The `lua-tikz3dtools` package v2.2.0

https://github.com/Pseudonym321/TikZ-Animations/tree/master1/TikZ/lua-tikz3dtools

Jasper Nice

November 29nd, 2025

This work was typeset using TeX, the typesetting system created by Donald E. Knuth, along with various extensions and packages developed by the TeX community. I am grateful to the vibrant TeX Stack Exchange community for their ongoing support and resources. For those interested, my contributions can be found at Jasper

Jasper Nice

Contents

1 What the heck is a projective transformation? 1

2 Getting started: drawing a sphere 2

3 Filtering surfaces: problems and possibilities 2

4 Documentation of Commands and Keys 2
4.1 Setting Objects 4
4.2 Appending Points and Labels 4
4.3 Appending Curves, Surfaces, and Solids 4
4.4 Package Options and Keys 5

5 Matrix Operations and Transformations in Parametric Code 5
5.1 Core Matrix Operations 5
5.2 Standard 3D Transformations 6

1 What the heck is a projective transformation?

Unfortunately, in order to use `lua-tikz3dtools`, you need to know how to do matrix multiplications. This can be learned in one semester of linear algebra—which

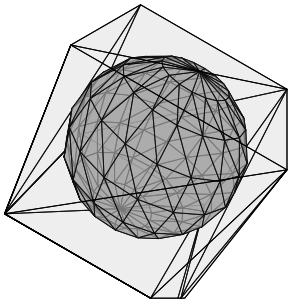

Figure 1: A sphere inside a cube, in perspective

is all I currently have. Linear algebra involves linear transformations, which exclude translations and perspective transformations. These linear transformations are encoded in 3×3 matrice (for 3D). This package also uses row-vector convention (because it is more convenient to code with), so our vectors are multiplied on the left of the transformation matrix. Using a homogeneous component, these linear transformation matrices can be transformed into affine and projective transformation matrices. I suggest the mathematical elements for computer graphics book by David Rogers (I recommend the first edition; it is free on archive.org) for learning about projective transformations. Read chapters two and three, and you'll be set.

2 Getting started: drawing a sphere

Before I drown you in documentation, here are some simple diagrams to get you started (see the source for the code):

3 Filtering surfaces: problems and possibilities

Filtering surfaces works when we don't use perspective. Currently, due to a bug, perspective breaks the filtering. I'm open to hear from anyone if they have a fix.

Additionally, the partitioning still has a bug due to degenerate triangles, so I'm all ears on that too.

4 Documentation of Commands and Keys

This section summarizes the main commands and configuration keys of the `lua-tikz3dtools` package.

This section is ChatGPT generated, and looks OK to me.

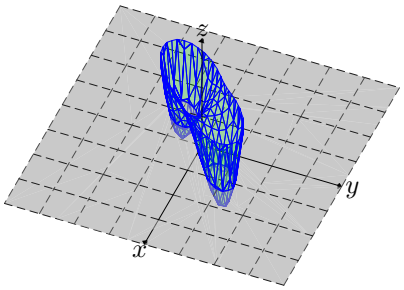

Figure 2: A case in volving filtering and partitioning

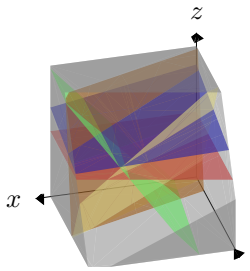

Figure 3: Intersecting planes

4.1 Setting Objects

- `\setobject[<options>]` Defines a 3D object with a transformation matrix. Options are passed as TikZ keys:
  - `name` — Name of the object.
  - `object` — Transformation matrix (default: `identity_matrix()`).

4.2 Appending Points and Labels

- `\appendpoint[<options>]` Adds a point in 3D space.
  - `x, y, z` — Coordinates of the point (default: 0,0,0).
  - `fill options` — TikZ styling for the point (default: `fill`).
  - `transformation` — Transformation matrix applied to the point (default: identity).
- `\appendlabel[<options>]` Adds a label at a 3D position.
  - `x, y, z` — Coordinates of the label (default: 0,0,0).
  - `name` — Text of the label (default: `George`).
  - `transformation` — Transformation applied to the label (default: identity).

4.3 Appending Curves, Surfaces, and Solids

- `\appendcurve[<options>]` Adds a parametric 3D curve.
  - `ustart, ustop` — Parameter range for the curve (default: 0 to 1).
  - `usamples` — Number of samples along the curve (default: 64).
  - `x, y, z` — Parametric functions of the parameter u.
  - `transformation` — Transformation matrix applied to the curve.
  - `draw options` — TikZ styling.
  - `arrow tip/tail, arrow tip/tail options` — Optional arrowheads.
  - `filter` — Boolean or Lua condition for selective drawing.
- `\appendsurface[<options>]` Adds a parametric 3D surface.
  - `ustart, ustop, vstart, vstop` — Parameter ranges.
  - `usamples, vsamples` — Number of samples along u and v.
  - `x, y, z` — Parametric functions of u and v.
  - `transformation` — Transformation matrix.
  - `fill options` — TikZ styling for the surface.
  - `filter` — Condition to include/exclude surface points.

- `\appendsolid[<options>]` Adds a parametric 3D solid (volume).
  - `ustart, ustop, vstart, vstop, wstart, wstop` — Parameter ranges.
  - `usamples, vsamples, wsamples` — Sampling resolution.
  - `x, y, z` — Parametric functions of u, v, w.
  - `transformation` — Transformation matrix.
  - `fill options` — TikZ styling for the solid.
  - `filter` — Boolean or Lua condition for selective drawing.

4.4 Package Options and Keys

All keys are accessible through Ti*k*Z's path system, under the family `/lua-tikz3dtools`. Subcategories:

- `/parametric/matrix` — Transformation matrices.
- `/parametric/point` — Individual points.
- `/parametric/label` — Labels in 3D space.
- `/parametric/curve` — Parametric curves.
- `/parametric/surface` — Parametric surfaces.
- `/parametric/solid` — Parametric solids.

5 Matrix Operations and Transformations in Parametric Code

Again, this part is ChatGPT generated. Note that new objects can be made with the `\setobject` command.

Inside all parametric fields of `lua-tikz3dtools` (for instance in `\appendcurve`, `\appendsurface`, and filtering conditions), a small collection of matrix commands is available. These functions originate from the internal module `mm`, but inside parametric expressions they are used *without* any prefix.

All transformations below return 4×4 matrices acting on homogeneous row-vectors $(x, y, z, 1)$ using the row-vector convention adopted by the package.

5.1 Core Matrix Operations

`matrix_multiply(A,B)` Computes the product $A \cdot B$. All chained transformations are formed using this routine.

`matrix_inverse(A)` Returns the inverse of a non-singular square matrix using Gauss–Jordan elimination.

5.2 Standard 3D Transformations

`xrotation(angle)` Rotation about the x-axis by the given angle.

`yrotation(angle)` Rotation about the y-axis.

`zrotation(angle)` Rotation about the z-axis.

`euler(`α, β, γ**`)`** Returns the composed rotation

$$R_z(\gamma)\, R_y(\beta)\, R_z(\alpha).$$

`translate(x,y,z)` Translation by the vector (x, y, z).

`xscale(s)`, `yscale(s)`, `zscale(s)` Scaling in the respective coordinate direction.

`scale(s)` Uniform scaling in all coordinates.

`scale3(x,y,z)` General non-uniform scaling by three independent factors.

These commands can be freely combined using `matrix_multiply` to build arbitrary affine (and some projective) transformations directly inside parametric expressions.