Design Tools for Satellite Design

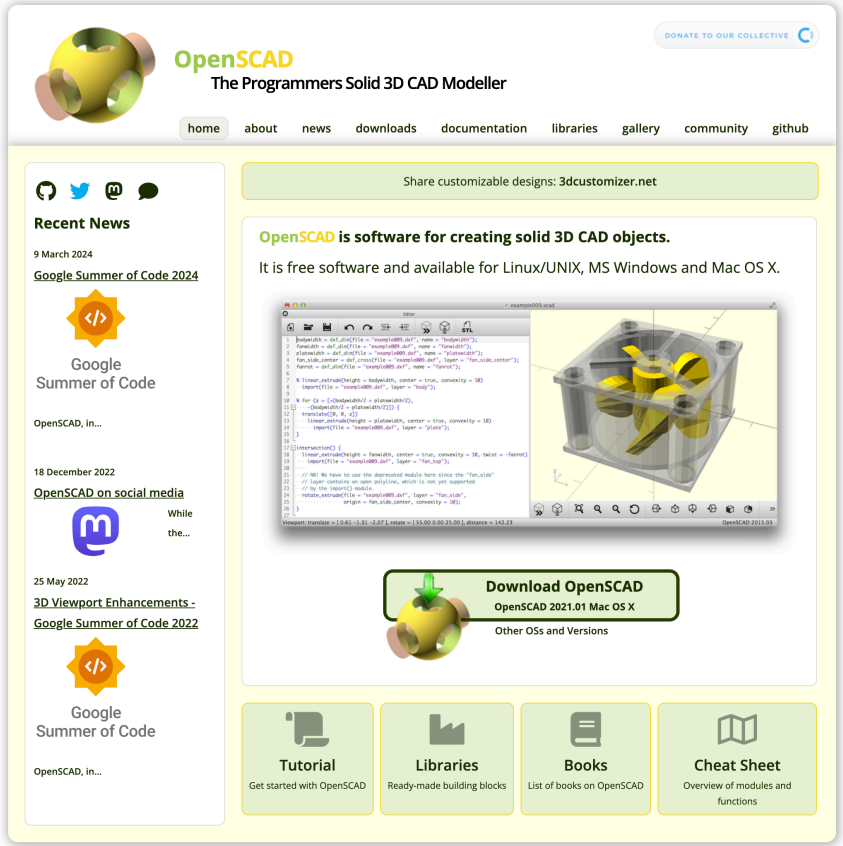
Daniel Topa
daniel.topa@hii-tsd.com

Huntington Ingalls Industries
Mission Technologies

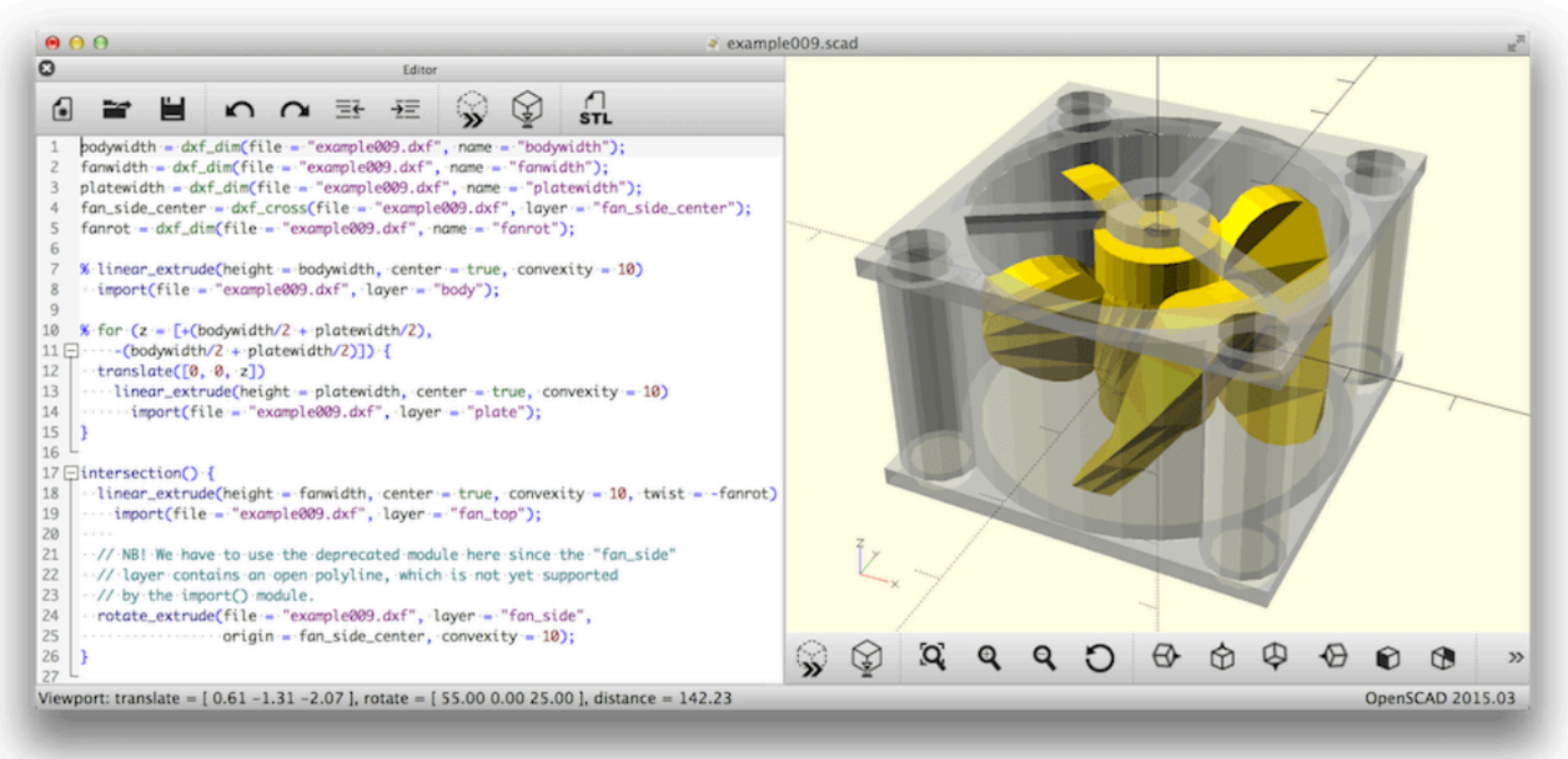
October 26, 2024

Overview

The Programmers Solid 3D CAD Modeller



OpenSCAD is Pythonic



OpenSCAD Tutorials

OpenSCAD Libraries

OpenSCAD Book

OpenSCAD Cheat Sheet

OpenSCAD v2021.01

Syntax

var = value;
var = cond ? value_if_true : value_if_false;
var = function (x) x + x;
module name(…) { … }
name();
function name(…) = …
name();
include <….scad>
use <….scad>

Constants

undef — undefined value
PI — mathematical constant π (~3.14159)

Operators

n + m — Addition
n - m — Subtraction
n * m — Multiplication
n / m — Division
n % m — Modulo
n ^ m — Exponentiation
n < m — Less Than
n <= m — Less or Equal
b == c — Equal
b != c — Not Equal
n >= m — Greater or Equal
n > m — Greater Than
b && c — Logical And
b || c — Logical Or
!b — Negation

Special variables

$fa — minimum angle
$fs — minimum size
$fn — number of fragments
$t — animation step
$vpr — viewport rotation angles in degrees
$vpt — viewport translation
$vpd — viewport camera distance
$vpf — viewport camera field of view
$children — number of module children
$preview — true in F5 preview, false for F6

Modifier Characters

* — disable
! — show only
— highlight / debug
% — transparent / background

2D

circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font,
halign, valign, spacing,
direction, language, script)
import("….ext", convexity)
projection(cut)

3D

sphere(radius | d=diameter)
cube(size, center)
cube([width,depth,height], center)
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, faces, convexity)
import("….ext", convexity)
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(angle,convexity)
surface(file = "….ext",center,convexity)

Transformations

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

Lists

list = […, …, …]; create a list
var = list[2]; index a list (from 0)
var = list.z; dot notation indexing (x/y/z)

Boolean operations

union()
difference()
intersection()

List Comprehensions

Generate [for (i = *range*|*list*) i]
Generate [for (*init*;*condition*;*next*) i]
Flatten [each i]
Conditions [for (i = …) if (condition(i)) i]
Conditions [for (i = …) if (condition(i)) x else y]
Assignments [for (i = …) let (assignments) a]

Flow Control

for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
for (i = […,…,…]) { … }
for (i = …, j = …, …) { … }
intersection_for(i = [start:end]) { … }
intersection_for(i = [start:step:end]) { … }
intersection_for(i = […,…,…]) { … }
if (…) { … }
let (…) { … }

Type test functions

is_undef
is_bool
is_num
is_string
is_list
is_function

Other

echo(…)
render(convexity)
children([idx])
assert(condition, message)
~~assign (…) { … }~~

Functions

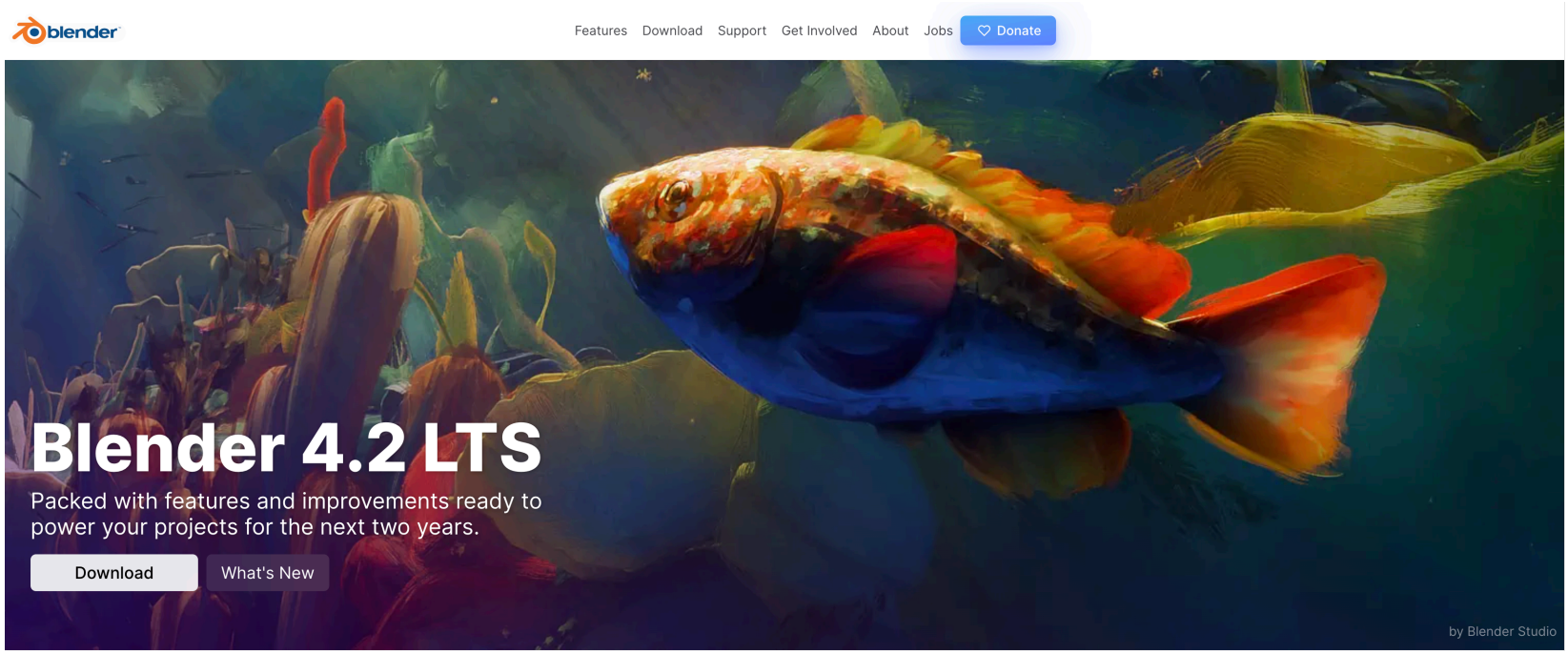
concat
lookup
str
chr
ord
search
version
version_num
parent_module(idx)

Mathematical

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

Links: Official website | Code | Issues | Manual | MCAD library | Mailing list | Other links

Sample Texts



User Manual **Developer Documentation** **Python API**

Primitives

Primitives

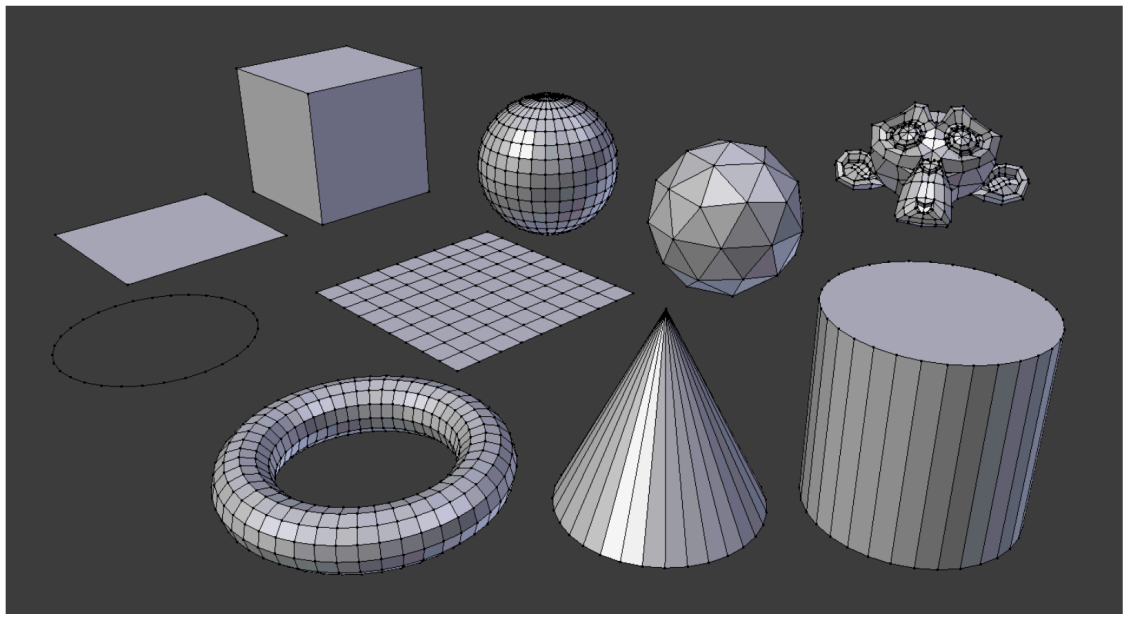
Reference

Mode: Object Mode and Edit Mode

Menu: Add ▸ Mesh

Shortcut: Shift−A

A common object type used in a 3D scene is a mesh. Blender comes with a number of "primitive" mesh shapes that you can start modeling from. You can also add primitives in Edit Mode at the 3D cursor.



Blender's standard primitives.

Stereolithography File Format

STL

Reference

Category: Import-Export

Menu: File ▸ Import/Export ▸ Stl (.stl)

The STL-file format is useful if you intend to import/export the files for CAD software. It is also commonly used for loading into 3D printing software.

Importing

General

Scale

Value by which to scale the imported objects in relation to the world's origin.

Scene Unit

Apply current scene's unit (as defined by unit scale) to imported data.

Forward / Up Axis

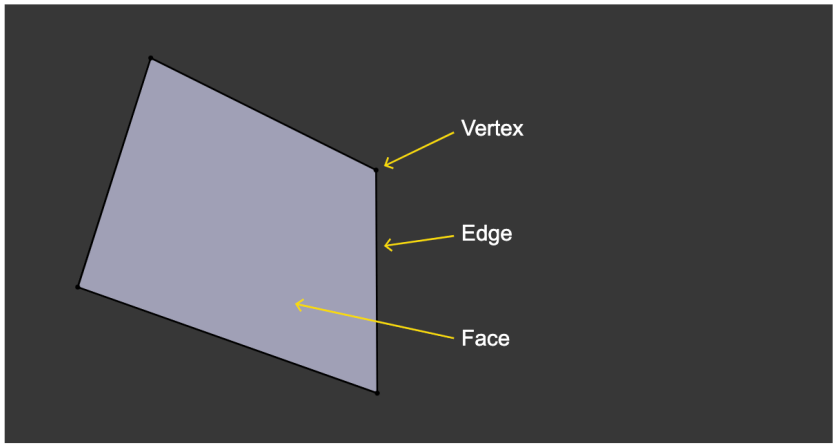
Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, it is common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Mesh Basics

Structure

With meshes, everything is built from three basic structures: *vertices*, *edges* and *faces*.



Example of mesh structure. #

Normals

Editing Custom Split Normals

Reference

Mode: Edit Mode
Menu: Mesh ▸ Normals
Shortcut: Alt–N

There are a number of tools for editing custom split normals. The custom normal mesh edit tools can affect all normals (the default), or only selected ones. To select a custom normal associated with a particular vertex and face:

- Make the element selection mode both Vertex and Face (use Shift–LMB to enable the second one).
- Select one or more vertices, then select a face. This can be repeated to select more vertices and a different face and so on. It is easiest to see the effect of these tools if you turn on the Edit Mode Overlays option *Display vertex-per-face normals as lines*.

See also

Editing Normals.

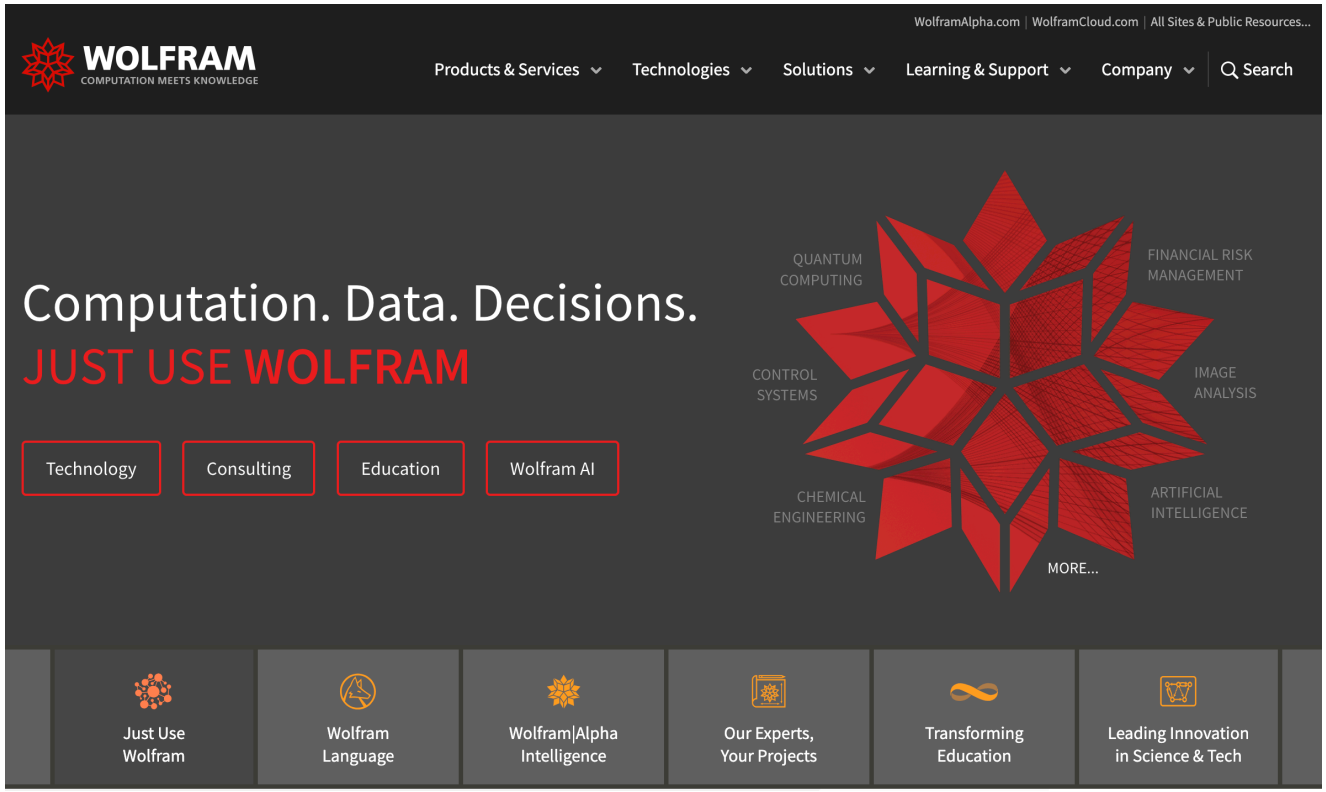
Importing Custom Split Normals

Some tools, particularly those used in CAD, tend to generate irregular geometry when tessellating their objects into meshes (very thin and long triangles, etc.). Auto-computed normals on such geometry often gives bad artifacts, so it is important to be able to import and use the normals as generated by the CAD tool itself.

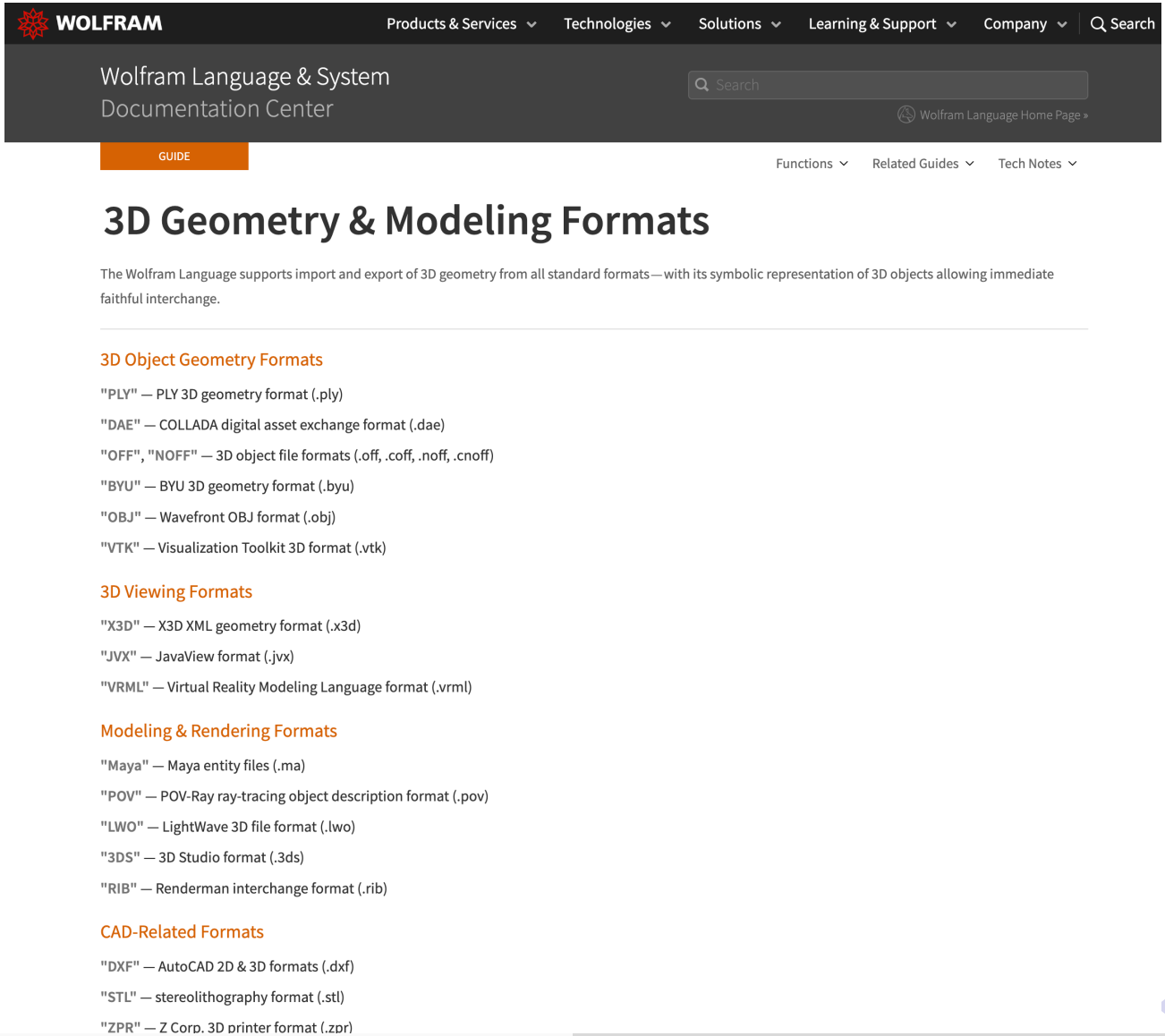
Note

Currently, only the FBX Importer and Alembic Importer are capable of importing custom normals.

Wolfram - née Mathematica



3D Geometry and Modeling Formats



WOLFRAM | Products & Services | Technologies | Solutions | Learning & Support | Company | Search

Wolfram Language & System
Documentation Center

Search

Wolfram Language Home Page »

GUIDE

Functions | Related Guides | Tech Notes

3D Geometry & Modeling Formats

The Wolfram Language supports import and export of 3D geometry from all standard formats—with its symbolic representation of 3D objects allowing immediate faithful interchange.

3D Object Geometry Formats

"PLY" — PLY 3D geometry format (.ply)

"DAE" — COLLADA digital asset exchange format (.dae)

"OFF", "NOFF" — 3D object file formats (.off, .coff, .noff, .cnoff)

"BYU" — BYU 3D geometry format (.byu)

"OBJ" — Wavefront OBJ format (.obj)

"VTK" — Visualization Toolkit 3D format (.vtk)

3D Viewing Formats

"X3D" — X3D XML geometry format (.x3d)

"JVX" — JavaView format (.jvx)

"VRML" — Virtual Reality Modeling Language format (.vrml)

Modeling & Rendering Formats

"Maya" — Maya entity files (.ma)

"POV" — POV-Ray ray-tracing object description format (.pov)

"LWO" — LightWave 3D file format (.lwo)

"3DS" — 3D Studio format (.3ds)

"RIB" — Renderman interchange format (.rib)

CAD-Related Formats

"DXF" — AutoCAD 2D & 3D formats (.dxf)

"STL" — stereolithography format (.stl)

"ZPR" — Z Corp. 3D printer format (.zpr)

Algorithmically Generate 3D Models

‹ 3D Printing

Algorithmically Generate 3D Models

The Wolfram Language core principles of automation and unification make it easy to generate geometry and 3D-printable models algorithmically better than ever.

Create 3D-printable objects from 2D images.

```
In[1]:= extrudeImage[image_] :=
          Block[{res, img},
            img = DeleteSmallComponents[Binarize[image, 0.9], 500];
            res = ImageMesh[ColorNegate[img]];
            RegionProduct[res, Line[{{0.}, {50.}}]]
          ]
```

```
In[2]:= extrudeImage /@ {
```

```
}
```

```
Out[2]= {
```

```
}
```

Mesh Repair

‹ 3D Printing

Automatically Detect Defects and Repair Meshes

Version 11 integrates fully automated detection of mesh defects in 3D models and provides repair functionality.

HoleEdges edges around a hole in the surface	**TJunctionEdges** edges that form a T junction
TinyFaces faces with near-zero area	**OverlappingFaces** faces that overlap
IsolatedVertices vertices without incident edges	**DanglingEdges** edges without incident faces
SingularEdges edges with more than two incident faces	**SingularVertices** vertices with a non-disc neighborhood
TinyComponents tiny connected mesh components	**FlippedFaces** faces that point inward

Find defects in a 3D model.

```
In[1]:= mesh = ExampleData[{"Geometry3D", "StanfordBunny"}, "Region"];
        FindMeshDefects[mesh]
```

Out[1]=

● hole edges

Reconstruct Models from 3D Data

‹ 3D Printing

Reconstruct Models from 3D Data

Version 11 includes state-of-the-art surface reconstruction from arbitrary 3D data.

```
In[1]:= pointcloud = ExampleData[{"Geometry3D", "StanfordBunny"}, "VertexData"];
        Graphics3D[Point[RandomSample[pointcloud, 1000]]]
```

Out[1]=

Reconstruct a 3D-printable Stanford bunny.

```
In[2]:= ListSurfacePlot3D[pointcloud, MaxPlotPoints → 50, Axes → None,
          Boxed → False, Mesh → None]
```

Out[2]=

Properties of Models, Materials, and Costs

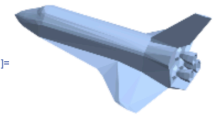
‹ 3D Printing

Get Properties of Models, Materials, and Costs

Leveraging built-in geometric computation and the extensive Wolfram Knowledgebase, properties of 3D models and materials can be computed.

```
In[1]:= model = ExampleData[{"Geometry3D", "SpaceShuttle"}, "Region"]
```

Out[1]=



Mass of the model in aluminum.

```
In[2]:= volume = Quantity[Volume[model], "Centimeters"^3]
Out[2]= 55.5217 cm^3
In[3]:= aluminum (element) ["Density"] * volume
Out[3]= 149.909 g
```

Import 3D Models

‹ 3D Printing

Import 3D Models

Version 11 supports import and export of 3D geometry from all standard formats.

STL	PLY	OBJ	OFF	DXF	DAE	ZPR	VTK	BYU
X3D	JVX	VRML	Maya	POV	LWO	3DS	RIB	XYZ

Import the *Bust of Eleonora Duse*—at the Gallery of Modern Art of the Palazzo Pitti in Florence, Italy—from My Mini Factory repository.

```
In[1]:= Import["https://myminifactory.net/uploads/object-files/1407
          d8e1e51a0cb2fe013fe1fff83fea31cbd224.stl"]
```

Out[1]=

Ready-Made 3D-Printable Models

‹ 3D Printing

Ready-Made 3D-Printable Models

Direct access to a large volume of curated 3D-printable objects, specially organized and created for the Wolfram Language.

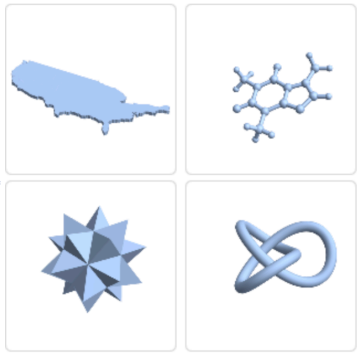
In[1]:= **skull** (anatomical structure) ··· ✓ ["Region"]

Out[1]=

From geographical features to knots.

⊞ show complete Wolfram Language input

Out[2]=

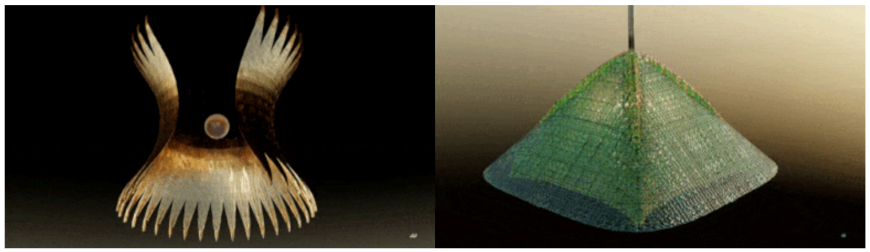


Animation with Blender

Animating Wolfram surfaces with Blender: Mesh deforming & Superellipse

Guenther Gsaller, Retired

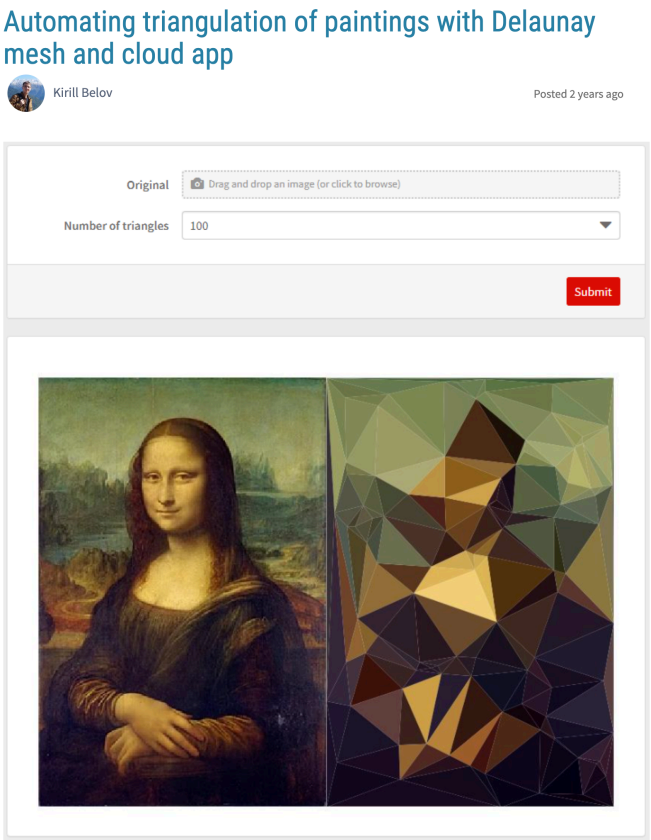
Posted 2 years ago



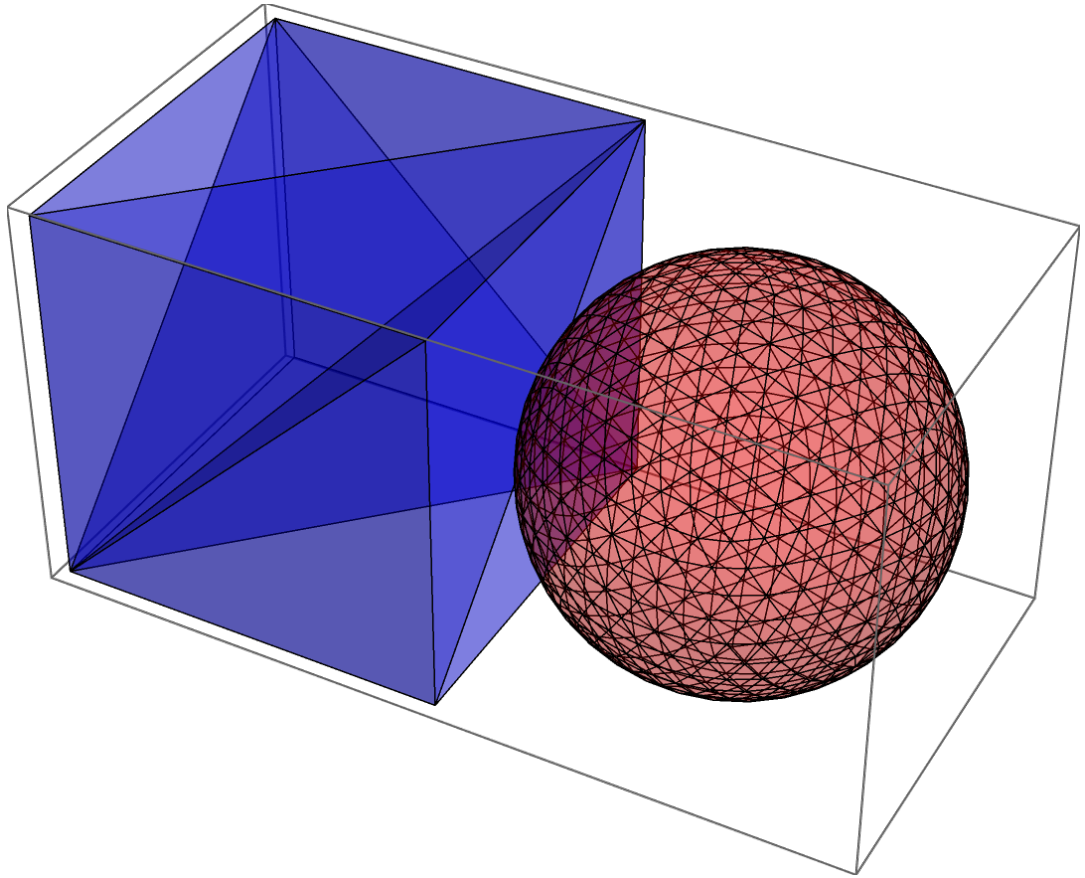
Animating Wolfram surfaces with Blender: Mesh deforming & Superellipse
by Guenther Gsaller

This is a continuation to the posts listed at: https://community.wolfram.com/web/ggoff/

Triangulation of paintings with Delaunay mesh

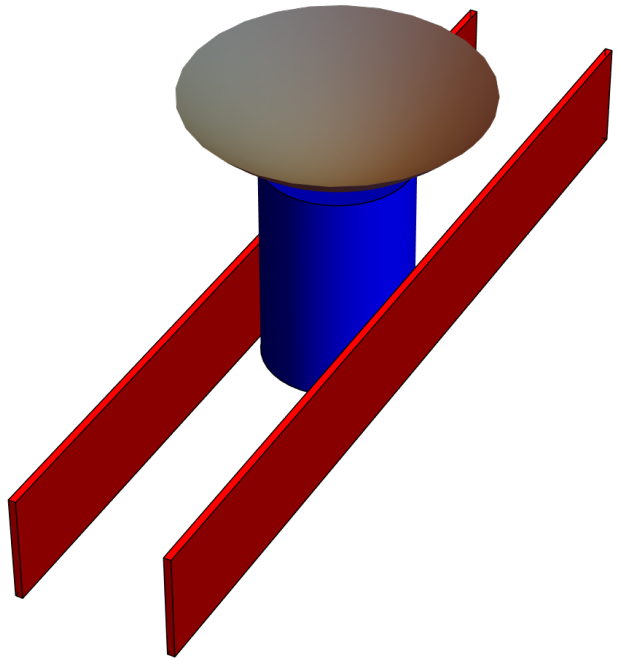


Simple Shapes



Simple Shapes

DanielSat©



Bibliography I

Design Tools for Satellite Design

Daniel Topa
daniel.topa@hii-tsd.com

Huntington Ingalls Industries
Mission Technologies

October 26, 2024