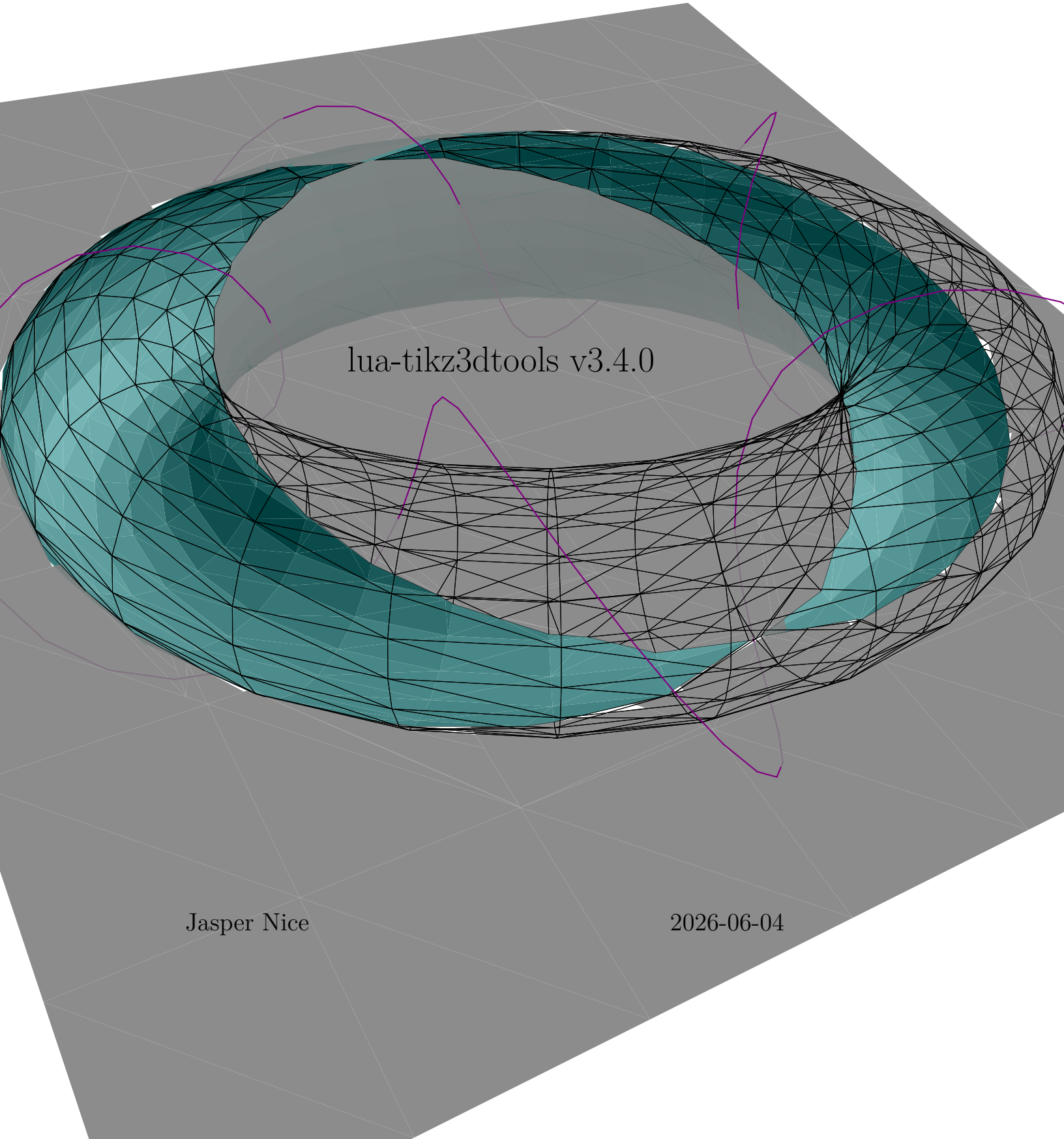
lua-tikz3dtools v3.4.0



Jasper Nice

2026-06-04

For this package, I have chosen the LaTeX Project Public License 1.3c or later.

My hope is that this book will benefit anyone who wants to learn projective transformations, simplicial occlusion, or simplicial partitioning. This package is good for learning these concepts, though I would always recomment building tools yourselves and using less restricted tools also. Still, quite a bit can be learned in just Lua and Ti*k*Z.

"I long to accomplish great and noble tasks but it is my duty to accomplish small tasks as though they were great and noble." —Sylvia Fedoruk

Contents

Preface

0.1 Who I am

I am Jasper from TEX.SX. I spent years learning Ti*k*Z and 3D, under the mentorship of helpful people on the internet, and eventually formed some ideas of my own. I am especially grateful to early illustrations from the internet which inspired me to learn to make my own.

Aside from 3D and ti*k*Z, I enjoy animals and nature. I spent 7 years volunteering in therapeutic horse riding, as well as many summers working in animal care. One of my favourite experiences was getting to work with at a bird rehabilitation centre—with skunks, squirrels, owls, a bald eagle, and yes, even ducks. Most of my work was with horses though.

0.2 What this manual is about

This is the manual for lua-tikz3dtools, a package which is designed to teach 3D philosophy in a restricted yet still capable evironment. lua-tikz3dtools is a set of tools for drawing projectively transformed, and properly occluded simplicial scenes. This is a lot of jargon, and we will soon learn what to do to learn it and then how to use it.

Simplicial scenes are scenes composed of simplices, which is plural for simplex. A simplex in 3D is any element of the collection of all points, line segments, and triangles in 3D. In lua-tikz3dtools, we typically focus on scenes composed of line segments and triangles.

Occlusion is the problem of determining what should be seen first if there is visual overlap between the simplices. Projective transformations are the collection of all translations, rotations, shears, reflections, and perspective transformations in 3D. It's a superset of the affine transformations, which are themselves a superset of the linear transformations.

All the things which I think are prerequisite to this book are in the appendix, and I recommend that readers start there for their journey. If you dive into chapter 1 immediately, you will likely have difficulty with things like affine algebra later on. It's worth it to have your ducks in a row before going off on an adventure.

0.3 The scope of this manual

This manual is all about learning my paradigms for projectively transformed and properly occluded simplicial illustrations. This is just a slice of the broader 3D graphics scene, albeit a pedagogically valuable slice. We will not cover things like pixelated, spline-based, or implicit 3D illustrations. We do hope though that our affine algebraic—an important projective subset—principles will aid you in synthesizing your own ideas.

Part I

Foundations

Chapter 1

Basic linear algebra

Basic linear algebra is super fundamental to what this package does, so we will give an overview of the prerequisites here. My hope is that this chapter will be short and accessible, with just enough core concepts to do pretty much whatever we want down the road. Linear algebra can become quite memory intensive if we look at a typical university curriculum. I have deliberately chosen to write this chapter in a more immediately accessible way, with just some core concepts.

1.1 Linear operations on geometric-coordinate vectors

In basic linear algebra, vectors are typically introduced in a more conceptually intuitive way. Specifically, they are commonly represented as directed line segments. This is a geometric-coordinate vector. Vectors can be much more abstract than this, but that is not basic linear algebra.

Then the student is introduced to operations on these vectors. An operation can take one or more operand, and in the case of linear operations, these operands are vectors. However, there are operations later on down the road which accept operands of different types, so this isn't a rule.

The linear operations achieve intuitive geometric effects, such as measuring distances, angles, or even producing new vectors with special properties relative to the operands; e.g., orthogonality.

1.1.1 Geometric-coordinate vectors in 3D

A geometric-coordinate vector in 3D is any element of the collection of 3-arrays of numbers and unique coordinates in 3-space, which are bijectively related. This is a bunch of jargon which says that every sequence of 3 numbers corresponds to exactly one coordinate in 3-space. This is not a rigorous explanation, but I hope that it lands instead on a more intuitive basis. See Figure 1.1.

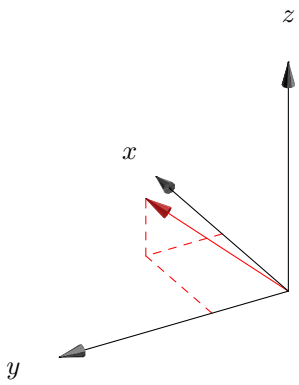


Figure 1.1: A geometric-coordinate vector in 3D

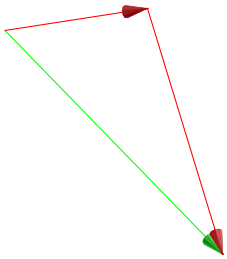


Figure 1.2: Geometric-coordinate vector addition in 3D

1.1.2 Addition and scalar multiplication

We can add and scale geometric-coordinate vectors. When we add them coordinate-wise, we add the components. This corresponds to geometric-wise addition where we connect the tail of the second vector to the tip of thefirst vector and produce the vector from the first's tail to the second's tip. For example,

$$\begin{bmatrix}1\\2\\3\end{bmatrix} + \begin{bmatrix}2\\2\\1\end{bmatrix} = \begin{bmatrix}3\\4\\4\end{bmatrix}.$$

See Figure 1.2. Similarly, we scale vectors coordinate-wise by scaling each element uniformly. Geometrically, this produces a vector which points in the same direction, and has its length uniformly scaled.

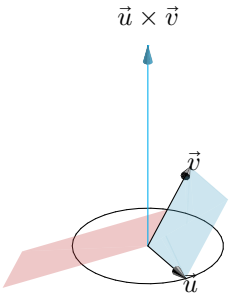


Figure 1.3: The dot product (red) and cross product (blue)

1.1.3 Dot and cross products

The dot product is useful for measuring angles and directional distances. It is written as follows;

$$\begin{bmatrix}1\\2\\3\end{bmatrix} \cdot \begin{bmatrix}2\\2\\1\end{bmatrix} = \begin{bmatrix}1\cdot 2\\2\cdot 2\\3\cdot 1\end{bmatrix} = \left\| \begin{bmatrix}1\\2\\3\end{bmatrix} \right\| \left\| \begin{bmatrix}2\\2\\1\end{bmatrix} \right\| \cos(\text{ least angle }).$$

The cross product is a means of achieving orthogonality. It has a formula, but the formula is unmemorizable. It could be derived by a determinant of a vector valued matrix, but that would be too expensive in terms of speed. So, we will be geometric here, instead of numeric. In Figure 1.3, the area of the red region is the dot product, and the cross product is the blue vector and its length is the area of the blue region.

1.2 Linear combinations of geometric vectors

Arrows (an informal, yet intuitive terminology) are very useful for **navigating subspaces**. A subspace can be a point, line, or plane which passes through the origin. In the chapter on affine algebra, we will extrapolate this concept to affine subspaces which do not necessarily coincide the origin. The steering wheel we use is called a linear combination of a linear basis. The linear basis is the set of directions we can travel, and the combination is the sum of them, each scaled by various lengths. Recall that summing scaled vectors lets us traverse space by their length and direction.

1.3 Navigating subspaces using bases

A 3D linear subspace basis is a matrix of zero, one, or two 3-tuples. These vectors absolutely cannot be redundant. There must be a bijection between every linear combination and every point in the subspace. That's a bit fancy language, but it means they must be one for one.

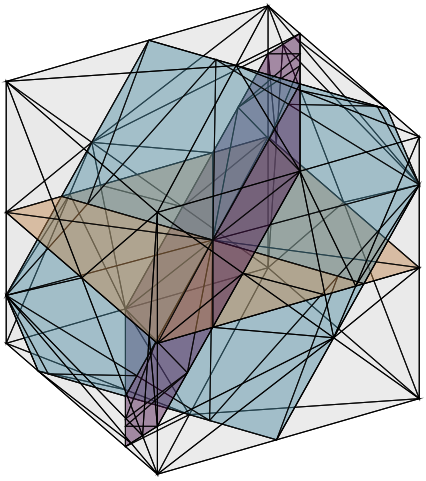


Figure 1.4: Intersecting subspaces

1.4 Intersections of subspaces

We use Gauss Jordan elimination to identify intersections of subspaces. You do not need to know how to compute column reduction fot this software. See Figure 1.4. For example, if we have two plane equations, we can use column reduction—row reduction is more common, but harder to type— to determine the subspace through which they intersect. Subspaces always intersect in lower dimensional subspaces.

1.5 Linear transformations and their inverses

A linear transformation is a change of basis vectors, relative to the current ones! That's all there is to it. If we have a basis, and another matrix, their product is the basis of the second matrix, relative to the first basis. We can do things like shears, and reflections; e.g., rotations. But linear algebra does not give us translation. For that we need affine algebra.

Chapter 2

Affine Algebra

This chapter explores affine algebra, which is an extension of linear algebra that includes translations. Translations enable us to navigate affine subspaces, which are produced from affine bases. An affine subspace is basically a translated linear subspace, so affine subspaces can be geometrically interpreted as points, lines, planes, and hyperplanes with a local origin which do not necessarily coincide the standard origin.

Affine bases The fundamental object of an affine subspace is its affine basis. An affine basis is a point and a set of zero or more vectors protruding from it. The point behaves as the origin, and the vectors are used to traverse the affine subspace which exists through their origin and is spanned by the *independent* vectors.

Affine simplices Every unique n-dimensional affine basis has exactly one corresponding n-dimensional affine simplex. An affine simplex is a point, or a line segment, or a triangle, or a tetrahedron, or any of their higher dimensional analogues. Curved mathematical objects are understood by tessellating them into affine simplices. By understanding the affine algebra of simplices, we can better understand curved objects.

Affine subspaces By going the other way and expressing a simplex as a basis, we can navigate spaces spanned by simplices using vector addition and scalar multiplication. We could even, for instance, take the intersection of a space formed by one simplex with a space formed by another simplex. Zooming out, we can imagine how this would be useful for understanding tessellated objects, which are composed of many simplices.

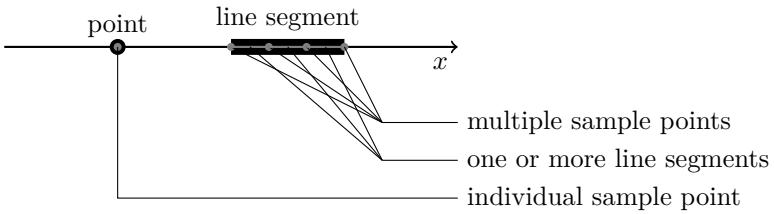


Figure 2.1: Tessellated parametric objects in $\mathbb{R}^1$.

2.1 Affine algebra in $\mathbb{R}^1$

We will begin our study of affine algebra with parametric objects in $\mathbb{R}^1$. This is because in order to illustrate affine transformations in higher dimensions, we will need to visualize an extra dimension. By starting in $\mathbb{R}^1$, we can do affine transformations entirely in two dimensions, which we can visualize easily.

2.1.1 Parametric objects in $\mathbb{R}^1$

There are exactly two classes of parametric objects which may exist in $\mathbb{R}^1$:

- those which are tessellated by line segments, and
- those which are tessellated by an individual point.

This is no coincidence, since these are all the simplices whose bases have maximum dimension one. In general, the maximum dimensional possible simplex in a space is equal to the dimension of the space itself.

For example, a parametric line segment—a curve in $\mathbb{R}^1$—is tessellated by line segments, and a parametric point is tessellated by an individual point. Points and line segments are collectively referred to as the **0–1-dimensional simplices**. In general, an n-simplex is a set of $n+1$ points, such that their basis representation is affine independent—which is analogous to linear independence.

This might all seem a bit abstract, so let's look at a couple parametric objects in $\mathbb{R}^1$; remember, there are two types. All line segments in $\mathbb{R}^1$ are 1-dimensional parametric objects. All points in $\mathbb{R}^1$ are 0-dimensional parametric objects. In order to obtain a tessellation, we take a sequence of samples along the object, and connect the samples with simplices. For a line segment, these simplices would be line segments; for a point, it would be just the point itself. See Figure 2.1 for the visual analogue.

Example 2.1.1. Illustrate the tessellation of the parametric object defined by

$$f(u) = 1 + u, \ u \in [0, 2],$$

using five sample points in an arithmetic sequence.

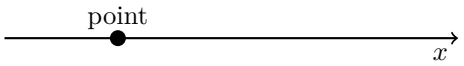


Figure 2.2: A point simplex in $\mathbb{R}^1$.

Solution 2.1.1. Every parametric object, in every dimension, is defined by two components. The first component is the parametric equation, and the second component is the domain. In this case, the parametric equation is $f(u) = 1 + u$, and the domain is $[0, 2]$.
A tessellated parametric object is a parametric object which is approximated by simplices. To obtain a tessellation by an arithmetic sequence, We find the start point a, the stop point b, and using the number of samples c, the step size

$$d = \frac{b-a}{c-1}.$$

With all this information, we can draw a number line containing both endpoints, and then place the sample points along it, starting from the start point and incrementing by the step size until we reach the stop point. Finally, we connect the sample points with line segments.

In this case, the start point is $f(0) = 1$, the stop point is $f(2) = 3$, and the step size is $(3-1)/(5-1) = 0.5$.

Remark. General methods for tessellating parametric objects become more advanced beyond $\mathbb{R}^1$. We will briefly discuss some of these in $\mathbb{R}^2$.

Now we will discuss each of the simplicial classes.

Points

The matrix representation of a point simplex a in $\mathbb{R}^1$ is

$$[a].$$

The matrix simply contains the point's coordinate as its only element. The affine basis representation of a point simplex is the same as its matrix representation. This is because the dimension of a point is zero, hence there being no protruding vectors in its basis—there's no space to span. See Figure 2.2 for the visual analogue.

Line segments

The matrix representation of a line segment simplex $\{a, b\}$ in $\mathbb{R}^1$ is represented by a 2×1 matrix. The elements of this matrix are simply the coordinates of the two endpoints.

To obtain the affine basis representation of this line segment matrix, we replace the second row by its difference with the first—turning it into a difference

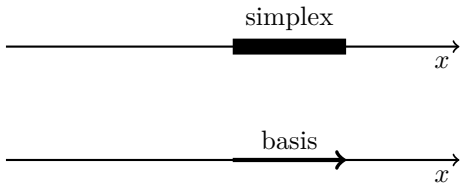


Figure 2.3: The simplex and basis of a line segment in $\mathbb{R}^1$.

vector—so that it can be added to the first to yield the original second point:

$$\underset{\text{simplex}}{\begin{bmatrix} a & b \end{bmatrix}} \sim \underset{\text{basis}}{\begin{bmatrix} a & b-a \end{bmatrix}}.$$

Now, we have two equivalent representations of the same line segment. See Figure 2.3 for the visual analogue. We usually start from a parametric object, and then inquire about its nature by tessellating it into simplices and using their affine basis representations. We will explore this further in $\mathbb{R}^2$ and $\mathbb{R}^3$ where the results become more interesting—involving solving systems of equations.

Remark. A matrix is only a simplex representation if its basis representation is affine independent; that is, if its protruding vectors are linearly independent.

2.1.2 Linear transformations in $\mathbb{R}^1$

A linear transformation in $\mathbb{R}^1$ is simply a scaling by some non-zero factor k. This is because the only linear transformation matrix which can exist in $\mathbb{R}^1$ is a 1×1 matrix, which is simply a scalar. This has the effect of scaling the real line, including reflection and degeneration to a point.

Linear transformations can be thought of as a change of basis. Again, change of basis might sound abstract, but it is really just taking one basis vector and changing it's displacement. Even more basically, take a nondegenerate vector from the origin, and make it's endpoint literally anything—even the origin, or even the original point in the case of the identity. Then, every other point in the line is scaled accordingly. See Figure 2.4 for the visual analogue.

2.1.3 Affine transformations in $\mathbb{R}^1$

Affine transformations are a strict superset of the linear transformations, in that all linear transformations are affine transformations even though not all affine transformations are linear. In plain speak, affine transformations add translation to our set of available transformations. This enables us to perform transformations with respect to points which are not the origin.

These translations are enabled by orthogonally projecting our space—normally by one unit—into a new dimension. By doing this, we can shear the extra dimension to move our origin. Finally, to obtain the result of the transformation, we inverse orthogonally project our space back onto the original region it occupied.

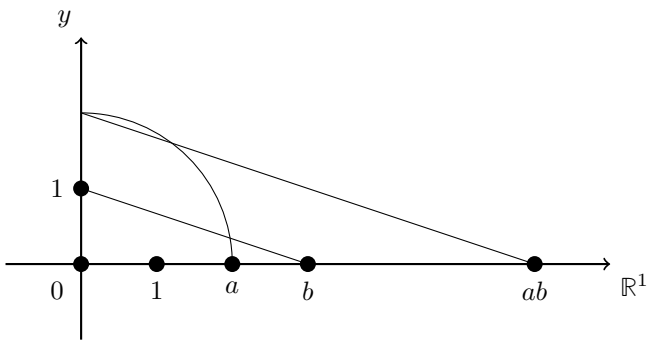


Figure 2.4: Scaling a by b in $\mathbb{R}^1$.

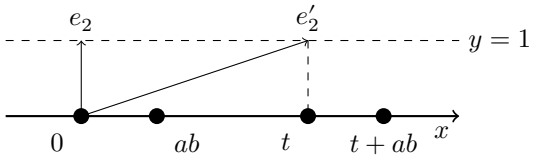


Figure 2.5: Translating $[0, ab]$ by t in $\mathbb{R}^1$.

Alternatively, one can thing of affine transformations as a composition of a linear transformation and a translation. First we perform the linear transformation, as in Figure 2.4, and then we perform the the translation, as in Figure 2.5. In particular, we project our space onto the line $y = 1$, shear the new dimension to move the origin, and then project back down to the x-axis. Shearing is a linear transformation in that higher dimensional space, and we will soon study geometric linear transformation when we examine Figure 2.10.

Remark. We are only concerned with the point $t+ab$ when translating a point. If we translate a line segment $[a, c]$, then we obtain the its affine image $[t+ab, t+cb]$. Each point would get its own construction.

Affine transformations in $\mathbb{R}^1$ can be represented by 2×2 matrices of the form

$$\begin{bmatrix} k & t \\ 0 & 1 \end{bmatrix},$$

where k is the scaling factor, and t is the translation amount. You might notice that a 2×2 matrix cannot directly transform a 1×1 or a 1×2 matrix. To resolve this, we augment the matrix by adding a second row with the value 1—projecting into a new dimension:

$$\underset{\text{point}}{\begin{bmatrix} a \\ 1 \end{bmatrix}} \qquad \underset{\text{line segment}}{\begin{bmatrix} a & b \\ 1 & 1 \end{bmatrix}}.$$

Now when we multiply the affine transformation matrix with the augmented matrices, we obtain

$$\begin{bmatrix} k & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ 1 \end{bmatrix} = \begin{bmatrix} ka + t \\ 1 \end{bmatrix}$$

for points, and

$$\begin{bmatrix} k & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a & b \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} ka + t & kb + t \\ 1 & 1 \end{bmatrix}$$

for line segments. In both cases, we can discard the last row—projecting back down to $\mathbb{R}^1$—to obtain the transformed object in $\mathbb{R}^1$.

The inverse of an affine transformation

The inverse of an affine transformation matrix is also an affine transformation matrix. Not only that, but the inverse is computed in exactly the same way as with linear transformation matrices.

2.1.4 Projective transformations in $\mathbb{R}^1$

Projective transformations are a strict superset of affine transformations, in that all affine transformations are projective transformations even though not all projective transformations are affine. In plain speak, projective transformations add perspective to our set of available transformations. This enables us to perform transformations which simulate the effect of viewing objects from different vantage points.

Projective transformations are enabled by both altering the values the bottom row of the affine transformation matrix, and by modifying how we multiply matrices—by adding an additional division step after the multiplication step. Specifically, we divide each matrix component by the homogeneous component of its vector.

Remark. We only perform the projective step when we are transforming a parametric object—i.e., its tessellated simplices. We **do not** perform the projective step when composing multiple projective transformation matrices together.

A projective transformation in $\mathbb{R}^1$ can be represented by 2×2 matrices of the form

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

where c is not zero. When we multiply this matrix with the augmented matrix of a line segment, we obtain

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} ae + b & af + b \\ ce + d & cf + d \end{bmatrix}.$$

by regular matrix multiplication. However, to complete the projective transformation, we must divide each element in the first row by the corresponding

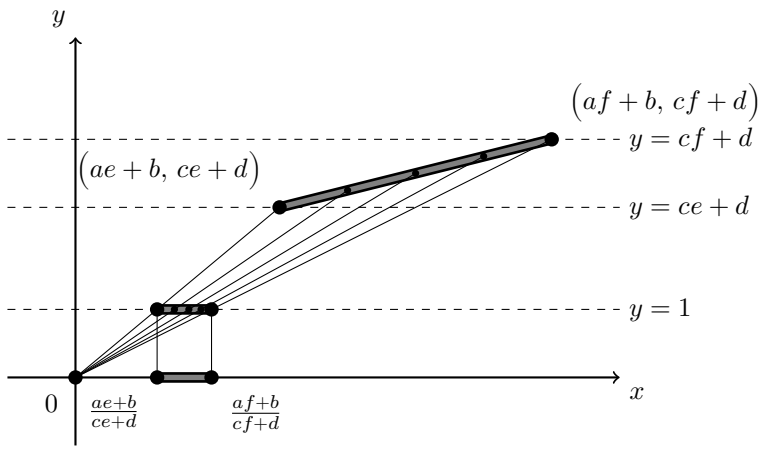


Figure 2.6: Projective transformation of a line segment in $\mathbb{R}^1$.

element in the second row:

$$\cdots = \begin{bmatrix} \frac{ae+b}{ce+d} & \frac{af+b}{cf+d} \\ 1 & 1 \end{bmatrix}.$$

Again, we can discard the last row to obtain the transformed object.

This all can seem very abstract, so let's analyze some what happens because of the c component. Notice that it scales the homogeneous component by the product of itself with the coordinate. This means that we are getting a scaling which is linearly dependent on the coordinate itself. Of course, this means that the equation is linear—with a constant term. When we divide by the homogeneous component afterward, we are effectively scaling the coordinate by the multiplicative inverse of a linear function. See Figure 2.6 for a visual analogue of transforming a line segment by a projective transformation.

Remark. Functions of the form

$$\frac{ax+b}{cx+d}$$

are called **Möbius transformations**. They are also called **fractional linear transformations**, or the **bilinear transformations**. We are essentially dealing with real-valued Möbius transformations in this chapter.

The inverse of a projective transformation

The inverse of a projective transformation matrix is also a projective transformation matrix. Not only that, but the inverse is computed in exactly the same way as with affine and linear transformation matrices. We just perform the homogeneous divide each pass.

2.2 Affine algebra in $\mathbb{R}^2$

There are three classes of parametric objects which may exist in $\mathbb{R}^2$: those which are tessellated by triangles, those which are tessellated by line segments, and those which are tessellated by individual points. This is no coincidence, since these are all the simplices whose affine bases have maximum dimension two.

A parametric curve is tessellated by line segments, and a parametric surface is tessellated by triangles. A parametric point, of course, is tessellated by an individual point. There can also be individual triangles, but they are non-parametric objects.

2.2.1 Simplices in $\mathbb{R}^2$

Points, line segments, and triangles are collectively referred to as the **0–2-dimensional simplices**. In general, a simplex is a set of one or more affine independent points. A set of points is affine independent if the protruding vectors of their affine basis are linearly independent.

Points

A point simplex in $\mathbb{R}^2$ is represented by a 1 2×1 matrix

$$\begin{bmatrix} a \\ b \end{bmatrix}.$$

The matrix representation of a point simplex is the same as its affine basis representation.

Line segments

A line segment simplex in $\mathbb{R}^2$ is represented by a 2×2 matrix. To obtain the affine basis representation of this line segment, we replace the second column by its difference with the first—turning it into a difference vector—so that it can be added to the first to yield the original second point:

$$\underset{\text{simplex}}{\begin{bmatrix} a & c \\ b & d \end{bmatrix}} \sim \underset{\text{basis}}{\begin{bmatrix} a & c-a \\ b & d-b \end{bmatrix}}.$$

Triangles

In the same way, a triangle simplex in $\mathbb{R}^2$ is represented by a 2×3 matrix. To obtain the affine basis representation of this triangle, we replace the second and third rows by their differences with the first—turning them into difference vectors:

$$\underset{\text{simplex}}{\begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}} \sim \underset{\text{basis}}{\begin{bmatrix} a & c-a & e-a \\ b & d-b & f-b \end{bmatrix}}.$$

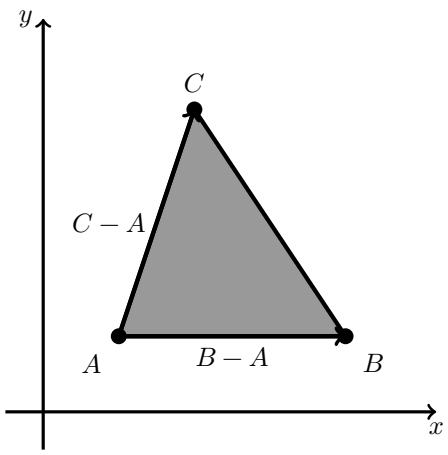


Figure 2.7: The simplex and basis of a triangle in $\mathbb{R}^2$.

See Figure 2.7 for the visual analogue of the simplex and basis of a triangle in $\mathbb{R}^2$. The lower dimensional analogues are found the same way.

2.2.2 Parametric objects in $\mathbb{R}^2$

Parametric objects in $\mathbb{R}^2$ are tessellated by simplices. Tessellated parametric objects are created by taking a domain space, sampling it, and tiling between the sample points with simplices. Logistical sampling and tessellation of parametric objects is advanced. We focus on the case where an arithmetic sequence of samples is taken along each dimension.

Parametric curves

A parametric curve in $\mathbb{R}^2$ is defined by an ordered list of two parametric equations, which both accept one parameter—on a common domain. For example, a circle is commonly defined by the parametric equation

$$f(u) = \begin{bmatrix} \cos(u) \\ \sin(u) \end{bmatrix}, \ u \in [0, \tau].$$

If we traverse this parametric equation in even amounts—an arithmetic sequence along the domain, we will obtain a regular polygon which is composed of line segments. The number of line segments comprising the regular polygon—which approximate the circle—is one less than the number of samples. See Figure 2.8 for the visual analogue of parametric curves with different sample counts.

More generally, we have a formula of the form

$$f(u) = \begin{bmatrix} g(u) \\ h(u) \end{bmatrix}, \ u \in D, \tag{2.1}$$

where D is a continuous finite subset of $\mathbb{R}$.

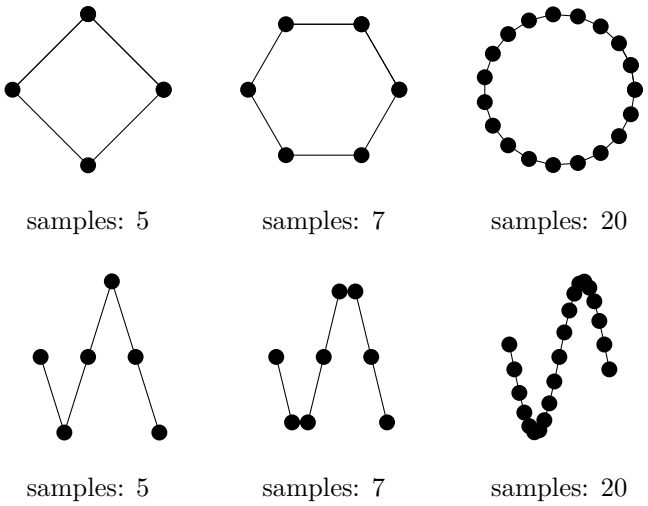


Figure 2.8: Parametric curves in $\mathbb{R}^2$ with different sample counts.

Parametric surfaces

A parametric surface in $\mathbb{R}^2$ is defined by an ordered list of two parameric equations, which both accept two parameters—on a common domain. For example, a disk is commonly defined by the parametric equation

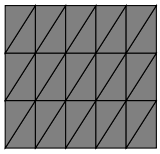
$$f(u,v) = \begin{bmatrix} v\cos(u) \\ v\sin(u) \end{bmatrix},\ u \in [0,\tau],\ v \in [0,1].$$

If we traverse this parametric equation in even amounts—an arithmetic sequence along each dimension of the domain, we will obtain a filled regular polygon composed of triangles. The number of triangles comprising the filled regular polygon—which approximates the disk—is one less than the number of u samples—u_s—multiplied by one less than the number of v samples—v_s, all multiplied by two—because there are two triangles for every small rectangle on the domain. Of course, in the case of triangulation, there may be degenerate triangles, so the actual number of triangles is the theoretical number, minus the number of degenerate triangles. See Figure 2.12 for the visual analogue of parametric surfaces with different sample counts.

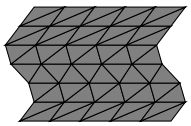
More generally, we have a formula of the form

$$f(u,v) = \begin{bmatrix} g(u,v) \\ h(u,v) \end{bmatrix},\ u \in D_1,\ v \in D_2 \tag{2.2}$$

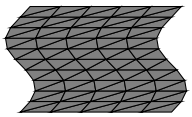
where D_1, D_2 are continuous finite subsets of $\mathbb{R}$.



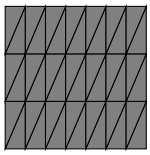
Samples: 5:3
$(u + \sin(v), v)$
$u \in [-4, 4],\ v \in [0, \tau]$



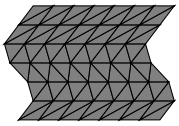
Samples: 5:6
$(u + \sin(v), v)$
$u \in [-4, 4],\ v \in [0, \tau]$



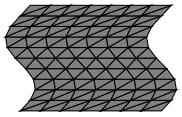
Samples: 5:10
$(u + \sin(v), v)$
$u \in [-4, 4],\ v \in [0, \tau]$



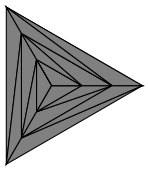
Samples: 7:3
$(u + \sin(v), v)$
$u \in [-4, 4],\ v \in [0, \tau]$



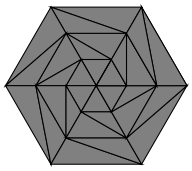
Samples: 7:6
$(u + \sin(v), v)$
$u \in [-4, 4],\ v \in [0, \tau]$



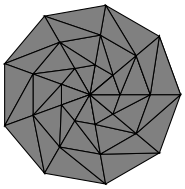
Samples: 7:10
$(u + \sin(v), v)$
$u \in [-4, 4],\ v \in [0, \tau]$



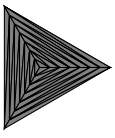
Samples: 3:4
$(u\cos(v), u\sin(v))$
$u \in [0, 4],\ v \in [0, \tau]$



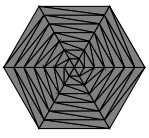
Samples: 3:7
$(u\cos(v), u\sin(v))$
$u \in [0, 4],\ v \in [0, \tau]$



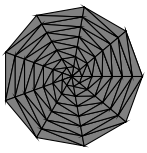
Samples: 3:10
$(u\cos(v), u\sin(v))$
$u \in [0, 4],\ v \in [0, \tau]$



Samples: 7:4
$(u\cos(v), u\sin(v))$
$u \in [0, 4],\ v \in [0, \tau]$



Samples: 7:7
$(u\cos(v), u\sin(v))$
$u \in [0, 4],\ v \in [0, \tau]$



Samples: 7:10
$(u\cos(v), u\sin(v))$
$u \in [0, 4],\ v \in [0, \tau]$

Figure 2.9: Parametric surfaces in $\mathbb{R}^2$ with different sample counts.

Parametric points

A parametric point in $\mathbb{R}^2$ is defined by an ordered list of two parameric equations, which both accept zero parameters; they just output constants. For example, a point centered at $(1, 2)$ is defined by the parametric equation

$$f = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

The general parameterization of a point is

$$f = \begin{bmatrix} g \\ h \end{bmatrix}, \tag{2.3}$$

where g, h have no dependencies.

2.2.3 Linear transformations in $\mathbb{R}^2$

In general, a linear transformation is a change of linear basis vectors. Nondegenerate transformations have the property that they transform simplices into simplices.

Linear transformations on simplices

Suppose we have a tile—for instance, a triangle—which is represented by some matrix

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix},$$

and we wanted to transform it by a 2×2 linear transformation matrix

$$\begin{bmatrix} g & h \\ i & j \end{bmatrix}.$$

This would yield the product

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \begin{bmatrix} g & h \\ i & j \end{bmatrix} = \begin{bmatrix} ag + bi & ah + bj \\ cg + di & ch + dj \\ eg + fi & eh + fj \end{bmatrix}.$$

This product retains the property of being a triangle in the plane, unless the transformation has determinant zero, which would be a projection onto a lower-dimensional subspace.

Let's analyze the effect of a linear transformation on a point. The point is represented by a 2×1 matrix

$$\begin{bmatrix} a \\ b \end{bmatrix},$$

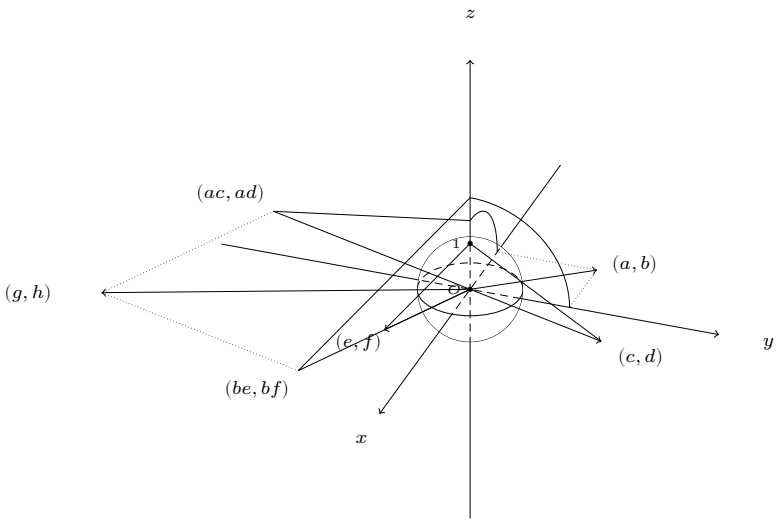


Figure 2.10: A general linear transformation of a point in $\mathbb{R}^2$.

and the linear transformation is represented by a 2×2 matrix

$$\begin{bmatrix} g & h \\ i & j \end{bmatrix}.$$

The product of these two matrices is

$$\begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} g & h \\ i & j \end{bmatrix} = \begin{bmatrix} ag + bi \\ ah + bj \end{bmatrix}.$$

See Figure 2.10 for the visual analogue of a linear transformation of a point in $\mathbb{R}^2$.

Elementary transformations

The elementary transformations arise from the elementary row operations. All linear transformations can be constructed from compositions of these elementary transformations.

The identity transformation is represented by the matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

There are three classes of elementary transformations:

1. row scaling,
2. row swapping, and

3. addition of a row's scalar multiple to another row.

This yields three types of elementary matrices:

$$\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & m \\ 0 & 1 \end{bmatrix}.$$

These correspond to scaling along the x-axis, reflecting in the x-dimension, and shearing the x-axis, respectively.

Transformations as a change of basis A linear transformation is basically just a swap of basis vectors. We can use vector-valued functions to define new bases. For example, the function

$$f(u) = \begin{bmatrix} \cos(u) & \cos(u + \tau/4) \\ \sin(u) & \sin(u + \tau/4) \end{bmatrix}$$

defines a new basis for every value of u, based on a parameterization—in this case, that of a circle. If we were to animate the effect of transforming a plane by this changing basis, we would see the plane rotating. We could do this with any parametric curve which defines two linearly independent vectors for every value of its parameter.

2.2.4 Affine transformations in $\mathbb{R}^2$

Just like in $\mathbb{R}^1$, affine transformations in $\mathbb{R}^2$ are enabled by orthogonally projecting our space—normally by one unit—into a new dimension. By doing this, we can shear the extra dimension to move our origin. Finally, to obtain the result of the transformation, we inverse orthogonally project our space back onto the original region it occupied. If we take the vector (g, h) from Figure 2.10, and we add a translation vector (t_x, t_y) to it, we will have the effect of an affine transformation. For example, the affine transformation

$$\begin{bmatrix} c & e & t_x \\ d & f & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = \begin{bmatrix} ca + eb + t_x \\ da + fb + t_y \\ 1 \end{bmatrix}$$

is illustrated in Figure 2.11. Just like in $\mathbb{R}^1$, the affine transformation is a linear transformation in an extra dimension in combination with a projection. We illustrated linear transformations in $\mathbb{R}^1$ and $\mathbb{R}^2$ by using similar triangles in an extra dimension, as in Figure 2.10. To illustrate a linear transformation in $\mathbb{R}^3$, we would need a fourth dimension, which is difficult to visualize. However, The intuition we have built up for linear transformations in $\mathbb{R}^1$ and $\mathbb{R}^2$ is the same intuition we would use for linear transformations in $\mathbb{R}^3$ and higher dimensions.

An affine transformation in $\mathbb{R}^2$ can be represented by 3×3 matrices of the form

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix},$$

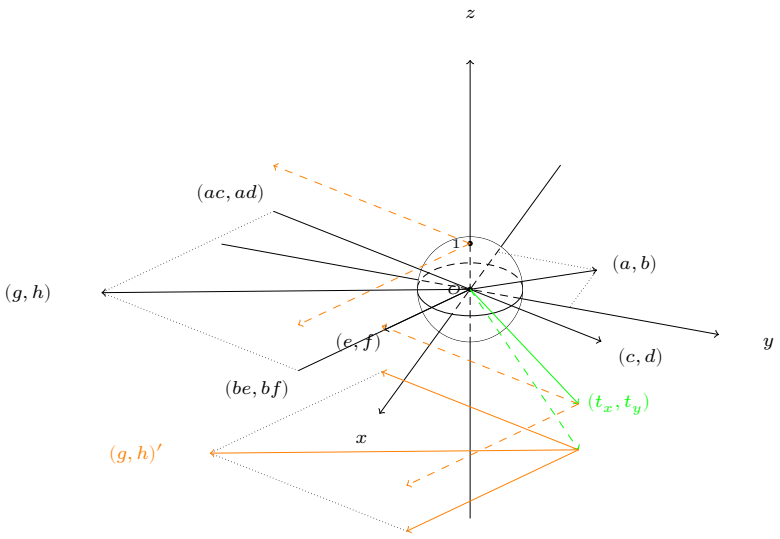


Figure 2.11: The effect of an affine transformation on a point in $\mathbb{R}^2$.

where the 2×2 submatrix in the upper-left corner is a linear transformation matrix, and t_x, t_y are the translation amounts along the x and y axes, respectively.

To apply this affine transformation matrix to a point, line segment, or triangle, we augment the matrix of the simplex with an extra row of ones. For example, the affine transformation of a triangle can be expressed as

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e & f & g \\ h & i & j \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} ae + ch + t_x & af + ci + t_x & ag + cj + t_x \\ be + dh + t_y & bf + di + t_y & bg + dj + t_y \\ 1 & 1 & 1 \end{bmatrix}.$$

2.2.5 Projective transformations in $\mathbb{R}^2$

Projective transformations in $\mathbb{R}^2$ are enabled by both altering the values the bottom row of the affine transformation matrix, and by modifying how we multiply matrices—by adding an additional division step after the multiplication step, as we did in $\mathbb{R}^1$.

A projective transformation in $\mathbb{R}^2$ can be represented by 3×3 matrices of the form

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix},$$

where at least one of g, h is not zero. When we multiply this matrix with the

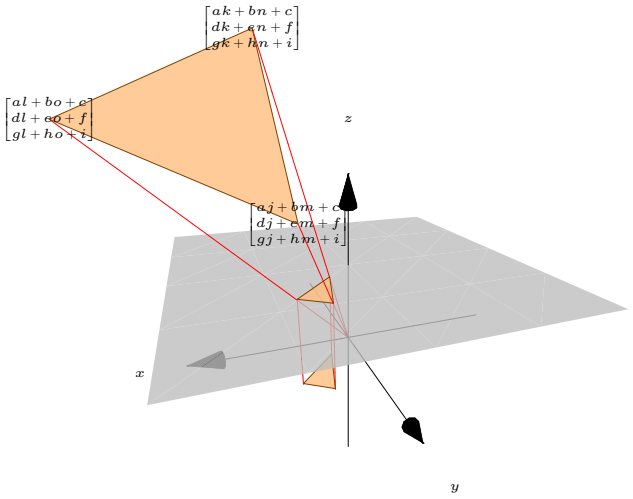


Figure 2.12: The effect of a projective transformation on a triangle in $\mathbb{R}^2$.

augmented matrix of a triangle, we obtain

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} j & k & l \\ m & n & o \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} aj+bm+c & ak+bn+c & al+bo+c \\ dj+em+f & dk+en+f & dl+eo+f \\ gj+hm+i & gk+hn+i & gl+ho+i \end{bmatrix}.$$

However, to complete the projective transformation, we must divide each element in the first and second rows by the corresponding element in the last row. The matrix then becomes

$$\cdots = \begin{bmatrix} \frac{aj+bm+c}{gj+hm+i} & \frac{ak+bn+c}{gk+hn+i} & \frac{al+bo+c}{gl+ho+i} \\ \frac{dj+em+f}{gj+hm+i} & \frac{dk+en+f}{gk+hn+i} & \frac{dl+eo+f}{gl+ho+i} \\ 1 & 1 & 1 \end{bmatrix}.$$

Figure 2.12 illustrates the effect of a projective transformation on a triangle in $\mathbb{R}^2$, analogously to how Figure 2.6 illustrated the effect of a projective transformation on a line segment in $\mathbb{R}^1$.

Exercises

2.3 Affine algebra in $\mathbb{R}^3$

There are four classes of parametric objects which may exist in $\mathbb{R}^3$: those which are tessellated by tetrahedra, those which are tessellated by triangles, those which are tessellated by line segments, and those which are tessellated by individual points. In this chapter, we will not explore tetrahedra.

The algebra in $\mathbb{R}^3$ is exactly the same as the algebra in $\mathbb{R}^2$. However, in $\mathbb{R}^3$, it is not so simple to illustrate parametric objects, because of the problem of occlusion. That will be the focus of this section.

Occlusion is composed of a few different problems, and we will explore one of them in depth, and outline the others briefly.

2.3.1 The problem of occlusion

Up until now we have exposed you to various three-dimensional illustrations. A lot of affine algebra goes into creating these illustrations, and we will briefly explore the overall structure of how these illustrations are created in this section. Afterward, we will focus on one piece of the overall structure with pedagogical value: an occlusion relation on a restricted, but fundamental set of 0–2-dimensional simplices in $\mathbb{R}^3$. The idea is that we can take any set of 0–2-dimensional simplices in $\mathbb{R}^3$, and perform a **partitioning operation** on them which produces a set of simplices which are of the aforementioned restricted type.

When we illustrate sets of parametric objects in $\mathbb{R}^2$, it is easy to look at the entire set at once. However, when we illustrate sets of parametric objects in $\mathbb{R}^3$, we have the problem of occlusion. Occlusion is a phenomenon which concerns the order in which simplices are drawn. For instance, if we draw the simplices in the order imposed by the traversal of their parametric equations, we will very often have diagrams which are obviously drawn incorrectly after a linear transformation, because some simplices which should be hidden behind other simplices are drawn on top of them. We literally need to reorder them before drawing.

In this subsection, we will present a method of ordering a restricted set 0–2-dimensional simplices in $\mathbb{R}^3$ which is pedagogical to implement because it relies entirely on affine algebra, and because it is of practical value for deterministic mathematics illustrations.

The restricted set of simplices

We have described a restriction on the set of 0–2-dimensional simplices in $\mathbb{R}^3$ which enables us to order the set with respect to occlusion. There is some nuanced terminology to describe this restriction. It is easiest to first describe the restriction incorrectly, but with intuitive appeal, and then to refine the description into a correct and precise one. The incorrect description is as follows: *A set of 0–2-dimensional simplices in $\mathbb{R}^3$ is occlusion-orderable if no simplices are intersecting one another, and if no simplices form a cycle of occlusion.* This makes sense intuitively because simplices passing through one another would be unorderable, and cycles of occlusion would be unorderable.

Refining the description

However, we used the word intersecting wrong. Formally speaking, two simplices intersect if they share at least one point in common. This is too strict of a

definition for our purposes, because two simplices which only share a vertex or an edge can still be ordered with respect to occlusion. The correct definition is as follows: A set of 0–2-dimensional simplices in $\mathbb{R}^3$ is occlusion-orderable if no simplices are capable of partitioning one another, and if there are no cycles of occlusion.

What is meant by cycle of occlusion

Occlusion ordering in $\mathbb{R}^3$ is based on a comparison between simplices of the restricted set. Given two simplices A and B, we say that A occludes B if there exists a ray off the viewing plane which intersects—in the set theoretic sense—both A and B, and the intersection point with A is closer to the viewing plane than the intersection point with B. If the intersection point with B is closer to the viewing plane than the intersection point with A, then we say that B occludes A. If neither of these conditions are met, then we cannot determine an occlusion relation between A and B until the entire set has been sorted. In fact, a consequence of this is that there will often be a multitude of valid occlusion orderings for a given set of simplices.

With this definition of occlusion, a cycle of occlusion is a set of simplices $\{S_1, S_2, \ldots, S_n\}$ such that S_1 occludes S_2, S_2 occludes S_3, ..., and S_n occludes S_1. Such a cycle makes it impossible to order the simplices with respect to occlusion.

We will not delve deep into resolving cyclic overlap in this chapter, but we will explain the basic idea. The idea is to partition one of the simplices in the cycle into two or more smaller simplices which do not participate in the cycle. This partitioning is done by finding a plane which intersects the simplex, and splitting the simplex along this plane.

2.3.2 Ordering a restricted set

0–2-dimensional restricted simplices in $\mathbb{R}^3$ have six relationships with respect to occlusion:

1. point occludes point,
2. point occludes line segment,
3. point occludes triangle,
4. line segment occludes line segment,
5. line segment occludes triangle, and
6. triangle occludes triangle.

Each of these relationships can be reasoned about with affine algebra. We will explore each of these relationships in technical detail in the following subsubsections.

Point versus point

One point occludes another point if they are non-coincident and if there is an orthogonal ray off the viewing plane which intersects both points, and the intersection point with the occluding point is closer to the viewing plane than the intersection point with the occluded point. If we cannot deduce a relationship, the result is inconclusive.

Point versus line segment

We first turn the line segment into its affine basis representation. Then, we find the orthogonal projection of the vector connecting the origin of the line segment basis to the point onto the line. If the xy-component of this projection is not the same as the xy-component of the original point, then the result is inconclusive. Otherwise, we check the z-component of the projection against the z-component of the point to determine which occludes which. Of course, if the point is coincident with the line segment, the result is inconclusive.

Point versus triangle

We first project the point and the triangle orthogonally onto the viewing plane. We then use barycentric coordinates to determine if the projected point is inside the projected triangle. If it is not, the result is inconclusive. Otherwise, we find the vertical projection of the point onto the plane of the triangle. We then order the simplices by comparing the z-component of the original point with its projection onto the triangle. Of course, if they are coincident, the result is inconclusive.

Line segment versus line segment

This case is interesting. We first project both line segments orthogonally onto the viewing plane. If the projections do not intersect, the result is inconclusive. Otherwise, we find the intersection point of the projections, and we find the vertical projections of this point onto both line segments. We then compare the z-components of these projections to determine which line segment occludes which. Of course, if the projections are coincident, the result is inconclusive.

This is a bit of a higher level overview though, and it omits some practical details. For instance, to determine if the line segments' projections intersect, we use Gauss-Jordan elimination to solve the system of equations defined by the two lines spanned by their affine basis representations. If there is a unique solution, then we check if this solution lies within both line segments. This means that both elements are between 0 and 1. If so, then the projections intersect, and we can proceed with the rest of the algorithm. If not, the result is inconclusive.

Line segment versus triangle

This case really illustrated how higher-dimensional algebra can be understood in terms of lower-dimensional algebra. This probably sounds like it will require an elaborate algorithm, but the algorithm is actually quite simple if we make use of our previous conclusions.

For every endpoint of the line segment, we perform the point versus triangle occlusion test. And for every edge of the triangle, we perform the line segment versus line segment occlusion test with the line segment. If any of these tests yield a conclusive result, we return that result immediately and skip the rest of the test. Otherwise, the result is inconclusive.

Triangle versus triangle

This case is similar to the line segment versus triangle case. We perform every combination of the point versus triangle and line segment versus line segment occlusion tests. If any of these tests yield a conclusive result, we return that result immediately and skip the rest of the test. Otherwise, the result is inconclusive.

2.3.3 Parametric objects in $\mathbb{R}^3$

Now that we have the tool of occlusion ordering for a restricted set of simplices in $\mathbb{R}^3$, we can use this tool to illustrate parametric objects in $\mathbb{R}^3$.

Since we are only interested in 0–2-dimensional simplices, we will only be able to illustrate parametric points, curves and surfaces. We will not illustrate parametric volumes.

2.3.4 A note on pre-transformation occlusion

Most 3D diagrams are drawn by first applying a linear transformation to the parametric objects, and then drawing the transformed objects. Prior to the linear transformation, the objects are in a state of occlusion which is easy to determine because the objects are axis-aligned.

2.3.5 Parametric points in $\mathbb{R}^3$

A parametric point in $\mathbb{R}^3$ is a tuple of the form

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

where x, y, z are constants with no parameters.

2.3.6 Parametric curves in $\mathbb{R}^3$

A parametric curve in $\mathbb{R}^3$ is a vector valued function of the form

$$c(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix},$$

where $x(t), y(t), z(t)$ are functions of a single parameter t.

2.3.7 Parametric surfaces in $\mathbb{R}^3$

A parametric surface in $\mathbb{R}^3$ is a vector valued function of the form

$$s(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix},$$

where $x(u, v), y(u, v), z(u, v)$ are functions of two parameters u and v.

2.3.8 Parametric volumes in $\mathbb{R}^3$

A parametric volume in $\mathbb{R}^3$ is a vector valued function of the form

$$v(u, v, w) = \begin{bmatrix} x(u, v, w) \\ y(u, v, w) \\ z(u, v, w) \end{bmatrix},$$

where $x(u, v, w), y(u, v, w), z(u, v, w)$ are functions of three parameters u, v, w. We will not illustrate parametric volumes in this chapter, but they are still important to be aware of.

2.3.9 Linear transformations in $\mathbb{R}^3$

In general, just like in $\mathbb{R}^2$, a linear transformation is a change of basis vectors. And, just like in $\mathbb{R}^2$, nondegenerate transformations have the property that they transform simplices into simplices.

Just like in $\mathbb{R}^2$, we have three classes of elementary transformations:

1. row scaling,
2. row swapping, and
3. addition of a row's scalar multiple to another row.

This yields three types of elementary matrices:

$$\begin{bmatrix} k & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

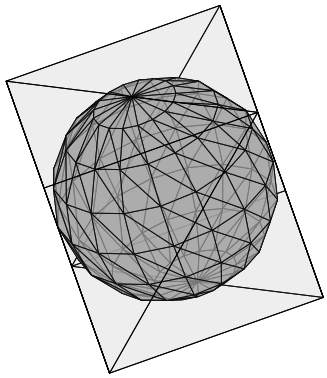


Figure 2.13: The effect of a linear transformation on a sphere and a cube in $\mathbb{R}^3$.

These correspond to scaling along a particular axis, reflecting in a particular dimension, and shearing an axis along one other dimension, respectively.

Again, just like in $\mathbb{R}^2$, a linear transformation is basically just a swap of basis vectors. We can use linearly independent vector-valued functions to define new bases. For example, the function

$$f(\alpha,\beta,\gamma) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

defines a new basis for every value of α, β, γ, based on the parameterization of Euler angles. If we were to animate the effect of transforming a volume by this changing basis, we would see the volume rotating. We could do this with any parametric solid which defines three linearly independent vectors for every value of its parameters.

In Figure 2.10 and Figure 2.4 we used similar triangles in an extra dimension to illustrate the effect of linear transformations on simplices in $\mathbb{R}^2$ and $\mathbb{R}^1$, respectively. Things get out of hand when we try to do this in $\mathbb{R}^3$ though, because cramming more dimensions into the viewing plane—which is linearly independent already—is not very effective. Instead, we will rely on the visual intuition we have built, and we will just imagine the similar triangles in a higher dimension. Of course, if we really wanted to, we could make multiple diagrams with different dimensions collapsed, so we could see each perspective. Figure 2.13 illustrates the effect of a linear transformation on a sphere and a cube in $\mathbb{R}^3$.

2.3.10 Affine transformations in $\mathbb{R}^3$

In $\mathbb{R}^1$ and $\mathbb{R}^2$, we were able to visualize the shear in of the extra dimension which causes the translation. Now, we will rely on the intuition we have built, and we will just make sense of affine transformations in $\mathbb{R}^3$. In Figure 2.14, we use many affine transformations to move spheres around in $\mathbb{R}^3$. We moved them into the shape of a larger sphere.

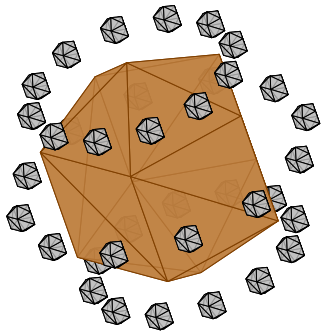


Figure 2.14: The effect of affine transformations on spheres in $\mathbb{R}^3$.

2.3.11 Projective transformations in $\mathbb{R}^3$

Projective transformations in $\mathbb{R}^3$ are enabled by both altering the values the bottom row of the affine transformation matrix, and by modifying how we multiply matrices—by adding an additional division step after the multiplication step, as we did in $\mathbb{R}^1$ and $\mathbb{R}^2$. A projective transformation in $\mathbb{R}^3$ can be represented by 4×4 matrices of the form

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix},$$

where at least one of m, n, o is not zero. When we multiply this matrix with the augmented matrix of a triangle, we obtain

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} q & r & s \\ t & u & v \\ w & x & y \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{aq+bt+cw+d}{mq+nt+ow+p} & \frac{ar+bu+cx+d}{mr+nu+ox+p} & \frac{as+bv+cy+d}{ms+nq+oy+p} \\ \frac{eq+ft+gw+h}{mq+nt+ow+p} & \frac{er+fu+gx+h}{mr+nu+ox+p} & \frac{es+fv+gy+h}{ms+nq+oy+p} \\ \frac{iq+jt+kw+l}{mq+nt+ow+p} & \frac{ir+ju+kx+l}{mr+nu+ox+p} & \frac{is+jv+ky+l}{ms+nq+oy+p} \\ 1 & 1 & 1 \end{bmatrix}$$

after the division step.

We have built the visual intuition for projective transformations in $\mathbb{R}^1$ and $\mathbb{R}^2$, so we will just make sense of projective transformations in $\mathbb{R}^3$ with the intuition we have built. We will just use them on parametric objects now. Figure 2.15 illustrates the effect of a projective transformation on a sphere and a cube in $\mathbb{R}^3$.

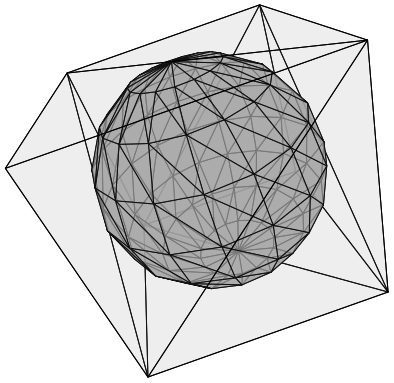


Figure 2.15: The effect of a projective transformation on a triangle in $\mathbb{R}^3$.

Chapter 3

Simplicial occlusion

NOTE: I have stolen this chapter verbatim from a tugboat article I wrote.

Occlusion ordering in our system is performed by first orthogonally projecting the two simplices onto the viewing plane. If a point of overlap is detected, we sort them according to the inverse orthogonal projection of that point back onto each shape.

3.1 Point versus point

If the points are not coincident but their projections coincide, then they are ordered by depth. Otherwise, the test is inconclusive.

3.2 Point versus line segment

This routine determines the occlusion relationship between a point and a line segment. It first expresses the line segment in terms of an affine basis, given by its origin and direction vector. The point is then projected orthogonally onto the line defined by the segment, producing a candidate projection. If the projection of the point onto the viewing plane—its xy-coordinates—is nearly identical to the projection of this candidate, the test proceeds; otherwise, the point and line segment are considered not to occlude each other. Next, the algorithm checks whether the projection lies within the bounds of the segment itself by comparing vector signs and computing a normalized coefficient. If the projection falls within the unit interval of the segment, the algorithm reduces the problem to comparing the depth of the point and its projection on the line. If these conditions fail, the test is inconclusive.

3.3 Point versus triangle

This routine compares the occlusion relationship between a point and a triangle. The point and the triangle are first projected orthogonally onto the viewing

plane, where a cross-product test is applied to check whether the projected point lies inside the projected triangle. If this test fails, the point and triangle are considered not to occlude one another. If the point lies inside the projection, the algorithm then projects the point vertically onto the plane of the triangle. Using an affine basis for the triangle, the coordinates of the point with respect to the triangle are solved via Gauss–Jordan elimination. If the solution lies within the unit square—ensuring the projection is inside the triangle—the algorithm reduces the problem to a point-point occlusion comparison between the original point and its projection on the triangle. If any step fails to satisfy these conditions, the test is inconclusive.

3.4 Line segment versus line segment

This routine determines the occlusion relationship between two line segments. First, both segments are projected onto the viewing plane, and their direction vectors are computed. If the direction vectors are not parallel, the algorithm solves for parameters t and s in the affine equations of the two lines using Gauss–Jordan elimination. If both parameters lie within the unit interval, the intersection point of the two line segments is found, and the occlusion is reduced to a point-point comparison of the inverse orthogonal projection of that point onto both original line segments.

If the direction vectors are parallel, the algorithm instead falls back to point-line segment tests: each endpoint of one segment is compared against the other segment using the point-line segment occlusion procedure. If any endpoint is found to occlude, the segments are ordered accordingly. If no consistent ordering can be determined, the test is inconclusive.

3.5 Line segment versus triangle

This routine compares the occlusion relationship between a line segment and a triangle. The algorithm begins by testing each endpoint of the segment against the triangle using the point-triangle occlusion procedure. It then tests the segment itself against each of the triangle's edges using the line-segment occlusion procedure. If any of these comparisons establishes a definite ordering, that result is returned. If no consistent conclusion can be drawn from the endpoint and edge tests, the test is inconclusive.

3.6 Triangle versus triangle

This routine compares the occlusion relationship between two triangles. The algorithm first tests each edge of the first triangle against the second using the line-segment–triangle occlusion procedure. If any edge establishes a definite ordering, that result is immediately returned. If these edge tests are inconclusive, the algorithm proceeds by testing the vertices of the first triangle against the

second, and the vertices of the second triangle against the first, using the point-triangle occlusion procedure. If any of these vertex tests determines a clear ordering, that result is returned. If neither the edge nor the vertex tests establish a consistent relationship, the routine concludes that the occlusion status is inconclusive.

Chapter 4

Simplicial partitioning

Tiles which are capable of partitioning one another are eliminated through minimal partitioning of simplices. To identify the cases, a bottom-up approach is taken: we first determine the meaningful ways to partition lower-dimensional simplices, and then extend this to higher dimensions. In the scope of this work, two tiles intersect if and only if one is capable of partitioning the other. Intersection of two tiles is not taken in the set theoretic sense, while intersections of two 0–2-dimensional affine subspaces are taken in the set theoretic sense.

There is no meaningful way to partition a point by another point, so that case is omitted. A point can divide a line segment into two if they intersect, so this case is retained. A point and a triangle have no meaningful partitioning with respect to each other.

A line segment can partition another line segment, and a line segment can also be partitioned by a triangle. Finally, a triangle can be partitioned by another triangle. For reasons of minimality, only one of the two intersecting simplices is partitioned in each case.

4.1 Line segment by point

We first check whether a point lies on a line segment, and if it does, we partition the line segment at that point. To perform this test, the line segment is expressed as a one-dimensional affine basis by replacing the second endpoint with its difference from the first. We then compute the vector from the affine origin to the point.

Next, we take the orthogonal projection of this vector onto the segment's direction vector. If the sum of this projection with the affine origin is nearly coincident with the point, we proceed with testing; otherwise, the point is not on the segment. To confirm, we apply Gauss–Jordan elimination to express the point in terms of the line's affine basis. If the resulting coordinate lies within the unit interval, the point is indeed on the segment, and the segment is partitioned.

4.2 Line segment by line segment

We first check whether two line segments intersect, and if they do, we partition one of them at the intersection point. To detect an intersection, each segment is expressed as a one-dimensional affine basis. The lines spanned by these bases are then intersected using Gauss–Jordan elimination. If the resulting parameters for both segments lie within their respective unit intervals, the segments intersect, and partitioning is performed.

4.3 Line segment by triangle

If an intersection between the line segment and the triangle can be detected, we partition the line segment at that point. Both simplices are expressed as affine bases in their respective dimensions, and the intersection is obtained by solving the resulting linear system via Gauss–Jordan elimination. If the line segment's coefficient lies within the unit interval, the candidate intersection is retained; otherwise, it is discarded.

When the line segment does intersect the plane of the triangle, we determine whether the intersection point lies inside the triangle using a common cross-product method. In this approach, the triangle is tessellated into one-dimensional affine bases—its edges, and each vertex is connected to the point being tested, forming another one-dimensional affine basis. For every pair of affine bases emanating from a common vertex, we compute their rotationally ordered cross product. If all such cross products point in the same direction, the point lies inside the triangle. If the point lies outside, at least one rotationally ordered cross product will traverse its angle in the opposite orientation, producing an anticommutative result.

4.4 Triangle by triangle

The partitioning of a triangle by another triangle builds upon the previous cases. The first step is to detect intersections between their edges. If two intersections are found—the expected outcome, though safeguards are included for degenerate cases—the cutting triangle partitions the other along the line defined by these two points. This produces a triangle and a quadrilateral, the latter of which is subdivided into two triangles. The intersections themselves are computed by treating each edge as a line segment and determining its intersection with the opposing triangle.

Part II

Manual

Chapter 5

Command reference

5.1 ltdtappendlabel

This command appends a label. Its keys are

- v: a homogeneous 3D vector (e.g., `Vector:new{1,2,3,1}`).
- text: a LaTeX label (e.g., `\LaTeX`).
- transformation: A homogeneous 3D transformation matrix.
- filter: A boolean on v.

5.2 ltdtappendlight

This command appends a light direction. I recommend using only one light direction, unless you have a good reason not to.

- v: a homogeneous 3D vector

5.3 ltdtappendcurve

This command appends a curve.

Figure 5.1: ltdtappendlabel

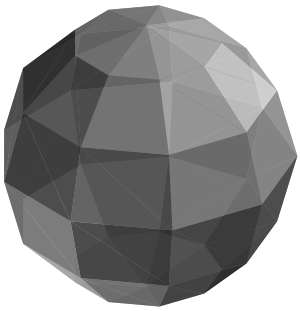


Figure 5.2: ltdtappendlight

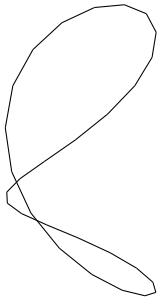


Figure 5.3: ltdtappendcurve

- uparams: A 3-vector of start, stop and step in u
- v: a homogeneous 3D vector
- transformation: A homogeneous 3D transformation matrix.
- draw options: a string of Ti*k*Z options for a path.
- arrow tip: a string of Ti*k*Z options for a path.
- arrow scale: a number to change the arrow's size
- filter: A boolean on v.

5.4 ltdtappendsurface

This command appends a surface.

- uparams: A 3-vector of start, stop and step in u
- vparams: A 3-vector of start, stop and step in v
- v: a homogeneous 3D vector

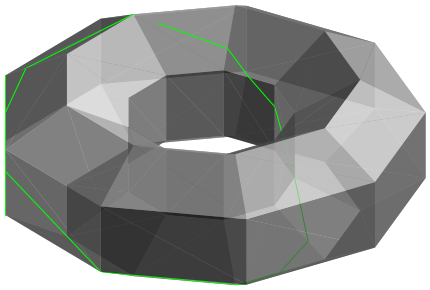


Figure 5.4: ltdtappendsurface

Figure 5.5: ltdtappendtriangle

- transformation: A homogeneous 3D transformation matrix.
- fill options: a string of Ti*k*Z options for a path.
- filter: A boolean on v.
- curve: If you know what you're doing, you can embed a table of UV line segments within a surface's simplices. See the example below.

5.5 ltdtappendtriangle

This command appends a triangle.

- m: A 3-vector of start, stop and step in u
- transformation: A homogeneous 3D transformation matrix.
- fill options: a string of Ti*k*Z options for a path.
- filter: A boolean on m.

Figure 5.6: ltdtappendsurface

5.6 ltdtappendsolid

This command appends the surface boundaries of a solid mapped from a cube.

- uparams: A 3-vector of start, stop and step in u
- vparams: A 3-vector of start, stop and step in v
- wparams: A 3-vector of start, stop and step in w
- v: a homogeneous 3D vector
- transformation: A homogeneous 3D transformation matrix.
- fill options: a string of Ti*k*Z options for a path.
- filter: A boolean on v.

5.7 ltdtdisplaysimplices

This command resolves the appended simplices, and displays the resulting output.

5.8 ltdtsetobject

This command sets a lua object

- name: the lua name of the object.
- object: the lua object (e.g., a function, or a matrix).