FEATPOST manual

L. Nobre G.

0.8.2

Abstract

FEATPOST is an extension of the METAPOST language that has a fairly large set of features to facilitate the production of schematic diagrams, both in three dimensions (3D) and two dimensions (2D). These schematic diagrams are vectorial and focus on the representation of edges (unlike ray-traced raster images that focus on surfaces).

Getting started

```
input featpost3Dplus2D;
```

Propaganda

METAPOST is based on mathematics and that is beautiful.
METAPOST is also the realm og graphic parametrization and that is beyond description.

A bit of history

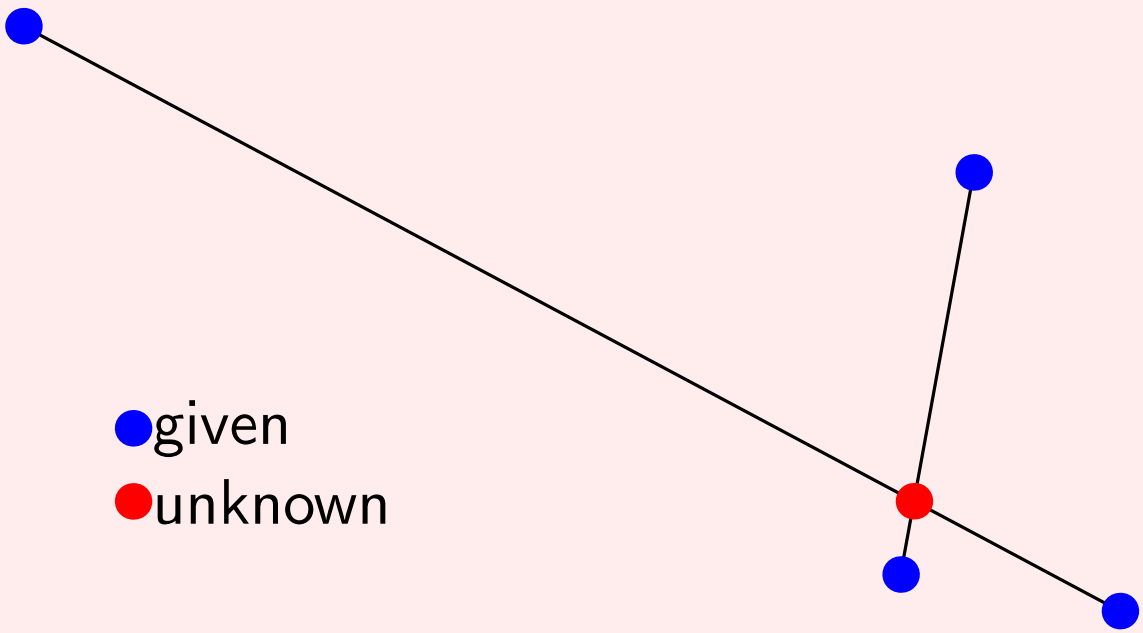
Donald Knuth – *The Art of Computer Programming*

- Volume1, 1969, hot metal
- Volume2, $2^{\underline{\text{nd}}}$ ed., 1977, photographic
- TEX, v1, 1978, $\text{v} \longrightarrow \pi$
- METAFONT, v1, 1979, $\text{v} \longrightarrow e$
- AMS, 1983
- APS, AIP, OSA, AAS, Springer-Verlag
- METAPOST, 1994, John Hobby

Main features of METAPOST

- It is easy to express geometry as METAPOST code
- METAPOST outputs `postscript`
- Very good control of 2D Bézier splines
- Special 2D operators
- Many operators work the same way in 1, 2, 3 or 4D
- Linear equations
- May include LaTeX
- May be included in LaTeX

Linear equations



- Constraints may be expressed as linear equations
- No need to explicitly assign calculations to unknowns

Other METAPOST built-ins

- `color` *(red,green,blue)* $\longrightarrow$ (X,Y,Z)

black	(0,0,0)
red	(1,0,0)
green	(0,1,0)
blue	(0,0,1)
white	(1,1,1)

- `cmykcolor` *(cyan,magenta,yellow,black)* $\longrightarrow$ (X,Y,Z,W)
  (quaternions, homogeneous coordinates, 4D splines, animation frames, straight line segments, etc.)
- `scantokens` ‘‘*string of commands*’’ (allows sorting of generic graphic elements)

Use METAPOST because

METAPOST helps the user to experiment different diagram layouts without changing specified geometric relationships among diagram elements.

First taste of FEATPOST

Each perspective depends on the point of view. FEATPOST uses the global variable `f`, of `color` type, to store the (X, Y, Z) space coordinates of the point of view. Also important is the aim of view (global variable `viewcentr`). Both define the line of view.

The perspective consists of a projection from space coordinates into planar (u, v) coordinates on the projection plane. FEATPOST uses a projection plane that is perpendicular to the line of view and contains the `viewcentr`. Furthermore, one of the projection plane axes is horizontal and the other is on the intersection of a vertical plane with the projection plane. "Horizontal" means parallel to the XY plane. The projection plane axes are perpendicular to each other.

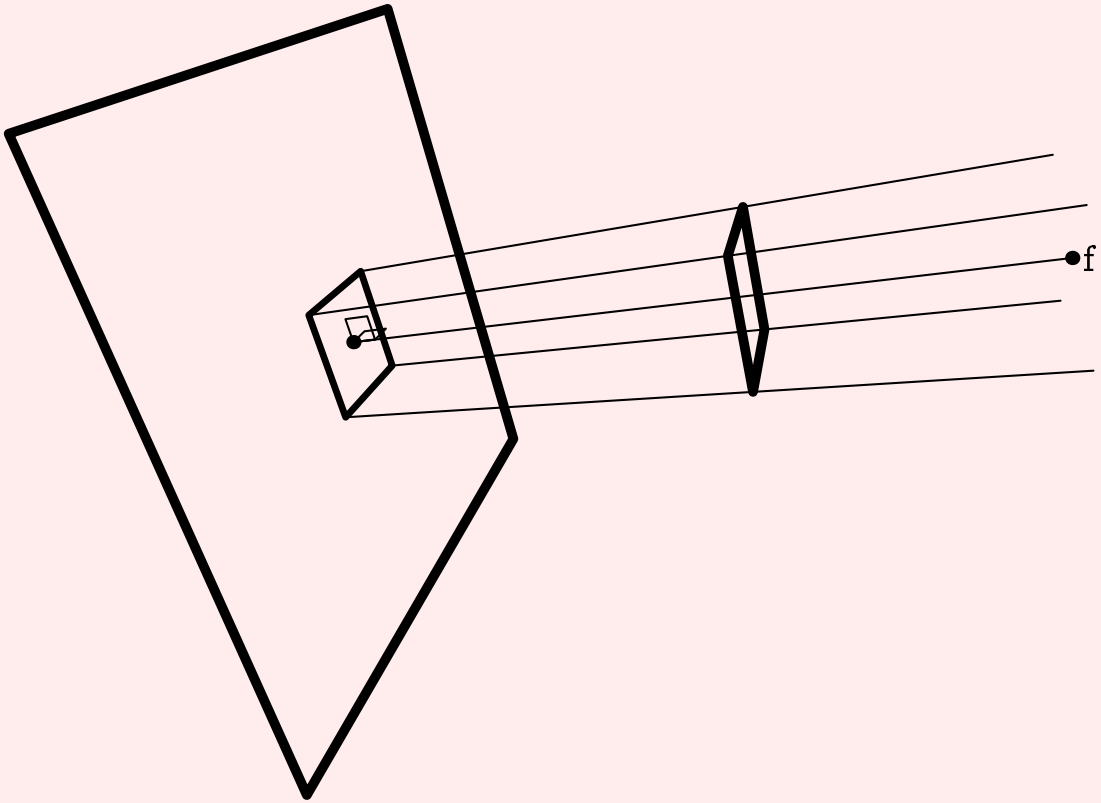


Figure: Parallel projection.

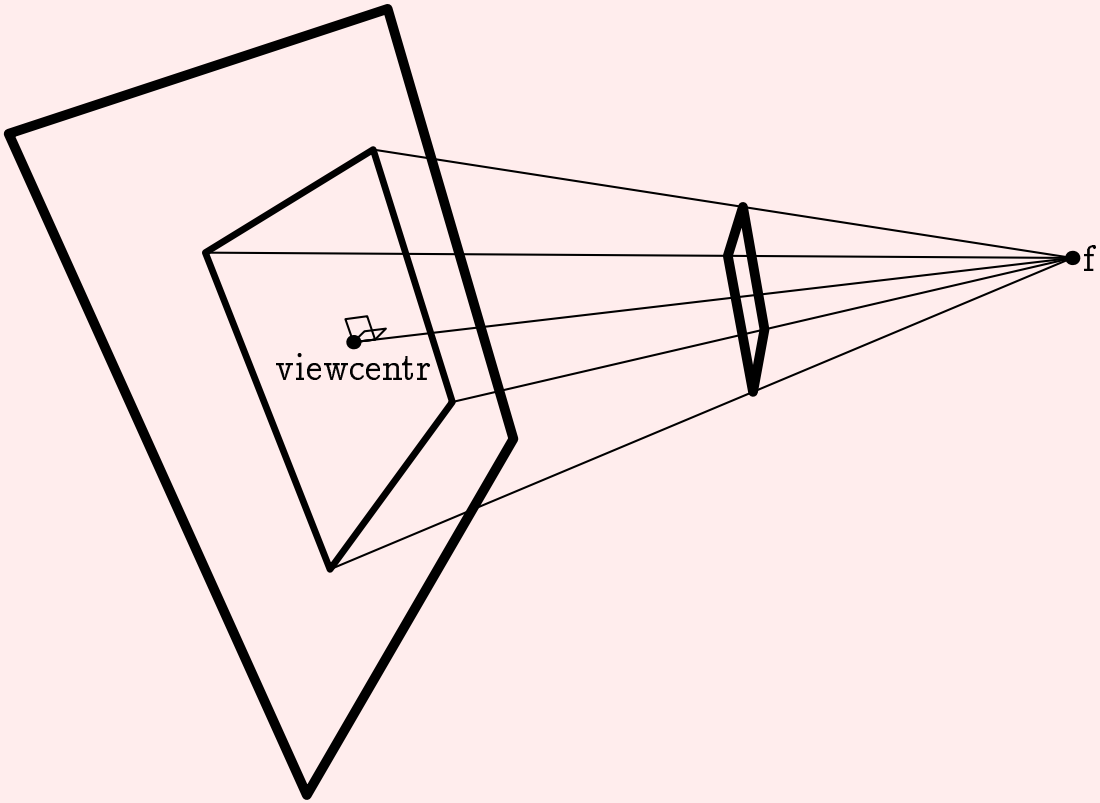


Figure: Central projection.

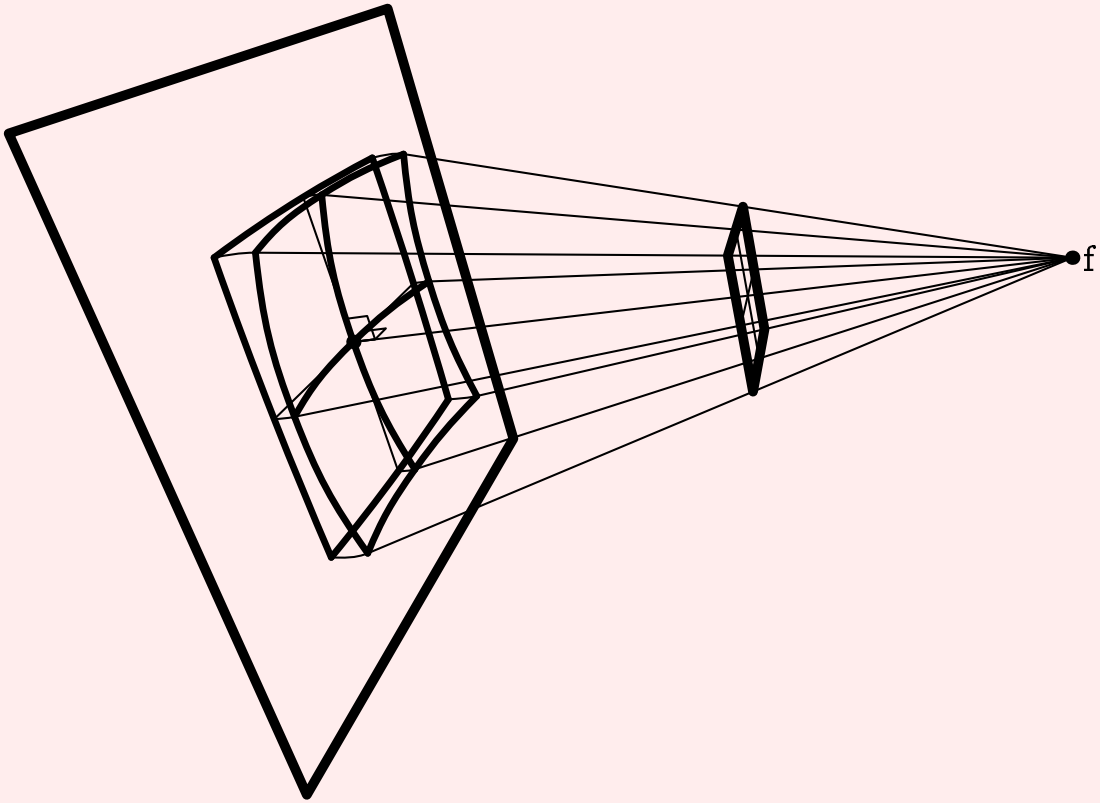


Figure: Spherical projection. The spherical projection is the composition of two operations: (i) there is a projection onto a sphere and (ii) the sphere is plaited onto the projection plane.

Some problems often require defining angles, and diagrams are needed to visualize their meanings. The `angline` and `squareangline` macros support this (see figure 4).

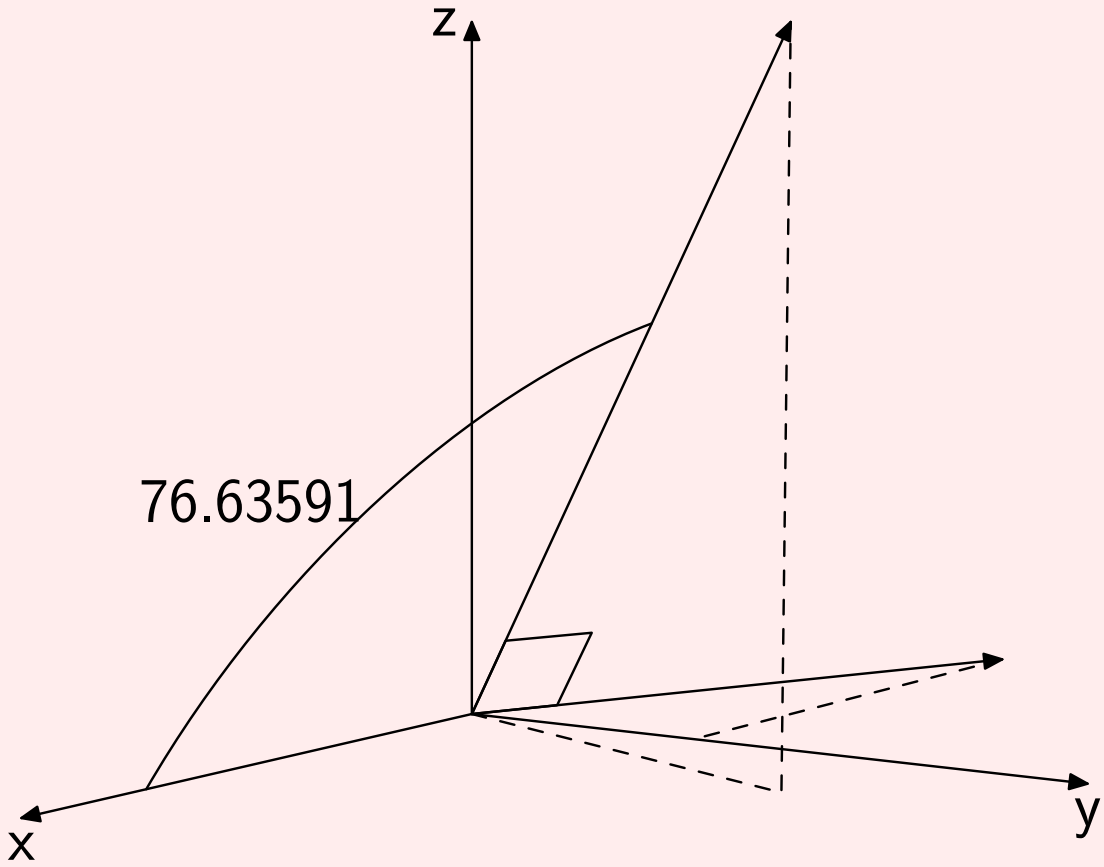


Figure: Example that uses `cartaxes`, `squareangline`, `angline` and `getangle`.

Visualizing parametric lines is another need. When two lines cross, one should be able to see which line is in front of the other. The macro `emptyline` can help here (see figure 5).

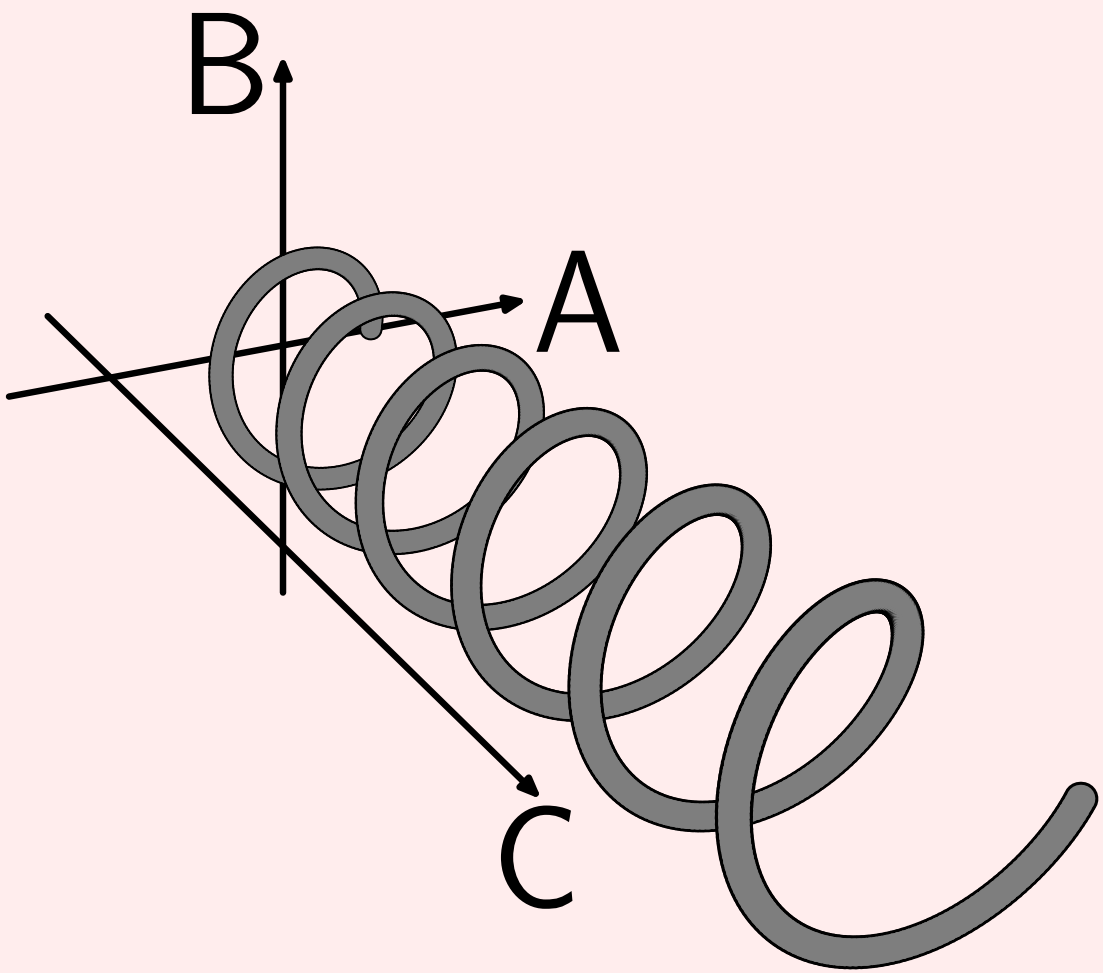


Figure: FEATPOST diagram using `emptyline`.

Cuboids and labels are always needed. The macros `kindofcube` and `labelinspace` fulfill this need (see figure 6). The macro `labelinspace` does not project labels from 3D into 2D. It only `Transforms` the label in the same way as its bounding box, that is, the same way as two perpendicular sides of its bounding box. This is only exact for parallel perspectives.

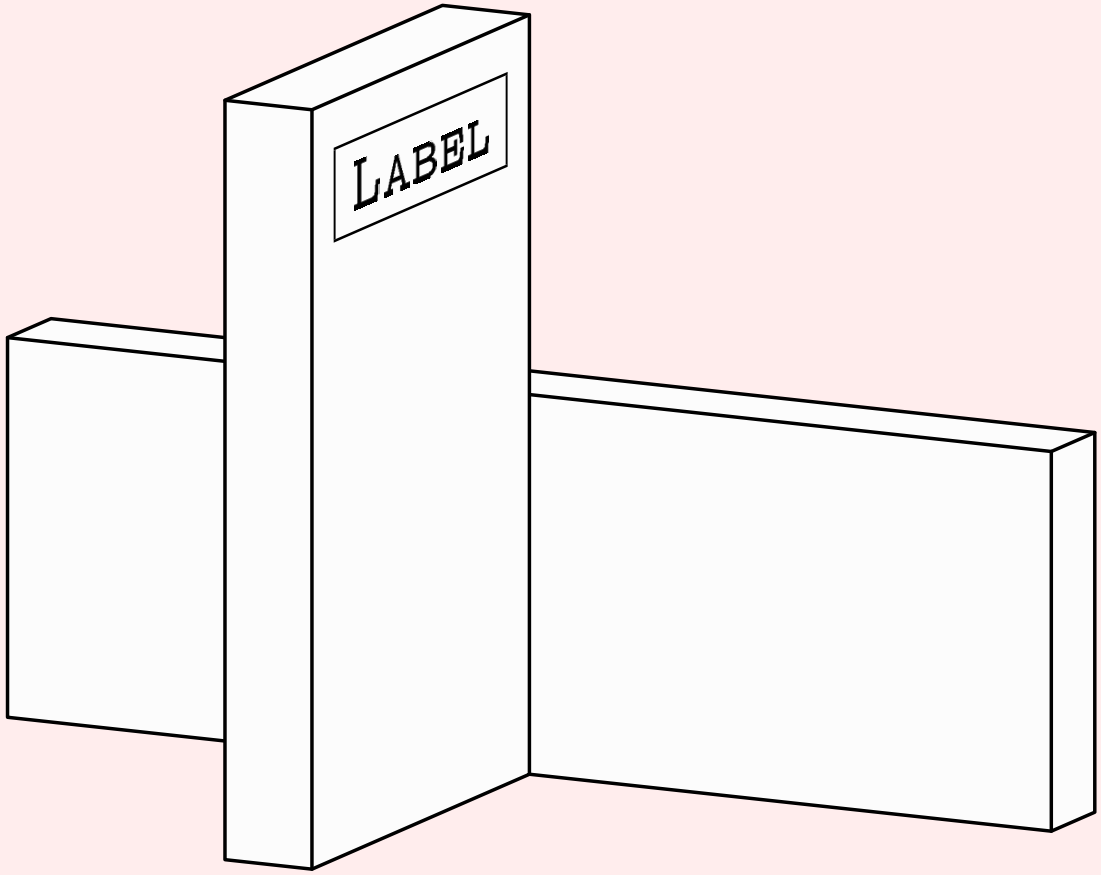


Figure: FEATPOST diagram using the macros `kindofcube` and `labelinspace`.

Some curved surface solid objects can be drawn with FEATPOST. Among them are cones (`verygoodcone`), cylinders (`rigorousdisc`) and globes (`tropicalglobe`). These can also cast their shadows on a horizontal plane (see figure 7). The production of shadows involves the global variables `LightSource`, `ShadowOn` and `HoriZon`.

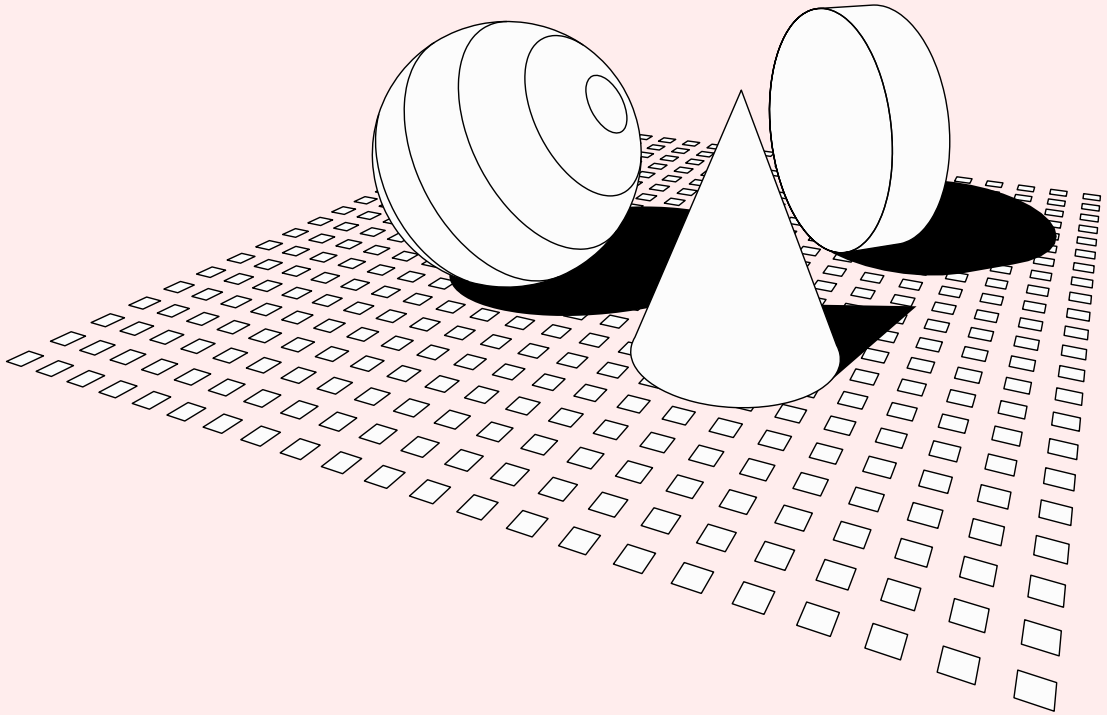


Figure: FEATPOST diagram using the macros `rigorousdisc`, `verygoodcone`, `tropicalglobe` and `setthestage`.

Another very common need is the plotting of functions, usually satisfied by software such as Gnuplot (`http://www.gnuplot.info/`) or Gri (`http://gri.sourceforge.net/`). Nevertheless, there are always new plots to draw. One specific FEATPOST kind of plot is the "triangular grid triangular domain surface" (see figure 8).

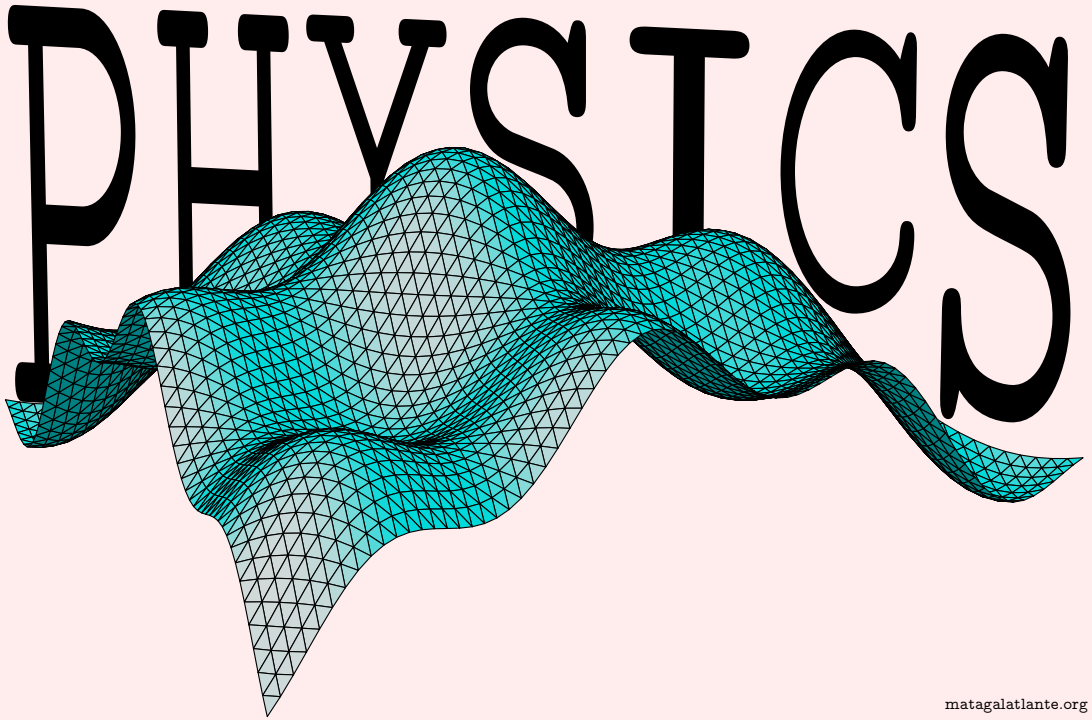


Figure: FEATPOST surface plot using the macro `hexagonaltrimesh`.

One feature that merges 2D and 3D involves what might be called "fat sticks". A fat stick resembles the Teflon magnets used to mix chemicals. They have volume but can be drawn like a small straight line segment stroked with a `pencircle`. Fat sticks may be used to represent direction fields (unitary vector fields without arrows). See figure 9.

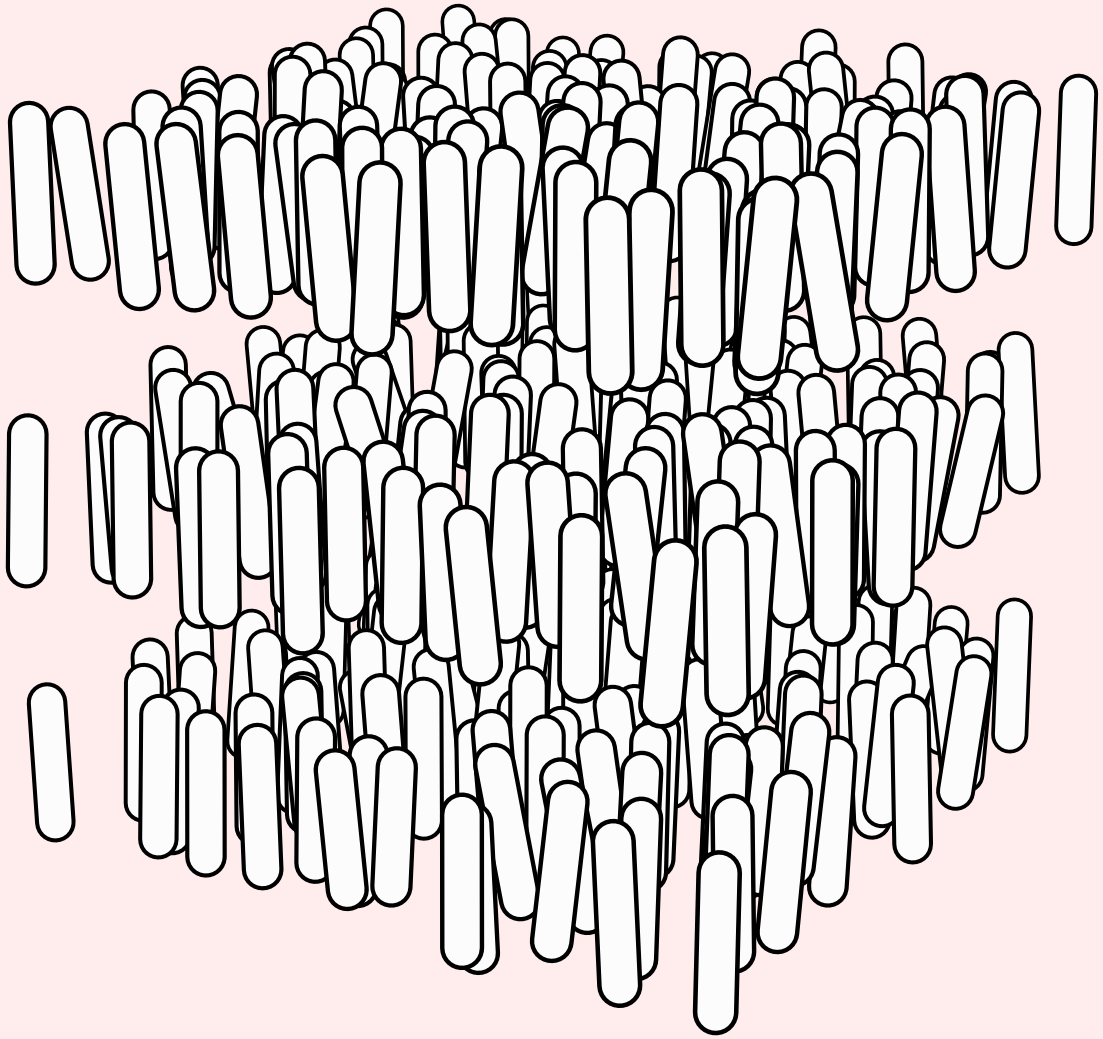


Figure: FEATPOST direction field macro `director_invisible` was used to produce this representation of the molecular structure of a Smectic A liquid crystal.

Finaly, it is important to remember that some capabilities of FEATPOST, although usable, may be considered "buggy" or only partially implemented. These include the calculation of intersections between polygons, as in figure 16, and the drawing of cylinders with axial holes, as in figure 10.

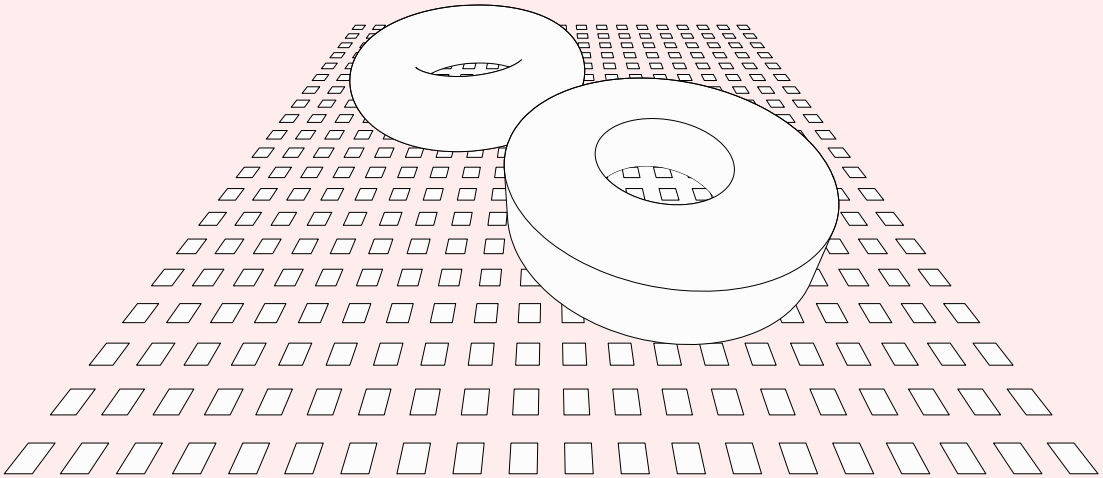


Figure: FEATPOST example containing a `smoothtorus` and a `rigorousdisc` with a hole.

It is highly beneficial to be able to understand and cope with METAPOST error messages as FEATPOST has no protection against mistaken inputs. One probable cause of errors is the use of variables with the name of procedures (macros), like

```
X, Y, Z, W, N, rp, cb, ps
```

All other procedure names have six or more characters.

One very minimalistic example program could be:

```
input featpost3Dplus2D; beginfig(1); cartaxes(1,1,1);
endfig; end;
```

where `cartaxes` is a FEATPOST macro that produces the Cartesian referential.

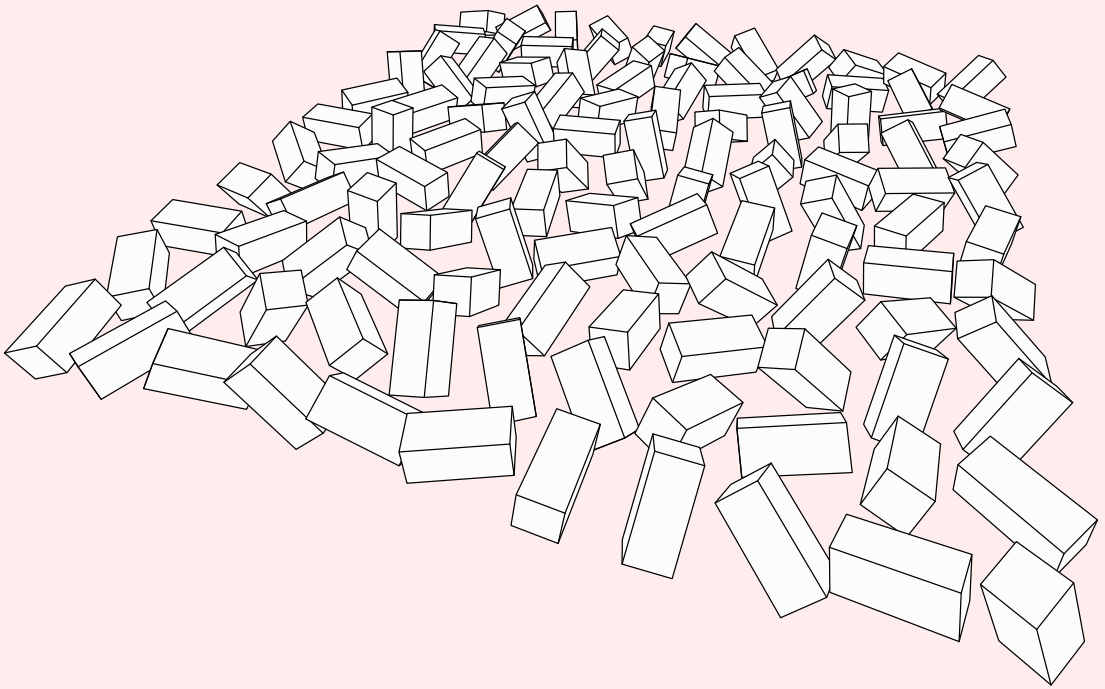


Figure: Example that uses `kindofcube`.

The main variable of any three-dimensional figure is the point of view. FEATPOST uses the variable `f` as the point of view. `Spread` is another global variable that controls the size of the projection.

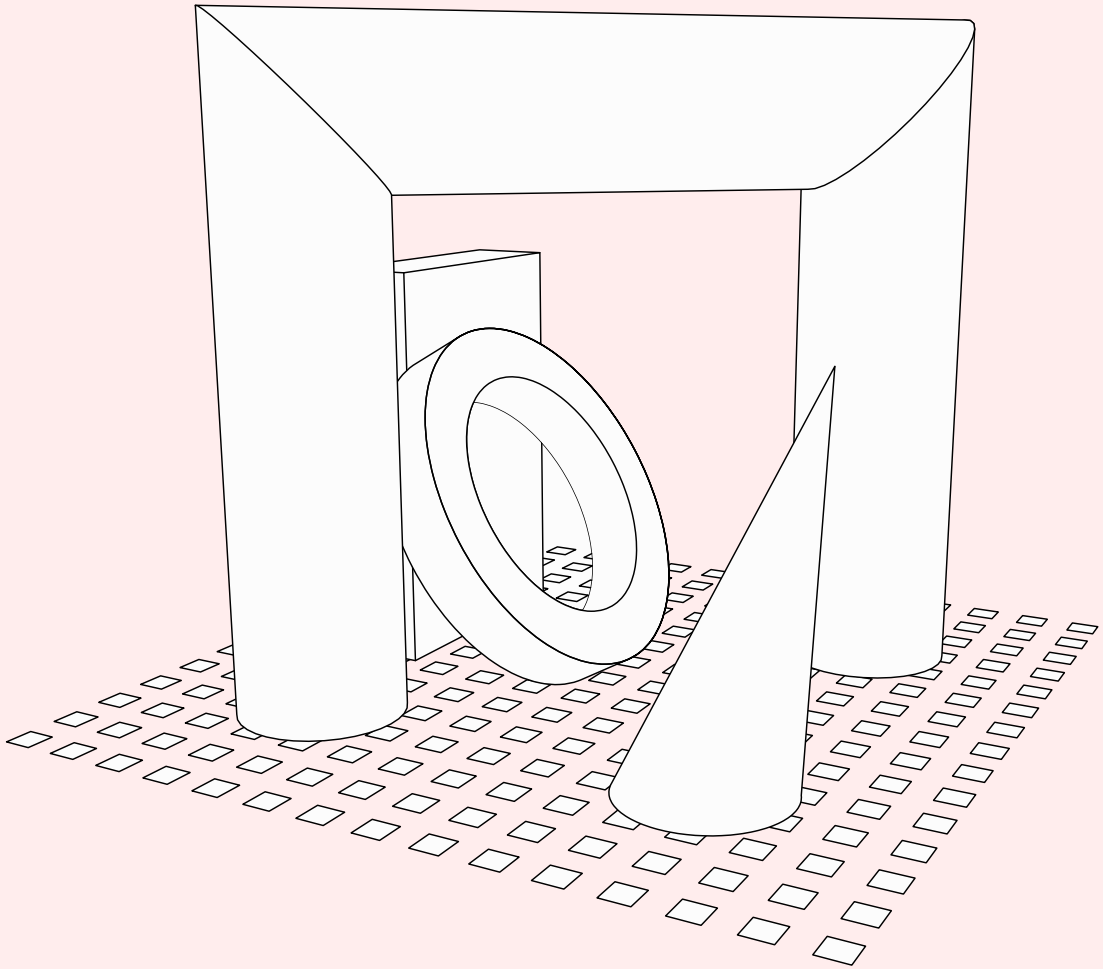


Figure: Example that uses `rigorousdisc` and `verygoodcone`.

Use FEATPOST because

FEATPOST has already been used in scientific publications:

- Figure 1 of *Phys. Rev. E*, **60**, 2985-2989 (1999).
- Figures 4, 6 and 8 of *Eur. Phys. J. E*, **2**, 351-358 (2000).
- Figures 8 and 12 of *Eur. Phys. J. E*, **20**, 55-61 (2006).

List of features

3D dots, vectors, flat arrows, angles, parametric lines, circles and ellipses, cuboids, cones, cylinders, cylindric holes, parts of cylindrical surfaces, spheres and spheroids, globes, hemispheres, torus, elliptical frusta, polygons, polyhedra and their planifications, functional and parametric surfaces, direction fields, field lines and trajectories in vector fields (differential equations), schematic automobiles, schematic electric charges, automatic perspective tuning, 2D representation of ropes, reference horizontal surfaces, hexagonal plots, schematic 2D springs, zig–zag lines, irregular circles, selective intersection of two circles, detection of tangency, paths for CNC machines, intersection of 2D areas, intersection of three spheres.

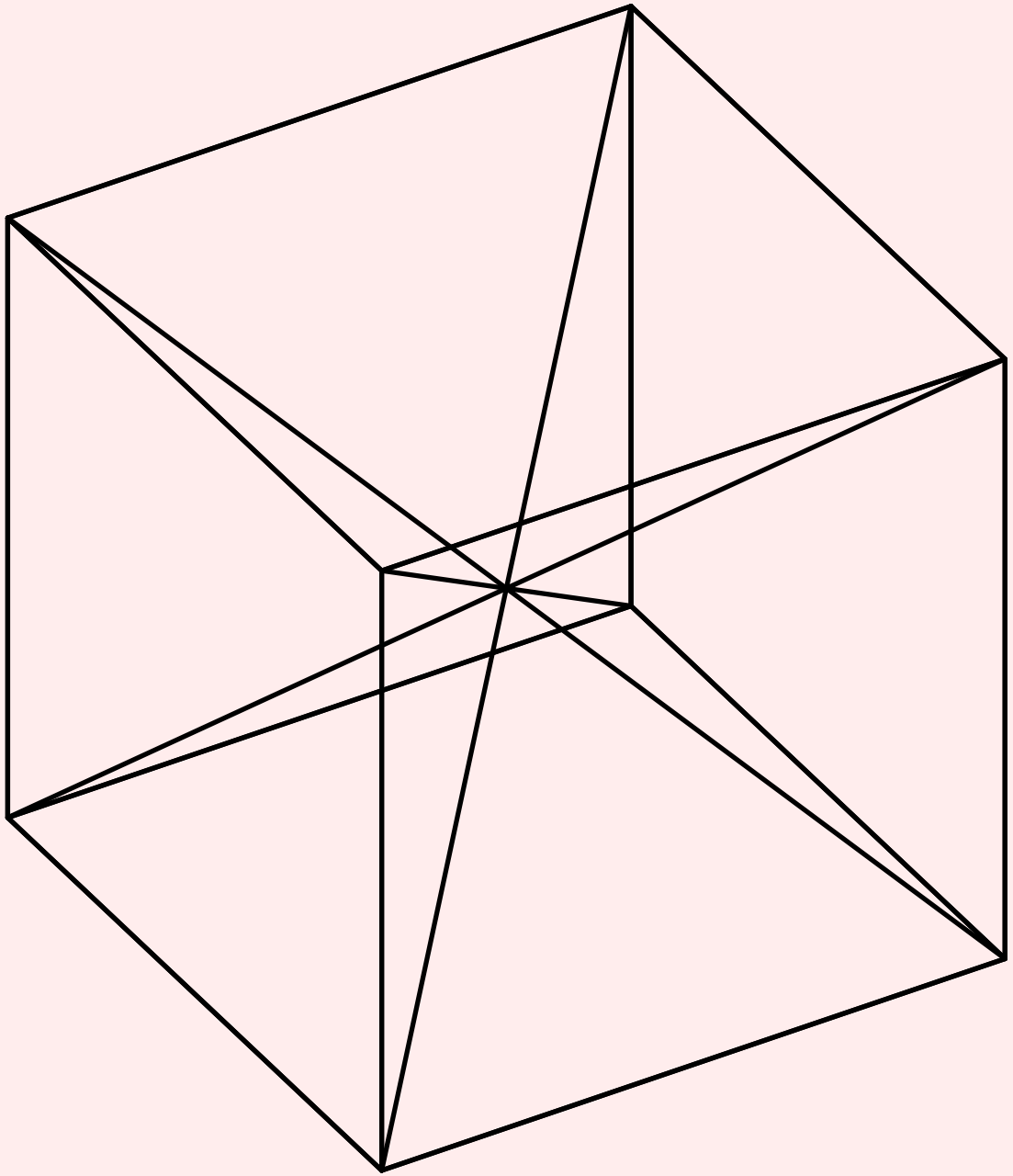


Figure: Orthogonal perspective.

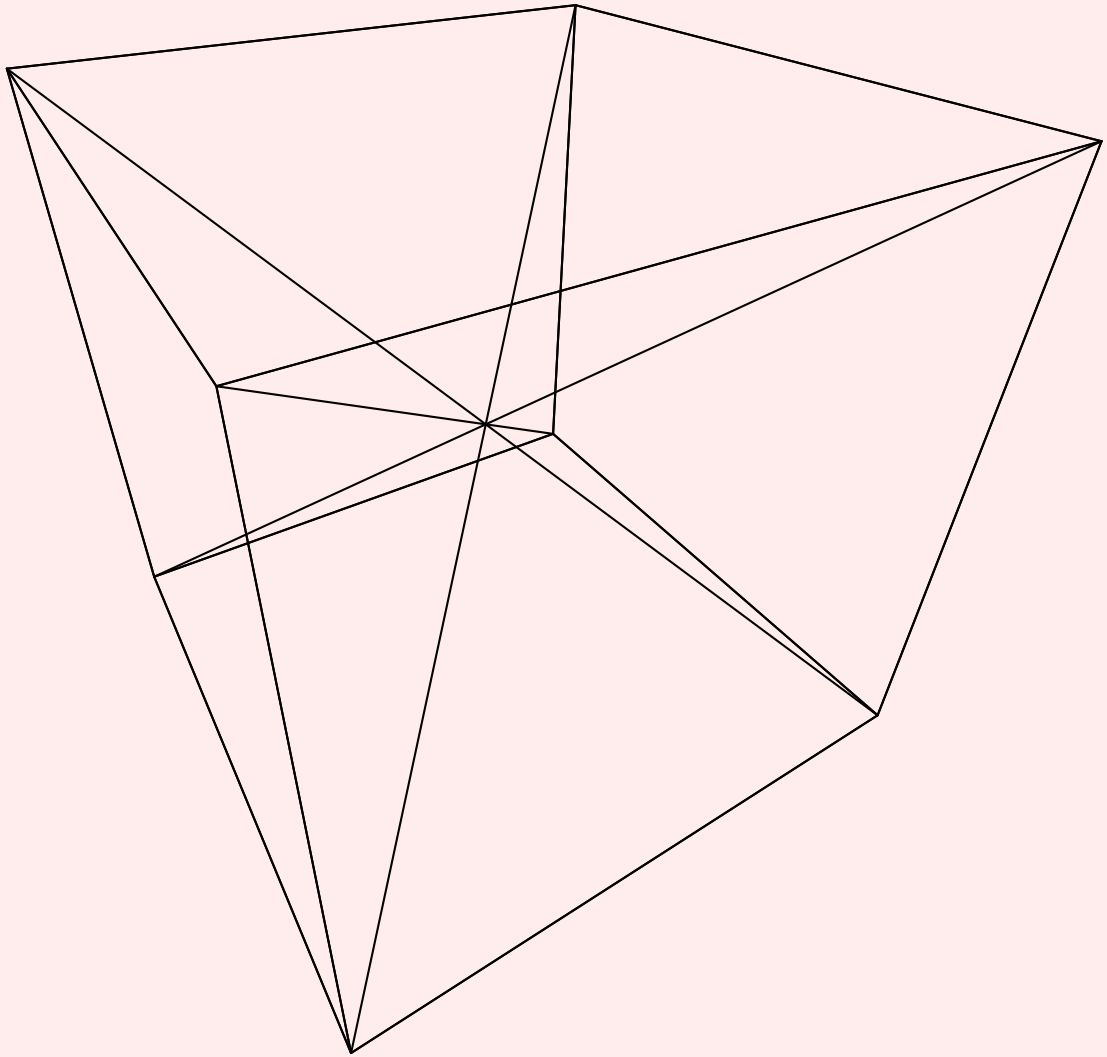


Figure: Rigorous perspective.

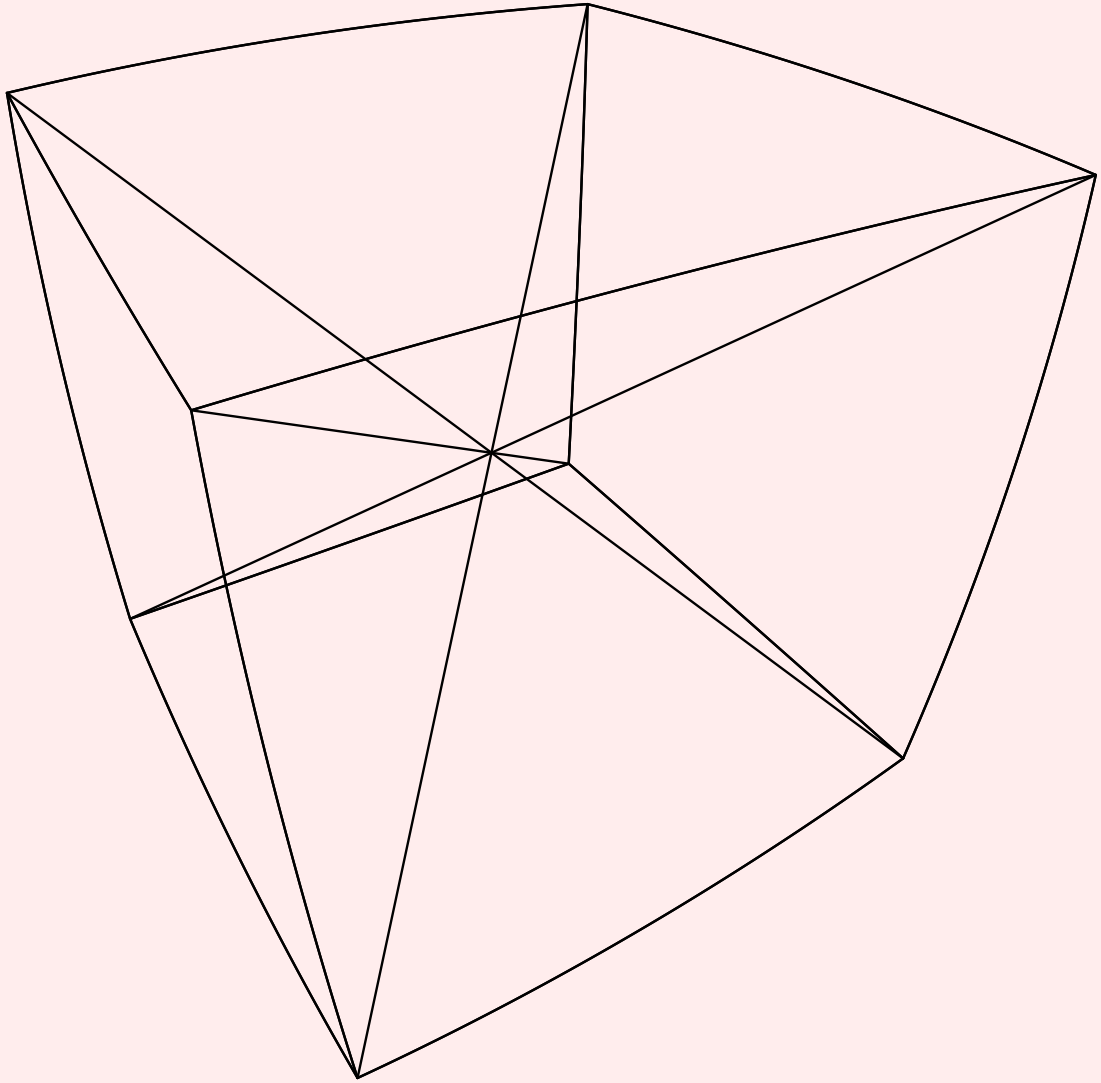


Figure: Fish-eye perspective.

From 3D to 2D

The most important macro is `rp` that converts 3D points to two-dimensional (2D) rigorous, orthogonal or fish-eye projections. To draw a line in 3D-space try

```
draw rp(a)--rp(b);
```

where `a` and `b` are points in space (of `color` type).

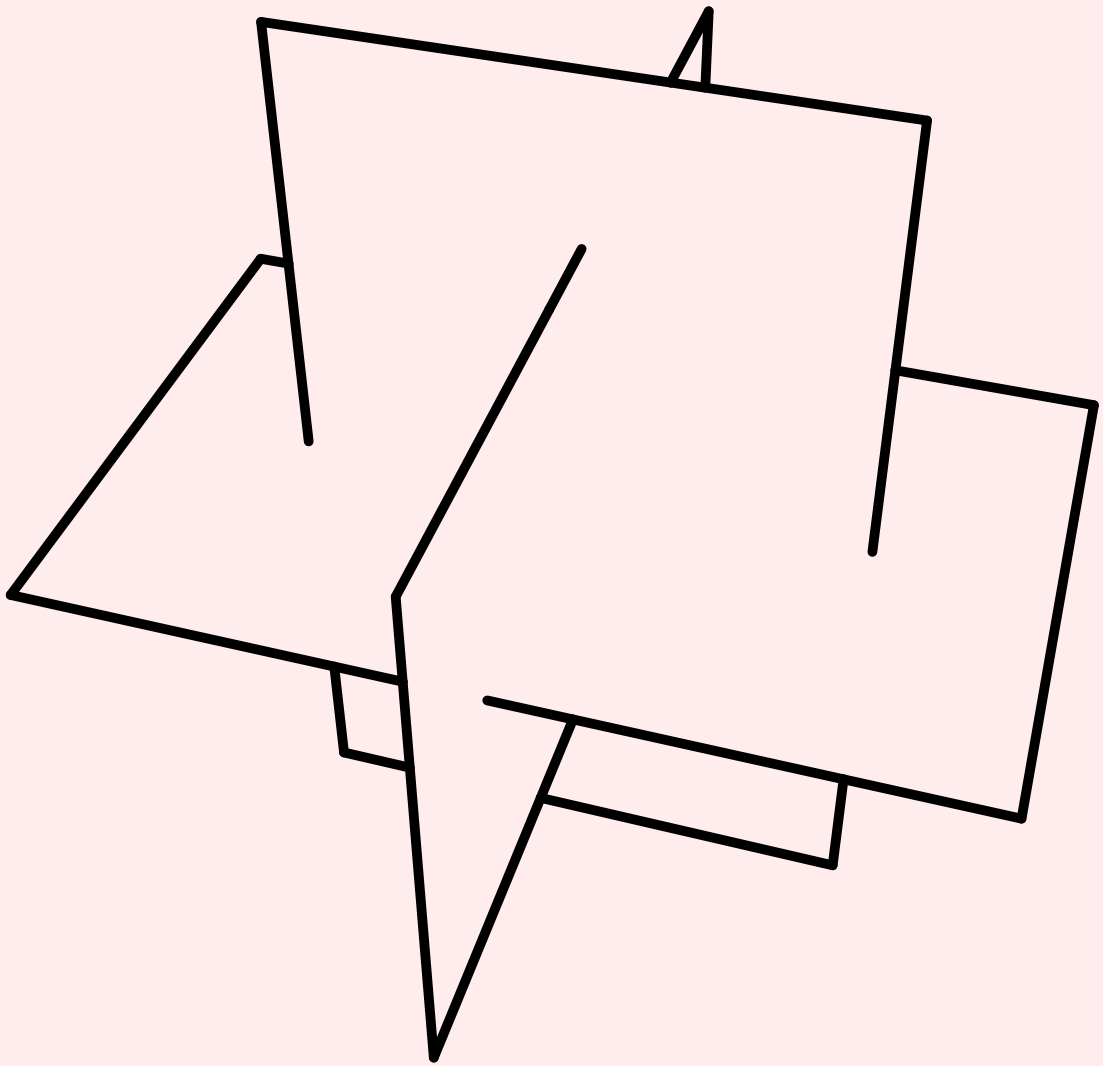


Figure: Intersecting polygons drawn with the macro `sharpraytrace`.

Coming back to 3D from 2D

It is possible to do an "automatic perspective tuning" with the aid of macro `photoreverse`. Please, refer both to example `photoreverse.mp` (see figure 17) and to the following web page: FeatPost Deeper Technicalities.

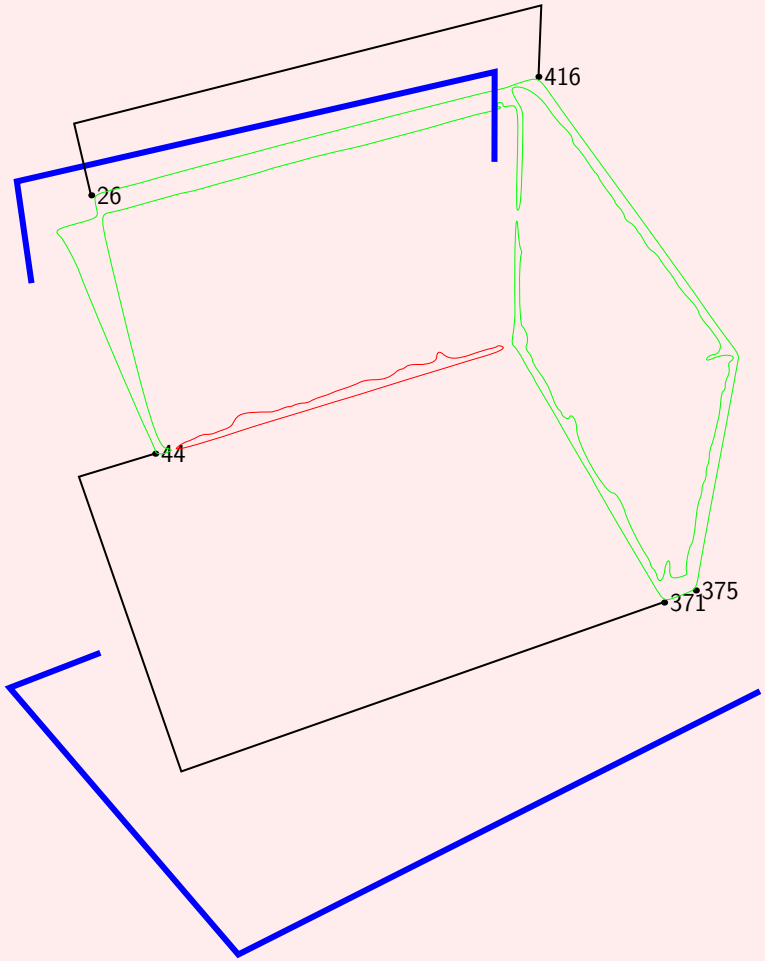


Figure: Example that uses `photoreverse`. It may not work when vertical lines are not vertical in average on the photo.

Coming back to 3D from 1D

Using the same algorithm of `photoreverse`, the macro `improvertex` allows one to approximate a point in 3D-space with given distances from three other points (an initial guess is required).

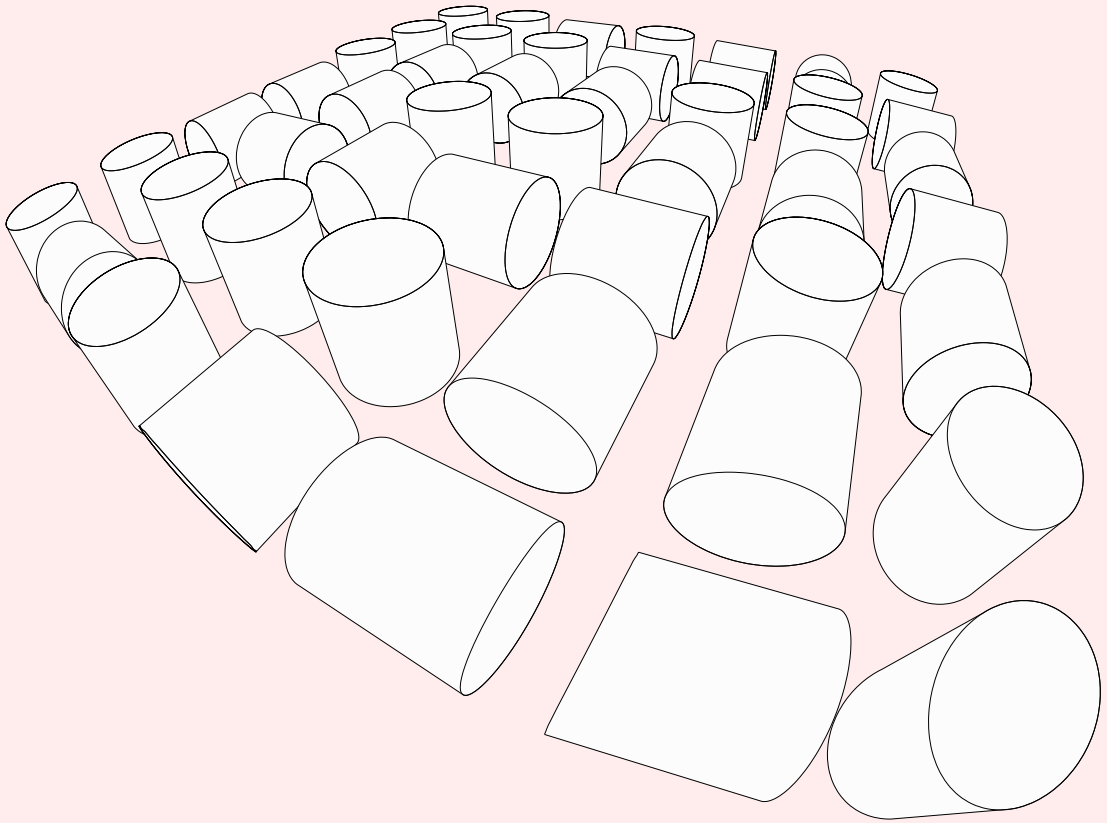


Figure: Figure that uses `SphericalDistortion:=true` and `rigorousdisc`.

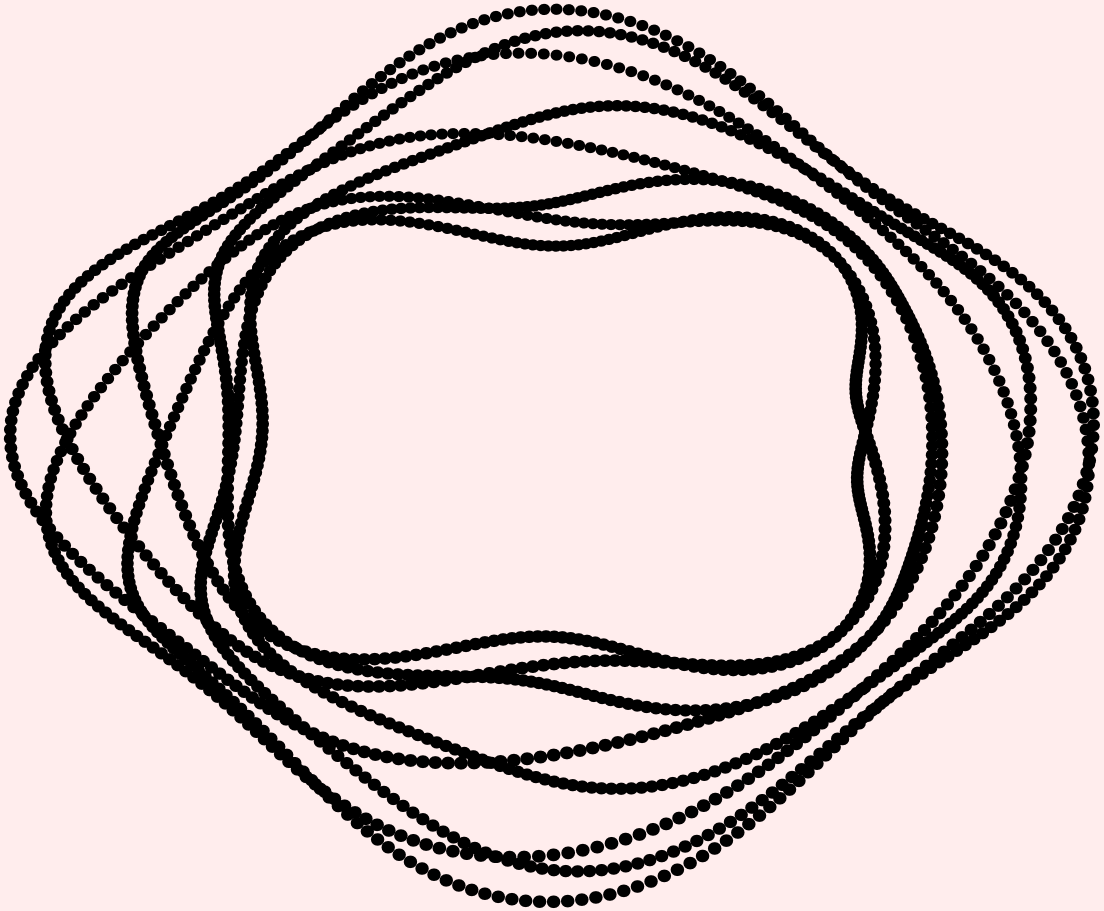


Figure: Figure that uses `signalvertex`.

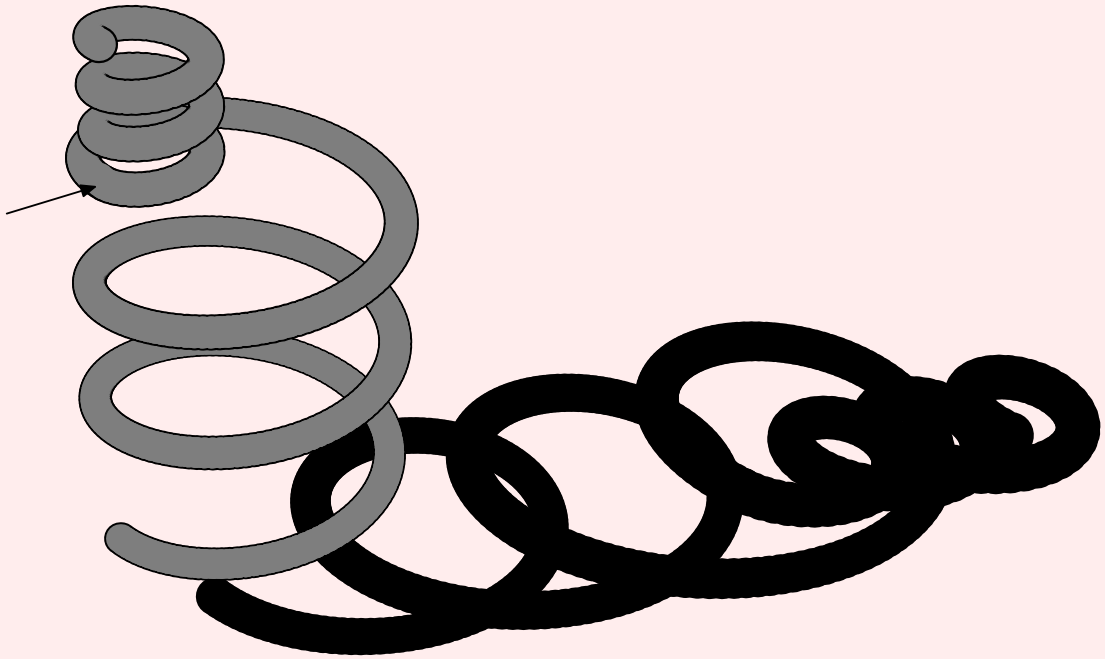


Figure: Figure that uses `emptyline`. The junction point of two different lines is indicated by an arrow.

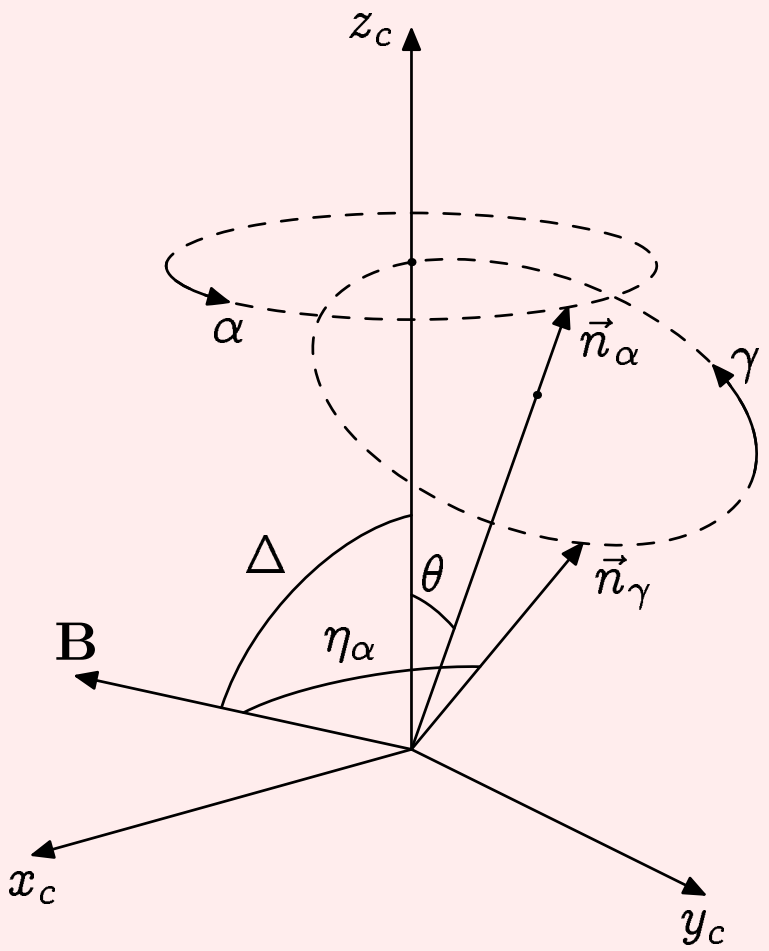


Figure: Figure that uses `anglinen` and `rigorouscircle`.

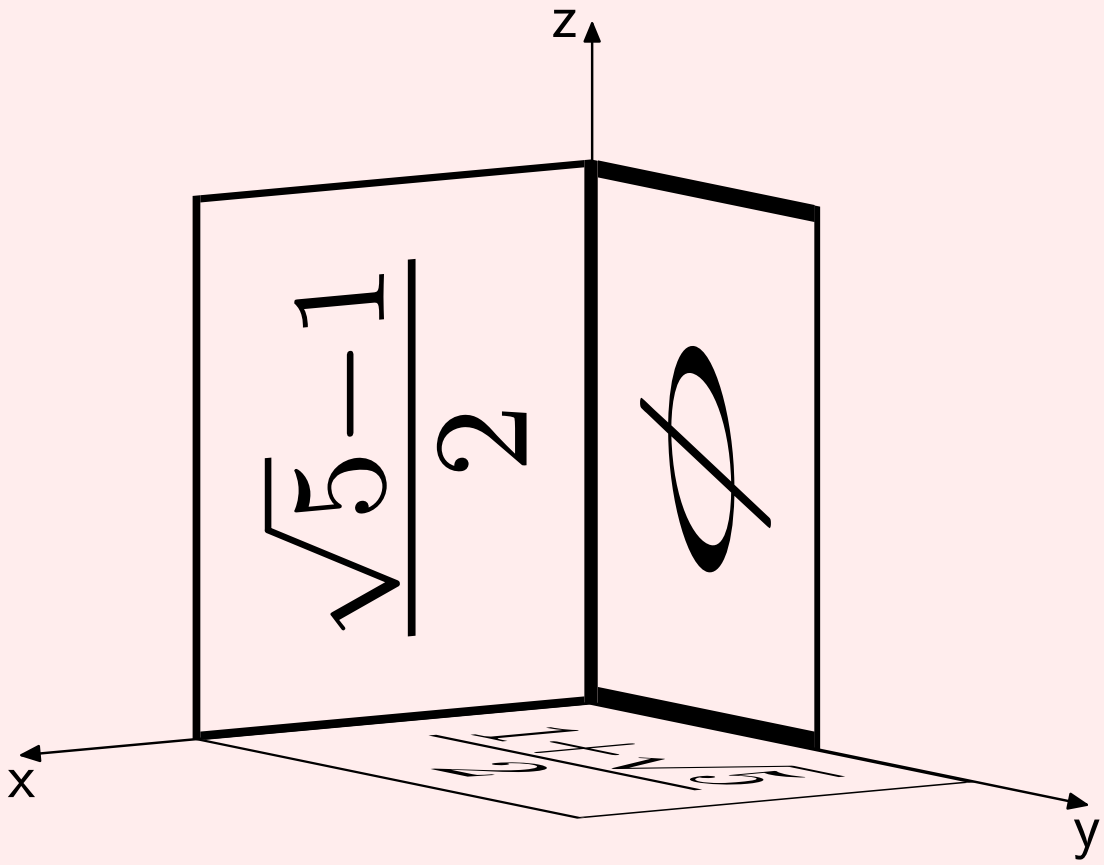


Figure: Example that uses `labelinspace`.

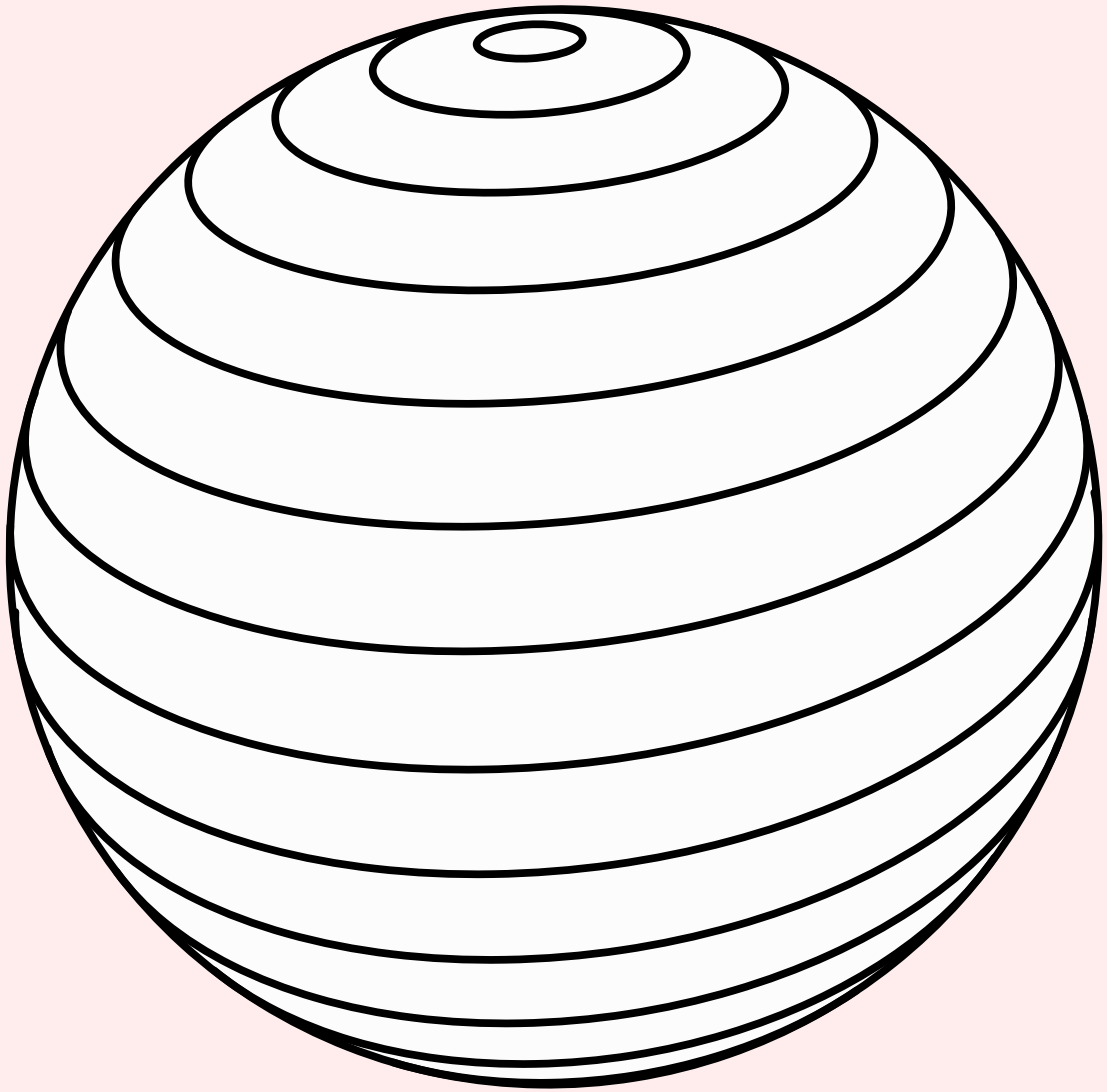


Figure: Figure that uses `tropicalglobe`.

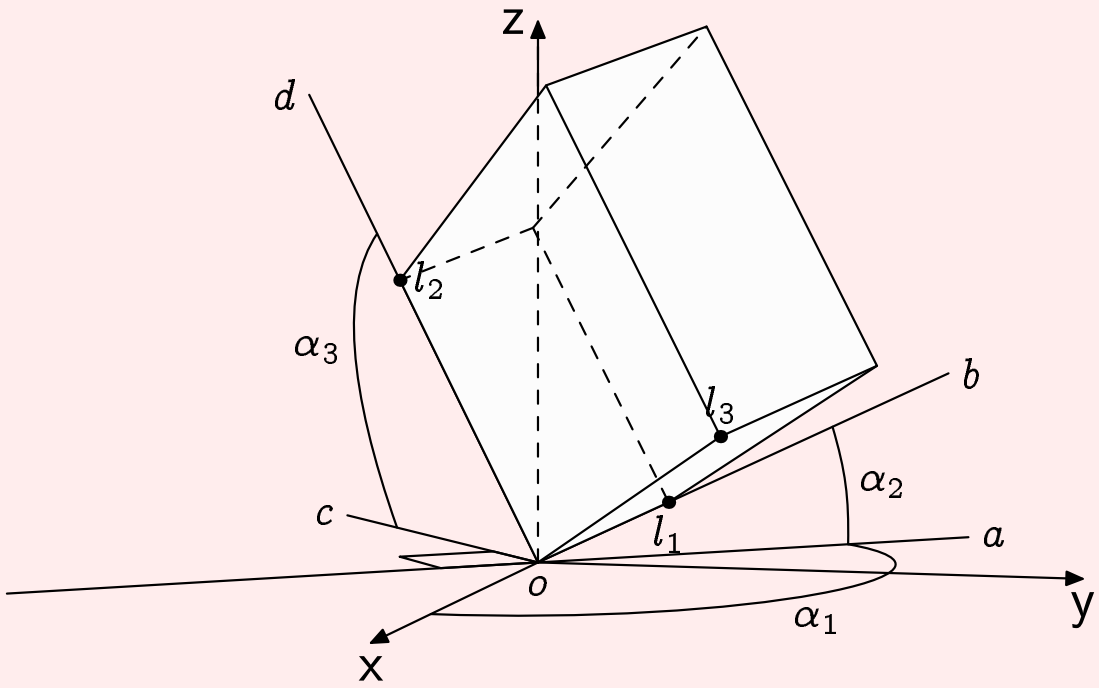


Figure: Figure that uses and explains `kindofcube`. Note that the three indicated angles may be used as arguments of `eulerrotation`.

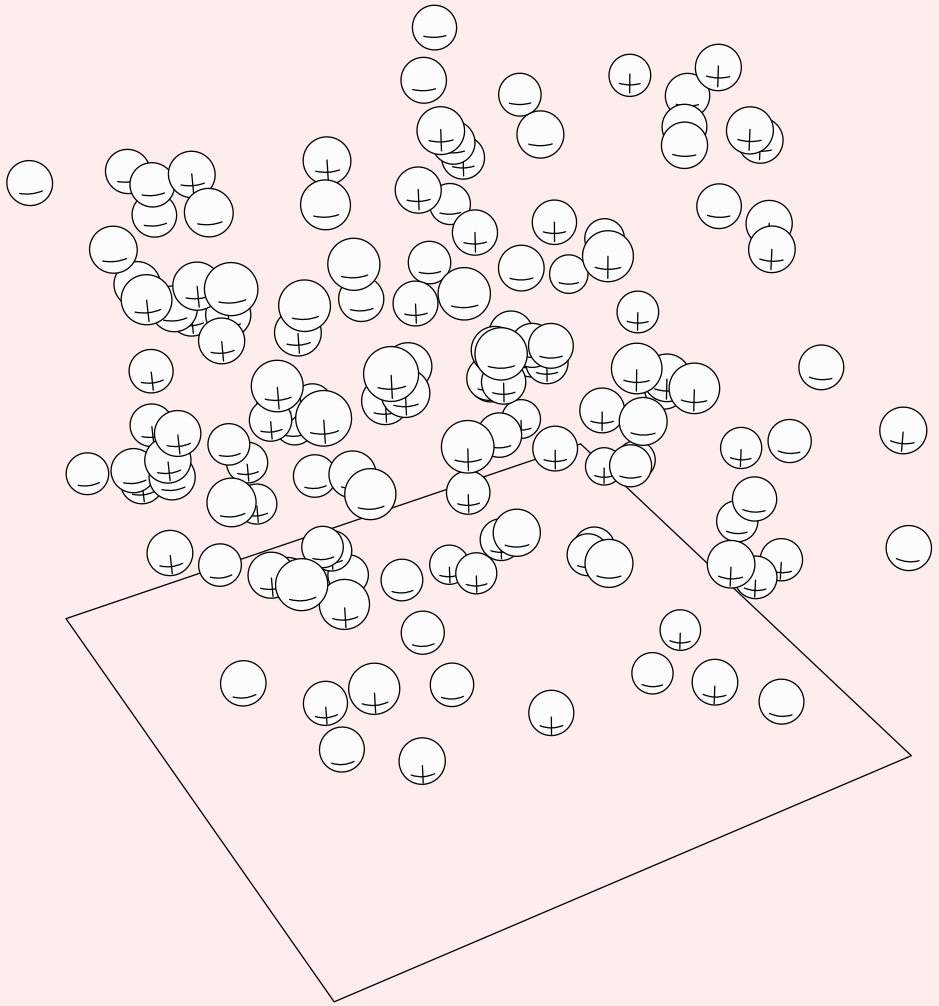


Figure: Figure that uses `positivecharge`, `getready` and `doitnow`.

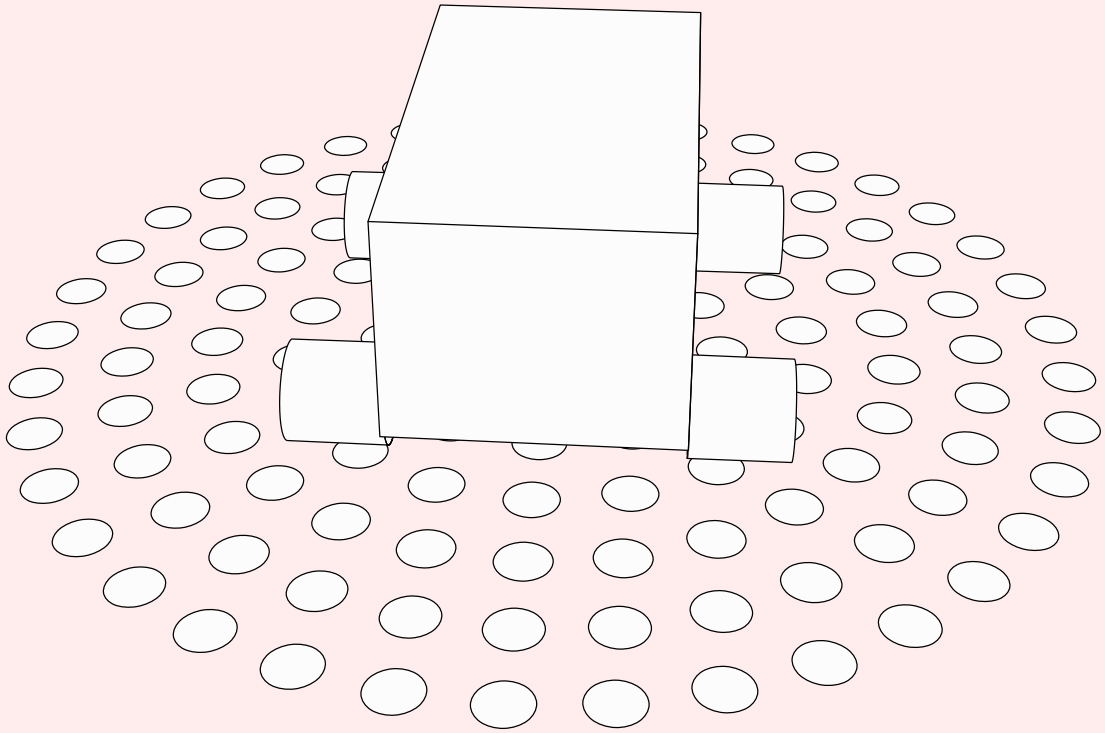


Figure: Figure that uses `setthearena` and `simplecar`.

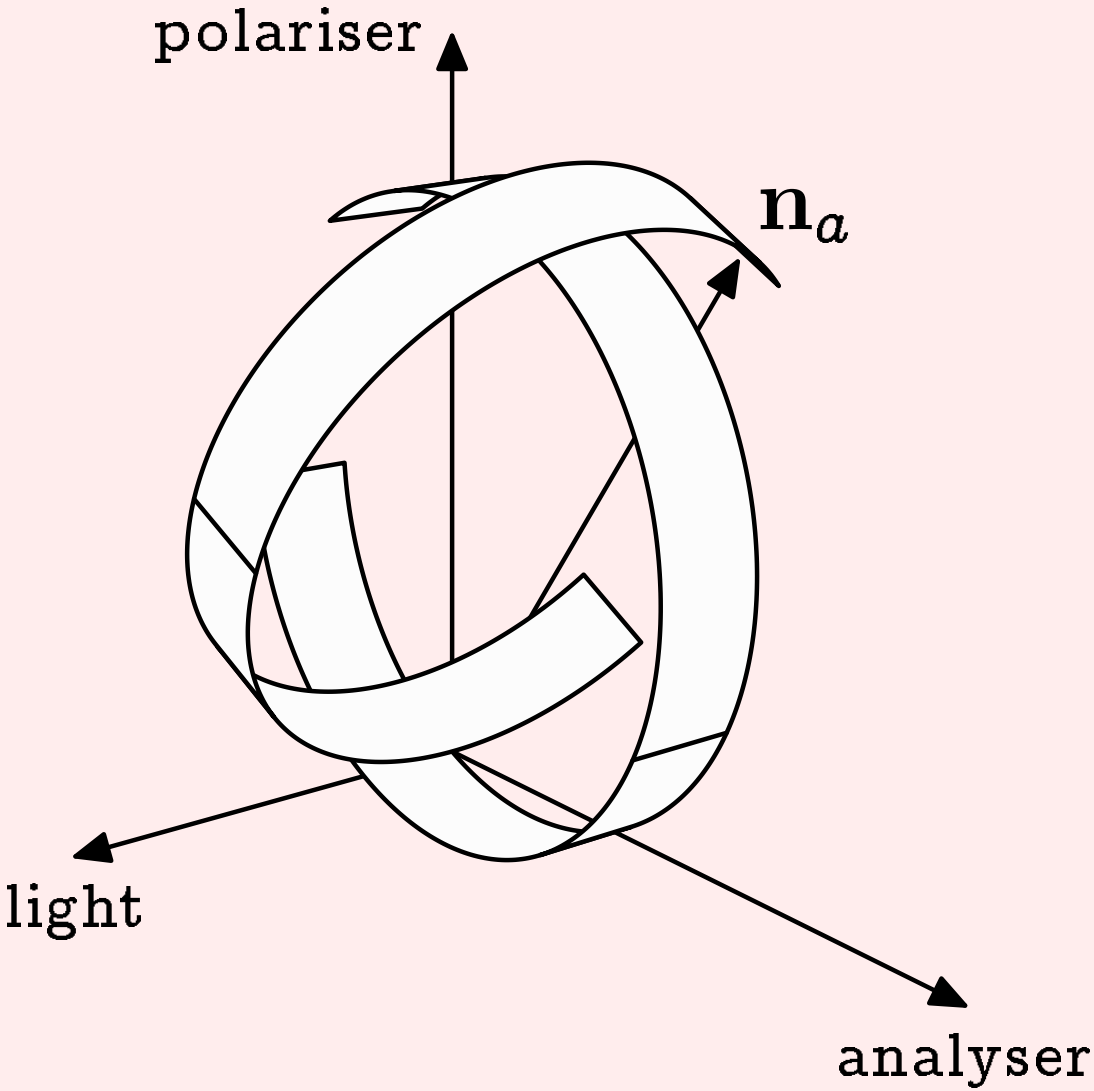


Figure: Figure that uses banana.

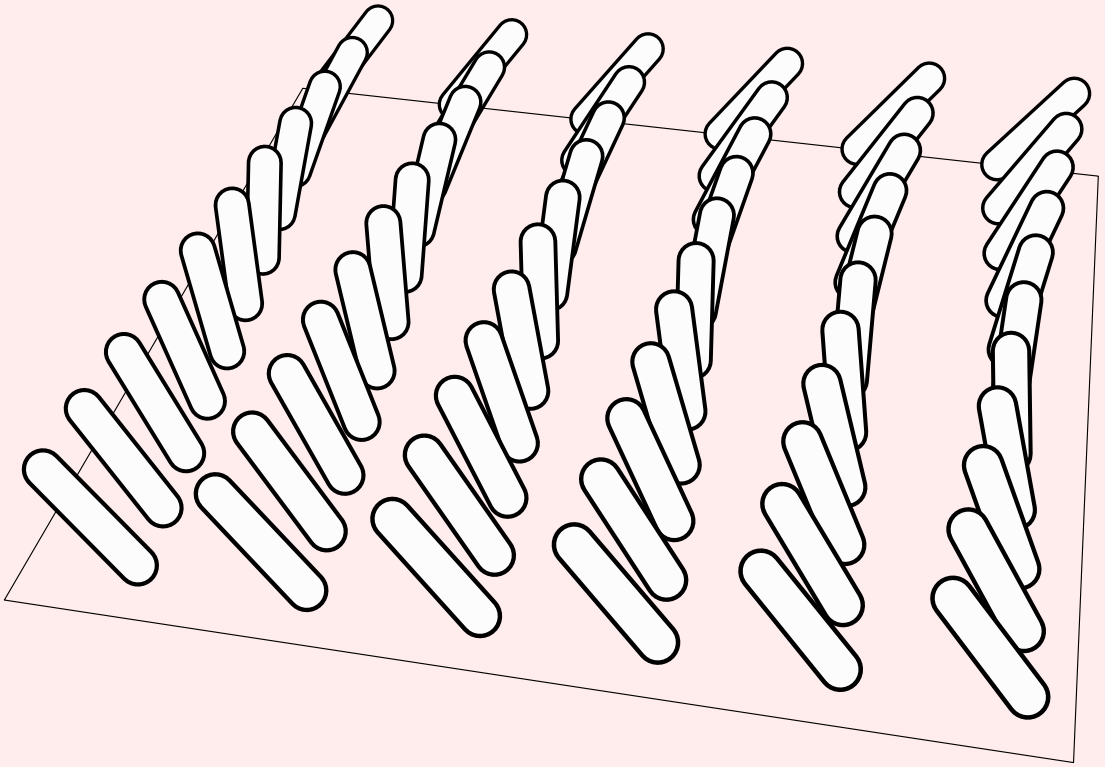


Figure: Figure that uses `director_invisible` and `generatedirline`.

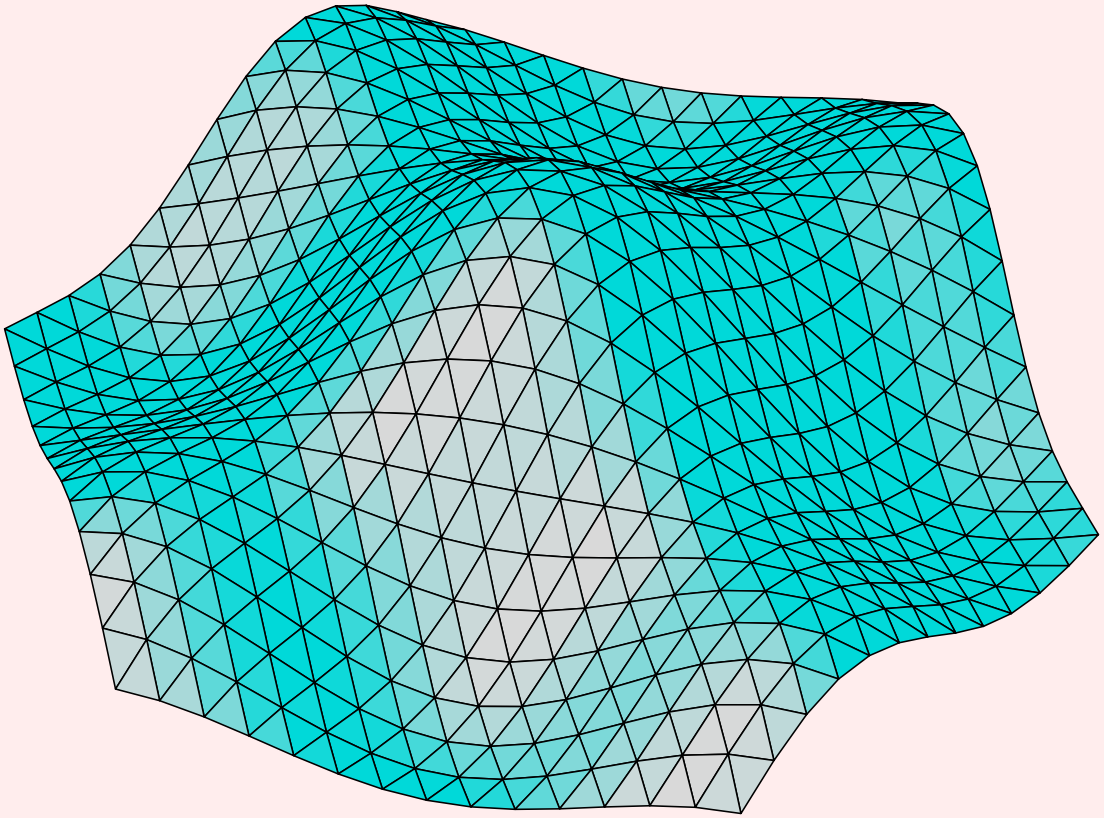
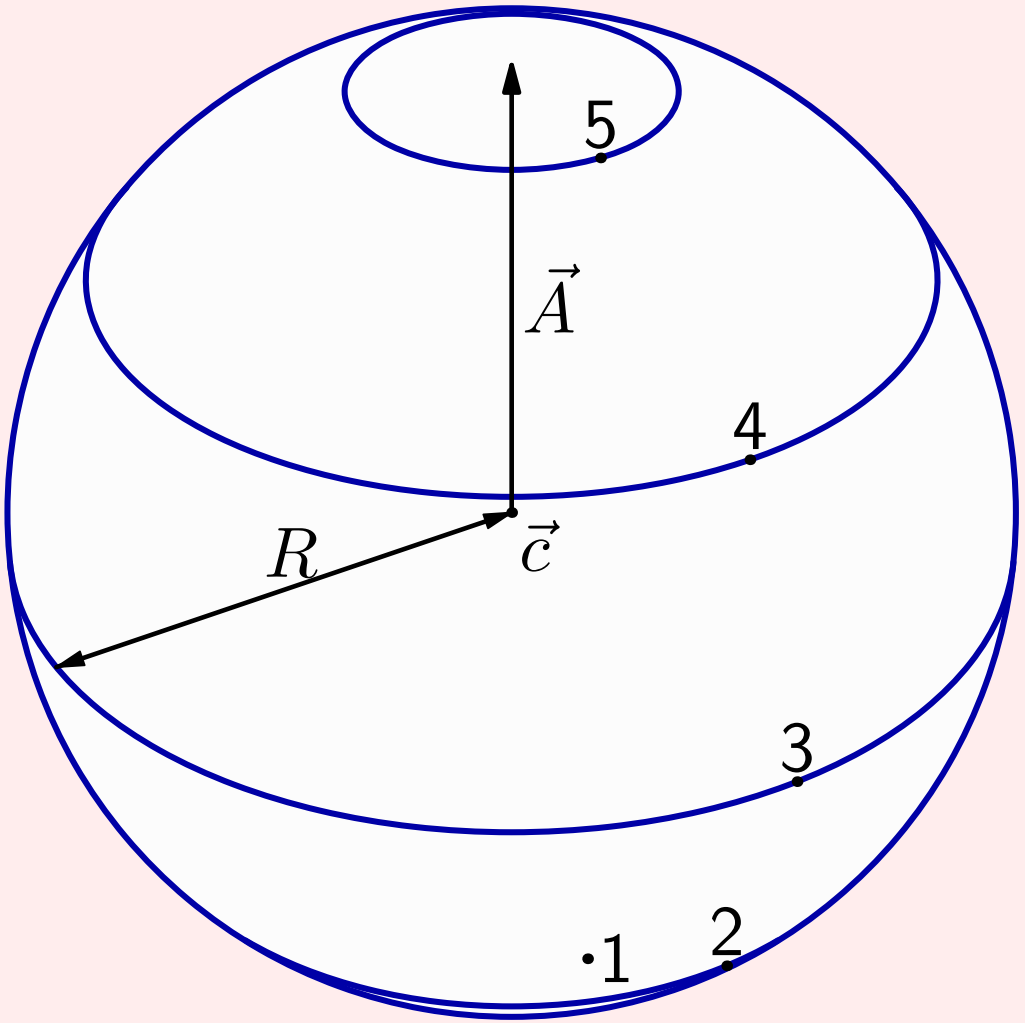


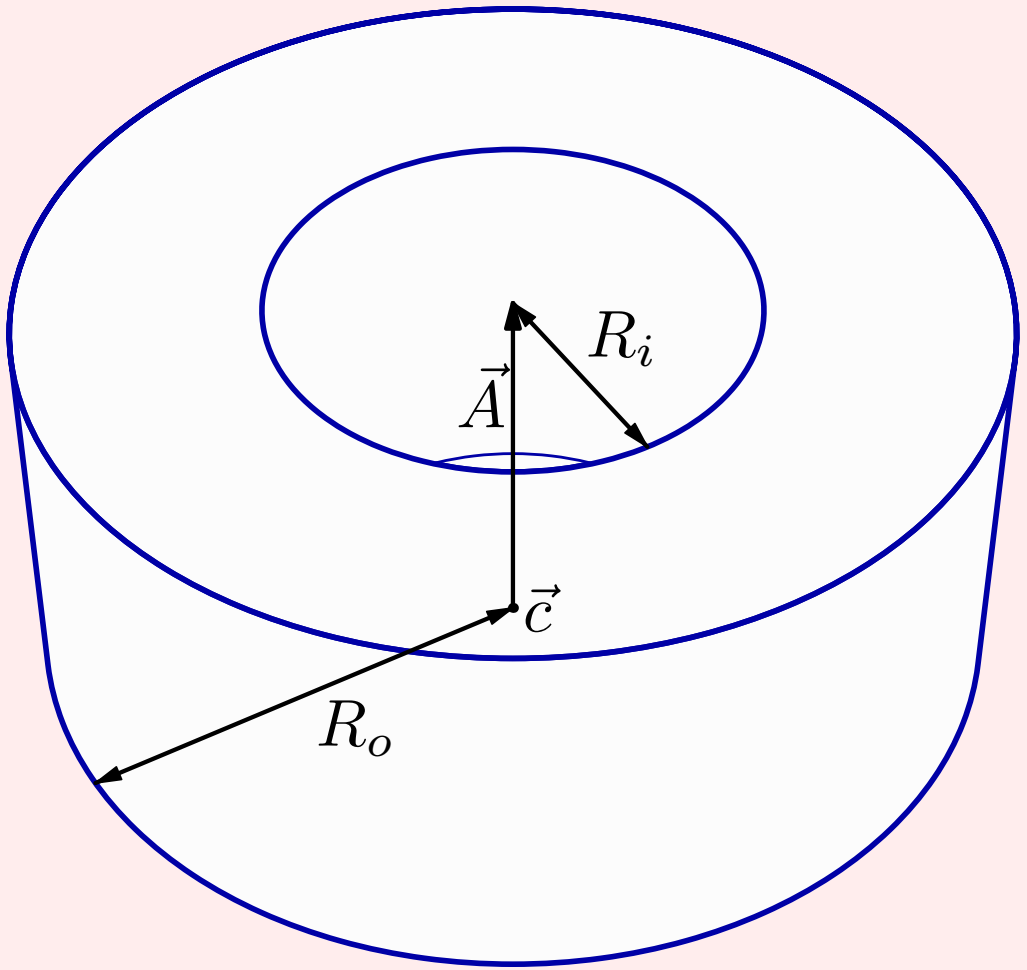
Figure: Figure that uses `hexagonaltrimesh`.

tropicalglobe(N, $\vec{c}$, R, $\vec{A}$)



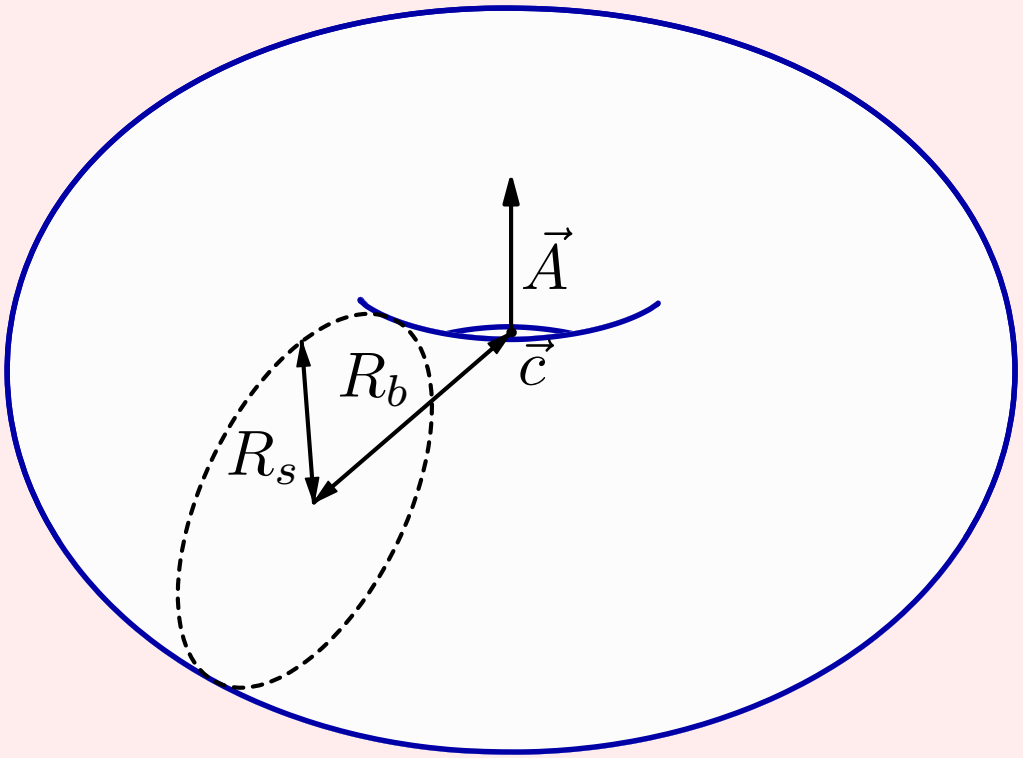

```
tropicalglobe( 5, black, 1, blue );
```

rigorousdisc(R_i, bool, $\vec{c}$, R_o, $\vec{A}$)



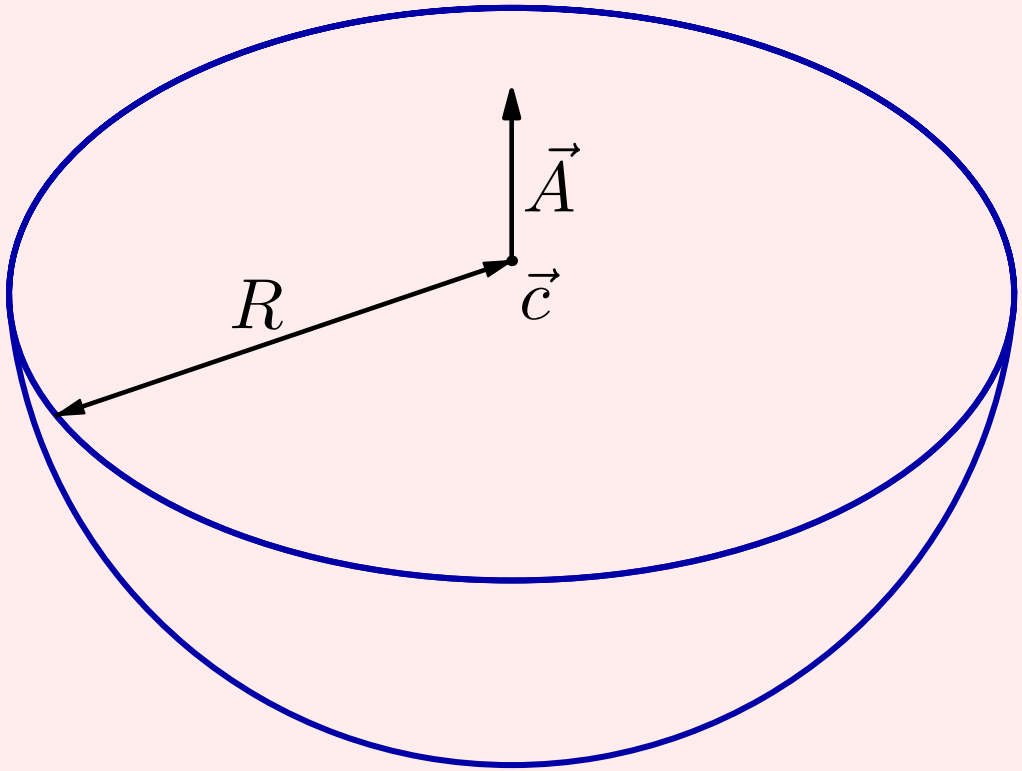

```
rigorousdisc( 0.5, true, black, 1, 0.85blue );
```

smoothtorus($\vec{c}$, $\vec{A}$, R_b, R_s)



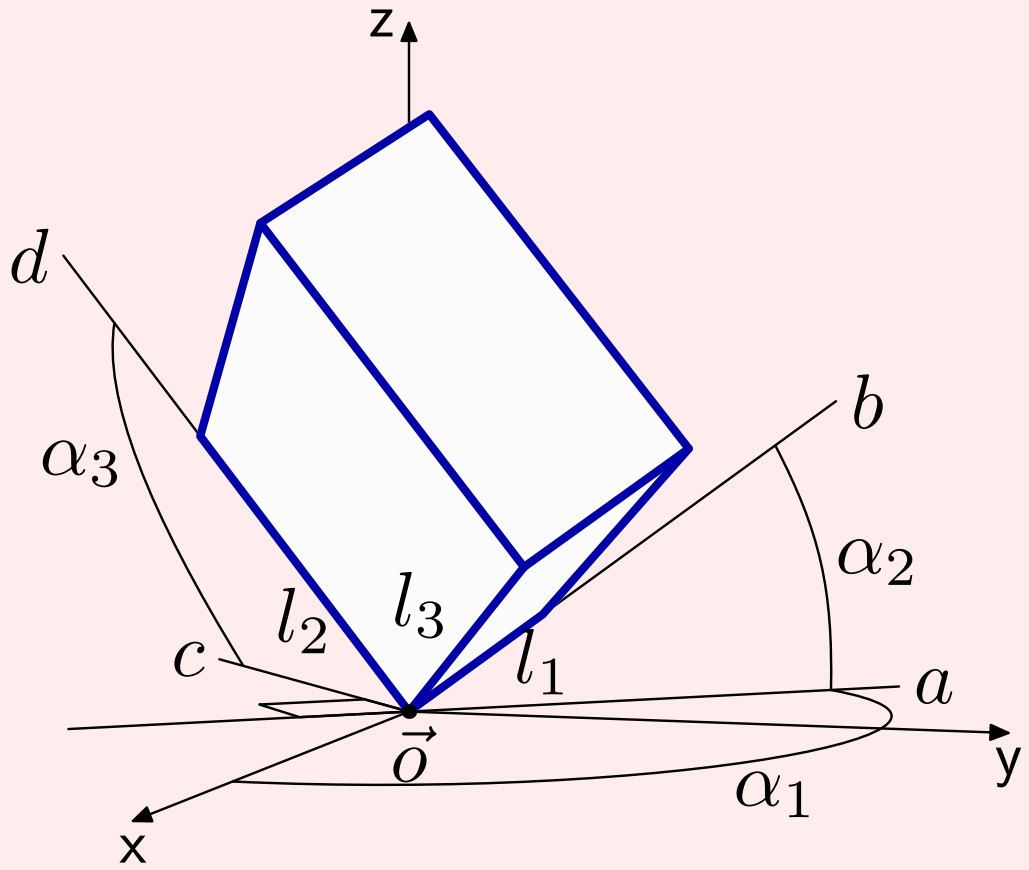

```
smoothtorus( black, blue, 0.7, 0.4 );
```

spatialhalfsfear($\vec{c}$, $\vec{A}$, R)



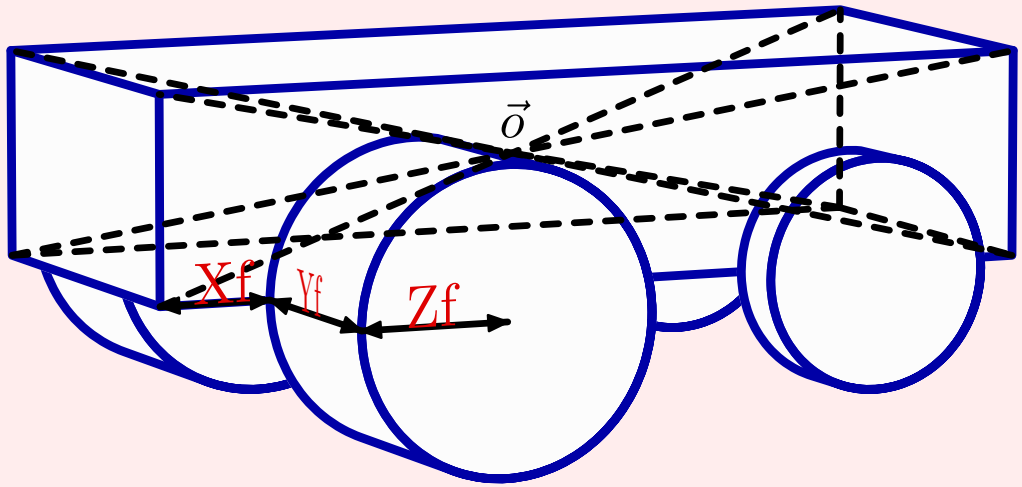

```
spatialhalfsfear( black, blue, 1 );
```

kindofcube(bool,bool,$\vec{o}, \alpha_1, \alpha_2, \alpha_3, l_1, l_2, l_3$)



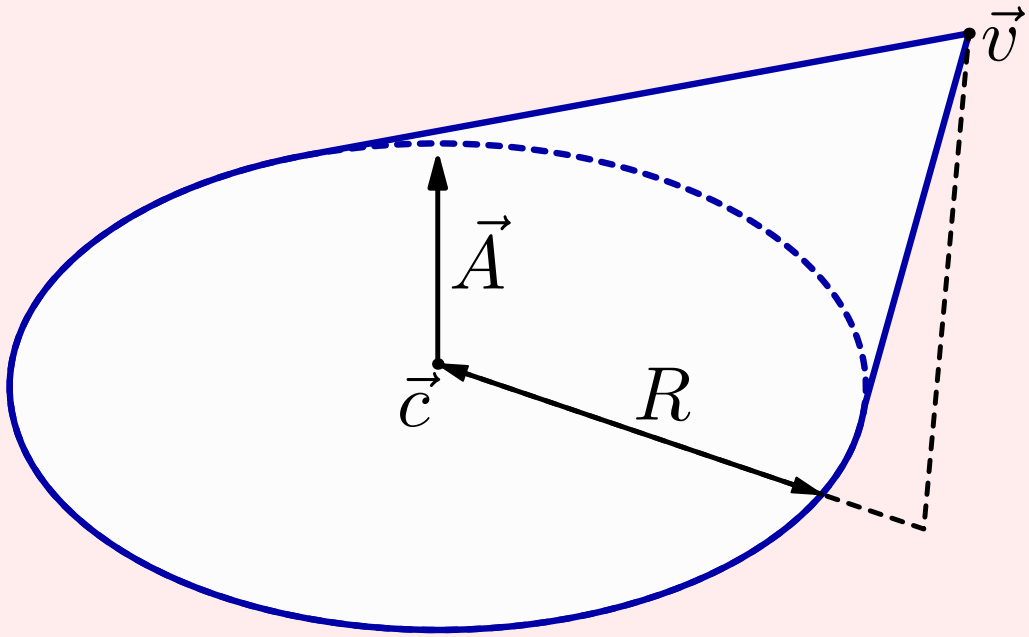

```
kindofcube( false, true, black, 130, 32, 67, 0.3,
                                0.6, 0.9 );
```

simplecar($\vec{o}$,($\alpha_1, \alpha_2, \alpha_3$), ($l_1, l_2, l_3$),
(Xf,Yf,Zf), (Xr,Yr,Zr))



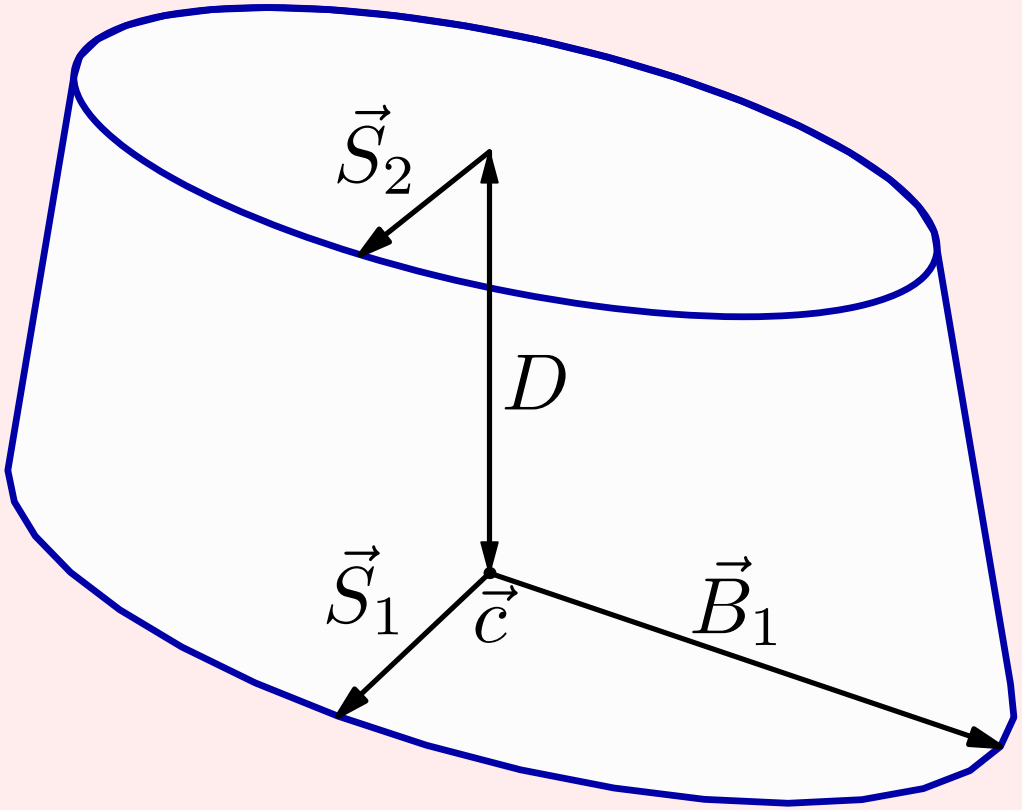

```
simplecar( black, black, (0.8,0.35,0.18),
   (0.1,0.2,0.132), (0.06,0.06,0.1) );
```

verygoodcone(bool, $\vec{c}$, $\vec{A}$, R, $\vec{v}$)



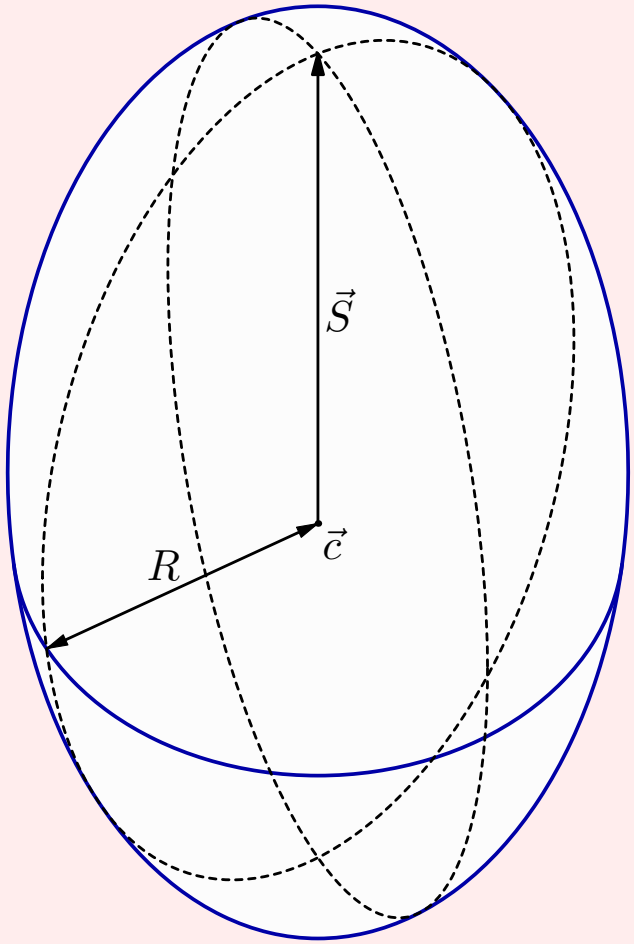

```
verygoodcone( true, black, blue, 0.8, blue+green );
```

whatisthis($\vec{c}$, $\vec{S}_1$, $\vec{B}_1$, D, $||\vec{S}_2||/||\vec{S}_1||$)



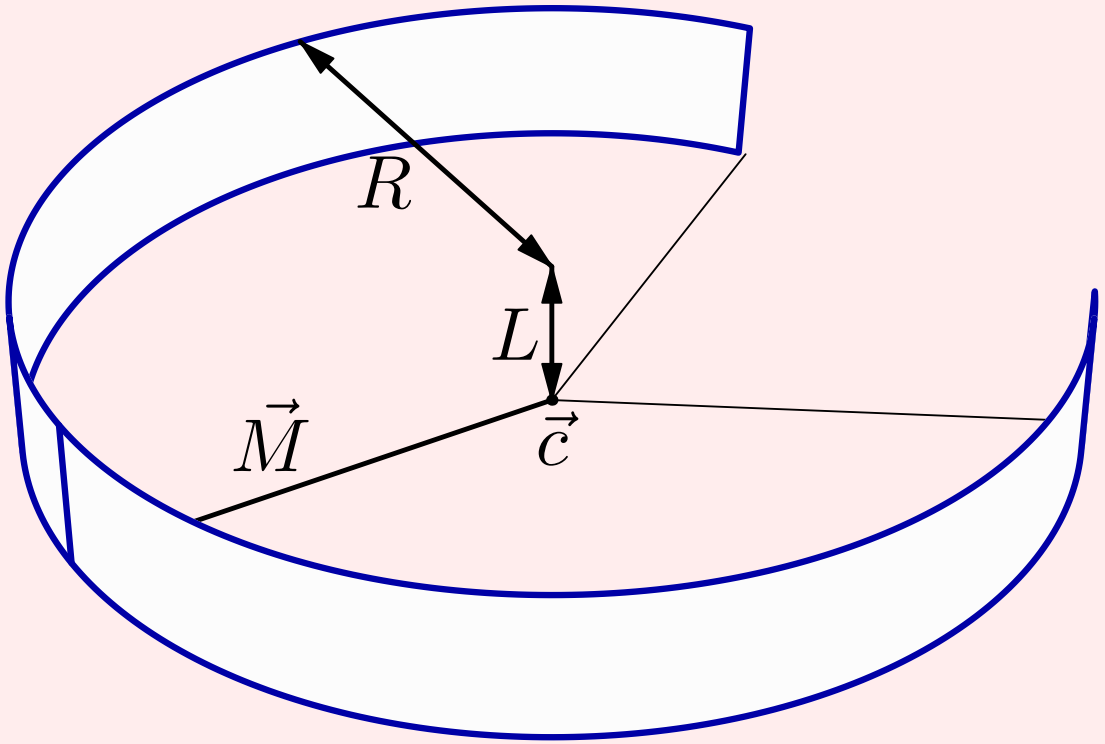

```
whatisthis( black, 0.5red, green, 0.85, 0.8 );
```

spheroid($\vec{c}$, $\vec{S}$, R)




```
spheroid( black, 2*blue, 1 );
```

banana($\vec{c}$, R, $(\alpha_M, \beta_M, \gamma_M)$, L, θ)




```
banana( black, 1, black, 0.3, 145 );
```

Acknowledgements

Many people have contributed to make FEATPOST what it is today. Perhaps it would have never come into being without the early intervention of Jorge Bárrios, providing access to his father's computer in 1986. Another important moment happened when José Esteves first spoke about METAPOST sometime in the late nineties.
Also, the very accurate criticism of Cristian Barbarosie has significantly contributed to fundamental improvements. Jens Schwaiger contributed new macros. Pedro Sebastião, João Dinis and Gonçalo Morais proposed challenging new features.