lua-tikz3dtools documentation v3.2.0

Jasper Nice

2026-05-21

Contents

1 What this package is

This package handles three main things; matrix based projective transformations, simplicial occlusion, and simplicial partitioning.

Users append optioned simplices and simplicially tessellated parametric objects to a big list, and lua-tikz3dtools handles the projective transformations, the simplical partitioning, and the simplicial occlusion.

2 Command reference

This section collects the public commands and their keys in reference form. Examples remain brief here; the larger workflow discussion is in the earlier sections.

2.1 \luatikztdtoolsset

Convenience wrapper for setting keys in the family `/lua-tikz3dtools/.cd`. Use it when document-wide or figure-wide defaults are worth centralizing.

2.2 \setobject

Key	Expected value	Meaning
`name`	nonempty string	Name stored in the sandbox object table.
`object`	Lua chunk	Evaluated object to store under `name`.

2.3 \appendlabel

Key	Expected value	Meaning
`v`	point expression	Three-dimensional label position.
`text`	TeX material	Node body emitted after the geometry.
`transformation`	matrix expression	Optional transform; default is identity.
`filter`	Lua block	Predicate on the projected point `A`; default returns true.

2.4 \appendlight

Key	Expected value	Meaning
`v`	vector expression	Directional light vector.

2.5 \appendtriangle

Key	Expected value	Meaning
`m`	matrix expression	Whole 3×4 triangle matrix, with one homogeneous vertex per row.
`transformation`	matrix expression	Optional transform; default is identity.
`fill options`	TikZ option list	Styling for the emitted triangle path.
`filter`	Lua block	Predicate on the triangle vertices `A`, `B`, `C`.

2.6 \appendcurve

Key	Expected value	Meaning
uparams	parameter triple	Start, stop, and sample count for u.
v	function body in u	Returns a point for each sample.
transformation	matrix expression	Optional transform; default is identity.
draw options	TikZ option list	Styling for emitted segments.
arrow tip	TikZ option list	Style for geometric arrowhead at the end.
arrow tail	TikZ option list	Style for geometric arrowhead at the start.
filter	Lua block	Predicate on segment endpoints A, B.

2.7 \appendsurface

Key	Expected value	Meaning
uparams, vparams	parameter triples	Start, stop, and sample counts for the two parameters.
v	function body in u,v	Returns the sampled point on the surface.
transformation	matrix expression	Optional transform; default is identity.
fill options	TikZ option list	Styling for emitted triangles.
filter	Lua block	Predicate on triangle vertices A, B, C.
curve	Lua chunk	Returns a table of parameter-space line segments to embed on the surface.

2.8 \appendsolid

Key	Expected value	Meaning
uparams, vparams, wparams	parameter triples	Parameter ranges and sample counts for the three directions.
v	function body in u,v,w	Returns the sampled point in space.
transformation	matrix expression	Optional transform; default is identity.
fill options	TikZ option list	Styling for emitted boundary triangles.
filter	Lua block	Predicate on triangle vertices A, B, C.

2.9 \displaysimplices

This command takes no keys. It partitions geometry if needed, applies filters, sorts by occlusion, emits TikZ paths, emits labels, and clears the accumulated scene and light list.

3 Example

```
\documentclass[tikz,border=1cm]{standalone}
\usepackage{lua-tikz3dtools} % https://github.com/Pseudonym321/TikZ-
Animations/tree/master1/TikZ/lua-tikz3dtools
\begin{document}
\foreach \CCC in {15} {
\begin{tikzpicture}
\useasboundingbox (-80mm,-45mm) rectangle (80mm,45mm);
\ltdtappendlight[v = {return Vector:new{1, 1, 1 , 1}}]
\def\SSS{3}
\foreach \AAA in {0,30,60} {
\pgfmathsetmacro{\AAA}{\AAA+\CCC}
\ltdtsetobject[
    name=a,object={return \AAA*pi/180}
]
\ltdtsetobject[
    name=R,
    object={return 1/cos(a)}
]
\ltdtsetobject[
    name=r,
    object={return sqrt(abs(1/cos(a)^2)-abs(cos(a)))}
]
\ltdtsetobject[
    name=view,
    object = {return Matrix.zyzrotation(Vector:new{pi/6, pi/3, pi/2})
    :multiply(Matrix.perspective(Vector:new{0,0,-1/8}))}
]
\pgfmathtruncatemacro{\BBB}{((\AAA-10)*10/7)}
\ltdtappendsurface[
    uparams={return Vector:new{pi/2,-pi,20}},
    vparams={return Vector:new{0,2*pi,50}},
    transformation = {return view},
    v = {
        return Vector.sphere(Vector:new{u, v, r})
        :hadd(Vector:new{R*cos(u), R*sin(u), 0, 1})
    },
    filter = {
```

```
        local M = A:hadd(B):hadd(C):hscale(1/3):multiply(view:inverse())
        return abs(M[1]) < \SSS.001 and abs(M[2]) < \SSS.001 and abs(M[3]) < \SSS.001
    },
    fill options = {fill={gray!\BBB!black!50!ltdtbrightness},fill opacity = 1},
    curve = {
        local segments = {}
        local domain_origin = pi/2
        local branches = 7
        local samples_per_branch = 40
        local samples = branches*samples_per_branch
        local step = tau/samples
        local function append_segment(s0, s1)
            if s1 <= s0 then
                return
            end
            local wrap = floor(branches*s0/tau + 1e-9)
            local u0 = domain_origin - s0
            local u1 = domain_origin - s1
            local v0 = branches*s0 - wrap*tau
            local v1 = branches*s1 - wrap*tau
            table.insert(segments, {
                Vector:new{u0, v0},
                Vector:new{u1, v1},
                drawoptions = "draw = green!50!black, thick"
            })
        end
        for i = 0, samples - 1 do
            local s0 = i*step
            local s1 = s0 + step
            local wrap0 = floor(branches*s0/tau + 1e-9)
            local wrap1 = floor(branches*s1/tau + 1e-9)
            if wrap0 == wrap1 then
                append_segment(s0, s1)
            else
                local seam = (wrap0 + 1)*tau/branches
                append_segment(s0, seam)
                append_segment(seam, s1)
            end
        end
        return segments
    }
]
} % ends foreach
\ltdtappendsolid[
    uparams={return Vector:new{-\SSS,\SSS,2}},
    vparams={return Vector:new{-\SSS,\SSS,2}},
```

```
    wparams={return Vector:new{-\SSS,\SSS,2}},
    transformation = {return view},
    filter = {return false},
    v = {return Vector:new{u, v, w, 1}}
]
\ltdtdisplaysimplices
\end{tikzpicture}} % ends \CCC
\end{document}
```