6.2.2.5 module ngon (n , r , vr)

An n-sided equiangular/equilateral regular polygon.

Parameters

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

Example

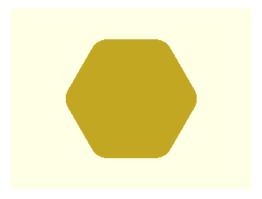


Figure 22: ngon

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

See Wikipedia for more information.

Definition at line 1054 of file shapes2d.scad.

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

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

Parameters

size	<pre><vector decimal> A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal></pre>
vr	<vector decimal="" =""> The corner rounding radius. A vector [v1r, v2r, v3r, v4r] of decimals or a single</vector>
	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>

Example

6.2 2D Shapes 65

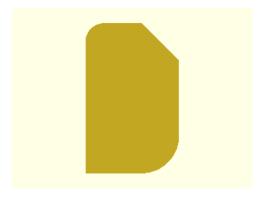


Figure 23: rectangle

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 112 of file shapes2d.scad.

6.2.2.7 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

<vector decimal> A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal>
<vector decimal> A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal>
<vector decimal> A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal>
<vector> Core offset. A vector [x, y] of decimals.</vector>
<decimal> Core z-rotation.</decimal>
<vector decimal="" =""> The default corner rounding radius. A vector [v1r, v2r, v3r, v4r] of decimals or</vector>
a single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.
<vector decimal> The outer corner rounding radius.</vector decimal>
<vector decimal> The core corner rounding radius.</vector decimal>
<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> .
<integer> The outer corner radius mode.</integer>
<integer> The core corner radius mode.</integer>
 boolean> Center about origin.

Thickness t

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

Example

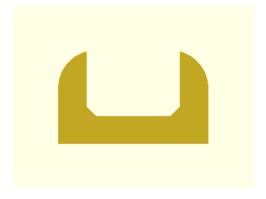


Figure 24: rectangle_c

```
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 232 of file shapes2d.scad.

```
6.2.2.8 module rhombus ( size , vr , center = false )
```

A rhombus.

Parameters

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

Example

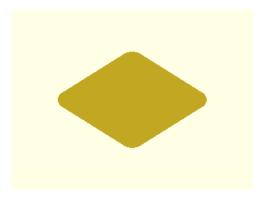


Figure 25: rhombus

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

See Wikipedia for more information.

Definition at line 297 of file shapes2d.scad.

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

A two dimensional star.

6.2 2D Shapes 67

Parameters

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

Example



Figure 26: star2d

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

Definition at line 1285 of file shapes2d.scad.

```
6.2.2.10 module triangle_aal ( 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.</decimal>
S	<decimal> The side length.</decimal>
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>

Example

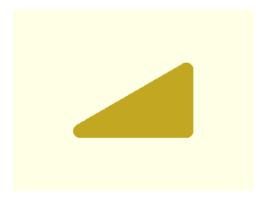


Figure 27: triangle_aal

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

See Wikipedia for more information.

Definition at line 882 of file shapes2d.scad.

6.2.2.11 module triangle_ala (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	<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.

Example



Figure 28: triangle_ala

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

6.2 2D Shapes 69

See Wikipedia for more information.

Definition at line 799 of file shapes2d.scad.

```
6.2.2.12 module triangle_la ( 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.

Example

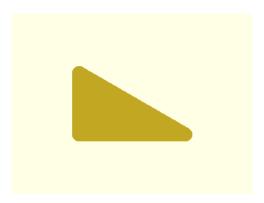


Figure 29: triangle_la

```
triangle_la( 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 1002 of file shapes2d.scad.

```
6.2.2.13 module triangle_lal ( 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	<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.
incenter	 boolean> Center incenter at origin.

Example

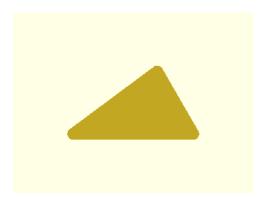


Figure 30: triangle_lal

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

See Wikipedia for more information.

Definition at line 730 of file shapes2d.scad.

```
6.2.2.14 module triangle_II ( 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.
incenter	<boolean> Center incenter at origin.</boolean>

Example

6.2 2D Shapes 71

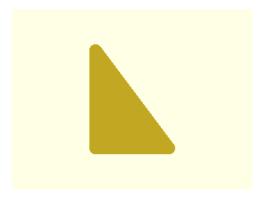


Figure 31: triangle_II

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

Definition at line 959 of file shapes2d.scad.

```
6.2.2.15 module triangle_III ( 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.
incenter	<boolean> Center incenter at origin.</boolean>

Example

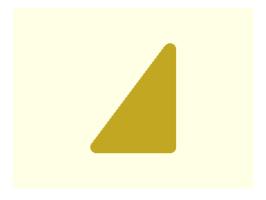


Figure 32: triangle_III

```
triangle_111( s1=30, s2=40, s3=50, vr=2, centroid=true );
```

See Wikipedia for more information.

Definition at line 524 of file shapes2d.scad.

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

A general triangle specified by three vertices.

Parameters

v1	<vector> A vector [x, y] for vertex 1.</vector>
v2	<vector> A vector [x, y] for vertex 2.</vector>
v3	<vector> A vector [x, y] for vertex 3.</vector>
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

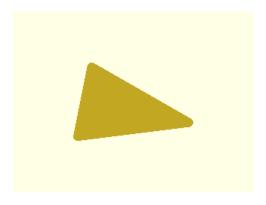


Figure 33: triangle_ppp

1 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 398 of file shapes2d.scad.

6.2.2.17 module triangle_vI (v , vr , centroid = false, incenter = false)

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

6.2 2D Shapes 73

Parameters

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

Example

```
t = triangle_1112vp( 3, 4, 5 );
s = triangle_vp2vl( t );
triangle_vl( v=s, vr=2, centroid=true );
```

Definition at line 584 of file shapes2d.scad.

```
6.2.2.18 module triangle_vl_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.

Parameters

VS	<pre><vector decimal> The size. A vector [s1, s2, s3] of decimals or a single decimal for (s1=s2=s3).</vector decimal></pre>
VC	<vector decimal> The core. A vector [s1, s2, s3] of decimals or a single decimal for (s1=s2=s3).</vector decimal>
со	<vector> Core offset. A vector [x, y] of decimals.</vector>
cr	<decimal> Core z-rotation.</decimal>
vr	<vector decimal> The default vertex rounding radius. A vector [v1r, v2r, v3r] of decimals or a</vector decimal>
	single decimal for (v1r=v2r=v3r).
vr1	<vector decimal> The outer vertex rounding radius.</vector decimal>
vr2	<vector decimal> The core vertex rounding radius.</vector decimal>
centroid	 boolean> Center centroid at origin.
incenter	 boolean> Center incenter at origin.

Example



Figure 34: triangle_vl_c

```
triangle_vl_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 638 of file shapes2d.scad.

6.2.2.19 module triangle_vp (v , vr , centroid = false, incenter = false)

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

6.2 2D Shapes 75

Parameters

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

Example

```
t = triangle_1112vp( 30, 40, 50 );
r = [2, 4, 6];
triangle_vp( v=t, vr=r );
```

Definition at line 474 of file shapes2d.scad.

6.3 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 tetrahedron (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.

6.3.1 Detailed Description

Three dimensional geometric shapes.

6.3 3D Shapes 77

6.3.2 Function Documentation

6.3.2.1 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 103 of file shapes3d.scad.

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

A cuboid with edge, fillet, or chamfer corners.

Parameters

size	<pre><vector decimal> A vector [x, y, z] of decimals or a single decimal for (x=y=z).</vector decimal></pre>
vr	<decimal> The 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	 doublean> Center about origin.

Example



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 168 of file shapes3d.scad.

6.3.2.3 module ellipsoid (size)

An ellipsoid.

Parameters

-:	cup atom de aireals. A supertou for left de aireada au a aireala de airead fou (su les
size	<pre><vector decimal> A vector [w, h] of decimals or a single decimal for (w=h).</vector decimal></pre>
0.20	trooter accumally it rooter [11, 11] or accumality or a configuration (11 11).

Example

6.3 3D Shapes 79

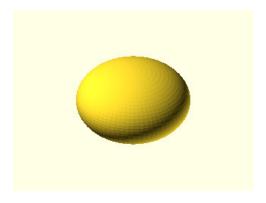


Figure 37: ellipsoid

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

Definition at line 253 of file shapes3d.scad.

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

A sector of an ellipsoid.

Parameters

size <vector decimal> A vector [w, h] of decimals or a single decimal for (w=h).</vector decimal>		<vector decimal> A vector [w, h] of decimals or a single decimal for (w=h).</vector decimal>
	a1	<decimal> The start angle in degrees.</decimal>
Ī	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 285 of file shapes3d.scad.

6.3.2.5 module pyramid_q (size , center = false)

A pyramid with quadrilateral base.

Parameters

size	<pre><vector decimal> A vector [x, y, z] of decimals or a single decimal for (x=y=z).</vector 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 387 of file shapes3d.scad.

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

A three dimensional star.

Parameters

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

Example



Figure 40: star3d

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

6.3 3D Shapes 81

Definition at line 433 of file shapes3d.scad.

6.3.2.7 module tetrahedron (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

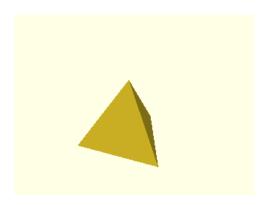


Figure 41: tetrahedron

1 tetrahedron(size=20, center=true);

Todo Support vertex rounding radius.

Definition at line 343 of file shapes3d.scad.

6.3.2.8 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.

Parameters

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

st_rotate_extrude_elongate 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

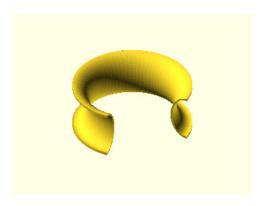


Figure 42: torus_ep

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

Definition at line 654 of file shapes3d.scad.

```
6.3.2.9 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.

Parameters

size	<vector decimal> The profile size. A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal>
core	<pre><vector decimal> The profile core. A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal></pre>
r	<decimal> The rotation radius.</decimal>
1	<pre><vector decimal> The elongation length. A vector [x, y] of decimals or a single decimal for (x=y)</vector decimal></pre>
t	<vector decimal> The profile thickness. A vector [x, y] of decimals or a single decimal for (x=y).</vector decimal>
со	<vector> Core offset. A vector [x, y] of decimals.</vector>

6.3 3D Shapes 83

cr	<decimal> Core z-rotation.</decimal>
vr	<vector decimal> The profile default corner rounding radius. A vector [v1r, v2r, v3r, v4r] of</vector decimal>
	decimals or a single decimal for (v1r=v2r=v3r=v4r). Unspecified corners are not rounded.
vr1	<vector decimal> The profile outer corner rounding radius.</vector decimal>
vr2	<vector decimal> The profile core corner rounding radius.</vector 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.
profile	 boolean> Show profile only (do not extrude).

See also

st_rotate_extrude_elongate 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

```
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 516 of file shapes3d.scad.

```
6.3.2.10 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	<vector decimal> The size. A vector [s1, s2, s3] of decimals or a single decimal for (s1=s2=s3).</vector decimal>
core	<vector decimal> The core. A vector [s1, s2, s3] of decimals or a single decimal for (s1=s2=s3).</vector decimal>
r	<decimal> The rotation radius.</decimal>
1	<pre><vector decimal> The elongation length. A vector [x, y] of decimals or a single decimal for (x=y)</vector decimal></pre>
со	<vector> Core offset. A vector [x, y] of decimals.</vector>
cr	<decimal> Core z-rotation.</decimal>
vr	<pre><vector decimal> The default vertex rounding radius. A vector [v1r, v2r, v3r] of decimals or a</vector decimal></pre>
	single decimal for (v1r=v2r=v3r).
vr1	<vector decimal> The outer vertex rounding radius.</vector decimal>
vr2	<vector decimal> The core vertex rounding radius.</vector 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.
incenter	
profile	 boolean> Show profile only (do not extrude).

See also

st_rotate_extrude_elongate for description of extrude parameters.

Example



Figure 44: torus_tp

1 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.

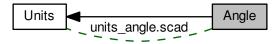
Definition at line 587 of file shapes3d.scad.

6.4 Angle 85

6.4 Angle

Angle units and conversions.

Collaboration diagram for Angle:



Files

· file units_angle.scad

Angle units and conversions.

Functions

• function unit_angle_name (units=base_unit_angle)

Return the name of the given angle unit identifier.

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

Convert the angle from from units to to units.

Variables

• base unit angle = "d"

<string> Base unit for angle measurements.

6.4.1 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 unit identifiers and their descriptions.

units id	description
r	radian
d	degree
dms	degree, minute, second

Example

```
include <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 = \mathbf{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]
```

6.4.2 Function Documentation

6.4.2.1 function convert_angle (angle , from = base_unit_angle, to = base_unit_angle)

Convert the angle from from units to to units.

Parameters

angle	angle <decimal vector> An angle to convert (dms angles are 3-tuple vector [d, m, s]).</decimal vector>	
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|vector> The conversion result (dms angles are 3-tuple vector [d, m, s]). Returns 'undef' for identifiers
that are not defined.

6.4.2.2 function unit_angle_name (units = base_unit_angle)

Return the name of the given angle unit identifier.

6.4 Angle 87

Parameters

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

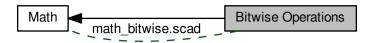
Returns

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

6.5 Bitwise Operations

Bitwise binary (base-two) operations.

Collaboration diagram for Bitwise Operations:



Files

• file math_bitwise.scad

Mathematical bitwise binary (base-two) 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 vector of bits.

function bitwise_v2i (v)

Decode a base-two vector 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_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.

6.5.1 Detailed Description

Bitwise binary (base-two) operations.

See Wikipedia binary numbers and operations for more information.

See validation results.

6.5.2 Function Documentation

6.5.2.1 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.

6.5.2.2 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

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

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

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

Parameters

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

Returns

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

6.5.2.4 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	 bit> The bit test value [0 1].

Returns

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

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

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

Parameters

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.

6.5.2.6 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.

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

Base-two bitwise OR 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 OR of v1 and v2. Returns undef when v1 or v2 is not an integer.

6.5.2.8 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.

6.5.2.9 function bitwise_s2i (v)

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

Parameters

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

Returns

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

6.5.2.10 function bitwise_v2i (v)

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

Parameters

Γ	V	<vector> A value encoded as a base-two vector of bits.</vector>
	=	

Returns

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

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

Base-two bitwise XOR operation for integers.

Parameters

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

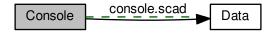
Returns

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

6.6 Console

Console message logging.

Collaboration diagram for Console:



Files

· file console.scad

Message logging functions.

Functions

• 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.

6.6.1 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 );

// information
log_info( message );

// warning
```

6.6 Console 93

```
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()"
7 ECHO: "# [ WARNING ] console log message #"
9 ECHO:
10 ECHO: "root()"
11 ECHO: "###################################
12 ECHO: "###################################
13 ECHO: "##
14 ECHO: "## [ ERROR ] console log message ##"
15 ECHO: "##
16 ECHO: "###################################
17 ECHO: "###################################
```

6.6.2 Function Documentation

6.6.2.1 module log_debug (m)

Output diagnostic message to console.

Parameters

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

When \$log_debug == true, message is written to the console. When false, output is not generated.

Definition at line 77 of file console.scad.

```
6.6.2.2 module log_echo ( m )
```

Output message to console.

Parameters

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

Definition at line 61 of file console.scad.

```
6.6.2.3 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 138 of file console.scad.

```
6.6.2.4 module log_info ( m )
```

Output information message to console.

Parameters

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

Definition at line 94 of file console.scad.

6.6.2.5 module log_warn (m)

Output warning message to console.

Parameters

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

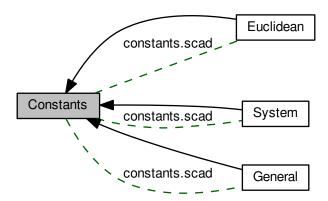
Definition at line 109 of file console.scad.

6.7 Constants 95

6.7 Constants

General design constants.

Collaboration diagram for Constants:



Modules

• Euclidean

Euclidean 2D/3D space mapping.

General

General design constants.

• System

System/Program limits.

Files

· file constants.scad

Mechanical design constants.

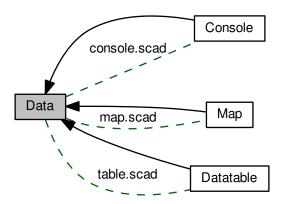
6.7.1 Detailed Description

General design constants.

6.8 Data

Data values and reference.

Collaboration diagram for Data:



Modules

• Console

Console message logging.

Datatable

Data table encoding and lookup.

• Map

Mapped data access via key-value pairs.

Files

· file console.scad

Message logging functions.

• file map.scad

Mapped key-value pair data access.

• file table.scad

Data table encoding and lookup.

6.8.1 Detailed Description

Data values and reference.

6.9 Datatable 97

6.9 Datatable

Data table encoding and lookup.

Collaboration diagram for Datatable:



Files

· file table.scad

Data table encoding and lookup.

Functions

function table_get_row_idx (rows, row_id)

Get the index for a table row identifier.

function table_get_row (rows, row_id)

Get the row for a table row identifier.

function table_get_col_idx (cols, col_id)

Get the index for a table column identifier.

function table_get_col (cols, col_id)

Get the column for a table column identifier.

function table_get (rows, cols, row_id, col_id)

Get the value for a table row and column identifier.

function table_get_row_cols (rows, cols, col_id)

Form a vector from the specified column of each table row.

function table_get_row_ids (rows)

Form a vector of each table row identifier.

function table_exists (rows, cols, row_id, col_id)

Test the existence of a table row and column identifier.

function table_size (rows, cols)

Get the size of a table.

• module table_check (rows, cols, verbose=false)

Perform some basic validation/checks on a table.

module table_dump (rows, cols, rows_sel, cols_sel, number=true)

Dump a table to the console.

function table_copy (rows, cols, rows_sel, cols_sel)

Create a copy of select rows and columns of a table.

• function table_sum (rows, cols, rows_sel, cols_sel)

Sum select rows and columns of a table.

6.9.1 Detailed Description

Data table encoding and lookup.

Example

```
use
          <table.scad>;
base unit length = "mm";
table_cols =
[ // id, description
  ["id",
            "row identifier"],
   ["ht",
             "head type [r|h|s]"],
  ["td",
["t1",
            "thread diameter"],
            "thread length"],
   ["hd", "head diameter"],
   ["hl",
             "head length"],
           "hex nut flat-to-flat width"],
   ["nd",
   ["nl", "hex nut length"]
1:
table_rows =
  // id, ht, td, tl, hd, hl, nd, nl
["m3r08r", "r", 3.000, 8.00, 5.50, 3.000, 5.50, convert_length(1.00, "in")],
["m3r14r", "r", 3.000, 14.00, 5.50, 3.000, 5.50, convert_length(1.25, "in")],
["m3r16r", "r", 3.000, 16.00, 5.50, 3.000, 5.50, convert_length(1.50, "in")],
["m3r20r", "r", 3.000, 20.00, 5.50, 3.000, 5.50, convert_length(1.75, "in")]
table_check( table_rows, table_cols, true );
table_dump( table_rows, table_cols );
m3r16r_tl = table_get( table_rows, table_cols, "m3r16r", "tl" );
if ( table_exists( cols=table_cols, col_id="nl" ) )
  echo ( "metric 'nl' available" );
table_ids = table_get_row_ids( table_rows );
table_cols_tl = table_get_row_cols( table_rows, table_cols, "tl" );
echo ( table_ids=table_ids );
echo ( table_cols_tl=table_cols_tl );
tnew = table_copy( table_rows, table_cols, cols_sel=["tl", "nl"] );
tsum = table_sum( table_rows, table_cols, cols_sel=["t1", "n1"] );
echo ( m3r16r_t1=m3r16r_t1 );
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]"
                                                        = [3]"
12 ECHO: "[m3r08r] [td] (thread diameter)
13 ECHO: "[m3r08r] [tl] (thread length)
                                                       = [8]"
14 ECHO: "[m3r08r] [hd] (head diameter)
                                                       = [5.5]"
15 ECHO: "[m3r08r] [h1] (head length)
                                                       = [3]"
16 ECHO: "[m3r08r] [nd] (hex nut flat-to-flat width) = [5.5]"
17 ECHO: "[m3r08r] [nl] (hex nut length)
                                                       = [25.4]"
18 ECHO: ""
19 ECHO: "row: 1"
20 ECHO: "[m3r14r] [id] (row identifier)
                                                       = [m3r14r]"
21 ECHO: "[m3r14r] [ht] (head type [rh|s])
22 ECHO: "[m3r14r] [td] (thread diameter)
                                                       = [r]"
                                                       = [3]"
23 ECHO: "[m3r14r] [t1] (thread length)
                                                        = [14]"
```

6.9 Datatable 99

```
24 ECHO: "[m3r14r] [hd] (head diameter)
25 ECHO: "[m3r14r] [hl] (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"
33 ECHO: "[m3r16r] [tl] (thread length)
                                                       = [16]"
34 ECHO: "[m3r16r] [hd] (head diameter)
35 ECHO: "[m3r16r] [h1] (head length)
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]"
42 ECHO: "[m3r20r] [td] (thread diameter)
                                                        = [3]"
43 ECHO: "[m3r20r] [t1] (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 ECHO: tsum = [58, 139.7]
```

6.9.2 Function Documentation

6.9.2.1 module table_check (rows , cols , verbose = false)

Perform some basic validation/checks on a table.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
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 254 of file table.scad.

```
6.9.2.2 function table_copy ( rows , cols , rows_sel , cols_sel )
```

Create a copy of select rows and columns of a table.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
rows_sel	<1d-vector> A n-tuple vector of row identifier to select.
cols_sel	<1d-vector> A n-tuple vector of column identifier to select.

Returns

<2d-vector> The selected rows and columns of the table.

6.9.2.3 $\,$ module table_dump (rows , cols , rows_sel , cols_sel , number = true)

Dump a table to the console.

6.9 Datatable 101

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
rows_sel	<1d-vector> A n-tuple vector of row identifier to select.
cols_sel	<1d-vector> A n-tuple vector of column identifier to select.
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 $rows \leftarrow _sel$ and $cols_sel$. For example to output only the column identifiers 'c1' and 'c2', assign $cols_sel = ["c1", "c2"]$.

Definition at line 337 of file table.scad.

6.9.2.4 function table_exists (rows , cols , row_id , col_id)

Test the existence of a table row and column identifier.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
row_id	<string> The row identifier string to locate.</string>
col_id	<string> The column identifier string to locate.</string>

Returns

true if the row and column identifier exists, otherwise returns false.

6.9.2.5 function table_get (rows , cols , row_id , col_id)

Get the value for a table row and column identifier.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
row_id	<string> The row identifier string to locate.</string>
col_id	<string> The column identifier string to locate.</string>

Returns

<decimal|string> The value at the located row_id and col_id. If it does not exists, returns undef.

6.9.2.6 function table_get_col (cols , col_id)

Get the column for a table column identifier.

Parameters

cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
col_id	<string> The column identifier string to locate.</string>

Returns

<vector> The column where the row identifier is located. If the identifier does not exists, returns undef.

6.9.2.7 function table_get_col_idx (cols , col_id)

Get the index for a table column identifier.

6.9 Datatable 103

Parameters

cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
col_id	<string> The column identifier string to locate.</string>

Returns

<decimal> The column index where the identifier is located. If the identifier does not exists, returns empty_v.

6.9.2.8 function table_get_row (rows , row_id)

Get the row for a table row identifier.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
row_id	<string> The row identifier string to locate.</string>

Returns

<vector> The row where the row identifier is located. If the identifier does not exists, returns undef.

6.9.2.9 function table_get_row_cols (rows , cols , col_id)

Form a vector from the specified column of each table row.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
col_id	<string> The column identifier string.</string>

Returns

<vector> The vector formed by selecting the col_id for each row in the table. If column does not exists, returns **undef**.

6.9.2.10 function table_get_row_ids (rows)

Form a vector of each table row identifier.

Parameters

rows <2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table row	WS.
--	-----

Returns

<vector> The vector of table row identifiers. If column "id" does not exists, returns undef.

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 table_get_row_cols(), of the form table_get_row_cols(rows, cols, "id"), may be used without this assumption.

6.9.2.11 function table_get_row_idx (rows , row_id)

Get the index for a table row identifier.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
row_id	<string> The row identifier string to locate.</string>

Returns

<decimal> The row index where the identifier is located. If the identifier does not exists, returns empty_v.

6.9.2.12 function table_size (rows , cols)

Get the size of a table.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.

Returns

<decimal> The table size.

The size is reported as: (1) The number of rows when only the rows parameter is specified. (2) The number of columns when only the rows parameter is specified. (3) The (rows * columns) when both parameters are specified.

6.9.2.13 function table_sum (rows , cols , rows_sel , cols_sel)

Sum select rows and columns of a table.

Parameters

rows	<2d-vector> A two dimensional vector (r-tuple x c-tuple) containing the table rows.
cols	<2d-vector> A two dimensional vector (c-tuple x 1-tuple) containing the table columns.
rows_sel	<1d-vector> A vector n-tuple of row identifier to select.
cols_sel	<1d-vector> A vector n-tuple of column identifier to select.

Returns

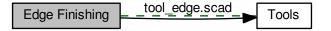
<1d-vector> The sum of the selected rows and columns of the table.

6.10 Edge Finishing 105

6.10 Edge Finishing

Shape edge finishing tools.

Collaboration diagram for Edge Finishing:



Files

• file tool_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.

6.10.1 Detailed Description

Shape edge finishing tools.

6.10.2 Function Documentation

6.10.2.1 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>

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

Example

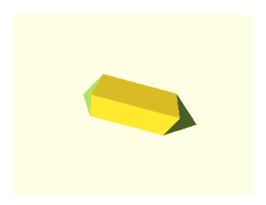


Figure 45: edge_add_r

1 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 204 of file tool_edge.scad.

6.10.2.2 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

6.10 Edge Finishing 107

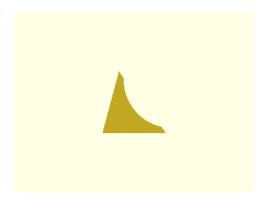


Figure 46: edge_profile_r

1 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

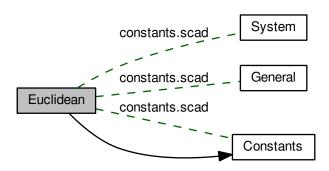
A offset factor greater than 1 moves the mid-point away from the profile edge-vertex. A factor less than 1 move it inwards towards the edge-vertex.

Definition at line 110 of file tool_edge.scad.

6.11 Euclidean

Euclidean 2D/3D space mapping.

Collaboration diagram for Euclidean:



Files

· file constants.scad

Mechanical design constants.

Variables

• x axis vi = 0

The vector index for the x-coordinate of a vector.

y_axis_vi = 1

The vector index for the y-coordinate of a vector.

• z_axis_vi = 2

The vector index for the z-coordinate of a vector.

• origin2d = [0, 0]

The origin coordinates in 2-dimensional Euclidean space.

• x_axis2d_uv = [1, 0]

The unit vector of the positive x-axis in 2-dimensional Euclidean space.

• y_axis2d_uv = [0, 1]

The unit vector of the positive y-axis in 2-dimensional Euclidean space.

• origin3d = [0, 0, 0]

The origin coordinates in 3-dimensional Euclidean space.

• $x_axis3d_uv = [1, 0, 0]$

The unit vector of the positive x-axis in 3-dimensional Euclidean space.

• y_axis3d_uv = [0, 1, 0]

The unit vector of the positive y-axis in 3-dimensional Euclidean space.

• z_axis3d_uv = [0, 0, 1]

The unit vector of the positive z-axis in 3-dimensional Euclidean space.

6.11 Euclidean 109

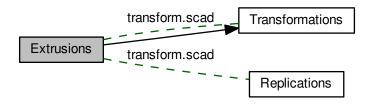
6.11.1 Detailed	Description
-----------------	-------------

Euclidean 2D/3D space mapping.

6.12 Extrusions

Shape Extrusions.

Collaboration diagram for Extrusions:



Files

· file transform.scad

Shape transformation functions.

Functions

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

Revolve the 2D shape about the z-axis.

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

Revolve the 2D shape about the z-axis with linear elongation.

• module st_linear_extrude_scale (h, center=false)

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

6.12.1 Detailed Description

Shape Extrusions.

6.12.2 Function Documentation

6.12.2.1 module st_linear_extrude_scale (h , center = false)

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

Parameters

h < vector decimal > A vector of decimals or a single decimal to specify simple extrusion height.

6.12 Extrusions 111

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 vector 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

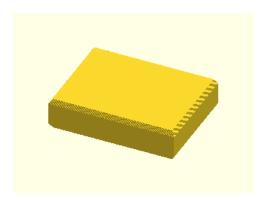


Figure 47: st_linear_extrude_scale

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

Note

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

Definition at line 230 of file transform.scad.

6.12.2.2 module st_rotate_extrude (r, pa = 0, ra = 360, profile = false)

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: st_rotate_extrude

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

Definition at line 103 of file transform.scad.

6.12.2.3 module st_rotate_extrude_elongate (r, I, pa = 0, ra = 360, m = 255, profile = false)

Revolve the 2D shape about the z-axis with linear elongation.

Parameters

r	<decimal> The rotation radius.</decimal>	
1	<pre><vector decimal> The elongation length. A vector [x, y] of decimals or a single decimal for (x=y)</vector decimal></pre>	
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

6.12 Extrusions 113



Figure 49: st_rotate_extrude_elongate

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

Note

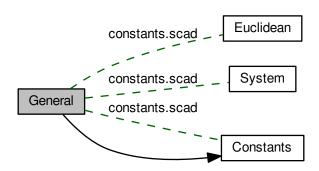
When elongating (1 > 0), ra is ignored. However, m may be used to control which complete revolution section to render.

Definition at line 139 of file transform.scad.

6.13 General

General design constants.

Collaboration diagram for General:



Files

· file constants.scad

Mechanical design constants.

Variables

- eps = 0.01
 - <decimal> Epsilon, small distance to deal with overlaping shapes
- pi = 3.1415926535897932384626433832795
 - <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

6.13.1 Detailed Description

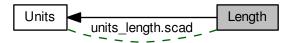
General design constants.

6.14 Length 115

6.14 Length

Length units and conversions.

Collaboration diagram for Length:



Files

· file units_length.scad

Length units and conversions.

Functions

• function unit_length_name (units=base_unit_length, d=1, w=false)

Return the name of the given unit identifier with dimension.

• function convert_length (value, from=base_unit_length, to=base_unit_length, d=1)

Convert the value from from units to to units with dimensions.

Variables

• base_unit_length = "mm"

<string> Base unit for length measurements.

6.14.1 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 unit identifiers and their descriptions.

units id	description
pm	picometer
nm	nanometer
um	micrometer
mm	millimeter
cm	centimeter

dm	decimeter
m	meter
km	kilometer
thou, mil	thousandth of an inch
in	inch
ft	feet
yd	yard
mi	mile

Example

```
include <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 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 \text{ ECHO: } c1 = 0.125
3 \text{ ECHO: } c2 = 0.125
4 \text{ ECHO: } c3 = 0.025
5 ECHO: c4 = 1728
6 ECHO: c5 = 0.01
7 \text{ ECHO: } c6 = 3048
```

6.14 Length 117

Example (equivalent lengths)



Figure 50: Unit Lengths

6.14.2 Function Documentation

6.14.2.1 function convert_length (value , from = base_unit_length, to = base_unit_length, d = 1)

Convert the value from from units to to units with dimensions.

Parameters

value	<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	<pre><decimal> The dimension set to one of [1 2 3].</decimal></pre>

Returns

<decimal> The conversion result. Returns 'undef' for identifiers or dimensions that are not defined.

6.14.2.2 function unit_length_name (units = base_unit_length, d = 1, w = false)

Return the name of the given unit identifier with dimension.

Parameters

W	<boolean> true: use word format, false: use symbol format.</boolean>
units	<string> A length unit identifier.</string>
d	<decimal $>$ A dimension set to one of [1 2 3].

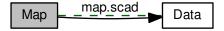
Returns

<string> The units name for the given length unit identifier with is specified dimension. Returns 'undef' for identifiers or dimensions that are not defined.

6.15 Map

Mapped data access via key-value pairs.

Collaboration diagram for Map:



Files

· file map.scad

Mapped key-value pair data access.

Functions

function map_get_idx (map, key)

Return the index for the storage location of a map key-value pair.

function map_exists (map, key)

Test if a key exists in a map.

• function map_get (map, key)

Get the value associated with a map key.

function map_get_keys (map)

Get a vector of the map entry identifier keys.

function map_get_values (map)

Get a vector of the map entry values.

• function map_size (map)

Get the number of key-value pairs stored in a map.

module map_check (map, verbose=false)

Perform some basic validation/checks on a map.

• module map_dump (map, sort=true, number=true, p=3)

Dump each map key-value pair to the console.

6.15.1 Detailed Description

Mapped data access via key-value pairs.

Example

6.15 Map 119

```
["screw10", [10, 10, -12]]],
                   ["top", "front", "rear"]], [21, 5, 0]],
  ["config",
   ["version",
  ["runid",
map_check(map, true);
echo( str("is part0 = ", map_exists(map, "part0")) );
echo( str("is part1 = ", map_exists(map, "part1")) );
p1 = map_get(map, "part1");
echo( c=second(p1) );
keys=map_get_keys(map);
parts = delete(keys, mv=["config", "version", "runid"]);
for (p = parts)
  echo
    n=p,
    p=first(map_get(map, p)),
    l=second(map_get(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."
```

6.15.2 Function Documentation

6.15.2.1 module map_check(map, verbose = false)

Perform some basic validation/checks on a map.

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.
verbose	<boolean> Be verbose during check.</boolean>

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 169 of file map.scad.

```
6.15.2.2 module map_dump ( map, sort = true, number = true, p = 3 )
```

Dump each map key-value pair to the console.

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.
sort	<boolean> Sort the output by key.</boolean>
number	<boolean> Output index number.</boolean>
р	<integer> Number of places for zero-padded numbering.</integer>

Definition at line 245 of file map.scad.

6.15.2.3 function map_exists (map , key)

Test if a key exists in a map.

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.
key	<string> A map entry identifier.</string>

Returns

<boolean> true when the key exists and false otherwise.

6.15.2.4 function map_get (map , key)

Get the value associated with a map key.

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.
key	<string> A map entry identifier.</string>

Returns

The map value associated with key. Returns **undef** if key does not exists.

6.15.2.5 function map_get_idx (map , key)

Return the index for the storage location of a map key-value pair.

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.
key	<string> A map entry identifier.</string>

Returns

<integer> The index of the value associated key in the map. Returns **undef** if key is not a string or does not exists.

6.15.2.6 function map_get_keys (map)

Get a vector of the map entry identifier keys.

6.15 Map 121

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.

Returns

<vector> A vector of keys that exist in the associative map.

Note

Uses function eselect to select the first column of the vector defining the map.

6.15.2.7 function map_get_values (map)

Get a vector of the map entry values.

Parameters

m	пар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
		elements.

Returns

<vector> A vector of values stored in the associative map.

Note

Uses function eselect to select the last column of the vector defining the map.

6.15.2.8 function map_size (map)

Get the number of key-value pairs stored in a map.

Parameters

тар	<2d-vector> A two dimensional vector (2-tuple x n-tuple) containing an associative map with n
	elements.

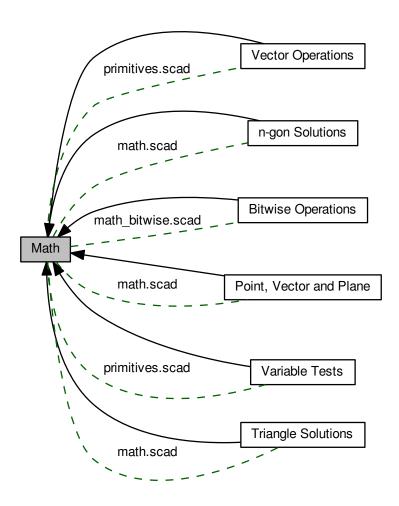
Returns

<integer> The number of key-value pairs stored in the map.

6.16 Math

Mathematical functions.

Collaboration diagram for Math:



Modules

• Bitwise Operations

Bitwise binary (base-two) operations.

• Point, Vector and Plane

Point, vector, and plane computations.

• Triangle Solutions

Triangle computations.

· Variable Tests

Variable property test primitives.

6.16 Math 123

Vector Operations

Vector operation primitives.

• n-gon Solutions

Regular n-sided polygon computations.

Files

· file primitives.scad

Mathematical primitive functions.

· file math.scad

Mathematical functions.

• file math_bitwise.scad

Mathematical bitwise binary (base-two) functions.

6.16.1 Detailed Description

Mathematical functions.

6.17 Parts

Parametric parts/components.

Parametric parts/components.

6.18 Point, Vector and Plane

Point, vector, and plane computations.

Collaboration diagram for Point, Vector and Plane:



Files

· file math.scad

Mathematical functions.

Functions

function distance_pp (p1, p2)

Compute the distance between two points in a Euclidean 1, 2, or 3D-space.

function dot_vv (v1t, v2t, v1i, v2i)

Compute the dot product of two vectors.

• function cross_vv (v1t, v2t, v1i, v2i)

Compute the cross product of two vectors in a Euclidean 3D-space (2D).

function striple_vvv (v1t, v2t, v3t, v1i, v2i, v3i)

Compute scalar triple product of two vectors in a Euclidean 3D-space.

function angle_vv (v1t, v2t, v1i, v2i)

Compute the angle between two vectors in a Euclidean 2 or 3D-space.

function angle_vvn (v1t, v2t, nvt, v1i, v2i, nvi)

Compute the angle between two vectors in a Euclidean 3D-space.

function unit_v (vt, vi)

Compute the normalized unit vector for a 1, 2, or 3 term vector.

function are_coplanar_vvv (v1t, v2t, v3t, v1i, v2i, v3i)

Test if three vectors are coplanar in Euclidean 3D-space.

6.18.1 Detailed Description

Point, vector, and plane computations.

See validation results.

6.18.2 Function Documentation

6.18.2.1 function angle_vv (v1t , v2t , v1i , v2i)

Compute the angle between two vectors in a Euclidean 2 or 3D-space.

Parameters

v1t	<vector> Vector 1 head. A 2 or 3-tuple of coordinates.</vector>
v2t	<vector> Vector 2 head. A 2 or 3-tuple of coordinates.</vector>
v1i	<vector> Vector 1 tail. A 2 or 3-tuple of coordinates.</vector>
v2i	<vector> Vector 2 tail. A 2 or 3-tuple of coordinates.</vector>

Returns

<decimal> The angle between the two vectors in degrees. Returns 'undef' when vector coordinates do not have same number of terms or when the vectors do not intersect.

Each vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

Note

For 3D vectors, a normal vector is required to uniquely identify the perpendicular plane and axis of rotation for the two vectors. 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_vvn().
```

6.18.2.2 function angle_vvn (v1t, v2t, nvt, v1i, v2i, nvi)

Compute the angle between two vectors in a Euclidean 3D-space.

Parameters

v1t	<vector> Vector 1 head. A 3-tuple of coordinates.</vector>
v2t	<vector> Vector 2 head. A 3-tuple of coordinates.</vector>
nvt	<vector> Normal vector head. A 3-tuple of coordinates.</vector>
v1i	<vector> Vector 1 tail. A 3-tuple of coordinates.</vector>
v2i	<vector> Vector 2 tail. A 3-tuple of coordinates.</vector>
nvi	<vector> Normal vector tail. A 3-tuple of coordinates.</vector>

Returns

<decimal> The angle between the two vectors in degrees. Returns **'undef'** when vector coordinates do not have same number of terms or when the vectors do not intersect.

Each vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

See also

angle_vv().

6.18.2.3 function are_coplanar_vvv (v1t , v2t , v3t , v1i , v2i , v3i)

Test if three vectors are coplanar in Euclidean 3D-space.

Parameters

v1t	<vector> Vector 1 head. A 3-tuple of coordinates.</vector>
v2t	<vector> Vector 2 head. A 3-tuple of coordinates.</vector>
v3t	<vector> Vector 3 head. A 3-tuple of coordinates.</vector>
v1i	<vector> Vector 1 tail. A 3-tuple of coordinates.</vector>
v2i	<vector> Vector 2 tail. A 3-tuple of coordinates.</vector>
v3i	<vector> Vector 3 tail. A 3-tuple of coordinates.</vector>

Returns

<boolean> true when all three vectors are coplanar, and false otherwise.

Each vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

See Wikipedia for more information.

Note

Coplanar vectors must all be within the same plane. However, this function can test if vectors are in a plane that is parallel to a coplanar plane by using non-zero vector tails.

6.18.2.4 function cross_vv (v1t , v2t , v1i , v2i)

Compute the cross product of two vectors in a Euclidean 3D-space (2D).

Parameters

v1t	<vector> Vector 1 head. A 2 or 3-tuple of coordinates.</vector>
v2t	<vector> Vector 2 head. A 2 or 3-tuple of coordinates.</vector>
v1i	<vector> Vector 1 tail. A 2 or 3-tuple of coordinates.</vector>
v2i	<vector> Vector 2 tail. A 2 or 3-tuple of coordinates.</vector>

Returns

<decimal> The cross product of the two vectors. Returns 'undef' when vector coordinates do not have same number of terms, n.

Each vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

See Wikipedia ${\hbox{\tt cross}}$ and ${\hbox{\tt determinant}}$ for more information.

Note

Although the cross product of two vectors is defined only in 3D space, this function will return the 2x2 determinant for a 2D vector.

6.18.2.5 function distance_pp (p1 , p2)

Compute the distance between two points in a Euclidean 1, 2, or 3D-space.

Parameters

р1	<vector> A 1, 2, or 3-tuple of coordinates.</vector>
p2	<vector> A 1, 2, or 3-tuple of coordinates.</vector>

Returns

<decimal> The distance between the two points. Returns **'undef'** when x and y do not have same number of terms or for n-tuple where n>3.

When p2 is not given, it is assumed to be at the origin.

6.18.2.6 function dot_vv (v1t , v2t , v1i , v2i)

Compute the dot product of two vectors.

Parameters

v1t	<vector> Vector 1 head. An n-tuple of coordinates.</vector>
v2t	<vector> Vector 2 head. An n-tuple of coordinates.</vector>
v1i	<vector> Vector 1 tail. An n-tuple of coordinates.</vector>
v2i	<vector> Vector 2 tail. An n-tuple of coordinates.</vector>

Returns

<decimal> The dot product of the two vectors. Returns 'undef' when vector coordinates do not have same number of terms, n.

Each vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

See Wikipedia for more information.

6.18.2.7 function striple_vvv (v1t , v2t , v3t , v1i , v2i , v3i)

Compute scalar triple product of two vectors in a Euclidean 3D-space.

Parameters

v1t	<vector> Vector 1 head. A 2 or 3-tuple of coordinates.</vector>
v2t	<vector> Vector 2 head. A 2 or 3-tuple of coordinates.</vector>
v3t	<vector> Vector 3 head. A 2 or 3-tuple of coordinates.</vector>
v1i	<vector> Vector 1 tail. A 2 or 3-tuple of coordinates.</vector>
v2i	<vector> Vector 2 tail. A 2 or 3-tuple of coordinates.</vector>
v3i	<vector> Vector 3 tail. A 2 or 3-tuple of coordinates.</vector>

Returns

<decimal> The scalar triple product of the three vectors. Returns 'undef' when vector coordinates do not have same number of terms, n.

Each vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

[v1, v2, v3] = v1 * (v2 x v3)

See Wikipedia for more information.

Warning

For 2D vectors, this function produces a 2D *non-scalar* vector result. The cross produce function computes the 2x2 determinant of the 2D vectors $(v2 \times v3)$, which is a scalar value, and this value is *multiplied* by v1, which results in a 2D vector.

6.18.2.8 function unit_v (vt, vi)

Compute the normalized unit vector for a 1, 2, or 3 term vector.

Parameters

vt	<vector> Vector head. A 1, 2, or 3-tuple of coordinates.</vector>
vi	<vector> Vector tail. A 1, 2, or 3-tuple of coordinates.</vector>

Returns

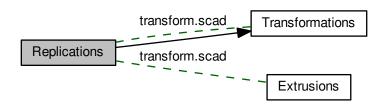
<vector> The vector normalized to its unit-vector. Returns **'undef'** when vector coordinates do not have same number of terms or for n-tuple where n>3.

The vector may be specified by both its head and tail coordinates. When specified by head coordinate only, the tail is assumed to be at origin.

6.19 Replications

Shape Replications and distribution.

Collaboration diagram for Replications:



Files

· file transform.scad

Shape transformation functions.

Functions

- module st_radial_copy (n, r=1, angle=true, move=false)
 Distribute copies of a 2D or 3D shape equally about a z-axis radius.
- module st_cartesian_copy (grid, incr, copy=1, center=false)

 Distribute copies of 2D or 3D shapes about Cartesian grid.

6.19.1 Detailed Description

Shape Replications and distribution.

6.19.2 Function Documentation

6.19.2.1 module st_cartesian_copy (grid , incr , copy = 1, center = false)

Distribute copies of 2D or 3D shapes about Cartesian grid.

Parameters

grid	<vector $ $ decimal $>$ A vector $[x, y, z]$ of decimals or a single decimal for $(x=y=z)$.
incr	<vector $ $ decimal $>$ A vector [x, y, z] of decimals or a single decimal for (x=y=z).
сору	<decimal> Number of times to iterate over children.</decimal>

6.19 Replications 131

center	<boolean> Center distribution about origin.</boolean>
--------	---

Example

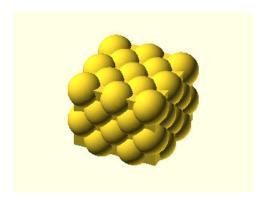


Figure 51: st_cartesian_copy

```
st_cartesian_copy( grid=[5,5,4], incr=10, copy=50, center=true ) {cube(10, center=true); sphere(10);}
```

Definition at line 352 of file transform.scad.

```
6.19.2.2 module st_radial_copy ( n, r = 1, angle = true, move = false )
```

Distribute copies of a 2D or 3D shape equally about a z-axis radius.

Parameters

n	<decimal> The number of equally spaced radii.</decimal>
r	<decimal> The shape move radius.</decimal>
angle	<boolean> Rotate each copy about z-axis.</boolean>
move	

Example

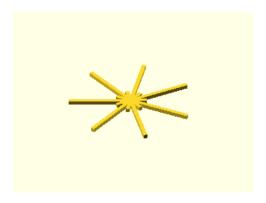


Figure 52: st_radial_copy

```
st_radial_copy( n=7, r=6, move=true ) square( [20,1], center=true );
```

Definition at line 322 of file transform.scad.

6.20 Resolution

Arch rendering resolution management.

Collaboration diagram for Resolution:



Files

· file resolution.scad

Arc rendering resolution abstraction.

Functions

function resolution_fn (radius)

Return facets number for the given arc radius.

• function resolution fs ()

Return minimum facets size.

• function resolution_fa (radius)

Return the minimum facets angle.

• function resolution_reduced ()

Return the radius at which arc resolution will begin to degrade.

• module resolution_info (radius)

Echo resolution information to the console for given radius.

• function resolution facets (radius)

Return facet count used to render a radius.

• function resolution_facetsv (radius)

Return facet count used to render a radius as vector triple.

Variables

• \$resolution_mode = "fast"

<string> Global special variable that configures the arc resolution mode.

• \$resolution_value = 0

<decimal> Global special variable for modes that use custom resolutions.

6.20 Resolution 133

6.20.1 Detailed Description

Arch rendering resolution management.

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 <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"
6.20.2 Function Documentation
6.20.2.1 function resolution_fa ( radius )
```

Return the minimum facets angle.

Parameters

radius	<decimal> An arc radius.</decimal>	
--------	------------------------------------	--

Returns

<decimal> Minimum facet angle to be assigned to \$fa.

The return result of this function can be assigned to the OpenSCAD special variables \$fa to render arcs.

6.20.2.2 function resolution_facets (radius)

Return facet count used to render a radius.

Parameters

radius	<decimal> An arc radius.</decimal>

Returns

<decimal> The number of fragments/facets that will be used to render a radius given the current values for \$fn, \$fa, and \$fs.

6.20.2.3 function resolution_facetsv (radius)

Return facet count used to render a radius as vector triple.

Parameters

radius	<decimal> An arc radius.</decimal>
--------	------------------------------------

Returns

A vector triple: [facets <decimal>,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.

6.20.2.4 function resolution_fn (radius)

Return facets number for the given arc radius.

Parameters

radius	<decimal> An arc radius.</decimal>

Returns

<decimal> The number of facets to be assigned to \$fn.

The return result of this function can be assigned to the special variables fn to render arcs according to the resolution mode set by $resolution_n$ and $resolution_n$.

The following table shows the modes that require \$resolution_value to be set prior to specify the custom values used during resolution calculation.

6.20 Resolution 135

\$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

6.20.2.5 function resolution_fs ()

Return minimum facets size.

Returns

<decimal> Minimum facet size to be assigned to \$fs.

The return result of this function can be assigned to the OpenSCAD 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

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

6.20.2.6 module resolution_info (radius)

Echo resolution information to the console for given radius.

Parameters

_		
	radius	<decimal> An arc radius.</decimal>

Definition at line 319 of file resolution.scad.

6.20.2.7 function resolution_reduced ()

Return the radius at which arc resolution will begin to degrade.

Returns

<decimal> Transition radius where resolution reduction begins.

The OpenSCAD special variables fs and fa work together when fa. 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.

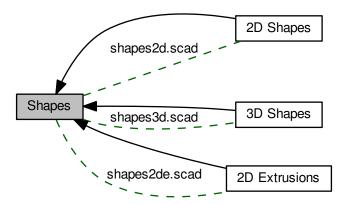
The return result of this function indicates the radius at which this enforced limiting begins. When \$fn != 0, returns 'undef'.

6.21 Shapes 137

6.21 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.

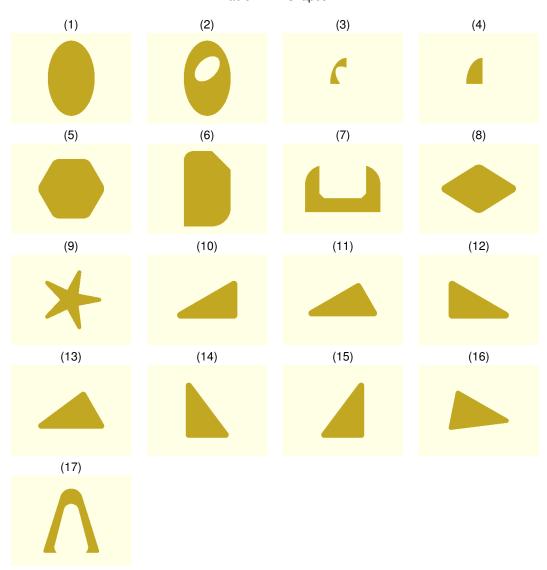
· file shapes3d.scad

Three-dimensional basic shapes.

6.21.1 Detailed Description

2D and 3D shapes.

Table 2: 2D Shapes



6.21 Shapes 139

Table 3: 2D Extrusions

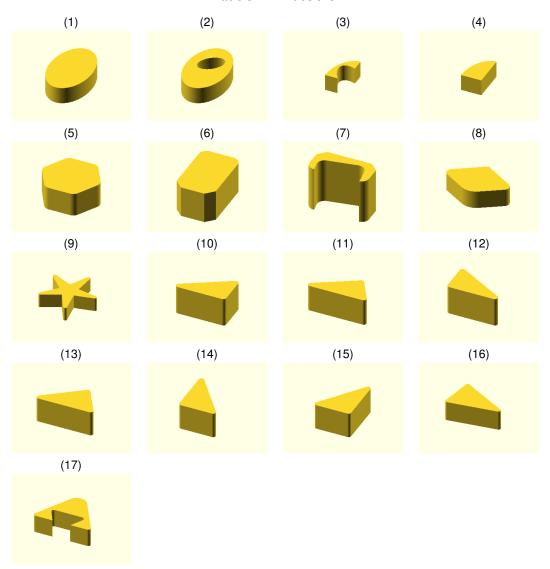
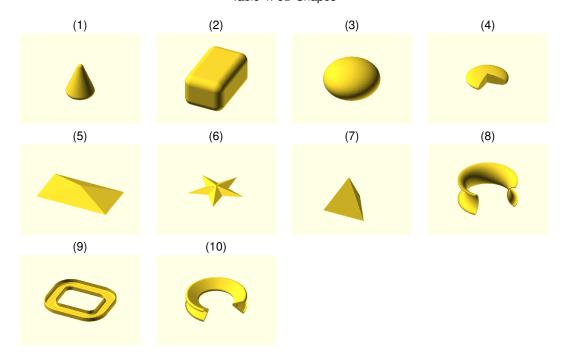


Table 4: 3D Shapes

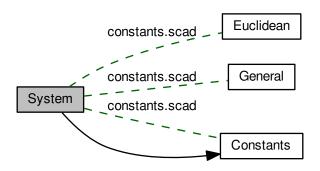


6.22 System 141

6.22 System

System/Program limits.

Collaboration diagram for System:



Files

· file constants.scad

Mechanical design constants.

Variables

• number_max = 1e308

The largest representable number in OpenSCAD scripts.

• number_min = -1e308

The smallest representable number in OpenSCAD scripts.

• empty_str = ""

A string with no content (the empty string).

• empty_v = []

A vector with no content (the empty vector).

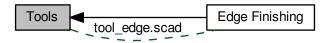
6.22.1 Detailed Description

System/Program limits.

6.23 Tools

Design tools and techniques.

Collaboration diagram for Tools:



Modules

• Edge Finishing

Shape edge finishing tools.

Files

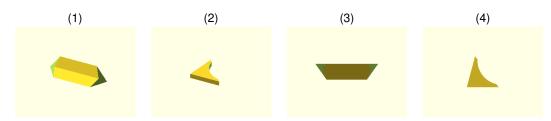
• file tool_edge.scad

Shape edge finishing tools.

6.23.1 Detailed Description

Design tools and techniques.

Table 5: Edge Finishing

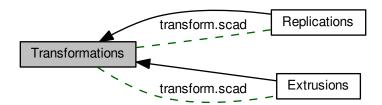


6.24 Transformations 143

6.24 Transformations

Shape transformations.

Collaboration diagram for Transformations:



Modules

• Extrusions

Shape Extrusions.

Replications

Shape Replications and distribution.

Files

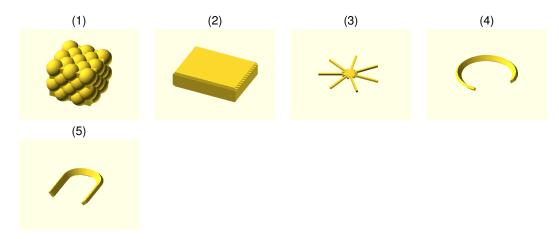
· file transform.scad

Shape transformation functions.

6.24.1 Detailed Description

Shape transformations.

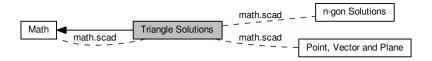
Table 6: Extrusions and Replications



6.25 Triangle Solutions

Triangle computations.

Collaboration diagram for Triangle Solutions:



Files

· file math.scad

Mathematical functions.

Functions

• function triangle_III2vp (s1, s2, s3)

Compute the vertices of a plane triangle given its side lengths.

• function triangle_vl2vp (v)

Compute the vertices of a plane triangle given its side lengths.

• function triangle_ppp2vl (v1, v2, v3)

Compute the side lengths of a triangle given its vertices.

function triangle_vp2vl (v)

Compute the side lengths of a triangle given its vertices.

function triangle_centroid_ppp (v1, v2, v3)

Compute the centroid (geometric center) of a triangle.

function triangle_centroid_vp (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_vp (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_vp (v)

Compute the inradius of a triangle's incircle.

6.25.1 Detailed Description

Triangle computations.

See Wikipedia for more information.

6.25.2 Function Documentation

6.25.2.1 function triangle_centroid_ppp (v1 , v2 , v3)

Compute the centroid (geometric center) of a triangle.

Parameters

	v1	<vector> A vector [x, y] for vertex 1 coordinates.</vector>
	v2	<vector> A vector [x, y] for vertex 2 coordinates.</vector>
ſ	v3	<vector> A vector [x, y] for vertex 3 coordinates.</vector>

Returns

<vector> A vector [x, y] of coordinates.

6.25.2.2 function triangle_centroid_vp (v)

Compute the centroid (geometric center) of a triangle.

Parameters

V	<pre><vector> A vector [v1, v2, v3] of vectors [x, y] coordinates.</vector></pre>

Returns

<vector> A vector [x, y] of coordinates.

6.25.2.3 function triangle_incenter_ppp (v1 , v2 , v3)

Compute the coordinate for the triangle's incircle.

Parameters

v1	<vector> A vector [x, y] for vertex 1 coordinates.</vector>
v2	<vector> A vector [x, y] for vertex 2 coordinates.</vector>
v3	<vector> A vector [x, y] for vertex 3 coordinates.</vector>

Returns

<vector> A vector [x, y] of coordinates.

The interior point for which distances to the sides of the triangle are equal.

6.25.2.4 function triangle_incenter_vp (v)

Compute the coordinate for the triangle's incircle.

Parameters

V	<vector> A vector [v1, v2, v3] of vectors [x, y] coordinates.</vector>

Returns

<vector> A vector [x, y] of coordinates.

The interior point for which distances to the sides of the triangle are equal.

6.25.2.5 function triangle_inradius_ppp (v1 , v2 , v3)

Compute the inradius of a triangle's incircle.

Parameters

v1	<vector> A vector [x, y] for vertex 1 coordinates.</vector>
v2	<vector> A vector [x, y] for vertex 2 coordinates.</vector>
<i>v3</i>	<vector> A vector [x, y] for vertex 3 coordinates.</vector>

Returns

<decimal> The incircle radius.

6.25.2.6 function triangle_inradius_vp (v)

Compute the inradius of a triangle's incircle.

Parameters

V	<pre><vector> A vector [v1, v2, v3] of vectors [x, y] coordinates.</vector></pre>
---	---

Returns

<decimal> The incircle radius.

6.25.2.7 function triangle_III2vp (s1 , s2 , s3)

Compute the vertices of a plane triangle given its side lengths.

Parameters

s1	<pre><decimal> The length of the side 1.</decimal></pre>
s2	<decimal> The length of the side 2.</decimal>
s3	<pre><decimal> The length of the side 3.</decimal></pre>

Returns

<vector> A vector [v1, v2, v3] of vectors [x, y] of coordinates.

Vertex v1 at the origin. Geometry required that s1 + s2 is greater then s3. Coordinates v3:[x, y] will be 'nan' when specified triangle does not exists.

Note

Side length s1 is measured along the positive x-axis. Sides are numbered counterclockwise.

6.25.2.8 function triangle_ppp2vl (v1, v2, v3)

Compute the side lengths of a triangle given its vertices.

Parameters

_		
	v1	<vector> A vector [x, y] for vertex 1 coordinates.</vector>

v2	<vector> A vector [x, y] for vertex 2 coordinates.</vector>
v3	<vector> A vector [x, y] for vertex 3 coordinates.</vector>

Returns

<vector> A vector [s1, s2, s3] of lengths.

Note

Vertices are numbered counterclockwise.

6.25.2.9 function triangle_vl2vp (v)

Compute the vertices of a plane triangle given its side lengths.

Parameters

V	<vector> of decimal side lengths.</vector>
V	vector > or decimal side lengths.

Returns

<vector> A vector [v1, v2, v3] of vectors [x, y] of coordinates.

Vertex vs[0] at the origin. Geometry required that vs[0] + vs[1] is greater then vs[2]. Coordinates v3:[x, y] will be 'nan' when specified triangle does not exists.

Note

Side length vs[0] is measured along the positive x-axis. Sides are numbered counterclockwise.

6.25.2.10 function triangle_vp2vl (v)

Compute the side lengths of a triangle given its vertices.

Parameters

V	<pre><vector> A vector [v1, v2, v3] of vectors [x, y] coordinates.</vector></pre>
---	---

Returns

<vector> A vector [s1, s2, s3] of lengths.

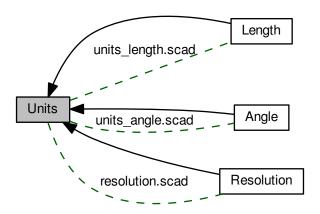
Note

Vertices are numbered counterclockwise.

6.26 Units

Units and unit conversions.

Collaboration diagram for Units:



Modules

Angle

Angle units and conversions.

Length

Length units and conversions.

• Resolution

Arch rendering resolution management.

Files

• file units_angle.scad

Angle units and conversions.

• file units_length.scad

Length units and conversions.

· file resolution.scad

Arc rendering resolution abstraction.

6.26.1 Detailed Description

Units and unit conversions.

6.27 Utilities 151

6.27 Utilities

General utilities.

Files

· file utilities.scad

Miscellaneous utilities.

· file validation.scad

Result validation functions.

• function stack (b=0, t=0)

Format the function call stack as a string.

• function validate (d, cv, t, ev, p=4, pf=false)

Compare a computed test value with an known good result.

6.27.1 Detailed Description

General utilities.

6.27.2 Function Documentation

6.27.2.1 function stack (
$$b = 0$$
, $t = 0$)

Format the function call stack as a string.

Parameters

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

Returns

<string> A 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: bo > \$parent_ \leftarrow modules-1).

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

6.27.2.2 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> Result reported as a 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

Example

```
use <validation.scad>;
2
       use <console.scad>;
3
4
5
       // function to validate
6
       function f1(x) = (x == undef) ? 1 : 2;
8
9
       farg = undef;
                           // function test argument
                            // correct expected function result
// incorrect expected function result
10
        erv1 = 1;
        erv2 = 3;
11
12
13
14
        // pass test example
15
        pass_result = validate("test-a f1(farg)", f1(farg), "equals", erv1);
16
17
18
        if (!validate(cv=f1(farg), t="equals", ev=erv1, pf=true) )
19
         log_warn( pass_result );
20
21
          log_info( pass_result );
22
23
24
        // fail test example
25
        fail_result = validate("test-b f1(farg)", f1(farg), "equals", erv2);
28
        if ( !validate(cv=f1(farg), t="equals", ev=erv2, pf=true) )
29
         log_warn( fail_result );
30
        else
31
          log_info( fail_result );
32
33
        // almost equal test example
34
35
        tvae1 = [[90.001], [[45.009], true]];
tvae2 = [[90.002], [[45.010], true]];
36
37
38
        log_info( validate("test-c", tvae1, "almost", tvae2, 3) );
log_warn( validate("test-d", tvae1, "almost", tvae2, 4) );
39
40
```

Result

```
1 ECHO: "[ INFO ] root(); passed: 'test-a f1(farg)'"
2 ECHO:
3 ECHO: "root()"
```

6.27 Utilities 153

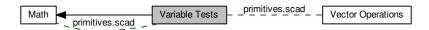
Note

When performing an almost equal validation type, 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.

6.28 Variable Tests

Variable property test primitives.

Collaboration diagram for Variable Tests:



Files

· file primitives.scad

Mathematical primitive functions.

Functions

• function is_defined (v)

Test if a value is defined.

function not_defined (v)

Test if a value is not defined.

• function is_empty (v)

Test if an iterable value is empty.

• 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_string (v)

Test if a value is a string.

function is_vector (v)

Test if a value is a vector.

• function is_boolean (v)

Test if a value is a boolean constant.

• function is_integer (v)

Test if a value is an integer.

• function is_decimal (v)

Test if a value is a decimal.

• function is_number (v)

Test if a value is a number.

function is_range (v)

Test if a value is a range definition.

• function is_nan (v)

Test if a numerical value is invalid.

function is_inf (v)

6.28 Variable Tests 155

Test if a numerical value is infinite.

function is_even (v)

Test if a numerical value is even.

• function is_odd (v)

Test if a numerical value is odd.

function all_equal (v, cv)

Test if all elements of a value equal a comparison value.

function any_equal (v, cv)

Test if any element of a value equals a comparison value.

function all defined (v)

Test if no element of a value is undefined.

• function any_undefined (v)

Test if any element of a value is undefined.

function all_scalars (v)

Test if all elements of a value are scalars.

function all_vectors (v)

Test if all elements of a value are vectors.

function all_strings (v)

Test if all elements of a value are strings.

• function all numbers (v)

Test if all elements of a value are numbers.

function all_len (v, l)

Test if all elements of a value have a given length.

function almost_equal (v1, v2, p=4)

Test if all elements of two values are approximately equal.

• function compare (v1, v2, s=true)

Compare any two values (may be iterable and/or of different types).

6.28.1 Detailed Description

Variable property test primitives.

See validation results group1 and group2.

6.28.2 Function Documentation

6.28.2.1 function all_defined (v)

Test if no element of a value is undefined.

Parameters

 $v \mid \langle value \rangle$ A value or an iterable value.

Returns

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

Warning

Always returns true when v is empty.

6.28.2.2 function all_equal (v , cv)

Test if all elements of a value equal a comparison value.

6.28 Variable Tests 157

Parameters

V	<value> A value or an iterable value.</value>
CV	<value> A comparison value.</value>

Returns

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

Warning

Always returns true when v is empty.

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

Test if all elements of a value have a given length.

Parameters

<value> A value or an iterable value.</value>
/ <integer> The length.</integer>

Returns

boolean> **true** when all elements have length equal to 1 and **false** otherwise. Returns the value of v when it is not defined.

Warning

Always returns **true** when v is empty.

6.28.2.4 function all_numbers (v)

Test if all elements of a value are numbers.

Parameters

V	<value> A value or an iterable value.</value>
---	---

Returns

boolean> **true** when all elements are numerical values and **false** otherwise. Returns **true** when v is a single numerical value. Returns the value of v when it is not defined.

Warning

Always returns \mbox{true} when ${\bf v}$ is empty.

6.28.2.5 function all_scalars (v)

Test if all elements of a value are scalars.

Parameters

V	<value> A value or an iterable value.</value>

Returns

 <boolean> **true** when all elements are scalar values and **false** otherwise. Returns **true** when v is a single scalar value. Returns the value of v when it is not defined.

Warning

Always returns true when v is empty.

6.28.2.6 function all_strings (v)

Test if all elements of a value are strings.

Parameters

V	<value> A value or an iterable value.</value>
---	---

Returns

boolean> **true** when all elements are string values and **false** otherwise. Returns **true** when v is a single string value. Returns the value of v when it is not defined.

Warning

Always returns true when v is empty.

6.28.2.7 function all_vectors (v)

Test if all elements of a value are vectors.

Parameters

V	<value> A value or an iterable value.</value>
-	Control of the following the f

Returns

boolean> **true** when all elements are vector values and **false** otherwise. Returns **true** when v is a single vector value. Returns the value of v when it is not defined.

Warning

Always returns true when v is empty.

6.28.2.8 function almost_equal (v1, v2, p = 4)

Test if all elements of two values are approximately equal.

6.28 Variable Tests 159

Parameters

v1	<value> A value or an iterable value 1.</value>
v2	<value> A value or an iterable value 2.</value>
р	<number> A numerical precision.</number>

Returns

Note

The parameter p indicated the number of digits of precision for each numerical comparison.

Warning

Always returns **true** when v is empty.

6.28.2.9 function any_equal (v , cv)

Test if any element of a value equals a comparison value.

Parameters

V	<value> A value or an iterable value.</value>
CV	<value> A comparison value.</value>

Returns

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

Warning

Always returns **false** when \vee is empty.

6.28.2.10 function any_undefined (v)

Test if any element of a value is undefined.

Parameters

V	<value> A value or an iterable value.</value>

Returns

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

Warning

Always returns **false** when \vee is empty.

6.28.2.11 function compare (v1, v2, s = true)

Compare any two values (may be iterable and/or of different types).

Parameters

v1	<value> A value or an iterable value 1.</value>
v2	<value> A value or an iterable value 2.</value>
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	string		lexical comparison
4	boolean		false < true
5	vector		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 vectors of equal length, the comparison continue element-by-element until an ordering can be determined. Two vectors are declared equal when all elements have been compared and no ordering has been determined.

Warning

The performance of element-wise comparisons of vectors degrades exponentially with vector size. The sum of a range may quickly exceeded the intermediate variable storage capacity for long ranges.

6.28.2.12 function is_boolean (v)

Test if a value is a 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.

6.28.2.13 function is_decimal (v)

Test if a value is a decimal.

6.28 Variable Tests 161

Parameters

V	<value> A value.</value>
---	--------------------------

Returns

<boolean> **true** when the value is a decimal and **false** otherwise.

6.28.2.14 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.

6.28.2.15 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.

6.28.2.16 function is_even (v)

Test if a numerical value is even.

Parameters

```
v <value> A numerical value.
```

Returns

<boolean> true when the value is determined to be even and false otherwise.

Note

The value must be valid and defined but may be positive or negative. Any value that is not an integer returns false.

6.28.2.17 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.

6.28.2.18 function is_integer (v)

Test if a value is an integer.

Parameters

Returns

<boolean> true when the value is an integer and false otherwise.

6.28.2.19 function is_iterable (v)

Test if a value has multiple parts and is iterable.

Parameters

Returns

<body><true when the value is an iterable multi-part value and false otherwise.</td>

value is	defined result
undef	false
inf	false
nan	false
integer	false
decimal	false
boolean	false
string	true
vector	true
range	not defined

6.28.2.20 function is_nan (v)

Test if a numerical value is invalid.

Parameters

ĺ	ν	<value> A numerical value.</value>

Returns

6.28.2.21 function is_number (v)

Test if a value is a number.

6.28 Variable Tests 163

Parameters

V	<value> A value.</value>
---	--------------------------

Returns

<boolean> true when the value is a number and false otherwise.

Warning

Returns true even for numerical values that are considered infinite and invalid.

6.28.2.22 function is_odd (v)

Test if a numerical value is odd.

Parameters

v <value> A numerical value.

Returns

<boolean> true when the value is determined to be odd and false otherwise.

Note

The value must be valid and defined but may be positive or negative. Any value that is not an integer returns false.

6.28.2.23 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.

6.28.2.24 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.

value is	defined result
undef	true
inf	true
nan	true
integer	true
decimal	true
boolean	true
string	false
vector	false
range	not defined

6.28.2.25 function is_string (v)

Test if a value is a string.

Parameters

v <value> A value.

Returns

<boolean> true when the value is a string and false otherwise.

6.28.2.26 function is_vector (v)

Test if a value is a vector.

Parameters

v <value> A value.

Returns

<boolean> true when the value is a vector and false otherwise.

6.28.2.27 function not_defined (v)

Test if a value is not defined.

Parameters

v <value> A value.

Returns

<boolean> true when the value is not defined and false otherwise.

6.29 Vector Operations

Vector operation primitives.

Collaboration diagram for Vector Operations:



Files

· file primitives.scad

Mathematical primitive functions.

Functions

• function consts (I, v)

Create a vector of constant elements.

• function vstr (v)

Convert all vector elements to strings and concatenate.

• function sum (v, i1, i2)

Compute the sum of a vector of numbers.

• 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 the existence of a match value in an iterable value.

function defined or (v, d)

Return a defined or default value.

function edefined_or (v, i, d)

Return a defined vector element or default value.

• function first (v)

Return the first element of an iterable value.

• function second (v)

Return the second element of an iterable value.

• function last (v)

Return the last element of an iterable value.

• function nfirst (v, n=1)

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

function nlast (v, n=1)

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

function nhead (v, n=1)

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

• function ntail (v, n=1)

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

function rselect (v, i)

Select a range of elements from an iterable value.

• function eselect (v, f=true, l=false, i)

Select an element from each iterable value.

• function ciselect (v, i)

Case-like select a value from a vector of ordered options by index.

• function cmvselect (v, mv)

Case-like select a value from a vector of identified options by match-value.

• function smerge (v, r=false)

Serial-merge vectors of iterable values.

• function pmerge (v, j=true)

Parallel-merge vectors of iterable values.

• function reverse (v)

Reverse the elements of an iterable value.

• function qsort (v, r=false)

Sort the numeric or string elements of a vector using quick sort.

• function qsort2 (v, d=0, r=false, s=true)

Hierarchically sort all elements of a vector using quick sort.

function strip (v, mv=empty_v)

Strip all matching values from 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 unique (v)

Return the unique elements of an iterable value.

6.29.1 Detailed Description

Vector operation primitives.

See validation results.

6.29.2 Function Documentation

6.29.2.1 function ciselect (v , i)

Case-like select a value from a vector of ordered options by index.

Parameters

V	<vector> A vector of values.</vector>
i	<integer> Element selection index.</integer>

Returns

<value> The value of the vector element at the specified index. Returns the default value when i does not map to an element of v or when i is undefined.

Behaves like a case statement for selecting values from a list of ordered options. The default value is: last (v).

Example

6.29.2.2 function cmvselect (v , mv)

Case-like select a value from a vector of identified options by match-value.

Parameters

V	<vector> A two dimensional vector of one or more identified values [[identifier, value],].</vector>
mv	<value> Element selection match value.</value>

Returns

<value> The value from the vector of identified elements with an identifier matching 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 identified options. The default value is \leftarrow : second (last (v)).

Example

6.29.2.3 function consts (I, v)

Create a vector of constant elements.

Parameters

1	<integer> The vector length.</integer>
V	<value> The element value.</value>

Returns

<vector> With 1 copies of the element value v. Returns **empty v** when 1 is not a number or if (1 < 1).

Note

When v is not specified, each element is assigned the value of its index position.

```
6.29.2.4 function count ( mv, v, s = true, i )
```

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	<boolean> Use search for element matching (false uses find).</boolean>
i	<integer> The element column index to match.</integer>

Returns

<integer> The number of times my occurs in v.

See find() for information on value matching.

6.29.2.5 function defined_or (v , d)

Return a defined or default value.

Parameters

V	<value> A value.</value>
d	<value> A default value.</value>

Returns

<value> v when it is defined or d otherwise.

6.29.2.6 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 vector integer> Deletion Indexes.</range vector integer>	
mv	<pre><vector string value> Match value candidates.</vector string value></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 (false uses find).</boolean>	
si	<integer> The element column index when matching.</integer>	

Returns

<vector> v 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 vector of index positions, an index range, an element match value, or a vector of element match values (when using search). When mv is a vector 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.

See find() for information on value matching.

6.29.2.7 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	<vector> A vector of values.</vector>	
r	<boolean> Reduce vector element value before appending.</boolean>	
j	 boolean> Join each appendage as a vector.	
1	<boolean> Append to last element.</boolean>	

Returns

<vector> 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

Note

Appending with reduction causes nv to be appended to the *elements* of each value of v that is a vector. Otherwise, nv is appended to the *vector* itself of each value of v that is a vector.

6.29.2.8 function edefined_or (v , i , d)

Return a defined vector element or default value.

Parameters

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

Returns

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

```
6.29.2.9 function eselect ( v , f = true, I = false, i )
```

Select an element from each iterable value.

Parameters

V	<vector> A vector of iterable values.</vector>

f	 boolean> Select the first element.
1	 boolean> Select the last element.
i	<integer> Select a numeric element index position.</integer>

Returns

<vector> Containing the selected element of each iterable value of v. Returns **empty_v** 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.

Check 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 (false uses find).</boolean>	
i	<integer> The element column index to match.</integer>	

Returns

boolean> true when mv exists in v and false otherwise.

See find() for information on value matching.

6.29.2.11 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	<pre><integer> A match count. For ($c>=1$), return the first c matches. For ($c<=0$), return all</integer></pre>
	matches.
i	<integer> The element column index to match.</integer>
i1	<integer> The element index where find begins (default: first).</integer>
i2	<integer> The element index where find ends (default: last).</integer>

Returns

<vector> Of indexes where elements match mv. Returns **empty_v** 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	vector of scalars	vector of iterables
scalar		(a)	(b) see note 1
string	(c)		(b) see note 1
vector of scalars			(b) see note 1
vector of iterables			(b) see note 1

Search:

mv / v	string	vector of scalars	vector of iterables
scalar		(a)	(b)
string	(d)	invalid	(e) see note 2
vector of scalars		(f)	(g)
vector 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_v** 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_v** is returned for each character of mv absent from v.
- (f) For each scalar of mv, identify where it exists in v. empty_v 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_v** 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 vector of one or more strings is valid. In which case, only the first character of each string element is considered.

6.29.2.12 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.

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

Insert a new value into an iterable value.

Parameters

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

Returns

<vector> 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 vector 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.

See find() for information on value matching.

6.29.2.14 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.

6.29.2.15 function nfirst (v, n = 1)

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

Parameters

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

Returns

<vector> Containing the first n elements of v. Returns **undef** when v is not defined, is not iterable, or is empty.

6.29.2.16 function nhead (v, n = 1)

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

Parameters

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

Returns

<vector> Containing all but the last n elements of v. Returns **empty_v** when v contains fewer than n elements. Returns **undef** when v is not defined, is not iterable, or is empty.

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

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

Parameters

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

Returns

<vector> Containing the last n elements of v. Returns undef when v is not defined, is not iterable, or is empty.

```
6.29.2.18 function ntail (v, n = 1)
```

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

Parameters

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

Returns

<vector> Containing all but the first n elements of v. Returns **empty_v** when v contains fewer than n elements. Returns **undef** when v is not defined, is not iterable, or is empty.

```
6.29.2.19 function pmerge (v, j = true)
```

Parallel-merge vectors of iterable values.

Parameters

V	<vector> A vector of iterable values.</vector>
j	<boolean> Join each merge as a vector.</boolean>

Returns

<vector> Containing the parallel-wise element concatenation of each iterable value in v. Returns **empty_v** 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 vector length will be limited by the iterable value with the shortest length. A string, although iterable, is treated as a merged unit.

6.29.2.20 function qsort (v, r = false)

Sort the numeric or string elements of a vector using quick sort.

Parameters

V	<vector> A vector of values.</vector>
r	 boolean> Reverse sort order.

Returns

<vector> With elements sorted in ascending order. Returns **undef** when v is not all strings or all numbers. Returns **undef** when v is not defined or is not a vector.

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 returns **undef** for vectors containing anything other than all scalar numbers or all strings.

See Wikipedia for more information.

6.29.2.21 function qsort2 (v, d = 0, r = false, s = true)

Hierarchically sort all elements of a vector using quick sort.

Parameters

V	<vector> A vector of values.</vector>
d	<integer> Recursive sort depth.</integer>
r	 boolean> Reverse sort order.
S	<boolean> Order ranges by their numerical sum.</boolean>

Returns

<vector> With all elements sorted in ascending order. Returns undef when v is not defined or is not a vector.

Elements are sorted using the compare function. 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.

6.29.2.22 function reverse (v)

Reverse the elements of an iterable value.

Parameters

<value> An iterable value.</value>	

Returns

<vector> Containing the elements of v in reversed order. Returns **empty_v** when v is empty. Returns **undef** when v is not defined or is not iterable.

6.29.2.23 function rselect (v , i)

Select a range of elements from an iterable value.

Parameters

V	<value> An iterable value.</value>
i	<range vector integer> Index selection.</range vector integer>

Returns

<vector> Containing the vector element indexes selected in i. Returns **undef** when i does not map to an element of v. Returns **empty_v** when v is empty. Returns **undef** when v is not defined or is not iterable.

6.29.2.24 function second (v)

Return the second element of an iterable value.

Parameters

.,	Zvalue \ An iterable value
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.

6.29.2.25 function smerge (v , r = false)

Serial-merge vectors of iterable values.

Parameters

V	<vector> A vector of iterable values.</vector>
r	

Returns

<vector> Containing the serial-wise element concatenation of each element in v. Returns **empty_v** when v is empty. Returns **undef** when v is not defined.

Note

A string, although iterable, is treated as a merged unit.

6.29.2.26 function strip (v , mv = empty_v)

Strip all matching values from an iterable value.

Parameters

V	<vector> A vector of values.</vector>
mv	<value> A match value.</value>

Returns

<vector> v with all elements equal to mv removed. Returns **undef** when v is not defined or is not iterable.

6.29.2.27 function sum (v , i1 , i2)

Compute the sum of a vector of numbers.

Parameters

V	ν <range vector=""> A vector of numerical values.</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

<decimal> The summation of elements over the index range. Returns \mathbf{v} when it is a scalar number. Returns 0 when \mathbf{v} is empty. Returns **undef** when \mathbf{v} is not defined or is not iterable and not a number.

6.29.2.28 function unique (v)

Return the unique elements of an iterable value.

Parameters

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

Returns

<vector> Of unique elements of v with order preserved. Returns **undef** when v is not defined or is not iterable.

6.29.2.29 function vstr (v)

Convert all vector elements to strings and concatenate.

Parameters

v <vector> A vector of values.</vector>

Returns

<string> Constructed by converting each element of the vector to a string and concatenating together. Returns **undef** when v is not defined.

Example

```
v1=["a", "b", "c", "d"];
v2=[1, 2, 3];
echo( vstr(concat(v1, v2)) );
```

Result

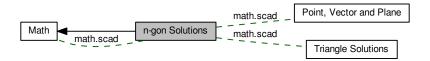
```
ECHO: "abcd123"
```

6.30 n-gon Solutions 179

6.30 n-gon Solutions

Regular n-sided polygon computations.

Collaboration diagram for n-gon Solutions:



Files

· file math.scad

Mathematical functions.

Functions

• function ngon_vp (n, r, vr)

Compute the vertices for an n-sided equiangular/equilateral regular polygon.

6.30.1 Detailed Description

Regular n-sided polygon computations.

6.30.2 Function Documentation

```
6.30.2.1 function ngon_vp ( n , r , vr )
```

Compute the vertices for an n-sided equiangular/equilateral regular polygon.

Parameters

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

Returns

<vector> A vector [v1, v2, ..., vn] of vectors [x, y] of coordinates.

Example

```
vr=5;
hull()
{
    for ( p = ngon_vp( r=20, n=5, vr=vr ) )
```

```
translate( p )
circle( r=vr );
```

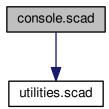
7 File Documentation 181

7 File Documentation

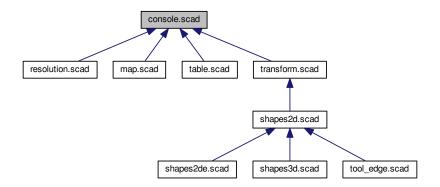
7.1 console.scad File Reference

Message logging functions.

#include <utilities.scad>
Include dependency graph for console.scad:



This graph shows which files directly or indirectly include this file:



Functions

• 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. 7.1.1 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 omd/ 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.

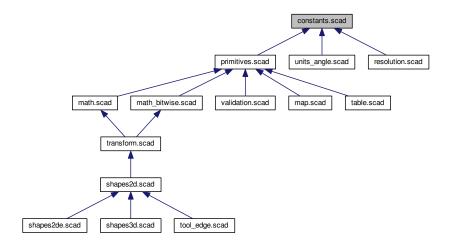
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

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/.

7.2 constants.scad File Reference

Mechanical design constants.

This graph shows which files directly or indirectly include this file:



Variables

• eps = 0.01

<decimal> Epsilon, small distance to deal with overlaping shapes

pi = 3.1415926535897932384626433832795

<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

• number_max = 1e308

The largest representable number in OpenSCAD scripts.

• number_min = -1e308

The smallest representable number in OpenSCAD scripts.

• empty_str = ""

A string with no content (the empty string).

• empty_v = []

A vector with no content (the empty vector).

• x_axis_vi = 0

The vector index for the x-coordinate of a vector.

• y_axis_vi = 1

The vector index for the y-coordinate of a vector.

• z axis vi = 2

The vector index for the z-coordinate of a vector.

• origin2d = [0, 0]

The origin coordinates in 2-dimensional Euclidean space.

x_axis2d_uv = [1, 0]

The unit vector of the positive x-axis in 2-dimensional Euclidean space.

• y_axis2d_uv = [0, 1]

The unit vector of the positive y-axis in 2-dimensional Euclidean space.

```
• origin3d = [0, 0, 0]
```

The origin coordinates in 3-dimensional Euclidean space.

• $x_axis3d_uv = [1, 0, 0]$

The unit vector of the positive x-axis in 3-dimensional Euclidean space.

• y axis3d uv = [0, 1, 0]

The unit vector of the positive y-axis in 3-dimensional Euclidean space.

• $z_{axis3d_uv} = [0, 0, 1]$

The unit vector of the positive z-axis in 3-dimensional Euclidean space.

7.2.1 Detailed Description

Mechanical design constants.

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.

7.3 mainpage.scad File Reference

Documentation main page.

7.3.1 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.

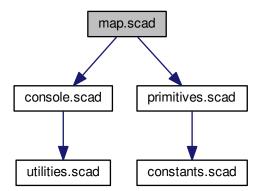
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

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. \leftarrow gnu.org/licenses/.

7.4 map.scad File Reference

Mapped key-value pair data access.

#include <console.scad>
#include <primitives.scad>
Include dependency graph for map.scad:



Functions

function map_get_idx (map, key)

Return the index for the storage location of a map key-value pair.

function map_exists (map, key)

Test if a key exists in a map.

function map_get (map, key)

Get the value associated with a map key.

function map_get_keys (map)

Get a vector of the map entry identifier keys.

function map_get_values (map)

Get a vector of the map entry values.

function map_size (map)

Get the number of key-value pairs stored in a map.

module map_check (map, verbose=false)

Perform some basic validation/checks on a map.

• module map_dump (map, sort=true, number=true, p=3)

Dump each map key-value pair to the console.

7.4.1 Detailed Description

Mapped key-value pair data access.

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.

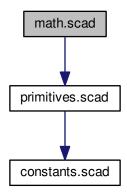
You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

Manage a collection of key-value pairs where keys are unique.

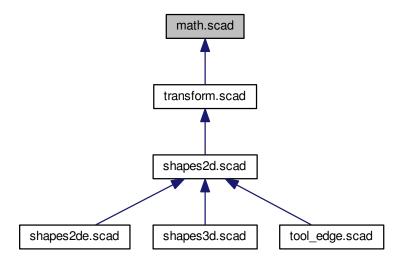
7.5 math.scad File Reference

Mathematical functions.

#include <primitives.scad>
Include dependency graph for math.scad:



This graph shows which files directly or indirectly include this file:



Functions

• function distance_pp (p1, p2)

Compute the distance between two points in a Euclidean 1, 2, or 3D-space.

function dot_vv (v1t, v2t, v1i, v2i)

Compute the dot product of two vectors.

function cross vv (v1t, v2t, v1i, v2i)

Compute the cross product of two vectors in a Euclidean 3D-space (2D).

function striple_vvv (v1t, v2t, v3t, v1i, v2i, v3i)

Compute scalar triple product of two vectors in a Euclidean 3D-space.

function angle vv (v1t, v2t, v1i, v2i)

Compute the angle between two vectors in a Euclidean 2 or 3D-space.

function angle vvn (v1t, v2t, nvt, v1i, v2i, nvi)

Compute the angle between two vectors in a Euclidean 3D-space.

• function unit v (vt, vi)

Compute the normalized unit vector for a 1, 2, or 3 term vector.

function are coplanar vvv (v1t, v2t, v3t, v1i, v2i, v3i)

Test if three vectors are coplanar in Euclidean 3D-space.

• function ngon_vp (n, r, vr)

Compute the vertices for an n-sided equiangular/equilateral regular polygon.

• function triangle III2vp (s1, s2, s3)

Compute the vertices of a plane triangle given its side lengths.

function triangle_vl2vp (v)

Compute the vertices of a plane triangle given its side lengths.

function triangle_ppp2vl (v1, v2, v3)

Compute the side lengths of a triangle given its vertices.

function triangle vp2vl (v)

Compute the side lengths of a triangle given its vertices.

function triangle centroid ppp (v1, v2, v3)

Compute the centroid (geometric center) of a triangle.

function triangle_centroid_vp (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_vp (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_vp (v)

Compute the inradius of a triangle's incircle.

7.5.1 Detailed Description

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.cognu.org/licenses/.

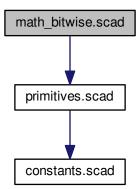
Note

Include this library file using the include statement.

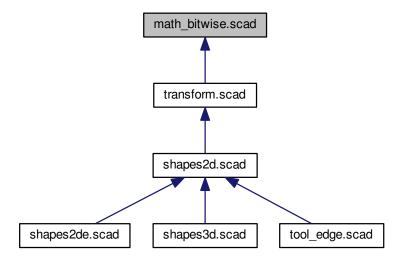
7.6 math bitwise.scad File Reference

Mathematical bitwise binary (base-two) functions.

#include <primitives.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 vector of bits.

• function bitwise_v2i (v)

Decode a base-two vector 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_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.

7.6.1 Detailed Description

Mathematical bitwise binary (base-two) 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/.

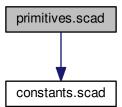
Note

Include this library file using the include statement.

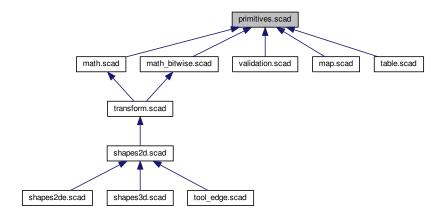
7.7 primitives.scad File Reference

Mathematical primitive functions.

#include <constants.scad>
Include dependency graph for primitives.scad:



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_empty (v)

Test if an iterable value is empty.

• 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 string (v)

Test if a value is a string.

• function is_vector (v)

Test if a value is a vector.

• function is_boolean (v)

Test if a value is a boolean constant.

• function is_integer (v)

Test if a value is an integer.

• function is_decimal (v)

Test if a value is a decimal.

function is_number (v)

Test if a value is a number.

function is range (v)

Test if a value is a range definition.

• function is_nan (v)

Test if a numerical value is invalid.

function is_inf (v)

Test if a numerical value is infinite.

• function is_even (v)

Test if a numerical value is even.

function is odd (v)

Test if a numerical value is odd.

function all_equal (v, cv)

Test if all elements of a value equal a comparison value.

function any_equal (v, cv)

Test if any element of a value equals a comparison value.

function all defined (v)

Test if no element of a value is undefined.

function any_undefined (v)

Test if any element of a value is undefined.

• function all_scalars (v)

Test if all elements of a value are scalars.

• function all_vectors (v)

Test if all elements of a value are vectors.

function all_strings (v)

Test if all elements of a value are strings.

• function all_numbers (v)

Test if all elements of a value are numbers.

• function all_len (v, I)

Test if all elements of a value have a given length.

• function almost equal (v1, v2, p=4)

Test if all elements of two values are approximately equal.

• function compare (v1, v2, s=true)

Compare any two values (may be iterable and/or of different types).

• function consts (I, v)

Create a vector of constant elements.

function vstr (v)

Convert all vector elements to strings and concatenate.

• function sum (v, i1, i2)

Compute the sum of a vector of numbers.

• 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 the existence of a match value in an iterable value.

• function defined or (v, d)

Return a defined or default value.

• function edefined_or (v, i, d)

Return a defined vector element or default value.

function first (v)

Return the first element of an iterable value.

function second (v)

Return the second element of an iterable value.

· function last (v)

Return the last element of an iterable value.

function nfirst (v, n=1)

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

function nlast (v, n=1)

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

function nhead (v, n=1)

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

function ntail (v, n=1)

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

• function rselect (v, i)

Select a range of elements from an iterable value.

function eselect (v, f=true, l=false, i)

Select an element from each iterable value.

• function ciselect (v, i)

Case-like select a value from a vector of ordered options by index.

• function cmvselect (v, mv)

Case-like select a value from a vector of identified options by match-value.

• function smerge (v, r=false)

Serial-merge vectors of iterable values.

function pmerge (v, j=true)

Parallel-merge vectors of iterable values.

• function reverse (v)

Reverse the elements of an iterable value.

function qsort (v, r=false)

Sort the numeric or string elements of a vector using quick sort.

• function qsort2 (v, d=0, r=false, s=true)

Hierarchically sort all elements of a vector using quick sort.

function strip (v, mv=empty_v)

Strip all matching values from 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 unique (v)

Return the unique elements of an iterable value.

7.7.1 Detailed Description

Mathematical primitive 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/.

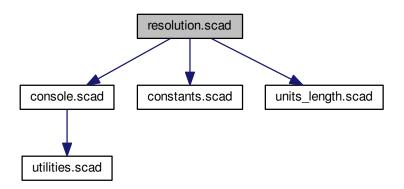
Note

Include this library file using the **include** statement.

7.8 resolution.scad File Reference

Arc rendering resolution abstraction.

```
#include <console.scad>
#include <constants.scad>
#include <units_length.scad>
Include dependency graph for resolution.scad:
```



Functions

· function resolution fn (radius)

Return facets number for the given arc radius.

• function resolution fs ()

Return minimum facets size.

function resolution_fa (radius)

Return the minimum facets angle.

· function resolution_reduced ()

Return the radius at which arc resolution will begin to degrade.

module resolution_info (radius)

Echo resolution information to the console for given radius.

function resolution facets (radius)

Return facet count used to render a radius.

function resolution_facetsv (radius)

Return facet count used to render a radius as vector triple.

Variables

• \$resolution mode = "fast"

<string> Global special variable that configures the arc resolution mode.

\$resolution value = 0

<decimal> Global special variable for modes that use custom resolutions.

7.8.1 Detailed Description

Arc rendering resolution abstraction.

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

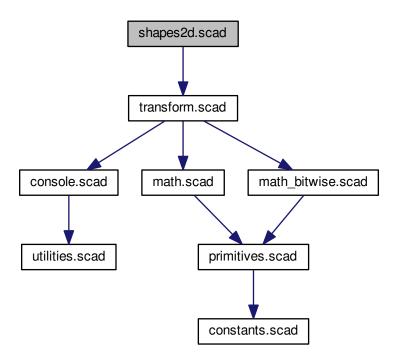
Include this library file using the include statement.

Todo Review model for accuracy.

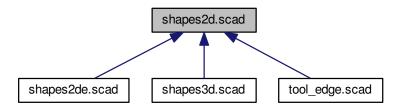
7.9 shapes2d.scad File Reference

Two-dimensional basic shapes.

#include <transform.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_vp (v, vr, centroid=false, incenter=false)

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

• module triangle III (s1, s2, s3, vr, v1r, v2r, v3r, centroid=false, incenter=false)

A general triangle specified by its three side lengths.

module triangle vl (v, vr, centroid=false, incenter=false)

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

module triangle_vl_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 lal (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_ala (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 aal (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_II (x, y, vr, v1r, v2r, v3r, centroid=false, incenter=false)

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

• module triangle_la (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.

7.9.1 Detailed Description

Two-dimensional basic shapes.

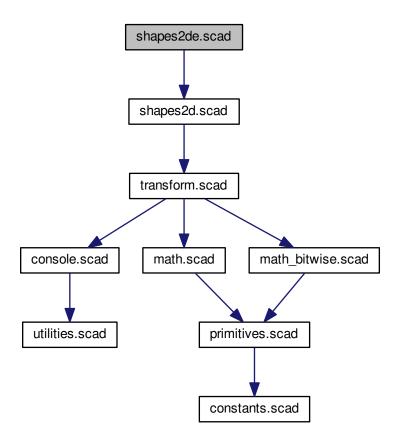
Author

Roy Allen Sutton

Date 0047
2015-2017
Copyright
This file is part of omdl, an OpenSCAD mechanical design library.
The <i>omdl</i> 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 <i>omdl</i> 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 <i>omdl</i> ; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.comu.org/licenses/.
7.10 shapes2de.scad File Reference
Linearly extruded two-dimensional basic shapes.

#include <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 vp (v, h, vr, centroid=false, incenter=false, center=false)
 - An extruded general triangle specified by a vector of its three vertices.
- module etriangle_III (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 vI (v, h, vr, centroid=false, incenter=false, center=false)

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

• module etriangle_vl_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_lal (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_ala (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_aal (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_II (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_la (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.

7.10.1 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.

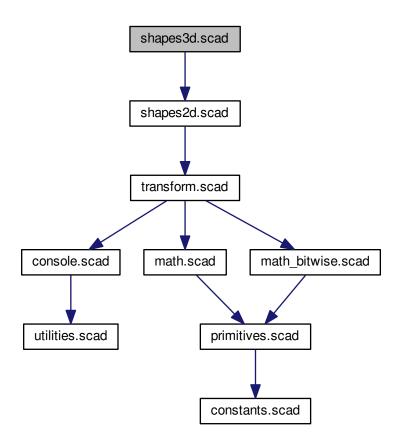
You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.⇔gnu.org/licenses/.

7.11 shapes3d.scad File Reference

Three-dimensional basic shapes.

#include <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 tetrahedron (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.

7.11.1 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.

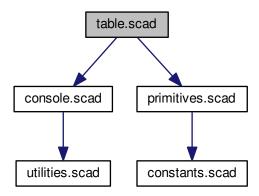
You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.←gnu.org/licenses/.

Todo Complete rounded cylinder.

7.12 table.scad File Reference

Data table encoding and lookup.

#include <console.scad>
#include <primitives.scad>
Include dependency graph for table.scad:



Functions

function table get row idx (rows, row id)

Get the index for a table row identifier.

function table_get_row (rows, row_id)

Get the row for a table row identifier.

function table_get_col_idx (cols, col_id)

Get the index for a table column identifier.

• function table_get_col (cols, col_id)

Get the column for a table column identifier.

function table_get (rows, cols, row_id, col_id)

Get the value for a table row and column identifier.

function table_get_row_cols (rows, cols, col_id)

Form a vector from the specified column of each table row.

function table_get_row_ids (rows)

Form a vector of each table row identifier.

• function table_exists (rows, cols, row_id, col_id)

Test the existence of a table row and column identifier.

• function table_size (rows, cols)

Get the size of a table.

module table check (rows, cols, verbose=false)

Perform some basic validation/checks on a table.

• module table_dump (rows, cols, rows_sel, cols_sel, number=true)

Dump a table to the console.

function table_copy (rows, cols, rows_sel, cols_sel)

Create a copy of select rows and columns of a table.

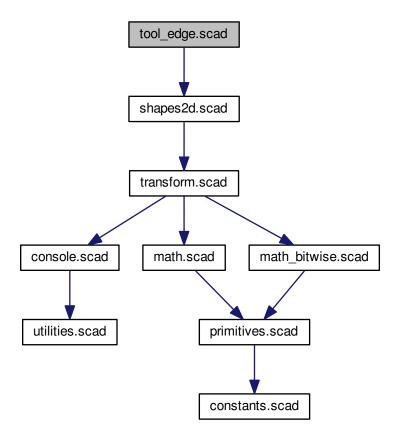
function table_sum (rows, cols, rows_sel, cols_sel)

Sum select rows and columns of a table.

7.12.1 Detailed Description
Data table encoding and lookup.
Author Roy Allen Sutton
Date 2015-2017
Copyright
This file is part of omdl, an OpenSCAD mechanical design library.
The <i>omdl</i> 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 <i>omdl</i> 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 <i>omdl</i> ; 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/ .
7.13 tool_edge.scad File Reference
Shape edge finishing tools.

#include <shapes2d.scad>

Include dependency graph for tool_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.

7.13.1 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.

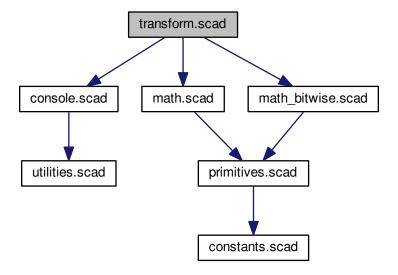
You should have received a copy of the GNU Lesser General Public License along with the *omdl*; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA; or see http://www.cognu.org/licenses/.

7.14 transform.scad File Reference

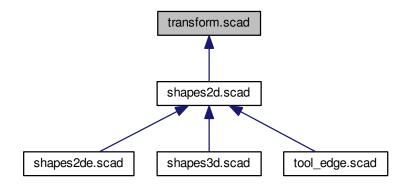
Shape transformation functions.

#include <console.scad>
#include <math.scad>
#include <math_bitwise.scad>

Include dependency graph for transform.scad:



This graph shows which files directly or indirectly include this file:



Functions

- module st_rotate_extrude (r, pa=0, ra=360, profile=false)
 - Revolve the 2D shape about the z-axis.
- module st_rotate_extrude_elongate (r, l, pa=0, ra=360, m=255, profile=false)
 - Revolve the 2D shape about the z-axis with linear elongation.
- module st_linear_extrude_scale (h, center=false)
 - Linearly extrude 2D shape with extrusion upper and lower scaling.
- module st_radial_copy (n, r=1, angle=true, move=false)
 - Distribute copies of a 2D or 3D shape equally about a z-axis radius.
- module st_cartesian_copy (grid, incr, copy=1, center=false)
 - Distribute copies of 2D or 3D shapes about Cartesian grid.

7.14.1 Detailed Description

Shape transformation 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.

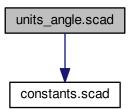
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

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/.

7.15 units angle.scad File Reference

Angle units and conversions.

#include <constants.scad>
Include dependency graph for units_angle.scad:



Functions

function unit_angle_name (units=base_unit_angle)

Return the name of the given angle unit identifier.

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

Convert the ${\it angle}$ from ${\it from}$ units to to units.

Variables

• base_unit_angle = "d"

<string> Base unit for angle measurements.

7.15.1 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.⇔gnu.org/licenses/.

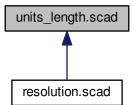
Note

Include this library file using the include statement.

7.16 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 (units=base_unit_length, d=1, w=false)

Return the name of the given unit identifier with dimension.

function convert length (value, from=base unit length, to=base unit length, d=1)

Convert the value from from units to to units with dimensions.

Variables

```
    base_unit_length = "mm"
    <string> Base unit for length measurements.
```

7.16.1 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/.

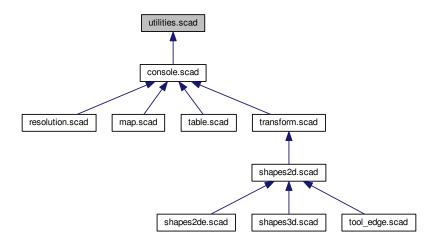
Note

Include this library file using the include statement.

7.17 utilities.scad File Reference

Miscellaneous utilities.

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.

7.17.1 Detailed Description

Miscellaneous utilities.

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.

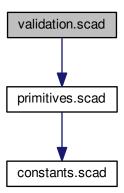
The *omdl* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

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/.

7.18 validation.scad File Reference

Result validation functions.

#include <primitives.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.

7.18.1 Detailed Description

Result validation 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. \leftarrow gnu.org/licenses/.

Index

2D Extrusions, 42	all_defined
eellipse, 43	Variable Tests, 155
eellipse_c, 44	all_equal
eellipse_cs, 44	Variable Tests, 155
eellipse_s, 45	all_len
engon, 46	Variable Tests, 157
erectangle, 47	all numbers
erectangle_c, 48	Variable Tests, 157
erhombus, 49	all scalars
estar2d, 50	Variable Tests, 157
etriangle_aal, 50	all strings
etriangle_ala, 51	Variable Tests, 158
etriangle_la, 52	all vectors
	-
etriangle_lal, 53	Variable Tests, 158
etriangle_II, 54	almost_equal
etriangle_III, 55	Variable Tests, 158
etriangle_ppp, 56	Angle, 85
etriangle_vl, 57	convert_angle, 86
etriangle_vl_c, 57	unit_angle_name, 86
etriangle_vp, 58	angle_vv
2D Shapes, 60	Point, Vector and Plane, 125
ellipse, 61	angle_vvn
ellipse_c, 61	Point, Vector and Plane, 126
ellipse_cs, 62	any_equal
ellipse_s, 63	Variable Tests, 159
ngon, 63	any undefined
rectangle, 64	Variable Tests, 159
rectangle_c, 65	are_coplanar_vvv
rhombus, 66	Point, Vector and Plane, 126
star2d, 66	Tomit, Toolor and Flame, 120
triangle_aal, 67	Bitwise Operations, 88
triangle_aai, 67 triangle_ala, 68	bitwise_and, 89
triangle_la, 69	bitwise i2s, 89
- —	- ·
triangle_lal, 69	bitwise_i2v, 89
triangle_II, 70	bitwise_is_equal, 89
triangle_III, 71	bitwise_lsh, 90
triangle_ppp, 72	bitwise_not, 90
triangle_vl, 72	bitwise_or, 90
triangle_vl_c, 73	bitwise_rsh, 90
triangle_vp, 73	bitwise_s2i, 91
3D Shapes, 76	bitwise_v2i, 91
cone, 77	bitwise_xor, 91
cuboid, 77	bitwise_and
ellipsoid, 78	Bitwise Operations, 89
ellipsoid_s, 79	bitwise_i2s
pyramid_q, 79	Bitwise Operations, 89
star3d, 80	bitwise_i2v
tetrahedron, 81	Bitwise Operations, 89
torus_ep, 81	bitwise_is_equal
torus_rp, 82	Bitwise Operations, 89
torus_tp, 83	bitwise_lsh
ισιασ_ιμ, σσ	DITMISE_ISH

Bitwise Operations, 90	table_get_row_ids, 103
bitwise_not	table_get_row_idx, 103
Bitwise Operations, 90	table_size, 104
bitwise_or	table_sum, 104
Bitwise Operations, 90	defined_or
bitwise_rsh	Vector Operations, 169
Bitwise Operations, 90	delete
bitwise_s2i	Vector Operations, 169
Bitwise Operations, 91	distance_pp
bitwise_v2i	Point, Vector and Plane, 127
Bitwise Operations, 91	dot_vv
bitwise_xor	Point, Vector and Plane, 128
Bitwise Operations, 91	
	eappend
ciselect	Vector Operations, 169
Vector Operations, 166	edefined_or
cmvselect	Vector Operations, 170
Vector Operations, 167	Edge Finishing, 105
compare	edge_add_r, 105
Variable Tests, 159	edge_profile_r, 106
cone	edge_add_r
3D Shapes, 77	Edge Finishing, 105
Console, 92	edge_profile_r
log_debug, 93	Edge Finishing, 106
log_echo, 93	eellipse
log_error, 93	2D Extrusions, 43
log_info, 93	eellipse_c
log_warn, 94	2D Extrusions, 44
console.scad, 180	eellipse_cs
Constants, 95	2D Extrusions, 44
constants.scad, 181	eellipse_s
consts	2D Extrusions, 45
Vector Operations, 167	
convert_angle	ellipse 2D Shapes, 61
Angle, 86	• •
convert_length	ellipse_c
Length, 117	2D Shapes, 61
count	ellipse_cs
Vector Operations, 167	2D Shapes, 62
cross_vv	ellipse_s
Point, Vector and Plane, 127	2D Shapes, 63
cuboid	ellipsoid
3D Shapes, 77	3D Shapes, 78
	ellipsoid_s
Data, 96	3D Shapes, 79
Datatable, 97	engon
table_check, 99	2D Extrusions, 46
table_copy, 99	erectangle
table_dump, 100	2D Extrusions, 47
table_exists, 101	erectangle_c
table_get, 101	2D Extrusions, 48
table_get_col, 101	erhombus
table_get_col_idx, 101	2D Extrusions, 49
table_get_row, 103	eselect
table_get_row_cols, 103	Vector Operations, 170

estar2d	is_nan
2D Extrusions, 50	Variable Tests, 162
etriangle_aal	is_number
2D Extrusions, 50	Variable Tests, 162
etriangle_ala	is_odd
2D Extrusions, 51	Variable Tests, 163
etriangle_la	is_range
2D Extrusions, 52	Variable Tests, 163
etriangle_lal	is_scalar
2D Extrusions, 53	Variable Tests, 163
etriangle_ll	is_string
2D Extrusions, 54	Variable Tests, 164
etriangle_III	is_vector
2D Extrusions, 55	Variable Tests, 164
etriangle_ppp	last
2D Extrusions, 56	Vector Operations, 173
etriangle_vl	Length, 115
2D Extrusions, 57	convert_length, 117
etriangle_vl_c 2D Extrusions, 57	unit length name, 117
etriangle_vp	log_debug
2D Extrusions, 58	Console, 93
Euclidean, 108	log_echo
exists	Console, 93
Vector Operations, 171	log_error
Extrusions, 110	Console, 93
st_linear_extrude_scale, 110	log_info
st_rotate_extrude, 111	Console, 93
st_rotate_extrude_elongate, 112	log_warn
	Console, 94
find	mainness and 100
Vector Operations, 171	mainpage.scad, 183
first	Map, 118 map check, 119
Vector Operations, 172	map_dump, 119
Compared 444	map_exists, 120
General, 114	map_get, 120
insert	map_get_idx, 120
Vector Operations, 172	map_get_keys, 120
is boolean	map_get_values, 121
Variable Tests, 160	map size, 121
is decimal	map.scad, 184
Variable Tests, 160	map_check
is_defined	Map, 119
Variable Tests, 161	map_dump
is_empty	Map, 119
Variable Tests, 161	map_exists
is_even	Map, 120
Variable Tests, 161	map_get
is_inf	Map, 120
Variable Tests, 161	map_get_idx
is_integer	Map, 120
Variable Tests, 162	map_get_keys
is_iterable	Map, 120
Variable Tests, 162	map_get_values

Map, 121	resolution_facetsv, 134
map_size	resolution_fn, 134
Map, 121	resolution_fs, 135
Math, 122	resolution_info, 136
math.scad, 186	resolution_reduced, 136
math_bitwise.scad, 188	resolution.scad, 194
	resolution fa
n-gon Solutions, 178	Resolution, 133
ngon_vp, 178	resolution_facets
nfirst	Resolution, 134
Vector Operations, 173	resolution facetsv
ngon	Resolution, 134
2D Shapes, 63	resolution fn
ngon_vp	Resolution, 134
n-gon Solutions, 178	resolution fs
nhead	Resolution, 135
Vector Operations, 173	resolution_info
nlast	
Vector Operations, 173	Resolution, 136
not_defined	resolution_reduced
Variable Tests, 164	Resolution, 136
ntail	reverse
Vector Operations, 174	Vector Operations, 175
vector Operations, 174	rhombus
Parts, 124	2D Shapes, 66
pmerge	rselect
Vector Operations, 174	Vector Operations, 175
·	
Point Vector and Plane 125	
Point, Vector and Plane, 125	second
angle_vv, 125	second Vector Operations, 176
angle_vv, 125 angle_vvn, 126	
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126	Vector Operations, 176
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127	Vector Operations, 176 Shapes, 137
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy Replications, 130 st_linear_extrude_scale
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy Replications, 130 st_linear_extrude_scale Extrusions, 110
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy Replications, 130 st_linear_extrude_scale Extrusions, 110 st_radial_copy Replications, 131
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy Replications, 130 st_linear_extrude_scale Extrusions, 110 st_radial_copy Replications, 131 st_rotate_extrude
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy Replications, 130 st_linear_extrude_scale Extrusions, 110 st_radial_copy Replications, 131 st_rotate_extrude Extrusions, 111 st_rotate_extrude_elongate Extrusions, 112
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge Vector Operations, 176 st_cartesian_copy Replications, 130 st_linear_extrude_scale Extrusions, 110 st_radial_copy Replications, 131 st_rotate_extrude Extrusions, 111 st_rotate_extrude_elongate Extrusions, 112 stack
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65 Replications, 130	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65 Replications, 130 st_cartesian_copy, 130	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65 Replications, 130 st_cartesian_copy, 130 st_radial_copy, 131	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65 Replications, 130 st_cartesian_copy, 130	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65 Replications, 130 st_cartesian_copy, 130 st_radial_copy, 131	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge
angle_vv, 125 angle_vvn, 126 are_coplanar_vvv, 126 cross_vv, 127 distance_pp, 127 dot_vv, 128 striple_vvv, 128 unit_v, 129 primitives.scad, 190 pyramid_q 3D Shapes, 79 qsort Vector Operations, 174 qsort2 Vector Operations, 175 rectangle 2D Shapes, 64 rectangle_c 2D Shapes, 65 Replications, 130 st_cartesian_copy, 130 st_radial_copy, 131 Resolution, 132	Vector Operations, 176 Shapes, 137 shapes2d.scad, 196 shapes2de.scad, 198 shapes3d.scad, 201 smerge

sum		
Vector Operations, 176 2D Shapes, 67 System, 141 triangle_ala 2D Shapes, 68 triangle_centroid_ppp table.scad, 202 triangle_centroid_ppp table_check Triangle Solutions, 146 Datatable, 99 triangle_centroid_yp table_dump Triangle Solutions, 146 Datatable, 100 triangle_incenter_yp Triangle Solutions, 146 triangle_inradius_pp Datatable, 101 triangle_inradius_pp table_get wists Triangle Solutions, 148 Datatable, 101 triangle_inradius_pp table_get_col Triangle Solutions, 148 Datatable, 101 triangle_la table_get_row 2D Shapes, 69 Datatable, 103 triangle_la table_get_row_cols 2D Shapes, 69 Datatable, 103 triangle_lill table_get_row_ids 2D Shapes, 70 Datatable, 103 triangle_lill table_get_row_ids 2D Shapes, 71 Datatable, 104 triangle_Dpp table_get_row_ids 2D Shapes, 72 Datatable, 104 tr	Point, Vector and Plane, 128	triangle_vp2vl, 149
System, 141 triangle_ala		- —
table.scad, 202 table_check Datatable, 99 table_copy Datatable, 99 table_copy Datatable, 99 table_dump Datatable, 100 table_get	•	•
table, scad, 202 table_check Datatable, 99 table_copy Datatable, 99 table_dump Datatable, 100 table_exists Datatable, 101 table_get_col idx Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 triangle_lll 2D Shapes, 70 Triangle Solutions, 148 triangle_ppp Triangle Solutions, 148 Triangle Solutions, 148 Triangle Solutions, 148 Triangle Solutions, 148 Triangle Solutions, 149 Triangle Solutions,	System, 141	-
table_check Datatable, 99 triangle_centroid_vp Triangle Solutions, 146 triangle_centroid_vp Triangle Solutions, 146 triangle_centroid_vp Triangle Solutions, 146 triangle_incenter_ppp Triangle Solutions, 146 triangle_incenter_vp Triangle Solutions, 146 triangle_incatics_vp Triangle Solutions, 146 triangle_inradius_pp Triangle Solutions, 146 triangle_inradius_pp Triangle Solutions, 146 triangle_inradius_vp Triangle Solutions, 146 triangle_inradius_vp Triangle Solutions, 148 triangle_la Triangle_la Solutions, 148 Triangle_get_col		•
Tatatable, 99 table_copy Datatable, 99 table_copy Triangle Solutions, 146 triangle_incenter_pp Triangle Solutions, 146 triangle_incenter_vp Triangle Solutions, 146 triangle_incenter_vp Triangle Solutions, 146 triangle_incenter_vp Triangle Solutions, 146 triangle_incations_pp Triangle Solutions, 146 triangle_inradius_pp Triangle Solutions, 146 triangle_inradius_vp Triangle Solutions, 148 triangle_get_col Datatable, 101 table_get_col_idx Datatable, 101 table_get_col_idx Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 triangle_lll 2D Shapes, 70 triangle_lll 2D Shapes, 71 triangle_Solutions, 148 triangle_pp 2D Shapes, 72 Triangle Solutions, 148 triangle_ppp 2D Shapes, 72 Triangle Solutions, 148 triangle_ppp 2D Shapes, 72 Triangle Solutions, 148 triangle_vl Triangle Solutions, 149 Triangle_ppp 1D Shapes, 73 Triangle Solutions, 145 Triangle_centroid_vp, 146 triangle_incenter_vp, 148 Units, 150		
table_copy Datatable, 99 Triangle Solutions, 146 triangle_incenter_ppp Triangle Solutions, 146 triangle_incenter_pp Triangle Solutions, 146 triangle_incenter_vp Triangle Solutions, 146 triangle_inradius_ppp Triangle Solutions, 146 triangle_inradius_ppp Triangle Solutions, 146 triangle_inradius_ppp Triangle Solutions, 146 triangle_inradius_vp Triangle Solutions, 148 triangle_ill 2D Shapes, 69 triangle_ill 2D Shapes, 69 triangle_ill 2D Shapes, 70 triangle_ill 2D Shapes, 70 triangle_ill 2D Shapes, 71 triangle Solutions, 148 triangle_ppt 2D Shapes, 72 triangle_vl2 Triangle Solutions, 148 triangle_vl 2D Shapes, 72 triangle_vl2 triangle_vl2 triangle_vl2 Triangle Solutions, 149 Triangle_vl2 Triangle_vl2 Triangle_vl2 Triangle_Solutions, 149 Triangle_vl2 Triangle_Solutions, 149 Triangle_solutions, 149 Triangle_vl2 Triangle_vl2 Triangle_vl2 Triangle_vl2 Triangle_Solutions, 149 Triangle_vl2 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_vp Triangle_solutions, 149 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_centroid_pp, 146 triangle_incenter_pp, 146 triangle_incenter_pp, 146 triangle_incenter_pp, 146 triangle_incenter_pp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 146 Triangle_inradius_pp, 148 Triangle_inradius_p	-	_
Datatable, 99 table_dump		·
table_dump Datatable, 100 table_exists Datatable, 101 table_get Datatable, 101 table_get Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_col_idx Datatable, 101 table_get_row_lods Datatable, 103 table_get_row_ids Datatable, 104 table_size Datatable, 104 table_size Datatable, 104 table_sum Datatable, 104 tetrahedron 2D Shapes, 81 tol_edge.scad, 204 Triangle_Dpp2vl triangle_Ul2vp Triangle Solutions, 148 triangle_Vl2vp Triangle Solutions, 149 triangle_Vl2vp Triangle Solutions, 149 Triangle Solutions, 149 Triangle_Vl2vp Triangle Solutions, 149 Triangle_vp 2D Shapes, 73 triangle_vp 2D Shapes, 73 triangle_vp 2D Shapes, 73 Triangle Solutions, 149 Triangle Solutions, 149 Triangle_Solutions, 149 Triangle_controid_vp, 146 triangle_centroid_vp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_inradius_vp, 148 Units, 150 Units, 150		
Datatable, 100 table_exists Datatable, 101 table_get Datatable, 101 table_get Datatable, 101 table_get Datatable, 101 table_get_col Datatable, 101 table_get_col_idx Datatable, 101 table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 triangle_ill2vp Triangle Solutions, 148 triangle_jill2vp Triangle_Solutions, 148 triangle_ppp 2D Shapes, 71 triangle_spp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_ppp 2U triangle_ppp 2U triangle_ppp 2U triangle_vl tetrahedron Datatable, 104 tetrahedron Datatable, 105 triangle_vl Datatable, 106 triangle_vl Datatable, 107 triangle_solutions, 148 triangle_vl Datatable, 109 triangle_vl Datatable, 103 tri		
table_exists Datatable, 101 table_get Datatable, 101 table_get Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_col, 01 Datatable, 101 table_get_row Datatable, 103 triangle_lal table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 triangle_lll 2D Shapes, 70 triangle_lll 2D Shapes, 71 triangle_lll2vp Triangle_Solutions, 148 triangle_ppp 2D Shapes, 71 triangle_solutions, 148 triangle_ppp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_ppp 3D Shapes, 81 tool_edge.scad, 204 Tools, 142 torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 81 torus_rp 3D Shapes, 82 torus_tp 3D Shapes, 83 triangle_vpv Triangle Solutions, 149 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incadius_ppp, 148 Units, 150 Units, 150	- ·	•
Datatable, 101 table_get Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 104 triangle_ppp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_ppp 2D Shapes, 72 triangle_vp 2D Shapes, 72 triangle_vl Triangle Solutions, 148 triangle_vl Triangle Solutions, 148 triangle_vl Triangle Solutions, 149 triangle_vl Tools, 142 triangle_vl Torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 82 trous_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 148 Units, 150 triangle_incenter_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 148 Units, 150		·
table_get Datatable, 101 table_get_col Datatable, 101 table_get_col Datatable, 101 table_get_colidx Datatable, 101 table_get_col_idx Datatable, 101 table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 104 table_get_row_idx Datatable, 105 table_get_row_idx Datatable, 106 Triangle Solutions, 148 Datatable, 107 triangle Solutions, 148 triangle_ppp Triangle Solutions, 148 triangle_vpp Triangle Solutions, 148 triangle_vl2vp Triangle Solutions, 148 triangle_vl2vp Triangle Solutions, 149 triangle_vl2vp Triangle Solutions, 149 triangle_vp 3D Shapes, 81 trus_rp 3D Shapes, 81 triangle_vp 2D Shapes, 73 triangle_vp 4D Shapes, 73 triangle_vp 4D Shapes, 73 triangle_vp 4D Shapes, 73 triangle_vp 4D Shapes, 73 triangle_vp2vl Triangle_solutions, 149 Triangle_centroid_ppp, 146 triangle_incenter_up, 146 triangle_incenter_up, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 148 Units, 150 Units, 150		
Datatable, 101 table_get_col Datatable, 101 table_get_col_idx Datatable, 101 table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 triangle_lll 2D Shapes, 70 triangle_lll2vp Triangle_Solutions, 148 triangle_pepp 2D Shapes, 71 triangle_ppp 2D Shapes, 72 Triangle Solutions, 148 triangle_ppp 2D Shapes, 72 triangle_ppp2vl table_sum Datatable, 104 tetrahedron 2D Shapes, 72 Triangle Solutions, 148 triangle_vl 2D Shapes, 72 triangle_vl triangle_vl triangle_vl triangle_vl Triangle Solutions, 149 trious_ep 3D Shapes, 81 triangle_vl triangle_vp 2D Shapes, 73 triangle_vp 2D Shapes, 73 triangle_vp 2D Shapes, 73 triangle_vp Triangle Solutions, 149 triangle_vp Triangle Solutions, 149 Triangle Solutions, 149 Triangle_vp Triangle Solutions, 149 triangle_vp 2D Shapes, 73 triangle_vp Triangle Solutions, 149 Triangle Solutions, 149 Triangle_vp Triangle Solutions, 149 Triangle_vp Triangle Solutions, 149 Triangle_vp Triangle Solutions, 149 Triangle Inradius_pp, 146 triangle_incenter_pp, 146 triangle_incenter_pp, 146 triangle_incenter_pp, 146 triangle_inradius_pp, 146 Triangle_in		
table_get_col Datatable, 101 table_get_col_idx Datatable, 101 table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 104 triangle_ppp table_size Datatable, 104 table_sum Datatable, 104 tetrahedron 2D Shapes, 72 Triangle Solutions, 148 triangle_vl2vp Triangle Solutions, 148 triangle_vl2vp Triangle Solutions, 149 triangle_vl2vp Triangle Solutions, 149 trius_ep 2D Shapes, 81 torus_ep 2D Shapes, 81 torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 82 torus_tp 3D Shapes, 82 torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_inradius_pp, 148 triangle_inradius_pp, 148 Triangle_inradius_pp, 148 Units, 150 Triangle Solution and Plane, 129 Units, 150		•
Datatable, 101 table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_dos Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 104 table_size Datatable, 104 tetrahedron Datatable, 105 triangle_vl triangle_vl Triangle Solutions, 148 triangle_vl triangle_vl triangle_vl triangle_vl triangle_vl Triangle Solutions, 149 triangle_vp Dabapes, 73 triangle_vp Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 145 triangle_centroid_pp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 146 triangle_inradius_pp, 148 Units, 150		· - ·
table_get_col_idx Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Triangle_Ill2vp Triangle_Solutions, 148 triangle_ppp table_size Datatable, 103 table_size Datatable, 104 table_sum Datatable, 104 tetrahedron 3D Shapes, 81 tool_edge.scad, 204 Triangle_vl2vp tool_edge.scad, 204 Triangle_vl2vp Triangle_vl_c 2D Shapes, 72 triangle_vl_c 2D Shapes, 73 triangle_vl_c 2D Shapes, 73 triangle_vp2vl Triangle_Solutions, 149 Trius_rp 3D Shapes, 82 triangle_vp2vl triangle_vp2vl triangle_vp2vl Triangle Solutions, 149 Triangle Solutions, 145 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_inradius_pp, 148 Triangle_inradius_pp, 146 Triangle_inradius_pp, 146 Triangle_inradius_pp, 148 Triangle_inradius_pp, 148 Triangle_inradius_pp, 148 Triangle_inradius_pp, 148 Triangle_inradius_pp, 146 Triangle_inradius_pp, 148		_
Datatable, 101 table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Triangle_Ill2vp Triangle_Solutions, 148 Datatable, 103 table_size Datatable, 104 table_sum Datatable, 104 tetrahedron Datatable, 104 tetrahedron Datatable, 104 tetrahedron Datatable, 104 tetrangle_vl tool_edge.scad, 204 Triangle Solutions, 149 Tools, 142 torus_ep DShapes, 81 triangle_vl torus_rp DShapes, 81 triangle_vl triangle_vl DShapes, 73 Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 149 Triangle_vp2vl triangle_vp2vl Triangle_Solutions, 149 Triangle_solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 148 Units, 150		-
table_get_row Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_cols Datatable, 103 triangle_lll 2D Shapes, 70 triangle_lll 2D Shapes, 71 triangle_lll2vp Triangle_lll2vp triangle_lll2vp triangle_lll2vp triangle_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_size Datatable, 104 table_size Datatable, 104 tetrahedron Datatable, 105 triangle_vl triangle_vl triangle_vl triangle_vl Triangle Solutions, 148 triangle_vl Dols, 142 triangle_vl Dols, 142 triangle_vl Dols, 142 triangle_vp Dabapes, 73 triangle_vp Dabapes, 73 triangle_vp Dabapes, 73 triangle_vp Unique Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 145 triangle_centroid_pp, 146 triangle_centroid_pp, 146 triangle_inradius_ppp, 146 Triangle_inradius_pp, 148 Units, 150		•
Datatable, 103 table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Datatable, 103 table_git_row_idx Datatable, 103 table_size Datatable, 104 table_size Datatable, 104 tetrahedron Datatable, 104 tetrahedron 3D Shapes, 81 tool_edge.scad, 204 Tools, 142 torus_ep 3D Shapes, 81 torus_rp 2D Shapes, 73 triangle_vp torus_tp 3D Shapes, 82 torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle_centroid_ppp, 146 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 148 Units, 150	•	-
table_get_row_cols Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_ids Datatable, 103 table_get_row_idx Datatable, 103 table_get_row_idx Triangle_lll2vp table_get_row_idx Datatable, 103 table_size Datatable, 104 table_sum Datatable, 104 tetrahedron Datatable, 104 tetrangle_vl zb shapes, 72 triangle_vl zvp Triangle Solutions, 148 triangle_vl zvp Triangle Solutions, 149 triangle_vp Datatable, 104 triangle_vl zb shapes, 72 triangle_vl zb shapes, 73 triangle_vl zc bhapes, 73 triangle_vp ZD Shapes, 73 triangle_vp ZD Shapes, 73 triangle_vp ZD Shapes, 73 triangle_vp Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 148 Units, 150	— -	•
Datatable, 103 table_get_row_ids		-
table_get_row_ids Datatable, 103 triangle_lll2vp Triangle Solutions, 148 Datatable, 103 triangle_ppp table_size Datatable, 104 table_sum Datatable, 104 tetrahedron 3D Shapes, 81 tool_edge.scad, 204 Tools, 142 torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 82 triangle_vp 3D Shapes, 82 triangle_vp torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 149 Triangle_vp 2D Shapes, 73 Triangle Solutions, 149 Triangle_vp Triangle_vp Triangle_Solutions, 149 Triangle_Solutions, 149 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_solutions, 149 Triangle_centroid_pp, 146 triangle_centroid_pp, 146 triangle_incenter_ppp, 146 triangle_incenter_pp, 146 triangle_incenter_pp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 148 Units, 150	— - — —	•
Datatable, 103 table_get_row_idx		- —
table_get_row_idx Datatable, 103 triangle_ppp table_size Datatable, 104 triangle_ppp table_size Datatable, 104 triangle_ppp2vl table_sum Datatable, 104 tetrahedron 3D Shapes, 81 triangle_vl2vp tool_edge.scad, 204 Triangle Solutions, 149 Tools, 142 torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 82 triangle_vp torus_tp 3D Shapes, 82 triangle_vp2vl torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_vpp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Triangle_inradius_vp, 148 Triangle_inradius_vp, 148 Units, 150 Triangle Solutions, 142 Units, 150	— - — —	•
Datatable, 103 table_size		- - ·
table_size	— - — —	_
Datatable, 104 table_sum Datatable, 104 tetrahedron 3D Shapes, 81 triangle_vl2vp tool_edge.scad, 204 Triangle Solutions, 149 Tools, 142 torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 82 torus_tp 3D Shapes, 82 torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 149 Triangle Solutions, 149 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 148 Units, 150		
table_sum Datatable, 104 tetrahedron 3D Shapes, 81 triangle_vl2vp tool_edge.scad, 204 Triangle_solutions, 149 Tools, 142 torus_ep 3D Shapes, 81 triangle_vl_c 2D Shapes, 73 3D Shapes, 81 triangle_vl_c 2D Shapes, 73 3D Shapes, 81 triangle_vp torus_rp 2D Shapes, 73 3D Shapes, 82 triangle_vp2vl torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Units, 150	-	•
tetrahedron 2D Shapes, 72 3D Shapes, 81 triangle_vlvp tool_edge.scad, 204 Triangle Solutions, 149 Tools, 142 triangle_vl_c torus_ep 2D Shapes, 73 3D Shapes, 81 triangle_vp torus_rp 2D Shapes, 73 3D Shapes, 82 triangle_vp torus_tp 2D Shapes, 73 triangle_vp2vl torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Vector Operations, 177 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Units, 150		
tetrahedron 2D Shapes, 72 3D Shapes, 81 triangle_vl2vp Tools, 142 torus_ep 2D Shapes, 73 3D Shapes, 81 triangle_vl_c 2D Shapes, 73 3D Shapes, 81 triangle_vp 2D Shapes, 73 3D Shapes, 81 triangle_vp 2D Shapes, 73 3D Shapes, 82 triangle_vp 2D Shapes, 73 Triangle Solutions, 149 3D Shapes, 82 triangle_vp2vl torus_tp Triangle Solutions, 149 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Units, 150		_
triangle_vl2vp tool_edge.scad, 204 Tools, 142 triangle_vl_c torus_ep	,	-
tool_edge.scad, 204 Triangle Solutions, 149 Tools, 142 torus_ep		
Tools, 142 torus_ep 3D Shapes, 81 torus_rp 3D Shapes, 82 torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_pp, 146 triangle_inradius_vp, 148 triangle_vp_c 2D Shapes, 73 triangle_vp_vp	•	- - •
torus_ep 2D Shapes, 73 3D Shapes, 81 triangle_vp torus_rp 2D Shapes, 73 3D Shapes, 82 triangle_vp2vl torus_tp Triangle Solutions, 149 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Unit_v Point, Vector and Plane, 129 Units, 150	_ · •	_
torus_rp 2D Shapes, 73 3D Shapes, 82 triangle_vp2vl torus_tp Triangle Solutions, 149 3D Shapes, 83 transform.scad, 206 Transformations, 143 Vector Operations, 177 Triangle Solutions, 145 unit_angle_name triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 Unit_v Triangle_inradius_vp, 148 Units, 150	Tools, 142	
torus_rp 2D Shapes, 73 3D Shapes, 82 triangle_vp2vl torus_tp Triangle Solutions, 149 3D Shapes, 83 transform.scad, 206 Transformations, 143 Vector Operations, 177 Triangle Solutions, 145 unit_angle_name triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 Unit_v Point, Vector and Plane, 129 triangle_inradius_vp, 148 Units, 150		
triangle_vp2vl torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148	·	
torus_tp 3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Triangle_inradius_vp, 148 Triangle_solutions, 147 Unit_angle_name Angle, 86 Unit_length_name Length, 117 Unit_v Point, Vector and Plane, 129 Units, 150	- ·	
3D Shapes, 83 transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Unit_v Point, Vector and Plane, 129 Units, 150	·	
transform.scad, 206 Transformations, 143 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Unit_v		Triangle Solutions, 149
Transformations, 143 Vector Operations, 177 Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Vector Operations, 177 unit_angle_name Angle, 86 unit_length_name Length, 117 unit_v Point, Vector and Plane, 129 Units, 150	•	
Triangle Solutions, 145 triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 148 vinit_v triangle_inradius_vp, 148 vinit_v vinit_s, 150		•
triangle_centroid_ppp, 146 triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Angle, 86 unit_length_name Length, 117 unit_v Point, Vector and Plane, 129 Units, 150		•
triangle_centroid_vp, 146 triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 unit_v Point, Vector and Plane, 129 Units, 150	-	_ • _
triangle_incenter_ppp, 146 triangle_incenter_vp, 146 triangle_inradius_ppp, 146 triangle_inradius_vp, 148 Length, 117 unit_v Point, Vector and Plane, 129 Units, 150	- · · ·	_
triangle_incenter_vp, 146 unit_v triangle_inradius_ppp, 146 Point, Vector and Plane, 129 triangle_inradius_vp, 148 Units, 150	· - ·	
triangle_inradius_ppp, 146 Point, Vector and Plane, 129 triangle_inradius_vp, 148 Units, 150		•
triangle_inradius_vp, 148 Units, 150		_
triangle_lll2vp, 148 units_angle.scad, 208	- - ·	
triangle_ppp2vl, 148 units_length.scad, 209		
triangle villum 140	triangle_vl2vp, 149	Utilities, 151
UNATION VIZVD, 145 UNITIONS LOT	···	

	stack, 151		pmerge, 174
	validate, 151		qsort, 174
utilit	ies.scad, 210		qsort2, 175
			reverse, 175
valic			rselect, 175
	Utilities, 151		second, 176
	dation.scad, 212		smerge, 176
Vari	able Tests, 154		strip, 176
	all_defined, 155		sum, 176
	all_equal, 155		unique, 177
	all_len, 157		vstr, 177
	all_numbers, 157	vstr	V - 0 - 1' - 477
	all_scalars, 157		Vector Operations, 177
	all_strings, 158		
	all_vectors, 158		
	almost_equal, 158		
	any_equal, 159		
	any_undefined, 159		
	compare, 159		
	is_boolean, 160		
	is_decimal, 160		
	is_defined, 161		
	is_empty, 161		
	is_even, 161		
	is_inf, 161		
	is_integer, 162 is_iterable, 162		
	is_nan, 162		
	is_number, 162		
	is_odd, 163		
	is_range, 163		
	is_scalar, 163		
	is_string, 164		
	is_vector, 164		
	not defined, 164		
Vect	for Operations, 165		
	ciselect, 166		
	cmvselect, 167		
	consts, 167		
	count, 167		
	defined_or, 169		
	delete, 169		
	eappend, 169		
	edefined_or, 170		
	eselect, 170		
	exists, 171		
	find, 171		
	first, 172		
	insert, 172		
	last, 173		
	nfirst, 173		
	nhead, 173		
	nlast, 173		
	ntail, 174		