



Design Tools for Satellite Design

Daniel Topa
daniel.topa@hii-tsd.com

Huntington Ingalls Industries
Mission Technologies

October 26, 2024



Overview

1 Radar Basics

2 CAD: Open Source

3 Mathematica



The Programmers Solid 3D CAD Modeller

The screenshot shows the official website for OpenSCAD. At the top, there's a navigation bar with links for home, about, news, downloads, documentation, libraries, gallery, community, and github. A "DONATE TO OUR COLLECTIVE" button is also present. Below the navigation, there's a "Recent News" section with items from March 2024, December 2022, and May 2022. To the right of the news is a large banner for "Share customizable designs: 3dcustomizer.net". The banner text says "OpenSCAD is software for creating solid 3D CAD objects. It is free software and available for Linux/UNIX, MS Windows and Mac OS X." Below the banner is a screenshot of the OpenSCAD software interface showing a 3D model of a complex mechanical part. A "Download OpenSCAD" button is located below the banner, along with links for "Other OSs and Versions". At the bottom of the page are four buttons for "Tutorial", "Libraries", "Books", and "Cheat Sheet".

OpenSCAD is Pythonic

The screenshot shows the OpenSCAD 2015.03 application window. On the left is the "Editor" pane displaying the SCAD source code for "example009.scad". On the right is the "Preview" pane showing a 3D rendering of a fan assembly within a rectangular frame. The code uses various OpenSCAD functions like `linear_extrude`, `intersection()`, and `rotate_extrude` to define the geometry.

```
1 bodywidth = dxf_dim(file = "example009.dxf", name = "bodywidth");
2 fanwidth = dxf_dim(file = "example009.dxf", name = "fanwidth");
3 platewidth = dxf_dim(file = "example009.dxf", name = "platewidth");
4 fan_side_center = dxf_cross(file = "example009.dxf", layer = "fan_side_center");
5 fanrot = dxf_dim(file = "example009.dxf", name = "fanrot");
6
7 % linear_extrude(height = bodywidth, center = true, convexity = 10)
8 import(file = "example009.dxf", layer = "body");
9
10 % For (z = [(bodywidth/2 + platewidth/2),
11 %----- (bodywidth/2 + platewidth/2)) {
12 %----- translate([0, 0, z])
13 %----- linear_extrude(height = platewidth, center = true, convexity = 10)
14 %----- import(file = "example009.dxf", layer = "plate");
15 }
16
17 intersection() {
18   linear_extrude(height = fanwidth, center = true, convexity = 10, twist = -fanrot)
19   import(file = "example009.dxf", layer = "fan_top");
20
21 // NB! We have to use the deprecated module here since the "fan_side"
22 // layer contains an open polyline, which is not yet supported
23 // by the import() module.
24 rotate_extrude(file = "example009.dxf", layer = "fan_side",
25                 origin = fan_side_center, convexity = 10);
26 }
27
```

Viewport: translate = [0.61 -1.31 -2.07], rotate = [55.00 0.00 25.00], distance = 142.23

OpenSCAD 2015.03



OpenSCAD Tutorials

The screenshot shows the official OpenSCAD website. At the top, there's a navigation bar with links for home, about, news, downloads, documentation (which is highlighted in blue), libraries, gallery, community, and github. To the left, there's a sidebar with links for Documentation, Books, Videos, and Articles / Blogs. The main content area features a large button labeled "Documentation" with a scroll icon, followed by "Books", "Videos", and "Articles / Blogs". Below this is a section titled "OpenSCAD Tutorial" with a "Table of Contents" containing 9 chapters. The footer of the website includes standard navigation icons for back, forward, search, and refresh.

OpenSCAD
The Programmers Solid 3D CAD Modeller

home about news downloads documentation libraries gallery community github

Documentation

- OpenSCAD Tutorial
- OpenSCAD User Manual
- OpenSCAD Language Reference
- Code Cheat Sheet

Books

- English
- German / Deutsch
- Spanish / Español

Videos

- OpenSCAD: Introduction

Articles / Blogs

- How to use OpenSCAD
- 3D Spielplatz (german)
- OpenSCAD Tutorial Series

Documentation

Books

Videos

Articles / Blogs

OpenSCAD Tutorial

Table of Contents

1. Chapter 1: A few words about OpenSCAD and getting started with the first object
2. Chapter 2: Scaling the model and first steps for parameterizing models
3. Chapter 3: Resizing models and more ways of combining objects
4. Chapter 4: Introducing modules to organize the code
5. Chapter 5: Using multiple scripts and libraries
6. Chapter 6: Control flow, conditional creation of objects
7. Chapter 7: Loops and creating more complex patterns
8. Chapter 8: Extruding 2D shapes into 3D objects
9. Chapter 9: Math, calculations and low level geometry creation



OpenSCAD Libraries

The screenshot shows the homepage of the OpenSCAD Libraries website. At the top, there's a navigation bar with links for home, about, news, downloads, documentation, libraries (which is the active tab), galleries, community, and github. A "DONATE TO OUR COLLECTIVE" button is also present. The main content area is divided into two columns.

Libraries

- General
 - BOSL
 - BOSL2
 - dotSCAD
 - NopSCADlib
 - UB.scad
 - Functional OpenSCAD
 - Constructive
 - BOLTS
 - Asset Collection
- Single Topic
 - Round Anything
 - Mark's Enclosure Helper
 - funcutlits
 - threads.scad Module
 - Smooth Primitives Library
 - Function Plotting Library
 - ClosePoints Library
 - Tray Library
 - YAPP Generator
 - STEMFIE Parts Library
 - Catch'n'Hole
 - Pathbuilder
 - Altair's 2D Library

General

BOSL
The Belfry OpenScad Library - A library of tools, shapes, and helpers to make OpenScad easier to use.
» Library
» Documentation
» License: BSD-2-Clause

BOSL2 (beta)
Belfry OpenScad Library v2 - A library of tools, shapes, and helpers to make OpenScad easier to use.
» Library
» Documentation
» Tutorials
» License: BSD-2-Clause

dotSCAD
Reduce the burden of 3D modeling in mathematics.
» Library
» Documentation
» License: LGPL-3.0-only

NopSCADlib
An ever expanding library of parts modelled in OpenSCAD useful for 3D printers and enclosures for electronics, etc.
» Library
» Documentation
» License: GPL-3.0-or-later

On the right side of the page, there are four thumbnail images illustrating different library components:

- BOSL_beziers: A yellow cone-like shape with a grid of points and lines.
- BOSL2_3d_attach: A yellow sphere attached to a base with a grid of points and lines.
- dotSCAD_helix_extrude: A yellow helical extrusion with a grid of points and lines.
- NopSCADlib: A complex assembly of various electronic components like resistors, capacitors, and a microcontroller.



OpenSCAD Book

The screenshot shows the OpenSCAD website's documentation page. At the top, there is a navigation bar with links for home, about, news, downloads, documentation (which is highlighted), libraries, gallery, community, and github. Below the navigation bar, there is a sidebar with links for Documentation, Books, Videos, and Articles / Blogs. The main content area features a section for 'English' with a book cover for 'Programming with OpenSCAD - A Beginner's Guide to Coding 3D-Printable Objects'. The book cover features a cartoon character working on a computer. Below the book cover, there is a brief description of the book and its authorship information.

Documentation

- OpenSCAD Tutorial
- OpenSCAD User Manual
- OpenSCAD Language Reference
- Code Cheat Sheet

Books

- English
- French / Français
- German / Deutsch
- Spanish / Español

Videos

- OpenSCAD: Introduction

Articles / Blogs

- How to use OpenSCAD
- 3D Spielplatz (german)
- OpenSCAD Tutorial Series
- Tutorial at EduTechWiki

Documentation

Books

Videos

Articles / Blogs

English

Programming with OpenSCAD - A Beginner's Guide to Coding 3D-Printable Objects

This book channels OpenSCAD's visual benefits and user-friendliness into a STEAM-focused, project-based tutorial that teaches the basics of coding, 3D printing, and computational thinking while you develop your spatial reasoning by creating 3D designs with OpenSCAD.

Accessibly written for a wide audience (advanced middle schoolers, high school students, college students, artists, makers and lifelong-learners alike), this is the perfect guide to becoming proficient at programming in general and 3D modeling in particular.

Author: Justin Gohde & Marius Kintel
Publisher: No Starch Press
Date: July 2021

Read more at programmingwithopenscad.github.io



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(...)= ...
include <...scad>;
use <...scad>
```

Constants

```
undefined: undefined value
PI: mathematical constant π (-3.14159)
```

Operators

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

Special variables

\$fa	minimum angle
\$fs	minimum size
\$fn	number of fragments
\$t	animation step
\$view	viewport rotation angles in degrees
\$view	viewport translation
\$view	viewport camera distance
\$view	viewport camera field of view
\$children	number of module children
\$isreview	true in FS preview, false for FG

Modifier Characters

- # disable
- ! show only
- E highlight / debug
- B transparent / background

2D

```
circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polyline([points])
text([text, font, height, align, spacing, direction, language, style])
import("...ext", convexity)
arcollection(cut)
```

3D

```
sphere(radius | d=diameter)
cube(size,depth,height, center)
cylinder([r,r,d],center)
cylindec([r,r,l,r2,d],center)
polyhedron(points, facets, 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])
translate([x,y,z],o)
rotate([x,y,z])
rotate([x,y,z],o)
scale([x,y,z])
rescale([x,y,z],auto,convexity)
matrix([x,y,z])
multimatrix()
color("colorname",alpha)
color("#hexcode")
color(r,g,b,a)
offset(r,delta,chanfer)
hull()
```

midowski(convexity)

Lists

```
list[...]: create a list
var = list[i]: Index a list (from 0)
var = list[z]: dot notation indexing (x/y/z)
```

Boolean operations

```
union()
difference()
intersection()
```

List Comprehensions

```
generate [ for (l = range|list) l ]
generate [ for (l1; condition; next) l ]
flatten [ each l ]
condition [ for (l = ...) if (condition(l)) l ]
conditions [ for (l = ...) if (condition(l)) x else y ]
assignments [ for (l = ...) let (assignments) a ]
```

Flow Control

```
for (i = [startvalue]) { ... }
for (i = [startvalue];cond) { ... }
for (i = ...,j = ...,) { ... }
for (i = ...,j = ...,k = ...) { ... }
intersection_for([ for (i=startvalue) { ... } ])
intersection_for([ for (i=startvalue);cond) { ... } ])
intersection_for([ for (i=...,j=...) { ... } ])
if (...) { ... }
let (...) { ... }
```

Type test functions

is_vector
is_bool
is_num
is_string
is_list
is_function

Other

edit()
render(convexity)
children([idx])
assert(condition, message)
exists({}) { ... }

Functions

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

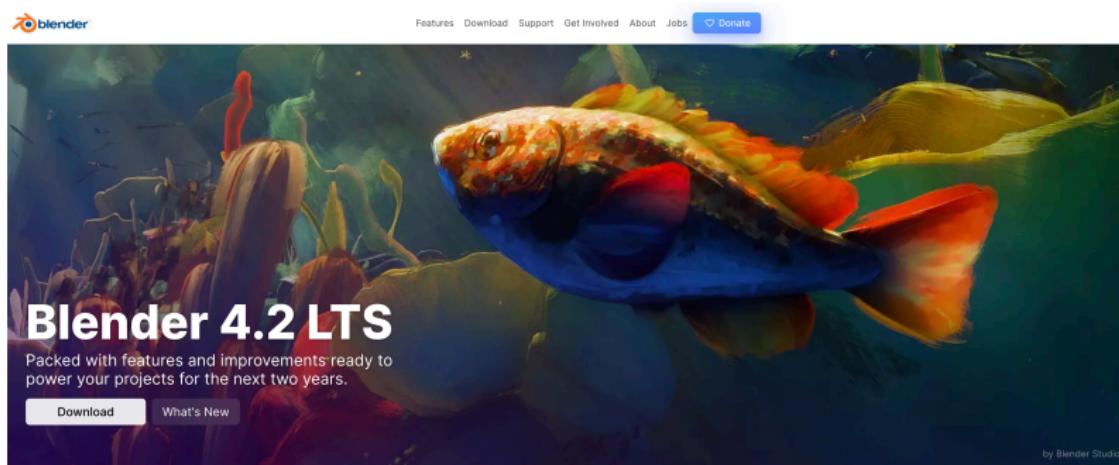
Mathematical

abs
sign
sin
cos
acos
asin
atan
atan2
floor
round
ceil
ln
log
lat
lat
log
now
sqrt
exp
cosh
sinh
tanh
norm
cross

Links: [official website](#) | [Code](#) | [Issues](#) | [Manuals](#) | [MCAD library](#) | [Mailing list](#) | [Other links](#)



Sample Texts



The screenshot shows the Blender 4.2 LTS landing page. At the top, there's a navigation bar with links for Features, Download, Support, Get Involved, About, Jobs, and a blue 'Donate' button. Below the navigation is a large, vibrant underwater scene featuring a large, multi-colored fish (orange, yellow, blue) swimming among coral reefs and smaller fish. In the lower-left corner of the main image, the text 'Blender 4.2 LTS' is displayed in large white letters. Below this, a subtitle reads 'Packed with features and improvements ready to power your projects for the next two years.' At the bottom left, there are two buttons: 'Download' and 'What's New'. A small 'by Blender Studio' credit is visible at the bottom right.

User Manual

Developer Documentation

Python API

Primitives

Primitives



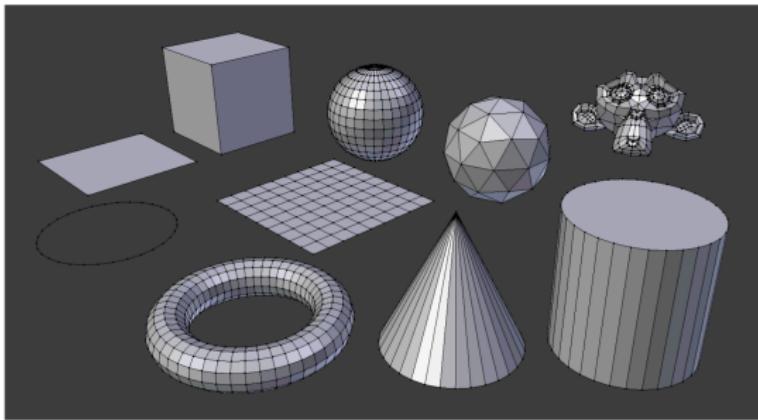
Reference

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

The screenshot shows a software interface with a navigation bar at the top. Under the 'Reference' section, it lists 'Category: Import-Export' and 'Menu: File > Import/Export > Stl (.stl)'. This indicates the STL format is used for importing and exporting CAD files.

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.

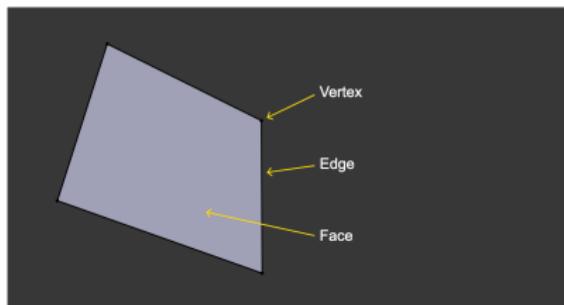


Mesh Basics

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

Wolfram Alpha.com | WolframCloud.com | All Sites & Public Resources.

WOLFRAM COMPUTATION MEETS KNOWLEDGE

Products & Services Technologies Solutions Learning & Support Company Search

Computation. Data. Decisions.
JUST USE WOLFRAM

Technology Consulting Education Wolfram AI

QUANTUM COMPUTING
FINANCIAL RISK MANAGEMENT
IMAGE ANALYSIS
ARTIFICIAL INTELLIGENCE
MORE...
CHEMICAL ENGINEERING
CONTROL SYSTEMS
ENGINEERING

Just Use Wolfram Wolfram Language Wolfram|Alpha Intelligence Our Experts, Your Projects Transforming Education Leading Innovation in Science & Tech



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)
- "JVR" — JavaView format (.jvr)
- "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)

Navigation icons: back, forward, search, etc.



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 TJunction
 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]:= 



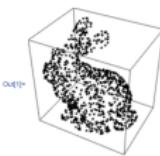
Reconstruct Models from 3D Data

↳ 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]]
```



Reconstruct a 3D-printable Stanford bunny.

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





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]=



```
In[2]:= volume = Quantity[Volume[model], "Centimeters"^3]
```

Out[2]= 55.5217 cm³

```
In[3]:= aluminum [element] \[Implies] ("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/1407d1e1e1a0cb2fe013fe1fffb3fea31cbd224.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]:=

Out[3]:=

Out[4]:=

Out[5]:=



Animation with Blender

The screenshot shows a post on the Wolfram Community website. The header features the Wolfram logo and the text "WOLFRAM COMMUNITY" with the subtitle "Connect with users of Wolfram technologies to learn, solve problems and share ideas". There are "Join" and "Sign In" buttons. Below the header is a navigation bar with "Dashboard", "Groups", "People", and a search bar. The main content area shows a post titled "Animating Wolfram surfaces with Blender: Mesh deforming & Superellipse" by Guenther Gsaller. The post has 8213 views, 1 reply, and 5 total likes. It includes buttons for "View groups...", "Follow this post", and "Share".

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

Automating triangulation of paintings with Delaunay
mesh and cloud app



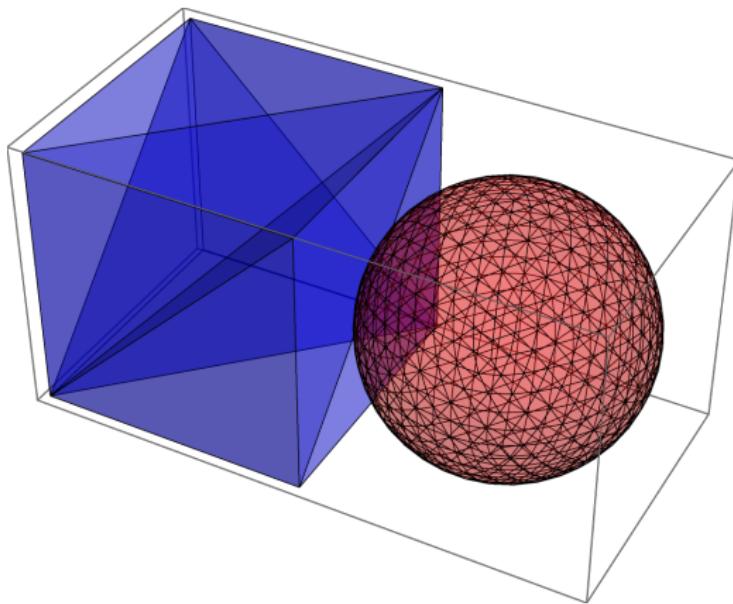
Kirill Belov

Posted 2 years ago

The screenshot shows a user interface for a web application. At the top, there is a form with two input fields: 'Original' (with a placeholder 'Drag and drop an image (or click to browse)') and 'Number of triangles' (set to 100). Below the form is a red 'Submit' button. The main area displays two versions of the Mona Lisa painting side-by-side. The left version is the original painting, and the right version is a low-polygonal Delaunay triangulation of the same image.



Simple Shapes

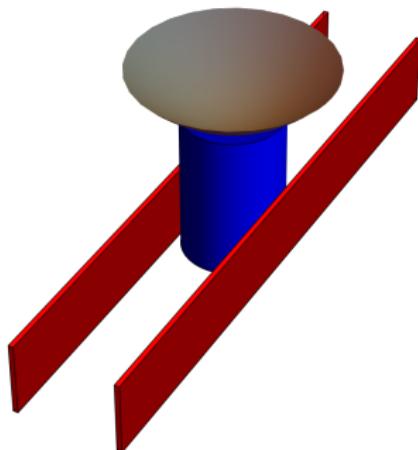




Simple Shapes



DanielSat©





Bibliography I



Design Tools for Satellite Design

Daniel Topa
daniel.topa@hii-tsd.com

Huntington Ingalls Industries
Mission Technologies

October 26, 2024