Modeling Affine and Projective Transformations in 3-Dimensions by Linear Transformations in 4-Dimensions

University of Waterloo Technical Report CS-2018-05

Juan Du and Ron Goldman and Stephen Mann

May 28, 2018

Abstract

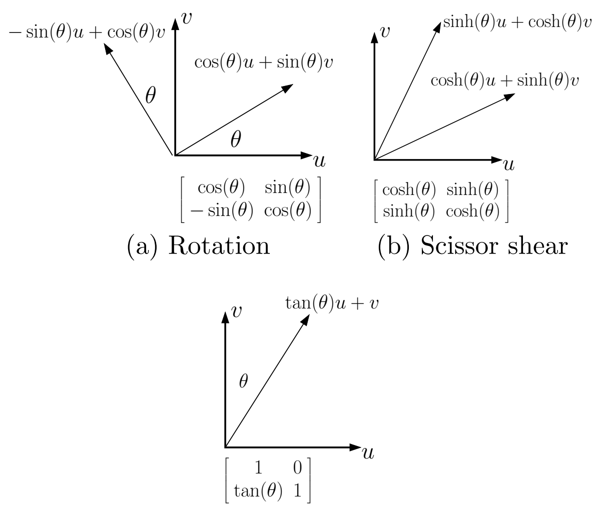
We provide a tutorial on how to use shears and rotations in 4-dimensions to model translations and perspective projections in 3-dimensions. We also explain how to uniformly scale points about the origin and mirror points in the origin in 3-dimensions using non-uniform scaling and reflection in 4-dimensions.

1 Introduction

Affine and projective transformations in 3-dimensions are often represented in computer graphics by 4×4 matrices [4]. But in linear algebra 4×4 matrices are typically used to represent linear transformations in 4-dimensions. Thus in computer graphics affine and projective transformations in 3-dimensions are usually represented by linear transformations in 4-dimensions.

The non-singular linear transformations in 4-dimensions are composites of four basics types of transformations: rotations, reflections, shears, and non-uniform scalings. The fundamental affine and projective transformations of interest in computer graphics are: translations, rotations, reflections, shears, uniform and non-uniform scalings and orthogonal and perspective projections. Most of these transformations are linear transformations in 3-dimensions and therefore could be represented by 3×3 matrices. Notice, however, that translations are affine, but not linear, transformations in 3-dimensions, and that perspective projections are projective transformations, but neither linear nor affine transformations in 3-dimensions [4]. Thus translations and perspective projections motivate the use of 4×4 matrices in computer graphics.

Figure 1 show how rotations, scissor shears [12], and classical shears are defined in the uv-plane. Notice that for rotation and scissor shear the effect of these transformations on the u-axis and v-axis is symmetric. But classical shear treats the u-axis and the v-axis very differently: the u-axis is left unchanged, while the v-axis picks up a component in the u-direction. Thus, a classical shear in the uv-plane behaves very differently from


$-\sin(\theta)u + \cos(\theta)v$
v
$\cos(\theta)u + \sin(\theta)v$
θ
θ
u
$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$
v
$\sinh(\theta)u + \cosh(\theta)v$
$\cosh(\theta)u + \sinh(\theta)v$
u
$\begin{bmatrix} \cosh(\theta) & \sinh(\theta) \\ \sinh(\theta) & \cosh(\theta) \end{bmatrix}$
v
$\tan(\theta)u + v$
θ
u
$\begin{bmatrix} 1 & 0 \\ \tan(\theta) & 1 \end{bmatrix}$

(a) Rotation (b) Scissor shear

(c) Classical shear

Figure 1: Three linear transformation and the matrices that represent these transformations in the uv-plane: (a) rotation, (b) scissor shear, and (c) classical shear. Note that rotation and scissors shear affect both axes by the same amount, while for classical shear only the v-axis is affect and the u-axis is unchanged.

a classical shear in the vu-plane. We shall take advantage of this asymmetry to use classical shears in planes in 4-dimensions to analyze matrices both for translations (Section 3) and for projections (Section 5) in 3-dimensions.

If the uv-plane lies in xyz-space, then rotations and shears represent linear transformations in R^3, where the vectors in the direction orthogonal to the uv-plane remain unchanged. But what if one of the uv-directions lies along the fourth axis, the w-axis, in R^4? Then we have a rotation or shear in a plane in R^4. Rotations and shears that affect vectors only in a single plane in R^4 and that do not affect vectors orthogonal to this plane are called *simple rotations* and *simple shears*. One of the goals of this paper is to show how to interpret these simple rotations and simple shears in R^4 as particular affine or projective transformations in R^3.

This paper investigates the geometric relationship between certain basic nonsingular linear transformations in 4-dimensions and the transformations most common in 3-dimensional computer graphics. The purpose of this paper is not a new or more efficient computational method, nor are the results intended for particular applications. Instead, we our goal is to provide a new geometric understanding of well known transformations in computer graphics. Thus the spirit of this technical report is more in the nature of a tutorial rather than a research monograph.

Our main contributions are to show that

1. Translations in 3-dimensions can be modeled by classical shears in 4-dimensions. These classical shears turn out to be the standard form of translations used in computer graphics. Thus, the standard computer graphics matrix for translation represents a shear in 4-dimensions.

2. Perspective projections in 3-dimensions can be modeled by (*i*) classical shears, (*ii*) scissor shears,[1] or (*iii*) rotations in 4-dimensions.

3. Pseudo-perspective projections in 3-dimensions can also be modeled by (*i*) classical shears, (*ii*) scissor shears, or (*iii*) rotations in 4-dimensions. The method using classical shear turns out to be the standard form of pseudo-perspective used in computer graphics; thus the standard computer graphics matrix for pseudo-perspective is a classical shear in 4-dimensions. The approaches to pseudo-perspective using either scissor shears or rotations are new and are presented here for the first time. The rotation form of pseudo-perspective is particularly noteworthy because it permits one to perform pseudo-perspective using sandwiching with unit quaternions.

We shall also show that

4. Orthogonal projections in 3-dimensions can be modeled by 90° rotations in a plane in 4-dimensions.

5. Uniformly scaling points about the origin in 3-dimensions can be modeled by scaling vectors non-uniformly along the w-axis in 4-dimensions.

6. Reflecting points about the origin in 3-dimensions can be modeled by mirroring vectors in the hyper-plane $w = 0$ in 4-dimensions.

Some of these techniques are inspired, in part, by comparable results concerning versors in the Clifford Algebra $R(4,4)$. One of the goals of this exposition is to reduce these results to matrix algebra for those uninitiated in Clifford algebra. The reader interested in using $R(4,4)$ in computer graphics is referred to [3, 8, 9].

[1]Scissor shears are also called *Lorentz boosts*.

This paper is organized in the following fashion. In Section 2 we introduce our conventions and fix our notation. Sections 3, 4, and 5 contain our main results.

In Section 3 we turn our attention to translation. We show how to apply classical shears in planes containing the w-axis in 4-dimensions to translate points along vectors in 3-dimensions.

In Section 4 we explore the effect on points in 3-dimensions of scaling vectors in 4-dimensions along the w-axis. We show that (*i*) scaling vectors in 4-dimensions along the w-axis by a scalar factor $s > 0$ scales the distance of points from the origin in 3-dimensions by the scale factor $1/s$ (Section 4.1); (*ii*) scaling vectors in 4-dimensions along the w-axis by the scalar factor $s = -1$—that is, reflecting vectors in 4-dimensions in the hyper-plane $w = 0$—reflects points about the origin in 3-dimensions (Section 4.2); and (*iii*) scaling vectors in 4-dimensions along the w-axis by a scalar factor $s = 0$—that is, projecting vectors from 4-dimensions to 3-dimensions—maps points P in 3-dimensions to vectors from the origin to P in 3-dimensions (Section 4.3). In Section 4.4 we show how to exploit 90° rotations in 4-dimensions together with projecting vectors from 4-dimensions to 3-dimensions to compute orthogonal projections in 3-dimensions.

Section 5 is devoted to perspective projection. In Section 5.1 we show how to project from R^3 to R^2 by classical shears, scissors shears, or rotations in 4-dimensions. In Section 5.2, we extend our results on perspective projections to pseudo-perspective projections, the kind of perspective projections found most often in computer graphics. Finally in Section 5.3 we observe that since rotations in 4-dimensions can be used to model perspective and pseudo-perspective in 3-dimensions, we can also use sandwiching with unit quaternions to compute both perspective and pseudo-perspective.

We close in Section 6 with a brief summary of our main results.

2 Points and Vectors in R^3 and R^4

We shall use four coordinates to represent points and vectors in 3-dimensions: A nonzero fourth coordinate represents a mass-point or weighted-point—that is, a scalar mass at a fixed location in R^3—and a zero fourth coordinate represents a vector in R^3. Alternatively, we shall also think of these four coordinates as representing vectors in 4-dimensions.

2.1 Points and Vectors in R^3

Although the visual world is 3-dimensional, contemporary computer graphics typically uses four coordinates to represent points and vectors and 4×4 matrices to represent affine and projective transformations.

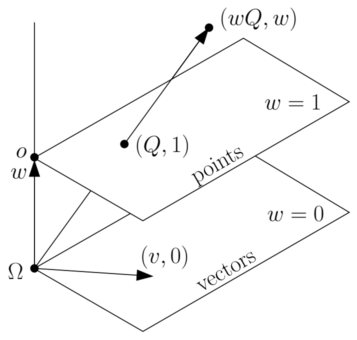


Figure 2: Points and vectors in 3-dimensions embedded in 4-dimensions

If $w \neq 0$, then the four coordinates (wx, wy, wz, w) represent a mass w located at the point with coordinates (x, y, z) in R^3. If P and Q both represent mass points, we write $P \equiv Q$ if $P = wQ$ for a non-zero scalar w because P and Q are located at the same location in R^3, they differ only in their masses. If $w = 0$, then the four coordinates $(x, y, z, 0)$ represent the vector from the origin to the point located at (x, y, z) in R^3.

We shall use the symbols $i = (1, 0, 0, 0)$, $j = (0, 1, 0, 0)$, $k = (0, 0, 1, 0)$ to represent unit vectors along x-, y-, z-axes. We use the symbol $o = (0, 0, 0, 1)$ to denote the origin for the points in 3-dimensions, and we use the symbol $\Omega = (0, 0, 0, 0)$ to denote the origin for the vectors in 4-dimensions. Note that in Figure 2 the arrow $(v, 0)$ corresponds to the vector v in 3-dimensions, the arrow $(Q, 1)$ corresponds to the point Q in 3-dimensions, and the arrow (wQ, w) also represents the point Q together with a mass $w \neq 0$ in 3-dimensions. In particular, if $Q = (q_1, q_2, q_3)$, then

$$\begin{aligned}(wQ, w) &= wq_1 i + wq_2 j + wq_3 k + wo \\ &\equiv q_1 i + q_2 j + q_3 k + o = (Q, 1)\end{aligned}$$

Notice that the map $(wQ, w) \to (Q, 1)$ is just the central projection of (wQ, w) from the origin Ω in 4-dimensions to the plane $w = 1$ (see Figure 2).

The vectors perpendicular to a fixed vector n in R^3 form a plane in R^3. Hence every vector v in R^3 can be written uniquely as $v = \lambda n + n_\perp$ for some constant λ and some vector $n_\perp$ perpendicular to n. Therefore, for every pair of points P, Q there is a constant λ and a vector $n_\perp$ perpendicular to n such that $P - Q = \lambda n + n_\perp$ so $P = Q + \lambda n + n_\perp$. It follows that every point P can be written in the form $P = o + \lambda n + n_\perp$ for some constant λ and some vector $n_\perp$ perpendicular to n.

2.2 Vectors in R^4

Alternatively, we shall also use four coordinates to represent vectors in 4-dimensions. Under this interpretation, i, j, k represent unit vectors along the x-, y-, and z-axes in 4-dimensions, and the origin $o = (0, 0, 0, 1)$ in 3-dimensions represents the unit vector along w-axis in 4-dimensions. The symbol Ω represents the zero vector in 4-dimensions. Let $\mathbf{n} = (n_1, n_2, n_3)$ be a unit vector in 3-dimensions. Then $n = (n_1, n_2, n_3, 0)$ represents $\mathbf{n}$ as a unit vector in 4-dimensions and $o \cdot n = 0$, so $o \perp n$. Thus o is orthogonal to every vector in R^3.

Notice again that o plays a dual role: in 4-dimensions $o = (0, 0, 0, 1)$ represents a unit vector along the w-axis, while in 3-dimensions o represents the point at the origin.

2.3 Notation

In the remainder of this paper, we shall adopt the following notation. We shall use upper case letters P, Q to denote points and lower case letters u, v to denote vectors. The maps $P \hookrightarrow (P, 1)$ and $v \hookrightarrow (v, 0)$ embed points and vectors in 3-dimensions as vectors in 4-dimensions. We shall use P to denote both the point P in R^3 and its embedding $(P, 1)$ in R^4. Similarly, shall use v to denote both the vector v in R^3 and its embedding $(v, 0)$ in R^4. The precise meaning will be clear from the context. When we want to emphasize that v is the embedding in R^4 of a vector in R^3, we shall use bold to represent the vector in R^3. Thus we shall let $\mathbf{v}$ be a vector in R^3 and set $v = (\mathbf{v}, 0)$ to denote the corresponding vector in R^4. Finally, we will use I to denote the 3×3 identity matrix.

3 Translation in 3-Dimensions by Classical Shear in 4-Dimensions

Here we show how to perform translations in 3-dimensions by using classical shears in 4-dimensions. Intuitively the reason a classical shear in 4-dimensions can be used to represent a translation in 3-dimensions is that a classical shear in the no-plane moves o along the direction n and leaves all the vectors in 4-dimensions orthogonal to o (i.e., all the vectors in 3-dimensions) fixed. Thus the effect of a classical shear in the no-plane on points $P = o + v$ in 3-dimensions is simply to translate P along the direction n by translating o and leaving v fixed. Therefore we have the following theorem:

Theorem 3.1 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions, and let* $n = (\mathbf{n}, 0)$. *The* 4×4 *matrix*

$$CShear(n, o, \theta) = \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix}$$

is a classical shear in the no-plane by the angle θ *in the direction* o *in 4-dimensions, and translates points in the direction* $\mathbf{n}$ *by the signed distance* $d = \tan(\theta)$ *in 3-dimensions.*

Proof. For any vector $\mathbf{n}_\perp$ perpendicular to $\mathbf{n}$, let $n_\perp = (\mathbf{n}_\perp, 0)$. Then

$$\begin{aligned} o * CShear(n,o,\theta) &= o * \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix} \\ &= o + \tan(\theta) n \\ n * CShear(n,o,\theta) &= (\mathbf{n},0) * \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix} = n \end{aligned}$$

and

$$n_\perp * CShear(n,o,\theta) = (\mathbf{n}_\perp, 0) * \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix} = n_\perp.$$

Therefore, $CShear(n,o,\theta)$ represents a classical shear in the no-plane by the angle θ in the direction o in 4-dimensions.

For any point Q in R^3, there exists a vector $n_\perp$ perpendicular to n and a constant λ such that $Q = o + \lambda n + n_\perp$. Now by linearity

$$Q * CShear(n,o,\theta) = (o + \lambda n + n_\perp) * \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix}$$

$$\begin{aligned} &= o + \tan(\theta)n + \lambda n + n_\perp \\ &= o + \lambda n + n_\perp + \tan(\theta) n \\ &= Q + \tan(\theta) n. \end{aligned}$$

And for any vector $v = \lambda n + n_\perp$ in R^3

$$\begin{aligned} v * CShear(n,o,\theta) &= (\lambda n + n_\perp) \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix} \\ &= \lambda n + n_\perp = v. \end{aligned}$$

Therefore, $CShear(n,o,\theta)$ translates points in 3-dimensions in the direction $\mathbf{n}$ by the signed distance $d = \tan(\theta)$ and leaves vectors in 3-dimensions unchanged.

$\diamondsuit$

Since in 3-dimensions $CShear(n,o,\theta)$ translates points P by the vector $\mathbf{v} = \tan(\theta)\mathbf{n}$, we often write

$$Trans(\mathbf{v}) = \begin{bmatrix} I & 0 \\ \mathbf{v} & 1 \end{bmatrix}$$

instead of

$$CShear(n,o,\theta) = \begin{bmatrix} I & 0 \\ \tan(\theta)\mathbf{n} & 1 \end{bmatrix}.$$

The basic geometric idea is illustrated in Figure 3: starting with a point in 3-dimensions represented by the vector Q in 4-dimensions, we perform a classical shear in the no-plane in the direction o in 4-dimensions. This transformation translates any point Q to a point $\bar{Q}$ in 3-dimensions.

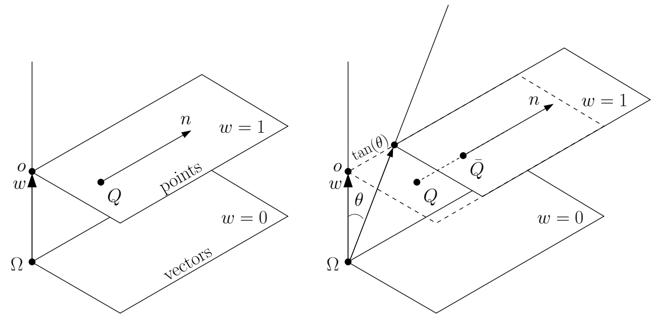


Figure 3: Translation in R^3 by classical shear in R^4

Example 3.2 *Let $n = i$ be the unit vector along the x-axis. Then the 4×4 matrix*

$$T_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t & 0 & 0 & 1 \end{bmatrix}$$

is a classical shear in the xw-plane by the angle $\arctan(t)$ *in the direction w in 4-dimensions, and translates points along the x-axis by the signed distance t in 3-dimensions.*

Similarly the matrices

$$\mathrm{T_y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & t & 0 & 1 \end{bmatrix}, \quad \mathrm{T_z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & t & 1 \end{bmatrix}$$

represent classical shears in the yw-plane and zw-planes in 4-dimensions that translates points along the y- and z-axes by the signed distance t in 3-dimensions.

Note that these translation matrices are the standard translation matrices used in computer graphics; thus, the transformations used in computer graphics for translations in 3-dimensions are classical shears in 4-dimensions.

Some of these ideas, such as shear in one higher dimension representing translation in one lower dimension are well known in the world of kinematics [11].

4 Scaling Along the w-Axis

In this section we explore the effect on points in 3-dimensions of scaling vectors in 4-dimensions along the w-axis. We shall show that scaling vectors in 4-dimensions along the w-axis by a scalar factor $s > 0$ scales the distance of points from the origin in 3-dimensions by the scale factor $1/s$. We shall also show that scaling vectors in 4-dimensions along the w-axis by the scalar factor $s = -1$—that is, reflecting vectors in 4-dimensions in the hyper-plane $w = 0$—reflects points about the origin in 3-dimensions. Finally we shall show that scaling vectors in 4-dimensions along the w-axis by the scalar factor $s = 0$—that is, projecting vectors from 4-dimensions to 3-dimensions—maps points P in 3-dimensions to vectors $P - o$ in 3-dimensions. To simplify our expressions, we shall adopt the following notation:

$$I(s) = \begin{bmatrix} I & 0 \\ 0 & s \end{bmatrix}.$$

4.1 Uniform Scaling in 3-Dimensions by Non-Uniform Scaling in 4-Dimensions

Theorem 4.1 *The* 4×4 *orthogonal matrix*

$$I(s) = \begin{bmatrix} I & 0 \\ 0 & s \end{bmatrix}, \quad s > 0$$

represents non-uniform scaling of vectors by the scale factor s *in the direction* o *in 4-dimensions, and represents uniform scaling of points about the origin* o *by the scale factor* $1/s$ *in 3-dimensions.*

Proof. For any vector $v = ai + bj + ck + do = (a, b, c, d)$ in R^4

$$v*I(s) = (a, b, c, d)*\begin{bmatrix} I & 0 \\ 0 & s \end{bmatrix} = (a, b, c, sd) = ai+bj+ck+sdo,$$

which is non-uniform scaling of vectors v by the scale factor s in the direction o in 4-dimensions.

Similarly, for any point $Q = ai+bj+ck+o = (a, b, c, 1)$ in R^3

$$\begin{aligned} Q * I(s) &= (a, b, c, 1) * \begin{bmatrix} I & 0 \\ 0 & s \end{bmatrix} = (a, b, c, s) \\ &= ai + bj + ck + so \\ &\equiv \frac{a}{s}i + \frac{b}{s}j + \frac{c}{s}k + o, \end{aligned}$$

which is uniform scaling by the scale factor $1/s$ of points from the origin o in 3-dimensions.

◇

The basic geometric idea is illustrated in Figure 4: starting with a point in 3-dimensions represented by the vector Q in 4-dimensions, we scale the vector Q non-uniformly by the factor s in the direction o in 4-dimensions, giving $\bar{Q}$. Then we project $\bar{Q}$ from the origin Ω to the plane $w = 1$ to get Q', which is the point Q scaled uniformly by the scale factor $1/s$ about the origin o in 3-dimensions.

The standard uniform scaling matrix in 3-dimensions is

$$U(1/s) = \begin{bmatrix} \frac{1}{s}I & 0 \\ 0 & 1 \end{bmatrix}.$$

The matrix $U(1/s)$ uniformly scales any point Q to $\frac{1}{s}Q$ and uniformly scales any vector v to $\frac{1}{s}v$. In contrast, the matrix presented in Theorem 4.1 that models uniform scaling in 3-dimensions by non-uniformly scaling in 4-dimensions is

$$I(s) = \begin{bmatrix} I & 0 \\ 0 & s \end{bmatrix}.$$

Notice that the matrix $I(s)$ uniformly scales any point Q to $\frac{1}{s}Q$ but leaves any vector v unchanged. Thus, the matrices $U(1/s)$ and $I(s)$ have the same effect on points but different effects on vectors. The matrix $U(1/s)$ is an affine transformation in 3-dimensions, while the matrix $I(s)$ is a linear transformation in 4-dimensions but not an affine transformation in 3-dimensions.

Example 4.2 *Consider the vector* $v = (2, 5, 4, 3)$ *in 4-dimensions, the point* $P = (2, 6, 4, 1)$ *and the vector* $u = (2, 6, 4, 0)$ *in 3-dimensions. Then the* 4×4 *orthogonal matrix*

$$I(2) = \begin{bmatrix} I & 0 \\ 0 & 2 \end{bmatrix}$$

scales v *non-uniformly by the scale factor* 2 *in the direction* o *in 4-dimensions. In 3-dimensions,* $I(2)$ *scales* P *uniformly about the origin* o *by the scale factor* $\frac{1}{2}$ *and leaves* u *unchanged.*

In 4-dimensions,

$$v * I(2) = (2, 5, 4, 3) * \begin{bmatrix} I & 0 \\ 0 & 2 \end{bmatrix} = (2, 5, 4, 6).$$

In 3-dimensions,

$$P*I(2) = (2, 6, 4, 1)*\begin{bmatrix} I & 0 \\ 0 & 2 \end{bmatrix} = (2, 6, 4, 2) \equiv (1, 3, 2, 1)$$

and

$$u * I(2) = (2, 6, 4, 0) * \begin{bmatrix} I & 0 \\ 0 & 2 \end{bmatrix} = (2, 6, 4, 0).$$

4.2 Reflection in the Origin in 3-Dimensions by Reflection in the hyper-plane $w = 0$ in 4-Dimensions

Theorem 4.3 *The* 4×4 *orthogonal matrix*

$$I(-1) = \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix}$$

represents reflection in the hyper-plane $w = 0$ *in 4-dimensions, and represents reflection in the origin* o *in 3-dimensions.*

Proof. For any vector $v = ai + bj + ck + do = (a, b, c, d)$ in R^4

$$\begin{aligned} v * I(-1) &= (a, b, c, d) * \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} = (a, b, c, -d) \\ &= ai + bj + ck - do \end{aligned}$$

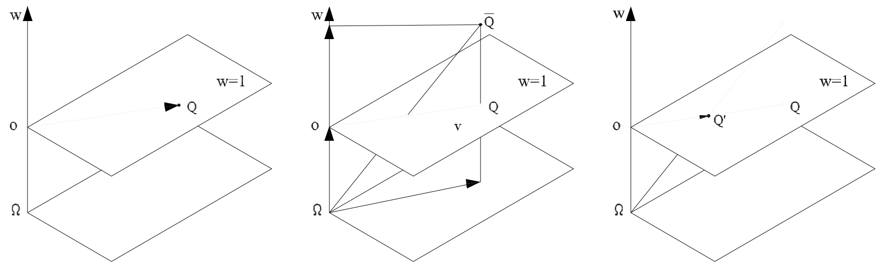


Figure 4: Uniform scaling in R^3 by non-uniform scaling in R^4

which is reflection in the hyper-plane $w = 0$ in 4-dimensions.

Similarly, for any point $Q = ai+bj+ck+o = (a,b,c,1)$ in R^3

$$\begin{aligned} Q * I(-1) &= (a,b,c,1) * \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} = (a,b,c,-1) \\ &= ai + bj + ck - o \equiv -ai - bj - ck + o \\ &= -Q, \end{aligned}$$

which is reflection in the origin o in 3-dimensions.

$\diamondsuit$

The basic geometric idea is illustrated in Figure 5: starting with a point in 3-dimensions represented by the vector Q in 4-dimensions, we reflect the 4-dimensional vector Q in the hyper-plane $w = 0$, giving $\bar{Q}$. Then we project $\bar{Q}$ through the origin Ω to the hyper-plane $w = 1$ to get $-Q$, which is the point Q reflected about the origin o in 3-dimensions.

Recall that the standard matrix for reflection in the origin in 3-dimensions is

$$R = \begin{bmatrix} -I & 0 \\ 0 & 1 \end{bmatrix}.$$

The matrix R reflects any point Q to $-Q$ and reflects any vector v to $-v$. In contrast, the matrix presented in Theorem 4.3 that models reflection about the origin in 3-dimensions by reflection in the hyper-plane $w = 0$ in 4-dimensions is

$$I(-1) = \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix}.$$

Notice that the matrix $I(-1)$ reflects any point Q to $-Q$ but leaves any vector v in R^3 unchanged because vectors in R^3 have no o component. Thus, the matrices R and $I(-1)$ have the same effect on points but different effects on vectors. The matrix R is an affine transformation in 3-dimensions, while the matrix $I(-1)$ is a linear transformation in 4-dimensions but not an affine transformation in 3-dimensions.

Example 4.4 *Consider the vector $v = (2,5,4,3)$ in 4-dimensions, the point $P = (2,6,4,1)$ and the vector $u = (2,6,4,0)$ in 3-dimensions. Then the 4×4 orthogonal matrix*

$$I(-1) = \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix}$$

reflects v in the hyper-plane $w = 0$ in 4-dimensions. In 3-dimensions, $I(-1)$ reflects P in the origin o and leaves u unchanged.

In 4-dimensions,

$$v * I(-1) = (2,5,4,3) * \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} = (2,5,4,-3).$$

In 3-dimensions,

$$\begin{aligned} P * I(-1) &= (2,6,4,1) * \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} = (2,6,4,-1) \\ &\equiv (-2,-6,-4,1) = P' \end{aligned}$$

and

$$u * I(-1) = (2,6,4,0) * \begin{bmatrix} I & 0 \\ 0 & -1 \end{bmatrix} = (2,6,4,0) = u.$$

4.3 Mapping Points to Vectors in 3-Dimensions by Orthogonal Projection Along the w-Axis in 4-Dimensions

Theorem 4.5 *The 4×4 matrix*

$$I(0) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$$

represents orthogonal projection from R^4 to R^3 for vectors in 4-dimensions and maps points Q to vectors $Q-o$ in 3-dimensions.

Proof. For any vector $v = ai + bj + ck + do = (a,b,c,d)$ in R^4

$$v * I(0) = (a,b,c,d) * \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} = (a,b,c,0) = ai+bj+ck,$$

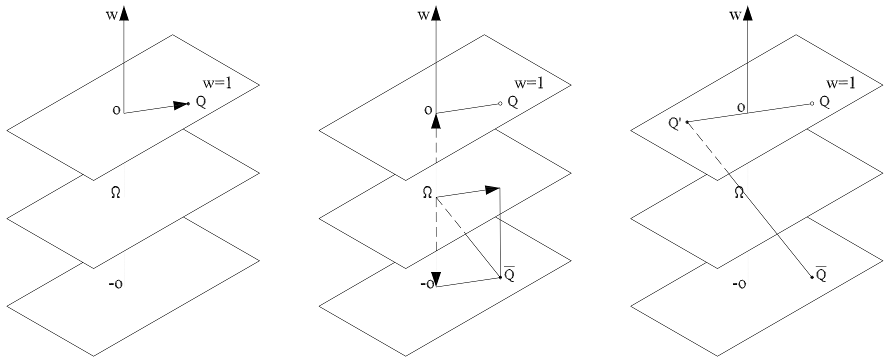


Figure 5: Reflection in the origin in R^3 by reflection in the $w = 0$ hyper-plane in R^4

which represents orthogonal projection from R^4 to R^3 for vectors in 4-dimensions.

Similarly, for any point $Q = ai+bj+ck+o = (a, b, c, 1)$ in R^3

$$\begin{aligned} Q * I(0) &= (a, b, c, 1) * \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} = (a, b, c, 0) \\ &= ai + bj + ck \end{aligned}$$

which maps points Q to vectors $Q - o$ in 3-dimensions.

♢

The basic geometric idea is as follows: starting with a point in 3-dimensions represented by the vector Q in 4-dimensions, we project the vector orthogonally from R^4 to R^3 in 4-dimensions, which maps points Q to vectors $Q - o$ in 3-dimensions. Notice, in particular, that the origin o in 3-dimensions is mapped to the origin Ω in 4-dimensions.

The matrix $I(0)$ in Theorem 4.5 is not an affine transformation and there is no comparable affine matrix in 3-dimensions. We shall see in Section 5.1 that the projection matrix $I(0)$ in Theorem 4.5 plays an important role in the representation of perspective projection.

4.4 Orthogonal Projection in 3-Dimensions by Rotation in 4-Dimensions

In this section we shall show how to use 90° rotations in 4-dimensions together with the matrix $I(0)$ projecting vectors from 4-dimensions to 3-dimensions to compute orthogonal projections in 3-dimensions.

In Lemma 4.6 and in subsequent theorems, we will use the following facts: Consider two row vectors $\mathbf{u}$ and $\mathbf{n}$ in R^3, and let $\mathbf{n^T}$ denote the column vector that is the transpose of the row vector $\mathbf{n}$. Then

1. $\mathbf{u} * \mathbf{n^T} = \mathbf{u} \cdot \mathbf{n}$;
2. $\mathbf{n^T} * \mathbf{n}$ is a 3×3 matrix;
3. $\mathbf{u} * (\mathbf{n^T} * \mathbf{n}) = (\mathbf{u} * \mathbf{n^T}) * \mathbf{n} = (\mathbf{u} \cdot \mathbf{n})\mathbf{n}$.

Lemma 4.6 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$. *The matrix*

$$Rot(n, o, \pm 90^\circ) = \begin{bmatrix} I - (\mathbf{n^T n}) & \pm\mathbf{n^T} \\ \mp\mathbf{n} & 0 \end{bmatrix}$$

represents a simple rotation in 4-dimensions that rotates vectors in the no-plane by $\pm 90^\circ$ *and leaves vectors in* R^4 *orthogonal to the no-plane unchanged.*

Proof. We give the proof for rotation by $+90^\circ$; the proof for rotation by -90° is similar. For any vector $\mathbf{n}_\perp$ perpendicular to $\mathbf{n}$ let $n_\perp = (\mathbf{n}_\perp, 0)$. Then

$$\begin{aligned} o * Rot(n, o, 90^\circ) &= o * \begin{bmatrix} I - (\mathbf{n^T n}) & \mathbf{n^T} \\ -\mathbf{n} & 0 \end{bmatrix} \\ &= -n \\ n * Rot(n, o, 90^\circ) &= (\mathbf{n}, 0) * \begin{bmatrix} I - (\mathbf{n^T n}) & \mathbf{n^T} \\ -\mathbf{n} & 0 \end{bmatrix} \\ &= o \end{aligned}$$

and

$$n_\perp * Rot(n, o, 90^\circ) = (\mathbf{n}_\perp, 0) * \begin{bmatrix} I - (\mathbf{n^T n}) & \mathbf{n^T} \\ -\mathbf{n} & 0 \end{bmatrix} = n_\perp.$$

Therefore it follows by linearity that $Rot(n, o, 90^\circ)$ rotates vectors in the *no*-plane by the angle 90° in 4-dimensions and leaves vectors in R^4 orthogonal to the *no*-plane unchanged.

♢

Theorem 4.7 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions, and let* $n = (\mathbf{n}, 0)$. *Then the matrix*

$$Ortho(n) = Rot(n, o, 90^\circ) * I(0) * Rot(n, o, -90^\circ)$$

$$= \begin{bmatrix} I - (\mathbf{n^T n}) & \mathbf{n^T} \\ -\mathbf{n} & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I - (\mathbf{n^T n}) & -\mathbf{n^T} \\ \mathbf{n} & 0 \end{bmatrix}$$
$$= \begin{bmatrix} I - (\mathbf{n^T n}) & 0 \\ 0 & 1 \end{bmatrix}$$

represents orthogonal projection into the plane π through the origin o orthogonal to the vector $\mathbf{n}$.

Proof. Let $\mathbf{n}_\perp$ be a vector perpendicular to $\mathbf{n}$ and let $n_\perp = (\mathbf{n}_\perp, 0)$. Then by Theorem 4.5 and Lemma 4.6

$$\begin{aligned} & o * Ortho(n) \\ & = o * Rot(n, o, 90^\circ) * I(0) * Rot(n, o, -90^\circ) \\ & = -n * I(0) * Rot(n, o, -90^\circ) \\ & = -n * Rot(n, o, -90^\circ) = o \\ & n * Ortho(n) \\ & = n * Rot(n, o, 90^\circ) * I(0) * Rot(n, o, -90^\circ) \\ & = o * I(0) * Rot(n, o, -90^\circ) \\ & = \Omega * Rot(n, o, -90^\circ) = \Omega \\ & n_\perp * Ortho(n) \\ & = n_\perp * Rot(n, o, 90^\circ) * I(0) * Rot(n, o, -90^\circ) \\ & = n_\perp * I(0) * Rot(n, o, -90^\circ) \\ & = n_\perp * Rot(n, o, -90^\circ) = n_\perp. \end{aligned}$$

Now any point P in R^3 can be written as $P = o + \lambda n + n_\perp$ where $n_\perp$ is a vector perpendicular to $\mathbf{n}$ and λ is a constant. Therefore by linearity

$$P * Ortho(n) = (o + \lambda n + n_\perp) * Ortho(n) = o + n_\perp.$$

$\diamondsuit$

The basic geometric idea here is that the matrix $Rot(n, o, 90^\circ)$ rotates the vector n to the origin o and rotates the origin o to the vector $-n$, while leaving the vectors $n_\perp$ perpendicular to n unchanged. The matrix $I(0)$ then collapses o to the zero vector Ω in 4-dimensions, effectively eliminating the original contribution of n. Finally, the matrix $Rot(n, o, -90^\circ)$ restores the origin o, which was temporarily stored as the vector $-n$.

Example 4.8 *Let the projection plane π be the xy-plane. Then the normal to the projection plane is* $\mathbf{n} = \mathbf{k} = (0, 0, 1)$, *so*

$$Ortho(k) = \begin{bmatrix} I - (\mathbf{k^T k}) & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We can also project orthogonally into arbitrary planes by translating these planes to and from a plane passing through the origin. Thus we have the following general result.

Corollary 4.9 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let $n = (\mathbf{n}, 0)$. The 4×4 matrix*

$$\begin{aligned} Ortho(n, Q) &= Trans(o - Q) * Ortho(n) \\ &\quad * Trans(Q - o) \\ &= \begin{bmatrix} I - (\mathbf{n^T n}) & 0 \\ ((Q - o) \cdot \mathbf{n})\mathbf{n} & 1 \end{bmatrix}, \end{aligned}$$

represents orthogonal projection into the plane π through the point Q orthogonal to the vector $\mathbf{n}$.

5 Perspective Projections

We now turn our attention to perspective projections, the projections needed to support realistic rendering. We shall investigate both standard perspective projections and pseudo-perspective projections, since both types of perspective maps are invoked in contemporary computer graphics.

5.1 Perspective Projection

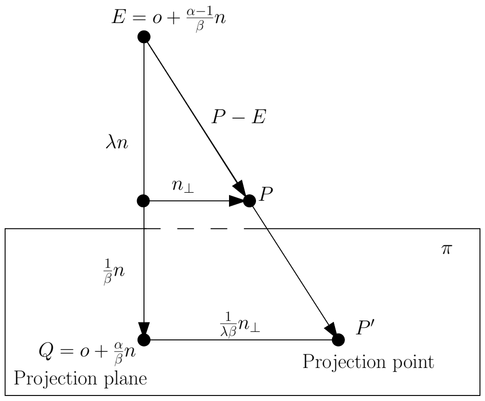
In this section we shall show how to use classical shears, scissor shears, or rotations in 4-dimensions to model perspective projections in 3-dimensions. In each case, starting with a fixed signed distance d from the eye to the projection plane, we initiate this study by selecting a special location for the eye. This location for the eye, the normal to the plane, and the signed distance d fixes the position and orientation of the perspective plane.

We begin with the following general lemma and its corollaries characterizing all the 4×4 matrices that can be used to compute perspective projections in 3-dimensions. In subsections 5.1.1, 5.1.2, 5.1.3 we shall see that the matrices for classical shear, scissor shear, and rotation in 4-dimensions that can be used to compute perspective projections in 3-dimensions are all special cases of these general matrices. Of course, for a fixed eye point and a fixed perspective plane, the matrix for perspective projection is unique. What we show here is that there are different ways to factor this perspective matrix in terms of different classical transformations in 4-dimensions.

Lemma 5.1 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let $n = (\mathbf{n}, 0)$. For $\alpha\delta - \beta\gamma \neq 0$, $\beta \neq 0$, the 4×4 matrix*

$$M = M(n, \alpha, \beta, \gamma, \delta) = \begin{bmatrix} I + (\alpha - 1)(n^T * n) & \beta n^T \\ \gamma n & \delta \end{bmatrix}$$

represents a non-singular linear transformation that maps vectors in the no-plane to vectors in the no-plane and leaves vectors $n_\perp$ in R^4 orthogonal to the no-plane fixed in 4-dimensions. In 3-dimensions, the matrix M represents perspective projection in the following way:



$E = o + \frac{\alpha-1}{\beta}n$
λn
$P - E$
$n_\perp$
P
π
$\frac{1}{\beta}n$
$\frac{1}{\lambda\beta}n_\perp$
P'
$Q = o + \frac{\alpha}{\beta}n$
Projection plane
Projection point

Figure 6: Perspective projection

let $E = o + \frac{\alpha-1}{\beta}n$ be the eye point and let π be the projection plane through the point $Q = o + \frac{\alpha}{\beta}n$ at a signed distance $d = \frac{1}{\beta}$ from the eye point E. Then for any point P, applying M to the vector $P - E$ represents the perspective projection of the point P to the plane π from the eye point E.

Proof. For any vector $\mathbf{n}_\perp$ perpendicular to $\mathbf{n}$, let $n_\perp = (\mathbf{n}_\perp, 0)$. Then

$$o * M = o * \begin{bmatrix} I + (\alpha-1)(n^T * n) & \beta n^T \\ \gamma n & \delta \end{bmatrix} = \gamma n + \delta o$$

$$n * M = n * \begin{bmatrix} I + (\alpha-1)(n^T * n) & \beta n^T \\ \gamma n & \delta \end{bmatrix} = \alpha n + \beta o$$

$$n_\perp * M = n_\perp * \begin{bmatrix} I + (\alpha-1)(n^T * n) & \beta n^T \\ \gamma n & \delta \end{bmatrix} = n_\perp$$

Therefore by linearity M represents a non-singular linear transformation that maps vectors in the *no*-plane to vectors in the *no*-plane and leaves vectors $n_\perp$ in R^4 orthogonal to the *no*-plane fixed.

Now for any point P in R^3, since $P - E$ is a vector in R^3, there exists a vector $n_\perp$ perpendicular to n and a constant λ such that $P - E = \lambda n + n_\perp$. Therefore,

$$(P - E) * M = (\lambda n + n_\perp) * M = \lambda(\alpha n + \beta o) + n_\perp$$
$$\equiv o + \frac{\alpha}{\beta}n + \frac{1}{\lambda\beta}n_\perp.$$

Let

$$P' = o + \frac{\alpha}{\beta}n + \frac{1}{\lambda\beta}n_\perp.$$

Then $P' \equiv (P - E) * M$. Moreover since $Q = o + \frac{\alpha}{\beta}n$,

$$P' - Q = \frac{1}{\lambda\beta}n_\perp \Rightarrow (P' - Q) \perp n.$$

Hence P' is on the projection plane π. Also since $E = o + \frac{\alpha-1}{\beta}n$,

$$P' = E + \frac{1}{\lambda\beta}(\lambda n + n_\perp) = E + \frac{1}{\lambda\beta}(P - E).$$

Hence P' is on the line through the eye point E in the direction $P - E$. Thus P' is located at the intersection of the line from the eye point E to the point P, and the plane π through the point Q at a signed distance $d = 1/\beta$ from the eye point E. Hence P' is the perspective projection of the point P from the eye point E into the projection plane π in 3-dimensions (see Figure 6). Note too that if P lies on the plane π, then $\lambda = \frac{1}{\beta}$ and $P' = P$.

$\diamondsuit$

Notice that the matrix M in Lemma 5.1 maps vectors in R^3 parallel to n into mass points in R^3; this mapping of vectors to mass points is important below in Corollary 5.2, where $I(0)$ is used to maps points to vectors.

To compute perspective projection, the matrix in Lemma 5.1 operates on vectors rather than points. But to implement perspective projection in the graphics pipeline, we need matrices that operate directly on points rather than on vectors from the eye to the points. Next we introduce such matrices. Notice that since these matrices operate directly on points P, the eye point E must now be incorporated inside these matrices.

Corollary 5.2 *Let $\mathbf{n}$ be a unit vector in 3-dimensions and let $n = (\mathbf{n}, 0)$. The 4×4 matrix*

$$\begin{aligned} &Persp(E, n, \theta) \\ &\quad = Trans(o - E) * I(0) * M(n, \alpha, \beta, \gamma, \delta) \\ &\quad = Trans(o - E) * I(0) * \begin{bmatrix} I + (\alpha-1)(n^T * n) & \beta n^T \\ \gamma n & \delta \end{bmatrix} \\ &\quad = \begin{bmatrix} I + (\alpha-1)(n^T * n) & \beta n^T \\ (o - E) + (\alpha-1)\big((o - E) \cdot n\big)n & \beta(o - E) \cdot n \end{bmatrix} \end{aligned}$$

projects points P from the eye point $E = o + \frac{\alpha-1}{\beta}n$ to the plane π with unit normal $\mathbf{n}$ at a signed distance $d = \frac{1}{\beta}$ from the eye. Notice, in particular, that if $\alpha = 1$, then $E = o$.

Proof. By Theorem 4.5:

$$P * Trans(o - E) * I(0) * M = (P - E) * M.$$

Therefore Corollary 5.2 follows from Lemma 5.1.

$\diamondsuit$

Corollary 5.2 shows how to compute perspective projection on arbitrary points when the eye point and the perspective plane are located in special canonical positions. Next we show how to compute perspective projection on arbitrary points when the eye point and the perspective plane are located in arbitrary positions by translating the scene to and from the canonical position.

Corollary 5.3 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$*. Set* $E = o + \frac{\alpha-1}{\beta} n$ *to the canonical eye point from Lemma 5.1. Then the* 4×4 *matrix*

$$Persp(E', n, \theta) = Trans(o - E') * I(0) * M(n, \alpha, \beta, \gamma, \delta) * Trans(E' - E)$$

projects points P *from the eye point* E' *to the plane* π' *with unit normal* $\mathbf{n}$ *at a signed distance* $d = \frac{1}{\beta}$ *from the eye.*

Proof. This result is a consequence of the following four observations.

1. $Trans(o - E') = Trans(E - E') * Trans(o - E)$.

2. The matrix $Trans(E - E')$ translates the entire scene by the vector $E - E'$. In particular, the matrix $Trans(E - E')$ translates the eye point E' to the canonical eye point E in 3-dimensions and translates the projection plane π' to the canonical plane π with unit normal $\mathbf{n}$ at a signed distance $d = \frac{1}{\beta}$ from the canonical eye point E.

3. By Corollary 5.2, the matrix $Trans(o - E) * I(0) * M(n, \alpha, \beta, \gamma, \delta)$ projects points from the canonical eye point E to the canonical plane π with unit normal n at a signed distance $d = \frac{1}{\beta}$ from the eye point E.

4. The matrix $Trans(E' - E)$ translates the entire scene back to its original position, mapping the canonical eye point E back to the original eye point E' and the canonical plane π with unit normal $\mathbf{n}$ at a signed distance $d = \frac{1}{\beta}$ from the canonical eye point E back to the original projection plane π'.

$\diamondsuit$

By Lemma 5.1 and Corollaries 5.2, 5.3, any non-singular matrix that in 4-dimensions maps the *no*-plane to the *no*-plane and fixes vectors orthogonal to the *no*-plane can be used to compute perspective projection in 3-dimensions into planes orthogonal to n, provided only that β, the entry in the upper right-hand corner of the matrix, is not zero. Thus simple classical shears, scissor shears, and rotations in the *no*-plane in 4-dimensions can all be used to compute perspective projections in 3-dimensions. We shall explore the 4×4 matrices corresponding to these three transformations in the next three subsections in more detail. Notice, however, that the 4×4 matrices corresponding to non-uniform scaling along directions in the *no*-plane cannot be used to compute perspective projections in 3-dimensions because for these non-uniform scaling matrices $\beta = 0$.

5.1.1 Perspective Projection in 3-Dimensions by Classical Shear in 4-Dimensions

In Section 3 we showed how to translate points in 3-dimensions in the direction $\mathbf{n}$ by using classical shears in the *no*-plane in the *o*-direction in 4-dimensions. Here we show how to perform perspective projections in 3-dimensions by using classical shears in the *on*-plane in the *n*-direction in 4-dimensions.

Theorem 5.4 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions, and let* $n = (\mathbf{n}, 0)$*. For* θ *not an integer multiple of* $90°$*, the* 4×4 *matrix*

$$CShear(o, n, \theta) = \begin{bmatrix} I & \tan(\theta)\mathbf{n}^{\mathbf{T}} \\ 0 & 1 \end{bmatrix}$$

represents a classical shear that shears vectors in the on-plane by the angle θ *in the direction* n *in 4-dimensions. In 3-dimensions, the matrix* $CShear(o, n, \theta)$ *represents perspective projection in the following way: place the eye point* E *at the origin* o*, and let* π *be the projection plane with unit normal vector* $\mathbf{n}$ *through the point* $Q = o + \cot(\theta) n$ *at a signed distance* $d = \cot(\theta)$ *from the eye point* E*. Then for any point* P*, applying* $CShear(o, n, \theta)$ *to the vector* $P - E$ *represents the perspective projection of the point* P *to the plane* π *from the eye point* E*.*

Proof. For any vector $\mathbf{n}_\perp$ perpendicular to $\mathbf{n}$ let $n_\perp = (\mathbf{n}_\perp, 0)$. Then

$$o * CShear(o, n, \theta) = o * \begin{bmatrix} I & \tan(\theta)\mathbf{n}^{\mathbf{T}} \\ 0 & 1 \end{bmatrix} = o$$

$$n * CShear(o, n, \theta) = (\mathbf{n}, 0) * \begin{bmatrix} I & \tan(\theta)\mathbf{n}^{T} \\ 0 & 1 \end{bmatrix} = n + \tan(\theta) o$$

$$n_\perp * CShear(o, n, \theta) = (\mathbf{n}_\perp, 0) * \begin{bmatrix} I & \tan(\theta)\mathbf{n}^{T} \\ 0 & 1 \end{bmatrix} = n_\perp.$$

Therefore $CShear(o, n, \theta)$ shears vectors in the *on*-plane by the angle θ in the direction n in 4-dimensions.

The result in 3-dimensions follows from Lemma 5.1 with $\alpha = 1$, $\beta = \tan(\theta)$, $\gamma = 0$, and $\delta = 1$.

$\diamondsuit$

Notice that the shear matrix in Theorem 5.4 that represents perspective projection in 3-dimensions is the transpose of the shear matrix in Theorem 3.1 that represents translation in 3-dimensions.

To compute perspective projection, the matrix in Theorem 5.4 operates on vectors rather than points. But as we observed in Section 5.1, to implement perspective projection in the graphics pipeline, we need matrices that operate directly on points rather than on vectors from the eye to the points. Next we introduce such matrices.

Corollary 5.5 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$*. For* θ *not an integer multiple of* $90°$*, the* 4×4 *matrix*

$$\begin{aligned} Persp(o, n, \theta) &= I(0) * CShear(o, n, \theta) \\ &= \begin{bmatrix} I & \tan(\theta)\mathbf{n}^T \\ 0 & 0 \end{bmatrix} \end{aligned}$$

projects points P *from the eye point* $E = o$ *to the plane* π *with unit normal* $\mathbf{n}$ *at a signed distance* $d = cot(\theta)$ *from the eye.*

Proof. Corollary 5.5 is a special case of Corollary 5.2 with $\alpha = 1$, $\beta = \tan(\theta)$, $\gamma = 0$, and $\delta = 1$. Notice that in Corollary 5.2 if $\alpha = 1$ then $E = o$.

$\diamondsuit$

The basic geometric idea is illustrated in Figure 7: starting with an eye point E at the origin o at an arbitrary signed distance $d = \cot(\theta)$ away from the projection plane, we project points from the plane $w = 1$ to the plane $w = 0$ in 4-dimensions using the projection $I(0)$ along the w-axis to map points P to vectors $P - o$ and map the eye point $E = o$ to the origin Ω in 4-dimensions. We then use the transformation $CShear(o, n, \theta)$ to shear in the *on*-plane along the direction n, leaving the direction o (the w-axis) fixed and mapping the projection plane back into the plane $w = 1$ and the vector $P - o$ to the mass-point $\bar{P}$ in 4-dimensions. As a final step, we project $\bar{P}$ from the origin Ω to the plane $w = 1$ in 4-dimensions to get P', which is the perspective projection of the point P into the projection plane from the eye point in 3-dimensions.

Example 5.6 *Let* π *be the projection plane with normal vector* $n = (0, 0, 1, 0) = k$ *through the point* $Q = (0, 0, 1, 1)$ *and let* $E = (0, 0, 0, 1) = o$ *be the eye point. Then the* 4×4 *matrix*

$$CShear(o, k, 45°) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is a classical shear in the wz-plane by the angle $45°$ *in the z-direction in 4-dimensions, and when composed with* $I(0)$ *projects points from the eye point* E *at the origin* o *to a perspective plane* π *parallel to the xy-plane in 3-dimensions. Thus*

$$\begin{aligned} &P * I(0) * CShear(o, k, 45°) \\ &= (P - o) * CShear(o, k, 45°) \\ &= (p_1, p_2, p_3, 0) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= (p_1, p_2, p_3, p_3) \equiv (\frac{p_1}{p_3}, \frac{p_2}{p_3}, 1, 1). \end{aligned}$$

Corollary 5.5 shows how to use classical shear to compute perspective projection on arbitrary points when the eye point and the perspective plane are located in special canonical positions. Next as in Section 5.1 we show how to compute perspective projection on arbitrary points when the eye point and the perspective plane are located in arbitrary positions by translating the scene to and from the canonical position.

Corollary 5.7 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$*. For* θ *not an integer multiple of* $90°$*, the* 4×4 *matrix*

$$\begin{aligned} Persp(E', n, \theta) = &Trans(o - E') * I(0) \\ &* CShear(o, n, \theta) * Trans(E' - o) \end{aligned}$$

projects points P *from the eye point* E' *to the plane* π' *with unit normal* $\mathbf{n}$ *at a signed distance* $d = cot(\theta)$ *from the eye.*

Proof. Corollary 5.7 is a special case of Corollary 5.3 with $\alpha = 1$, $\beta = \tan(\theta)$, $\gamma = 0$, and $\delta = 1$.

$\diamondsuit$

Example 5.8 *Let the projection plane* π *be the xy-plane and place the eye point at* $E = (0, 0, -1, 1)$ *a unit distance below the xy-plane. Then the normal to the projection plane is* $\mathbf{n} = \mathbf{k} = (0, 0, 1)$ *and the distance from the eye to the projection plane is* $d = 1$*. Therefore,* $\tan(\theta) = 1/d = 1$*, so* $\theta = 45°$*. Thus, in this case, the matrix representing perspective projection is*

$$\begin{aligned} &Persp(E, k, 45°) \\ &= Trans(\mathbf{k}) * I(0) * CShear(o, k, 45°) * Trans(-\mathbf{k}). \end{aligned}$$

Hence for any point $P = (x, y, z, 1)$

$$\begin{aligned} &P{*}Persp(E, k, 45°) \\ &\quad = (x, y, z, 1) * Trans(\mathbf{k}) * I(0) \\ &\qquad * CShear(o, k, 45°) * Trans(-\mathbf{k}) \\ &\quad = (x, y, z + 1, 1) * I(0) * CShear(o, k, 45°) \\ &\qquad * Trans(-\mathbf{k}) \\ &\quad = (x, y, z + 1, 0) * CShear(o, k, 45°) * Trans(-\mathbf{k}) \\ &\quad = (x, y, z + 1, z + 1) * Trans(-\mathbf{k}) \\ &\quad = (x, y, 0, z + 1) \\ &\quad \equiv \left(\frac{x}{z+1}, \frac{y}{z+1}, 0, 1\right) \end{aligned}$$

Alternatively, by direct computation

$$\begin{aligned} &Persp(E, k, 45°) \\ &\quad = Trans(\mathbf{k}) * I(0) * CShear(o, k, 45°) * Trans(-\mathbf{k}) \\ &\quad = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

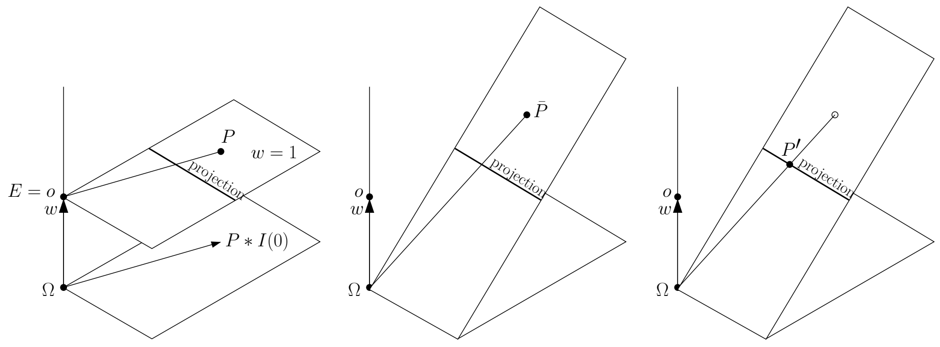


Figure 7: Perspective projection in R^3 by classical shear in R^4

so

$$
\begin{aligned}
&P * Persp(E,k,45^\circ)\\
&= (x,y,z,1) * \begin{pmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1 \end{pmatrix}\\
&= (x,y,0,z+1) \equiv (\frac{x}{z+1}, \frac{y}{z+1}, 0, 1).
\end{aligned}
$$

5.1.2 Perspective Projection in 3-Dimensions by Scissor Shear in 4-Dimensions

In Section 5.1.1 we showed how to compute perspective projection using classical shears in the *on*-plane in the n-direction in 4-dimensions. Here we show how to perform perspective projections in 3-dimensions by using scissor shears in the *no*-plane in 4-dimensions.

Theorem 5.9 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions, and let* $n = (\mathbf{n}, 0)$*. For* $\theta \neq 0$*, the* 4×4 *matrix*

$$
SShear(n,o,\theta) = \begin{bmatrix} I + (\cosh(\theta)-1)(\mathbf{n}^T\mathbf{n}) & \sinh(\theta)\mathbf{n}^T \\ \sinh(\theta)\mathbf{n} & \cosh(\theta) \end{bmatrix}
$$

represents a scissor shear that shears vectors in the no-plane by the angle θ *in 4-dimensions. In 3-dimensions, the matrix* $SShear(n,o,\theta)$ *represents perspective projection in the following way: let* $E = o + (\coth(\theta) - \operatorname{csch}(\theta))n$ *be the eye point, and let* π *be the projection plane with unit normal vector* n *through the point* $Q = o + \coth(\theta)n$ *at a signed distance* $d = \operatorname{csch}(\theta)$ *from the eye point* E*. Then for any point* P*, applying* $SShear(n,o,\theta)$ *to the vector* $P-E$ *represents the perspective projection of the point* P *to the plane* π *from the eye point* E*.*

Proof. For any vector $\mathbf{n}_\perp$ perpendicular to $\mathbf{n}$ let $n_\perp = (\mathbf{n}_\perp, 0)$. Then

$$
\begin{aligned}
&o * SShear(n,o,\theta)\\
&\quad = o * \begin{bmatrix} I + (\cosh(\theta)-1)(\mathbf{n^Tn}) & \sinh(\theta)\mathbf{n^T} \\ \sinh(\theta)\mathbf{n} & \cosh(\theta) \end{bmatrix}\\
&\quad = \cosh(\theta)o + \sinh(\theta)n\\
&n * SShear(n,o,\theta)\\
&\quad = (\mathbf{n},0) * \begin{bmatrix} I + (\cosh(\theta)-1)(\mathbf{n^Tn}) & \sinh(\theta)\mathbf{n^T} \\ \sinh(\theta)\mathbf{n} & \cosh(\theta) \end{bmatrix}\\
&\quad = \sinh(\theta)o + \cosh(\theta)n\\
&n_\perp * SShear(n,o,\theta)\\
&\quad = (\mathbf{n}_\perp,0) * \begin{bmatrix} I + (\cosh(\theta)-1)(\mathbf{n^Tn}) & \sinh(\theta)\mathbf{n^T} \\ \sinh(\theta)\mathbf{n} & \cosh(\theta) \end{bmatrix}\\
&\quad = n_\perp.
\end{aligned}
$$

Therefore $SShear(n,o,\theta)$ shears vectors in the *no*-plane by the angle θ in 4-dimensions.

The result in 3-dimensions follows from Lemma 5.1 with $\alpha = \cosh(\theta)$, $\beta = \sinh(\theta)$, $\gamma = \sinh(\theta)$, and $\delta = \cosh(\theta)$.

$\diamondsuit$

The basic geometric idea is illustrated in Figure 8: to locate the projection plane a signed distance $d = \operatorname{csch}(\theta)$ from the eye point, we start with the eye point $E = o + (\coth(\theta) - \operatorname{csch}(\theta))n$. We then translate the eye point E to the origin o, project the vector $P-E$ into the $w=0$ plane (mapping E from o to Ω), and perform a scissor shear that leaves the projection plane unchanged. This transformation also shears any vector $P = (v,1)$ to a vector $\bar{P} = (wv, w)$ in 4-dimensions. As a final step, we project $\bar{P}$ from the origin Ω to the plane $w=1$ in 4-dimensions to get P', which is the perspective projection of the point P into the projection plane from the eye point in 3-dimensions.

Once again to implement perspective projection in the graphics pipeline, we need matrices that operate directly

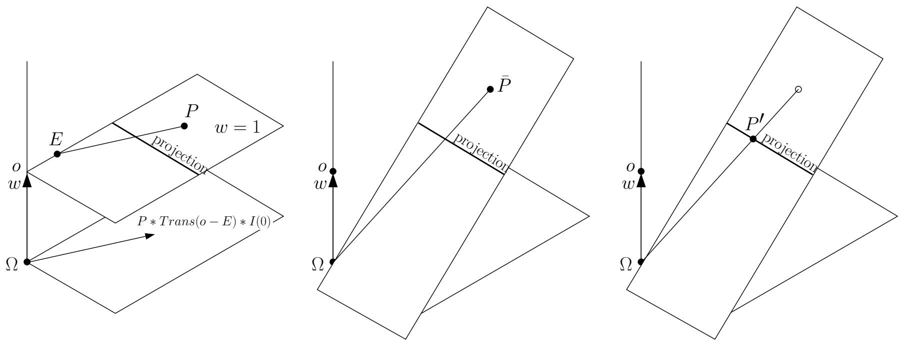


Figure 8: Perspective projection in R^3 by scissor shear in R^4

on points rather than on vectors from the eye to the points. Next we introduce such matrices.

Corollary 5.10 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$. *For* $\theta \neq 0$, *the* 4×4 *matrix*

$$Persp(n, o, \theta) = Trans(o - E) * I(0) * SShear(n, o, \theta)$$

projects points P *from the eye point* $E = o + (\coth(\theta) - \operatorname{csch}(\theta))n$ *to the plane* π *with unit normal* $\mathbf{n}$ *at a signed distance* $d = \operatorname{csch}(\theta)$ *from the eye.*

Proof. Corollary 5.10 is a special case of Corollary 5.2 with $\alpha = \cosh(\theta)$, $\beta = \sinh(\theta)$, $\gamma = \sinh(\theta)$, and $\delta = \cosh(\theta)$.

$\diamondsuit$

Example 5.11 *Let* π *be the perspective plane with normal vector* $n = (0, 0, 1, 0) = k$ *through the point* $Q = (0, 0, \coth(\theta), 1)$ *and let* $E = (0, 0, \coth(\theta) - \operatorname{csch}(\theta), 1)$ *be the eye point. Then for* $\theta \neq 0$, *the* 4×4 *matrix*

$$SShear(k, o, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cosh(\theta) & \sinh(\theta) \\ 0 & 0 & \sinh(\theta) & \cosh(\theta) \end{bmatrix}$$

scissor shears vectors in the zw-plane by the angle θ *in 4-dimensions, and when composed with* $Trans(o - E) * I(0) = Trans(-E)$ *projects points* P *from the eye point* E *on the z-axis to a perspective plane* π *parallel to the xy-plane in 3-dimensions.*

Corollary 5.10 shows how to use scissor shear to compute perspective projection on arbitrary points when the eye point and the perspective plane are in special canonical positions. Next as in Sections 5.1 and 5.1.1 we show how to compute perspective projection on arbitrary points for the eye point and the perspective plane in arbitrary positions by translating the scene to and from the canonical position.

Corollary 5.12 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$. *Set* $E = o + (\coth(\theta) - \operatorname{csch}(\theta))n$ *to the canonical eye point from Theorem 5.9. Then for* $\theta \neq 0$, *the* 4×4 *matrix*

$$\begin{aligned} &Persp(E', n, \theta) \\ &= Trans(o - E') * I(0) * SShear(n, o, \theta) * Trans(E' - E) \end{aligned}$$

projects points P *from the eye point* E' *to the plane* π' *with unit normal* $\mathbf{n}$ *at a signed distance* $d = \operatorname{csch}(\theta)$ *from the eye.*

Proof. Corollary 5.12 is a special case of Corollary 5.3 with $\alpha = \cosh(\theta)$, $\beta = \sinh(\theta)$, $\gamma = \sinh(\theta)$, and $\delta = \cosh(\theta)$.

$\diamondsuit$

Example 5.13 *Let the projection plane* π' *be the xy-plane and place the eye point at* $E' = (0, 0, -1, 1) = o - k$ *a unit distance below the xy-plane. Then the normal to the projection plane is* $\mathbf{n} = \mathbf{k} = (0, 0, 1)$ *and the distance from the eye to the projection plane is* $d = 1$. *Therefore,* $\sinh(\theta) = 1/d = 1$, *so* $\cosh(\theta) = \sqrt{2}$, *and the canonical eye point is located at* $E = o + (\coth\theta) - \operatorname{csch}(\theta))\mathbf{n} = o + (\sqrt{2} - 1)\mathbf{k}$ *Thus, in this case, the matrix representing perspective projection is*

$$\begin{aligned} &Persp(E', k, \theta) \\ &= Trans(\mathbf{k}) * I(0) * SShear(k, o, \theta) * Trans(-\sqrt{2}\mathbf{k}). \end{aligned}$$

Hence for any point $P = (x, y, z, 1)$

$$\begin{aligned}
&P * Persp(E', k, \theta) \\
&= (x, y, z, 1) * Trans(\mathbf{k}) * I(0) * SShear(k, o, \theta) \\
&\quad * Trans(-\sqrt{2}\mathbf{k}) \\
&= (x, y, z + 1, 1) * I(0) * SShear(k, o, \theta) \\
&\quad * Trans(-\sqrt{2}\mathbf{k}) \\
&= (x, y, z + 1, 0) * \begin{bmatrix} I + (\sqrt{2} - 1)\mathbf{k}^T\mathbf{k} & \mathbf{k}^T \\ \mathbf{k} & \sqrt{2} \end{bmatrix} \\
&\quad * \begin{bmatrix} I & 0 \\ -\sqrt{2}\mathbf{k} & 1 \end{bmatrix} \\
&= (x, y, \sqrt{2}(z + 1), z + 1) * \begin{bmatrix} I & 0 \\ -\sqrt{2}\mathbf{k} & 1 \end{bmatrix} \\
&= (x, y, 0, z + 1) \\
&\equiv (\frac{x}{z+1}, \frac{y}{z+1}, 0, 1)
\end{aligned}$$

Alternatively, by direct computation

$$\begin{aligned}
&Persp(E', k, \theta) \\
&= Trans(\mathbf{k}) * I(0) * SShear(k, o, \theta) * Trans(-\sqrt{2}\mathbf{k}) \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

so

$$\begin{aligned}
&P * Persp(E', k, \theta) \\
&= (x, y, z, 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= (x, y, 0, z + 1) \equiv (\frac{x}{z+1}, \frac{y}{z+1}, 0, 1)
\end{aligned}$$

Notice that we get the same matrix for perspective projection as in Example 5.8, but factored in a different way.

5.1.3 Perspective Projection in 3-Dimensions by Rotation in 4-Dimensions

Here we show how to perform perspective projections in 3-dimensions by using rotations in 4-dimensions. The proofs of the results in this section are essentially the same as the proofs in Section 5.1.2 with sinh and cosh replaced by sin and cos, and the identity $\cosh^2(\theta) - \sinh^2(\theta) = 1$ replaced by the identity $\cos^2(\theta) + \sin^2(\theta) = 1$. Therefore, in this section we shall omit the proofs. For an alternative discussion of this topic, which also includes quaternions, see [7].

Theorem 5.14 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions, and let* $n = (\mathbf{n}, 0)$. *For* θ *not an integer multiple of* $180°$, *the* 4×4 *matrix*

$$Rot(n, o, \theta) = \begin{bmatrix} I + (\cos(\theta) - 1)(\mathbf{n^T n}) & \sin(\theta)\mathbf{n^T} \\ -\sin(\theta)\mathbf{n} & \cos(\theta) \end{bmatrix}$$

represents a rotation that rotates vectors in the no-plane by the angle θ *in 4-dimensions. In 3-dimensions, the matrix* $Rot(n, o, \theta)$ *represents perspective projection in following way: let* $E = o + (\cot(\theta) - \csc(\theta))n$ *be the eye point, and let* π *be the projection plane with unit normal vector* n *through the point* $Q = o + \cot(\theta)n$ *at a signed distance* $d = \csc(\theta)$ *from the eye point* E. *Then for any point* P, *applying* $Rot(n, o, \theta)$ *to the vector* $P - E$ *represents the perspective projection of the point* P *to the plane* π *from the eye point* E.

The basic geometric idea is illustrated in Figure 9: to locate the projection plane a signed distance $d = \csc(\theta)$ from the eye, we start with the eye point at $E = o + (\cot(\theta) - \csc(\theta))n$. We then perform a rotation that leaves the projection plane unchanged and moves the eye to the origin Ω in 4-dimensions. This transformation also rotates any vector $P = (v, 1)$ to a vector $\bar{P} = (wv, w)$ in 4-dimensions. As a final step, we project $\bar{P}$ from the origin Ω to the plane $w = 1$ in 4-dimensions to get P', which is the perspective projection of the point P into the projection plane from the eye point in 3-dimensions.

Once again we introduce matrices to operate directly on points rather than on vectors from the eye to the points.

Corollary 5.15 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$. *For* θ *not an integer multiple of* $180°$, *the* 4×4 *matrix*

$$Persp(n, o, \theta) = Trans(o - E) * I(0) * Rot(n, o, \theta)$$

projects points P *from the eye point* $E = o + (\cot(\theta) - \csc(\theta))n$ *to the plane* π *with unit normal* $\mathbf{n}$ *at a signed distance* $d = \csc(\theta)$ *from the eye.*

Example 5.16 *Let* π *be the projection plane with normal vector* $n = (0, 0, 1, 0) = k$ *through the point* $Q = (0, 0, \cot(\theta), 1)$ *and let* $E = (0, 0, \cot(\theta) - \csc(\theta), 1)$ *be the eye point. For* θ *not an integer multiple of* $180°$, *the* 4×4 *matrix*

$$Rot(k, o, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

rotates vectors in the zw-plane by the angle θ *in 4-dimensions, and when composed with* $Trans(o - E) * I(0) = Trans(-E)$ *projects points* P *from an eye point* E *on the z-axis to a perspective plane* π *parallel to the xy-plane in 3-dimensions.*

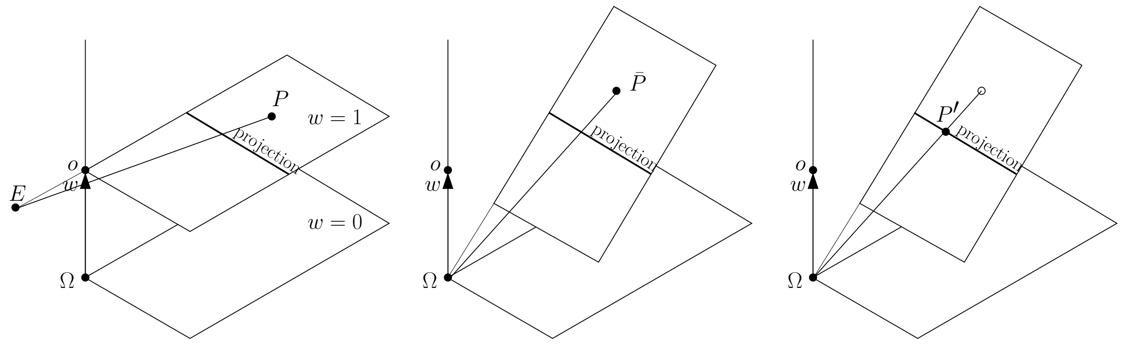


Figure 9: Perspective projection in R^3 by rotation in R^4

Next as in Section 5.1.1 and 5.1.2, we show how to compute perspective projection on arbitrary points for the eye point and the perspective plane in arbitrary positions by translating the scene to and from the canonical position.

Corollary 5.17 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$*. Let* $E = o + (\cot(\theta) - \csc(\theta))n$ *be the canonical eye point from Theorem 5.14. For* θ *not an integer multiple of* $180°$*, the* 4×4 *matrix*

$$\begin{aligned}&Persp(E', n, \theta)\\ &= Trans(o - E') * I(0) * Rot(n, o, \theta) * Trans(E' - E)\end{aligned}$$

projects points P *from the eye point* E' *to the plane* π' *with unit normal* $\mathbf{n}$ *at a signed distance* $d = \csc(\theta)$ *from the eye.*

Example 5.18 *Let the projection plane* π' *be the xy-plane and place the eye point at* $E' = (0, 0, -1, 1) = o - k$ *a unit distance below the xy-plane. Then the normal to the projection plane is* $\mathbf{n} = \mathbf{k} = (0, 0, 1)$ *and the distance from the eye to the projection plane is* $d = 1$*. Therefore,* $sin(\theta) = 1/d = 1$*, so* $\theta = 90°$ *and* $cos(\theta) = 0$*. Hence the canonical eye point is located at* $E = o + (cot(90°) - \csc(90°))\mathbf{n} = o - \mathbf{k}$*. Thus, in this case, the matrix representing perspective projection is*

$$\begin{aligned}Persp(E', k, 90°) &= Trans(\mathbf{k}) * I(0) * Rot(k, o, 90°)\\ &\quad * Trans(0)\\ &= Trans(\mathbf{k}) * I(0) * Rot(k, o, 90°)\\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}\\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.\end{aligned}$$

Notice that we get the same matrix as in Examples 5.8 and 5.13, but factored in yet another way.

Remark 5.19 *For rotation, unlike classical shear and scissor shear, since* $d = \csc(\theta)$ *and* $|\csc(\theta)| \geq 1$*, when the distance from the eye to the perspective plane is less than one, we cannot apply this method directly. Instead we must first scale the entire scene uniformly by* $1/|d|$*. Then we can perform perspective projection using rotations in 4-dimensions. To complete the transformation, we must rescale the resulting scene uniformly by* $|d|$*. In this way, this perspective projection matrix based on rotations in 4-dimensions can be made to work for eye points* E' *at an arbitrary distance from the projection plane.*

5.1.4 Reflections on Perspective Projection

We have presented three different techniques for computing perspective projections in 3-dimensions by using classical shears, scissor shears, or rotations in 4-dimensions. Of course, when the eye point and the perspective plane are fixed, the matrix representing perspective projection is unique because the action of the matrix is well-defined on a basis in 4-dimensions. Thus for a fixed eye point and perspective plane, each of these approaches yields the same 4×4 matrices for perspective projection, but each method factors these matrices in different ways (see Examples 5.8, 5.13, 5.18).

Remarkably, despite the fact that the perspective map projects a scene in 3-dimensions into a plane, no information is lost by these projections. To see why, let $\lambda = \lambda(P)$ be the distance from a point P to the plane that contains the eye point E and is parallel to the projection plane π—that is, the plane through E with unit normal $\mathbf{n}$—and let $d = \frac{1}{\beta}$ be the signed distance from the eye point E to the projection plane π (see Figure 6). Then the image of P under these perspective projections yields not only the location of the projection of P from the eye point E into the perspective plane π, but also the mass $\lambda(P)/d = \lambda\beta$ (see the proof of Lemma 5.1). Indeed the perspective projection of P is a mass-point. Since $d = \frac{1}{\beta}$ is a constant, we can recover the signed distance $\lambda(P)$ from this mass. Thus even though we are computing projections, the distance of a point P from the plane of the eye is preserved. Knowing this distance allows us to perform hidden surface algorithms: if two points project to the same point on the perspective plane, we see the point closest to the eye (for use in a Z-buffer scan line algorithm, the weight λ must be converted to *pseudo-depth* $= 1 - \frac{1}{\lambda}$) [5]. One reason that no information is lost is that the only projection we actually perform is the map $I(0)$, which projects orthogonally from 4-dimensions to 3-dimensions (see Corollaries 5.2, 5.3); the maps that project the scene from 3-dimensions to the perspective plane are all non-singular (see again Corollaries 5.2, 5.3).

Finally, notice that for arbitrary locations of the eye point E' and the perspective plane π', we perform perspective projection by mapping to and from the canonical positions of the eye point E and the canonical perspective plane π that appear in Lemma 5.1 and in Theorems 5.4, 5.9, 5.14 (see Corollaries 5.2, 5.3, 5.7, 5.12, 5.17). The final translation, $Trans(E' - E)$, simply maps the canonical perspective plane back to the original perspective plane; the projected scene already appears in the canonical perspective plane π. Thus, this final translation is not really necessary; we can display the projected scene directly from its image in the canonical plane π. Alternatively, once we have the projected scene in the canonical plane π, we can transform this image into any plane we desire by a rigid motion. For example, we can place the image in the xy-plane by translating the point Q on the canonical plane at a distance d from the canonical eye point E to the origin o and then rotating the normal vector $\mathbf{n}$ into the z-axis by rotating around the axis vector $n \times k$ by the angle θ, where $cos(\theta) = n \cdot k$. In this way, if we like, we can always recover any scene from the xy-plane.

5.2 Pseudo-Perspective

Pseudo-perspective maps the eye point to a point at infinity and a viewing frustum to a rectangular box [10]. Pseudo-perspective is used in computer graphics for three purposes: (i) to speed up clipping algorithms by replacing the viewing frustum with a rectangular box; (ii) to simplify projections by replacing perspective projection with orthogonal projection; (iii) to expedite scan converting triangles with hidden surface removal.

Just like we can compute perspective projection using classical shear, scissor shear or rotation in 4-dimensions, we can also compute pseudo-perspective using any one of these three transformations. But before we proceed to establish these results, we begin with the following lemma that characterizes all the 4×4 matrices that can be used to compute pseudo-perspective in 3-dimensions. Since these matrices are the same matrices as in Lemma 5.1 (but using a different location for the eye point), the matrices for classical shear, scissor shear, and rotation in 4-dimensions that can be used to compute perspective projections in 3-dimensions can also be used to compute pseudo-perspective in 3-dimensions, since once again these matrices are all special cases of the general 4×4 matrices for pseudo-perspective that appear in Lemma 5.20.

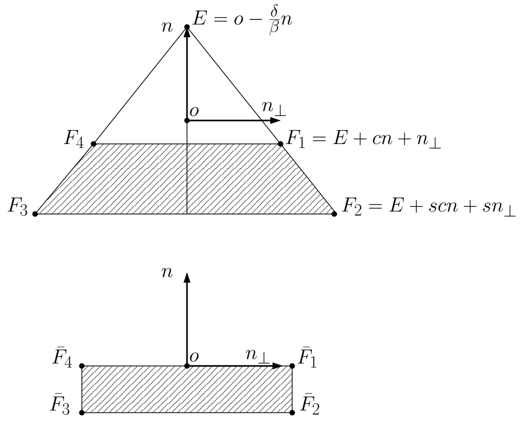


Figure 10: Mapping a viewing frustum to a rectangular box, with $c \neq 0$ and $s \neq 0$

Lemma 5.20 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions and let* $n = (\mathbf{n}, 0)$. *For* $\alpha\delta - \beta\gamma \neq 0$, $\beta \neq 0$, *the* 4×4 *matrix*

$$M = M(n, \alpha, \beta, \gamma, \delta) = \begin{bmatrix} I + (\alpha - 1)(n^T * n) & \beta n^T \\ \gamma n & \delta \end{bmatrix}$$

represents a non-singular linear transformation that maps vectors in the no-plane to vectors in the no-plane and leaves vectors $n_\perp$ *in* R^4 *orthogonal to the no-plane fixed in 4-dimensions. Moreover* $E = o - \frac{\delta}{\beta}n$ *is the unique point in* R^3 *such that* $E * M = \mu n$ *where* μ *is a real number. In 3-dimensions* M *maps a frustum with the eye point at* E *and with two faces orthogonal to* $\mathbf{n}$ *into a rectangular box.*

Proof. By Lemma 5.1, M is a non-singular linear transformation that maps vectors in the *no*-plane to vectors in the *no*-plane and leaves vectors $n_\perp$ in R^4 orthogonal to the *no*-plane fixed in 4-dimensions. Explicitly

$$o * M = \gamma n + \delta o, \quad n * M = \alpha n + \beta o, \quad n_\perp * M = n_\perp.$$

Suppose that E is a point in R^3 such that $E * M = \mu n$, where μ is a real number. Since E is a point in R^3, there is a constant λ and a vector $n_\perp$ such that $E = o + \lambda n + n_\perp$. Therefore by linearity

$$E * M = (o + \lambda n + n_\perp) * M = (\gamma n + \delta o) + \lambda(\alpha n + \beta o) + n_\perp.$$

Since by assumption $E * M = \mu n$, it follows that

$$n_\perp = 0 \quad \text{and} \quad \lambda = -\frac{\delta}{\beta},$$

so $E = o - \frac{\delta}{\beta} n$ and $E * M = \frac{\beta\gamma - \delta\alpha}{\beta} n$. Now consider the frustum in Figure 10.

$$\begin{aligned}
\bar{F}_1 &= F_1 * M = (E + cn + n_\perp) * M \\
&= \frac{\beta\gamma - \delta\alpha}{\beta} n + c(\alpha n + \beta o) + n_\perp \\
&\equiv o + \left(\frac{\beta\gamma - \delta\alpha}{c\beta^2} + \frac{\alpha}{\beta} \right) n + \frac{1}{c\beta} n_\perp \\
\bar{F}_2 &= F_2 * M = (E + scn + sn_\perp) * M \\
&= \frac{\beta\gamma - \delta\alpha}{\beta} n + sc(\alpha n + \beta o) + sn_\perp \\
&\equiv o + \left(\frac{\beta\gamma - \delta\alpha}{sc\beta^2} + \frac{\alpha}{\beta} \right) n + \frac{1}{c\beta} n_\perp
\end{aligned}$$

Similarly,

$$\begin{aligned}
\bar{F}_3 &= F_3 * M = (E + scn - sn_\perp) * M \\
&\equiv o + \left(\frac{\beta\gamma - \delta\alpha}{sc\beta^2} + \frac{\alpha}{\beta} \right) n - \frac{1}{c\beta} n_\perp \\
\bar{F}_4 &= F_4 * M = (E + cn - n_\perp) * M \\
&\equiv o + \left(\frac{\beta\gamma - \delta\alpha}{c\beta^2} + \frac{\alpha}{\beta} \right) n - \frac{1}{c\beta} n_\perp
\end{aligned}$$

Since $\frac{1}{c\,\beta}$ is a constant independent of s, it follows that M maps a frustum with the eye point at E and with two faces orthogonal to n into a rectangular box.

$\diamondsuit$

Remark 5.21 *The matrix $M(n, \alpha, \beta, \gamma, \delta)$ of Lemma 5.20 is identical to the matrix $M(n, \alpha, \beta, \gamma, \delta)$ of Lemma 5.1. Further, the transformations in Sections 4 and 3 are also special cases of the matrix $M = M(n, \alpha, \beta, \gamma, \delta)$ for the right settings of α, β, δ, γ. For example, for $\alpha = 1$, $\beta = 0$, $\gamma = 0$, and $\delta = s$, the matrix M is $I(s)$. The matrix M generalizes all of these transformations because M is the general form of the matrix that maps the no-plane to the no-plane and that leaves vectors in 4-dimensions orthogonal to the no-plane fixed, properties shared by all the transformations in this paper.*

5.2.1 Pseudo-perspective in 3-Dimensions by Classical Shear in 4-Dimensions

Theorem 5.22 *Let* $\mathbf{n}$ *be a unit vector in 3-dimensions, and let $n = (\mathbf{n}, 0)$. For θ not an integer multiple of $90°$, the 4×4 matrix*

$$CShear(o, n, \theta) = \begin{bmatrix} I & \tan(\theta)\mathbf{n}^{\mathbf{T}} \\ 0 & 1 \end{bmatrix}$$

represents a classical shear that shears vectors in the on−plane by the angle θ in the direction n in 4-dimensions. In 3-dimensions, $CShear(o, n, \theta)$ represents pseudo-perspective in the following way: let n be

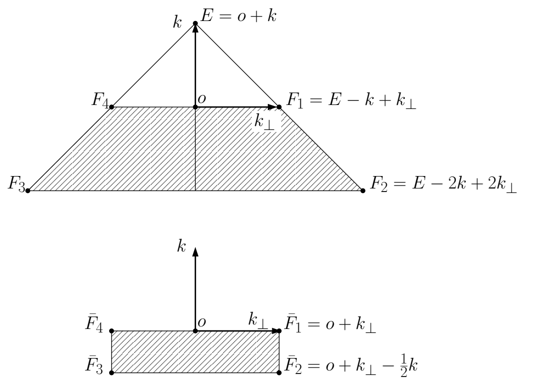


Figure 11: Pseudo-perspective in 3-dimensions by classical shear in 4-dimensions with $n = k$ and $\theta = 135°$ (see Example 5.23)

the unit normal vector to the projection plane and place the eye point at $E = o - \cot(\theta)n$. Then $CShear(o, n, \theta)$ maps the eye from the point E to a vector parallel to the vector n and maps a viewing frustum to a rectangular box.

Proof. By Theorem 5.4, the matrix $CShear(o, n, \theta)$ represents a classical shear that shears vectors in the *on*-plane by the angle θ in the direction n in 4-dimensions. The rest of Theorem 5.22 concerning the action of $CShear(o, n, \theta)$ in 3-dimensions follows from Lemma 5.20 with $\alpha = 1$, $\beta = \tan(\theta)$, $\gamma = 0$, and $\delta = 1$.

Now by linearity,

$$\begin{aligned}
E * CShear(o, n, \theta) &= (o - \cot(\theta)n) \begin{bmatrix} I & \tan(\theta)\mathbf{n}^{\mathbf{T}} \\ 0 & 1 \end{bmatrix} \\
&= o - \cot(\theta)(n + \tan(\theta)o) = -\cot(\theta)n.
\end{aligned}$$

Hence $CShear(o, n, \theta)$ maps the eye from the point E to a vector parallel to the vector n. Since $CShear(o, n, \theta)$ also maps vectors in the *no*-plane to vectors in the *no*-plane, and leaves vectors $n_\perp$ orthogonal to the *no*-plane fixed, by Lemma 5.20, $CShear(o, n, \theta)$ maps a viewing frustum to a rectangular box. Thus $CShear(o, n, \theta)$ represents pseudo-perspective for the eye point located at $E = o - \cot(\theta)n$ in 3-dimensions.

$\diamondsuit$

Example 5.23 *Let $n = (0, 0, 1, 0) = k$ be the unit normal vector to the projection plane. Let $\theta = 135°$ and let $E = o - \cot(135°)k = o + k$ be the eye point. Then the 4×4 matrix*

$$CShear(o, k, 135°) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

is a classical shear in the wz−plane by the angle 135° in the z-direction in 4-dimensions. In 3-dimensions,

$CShear(o,k,135°)$ is a pseudo-perspective that maps the eye from the point E to the vector k,

$$\begin{aligned} E * CShear(o,k,135°) &= (o+k) * CShear(o,k,135°) \\ &= (0,0,1,1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= (0,0,1,0) = k \end{aligned}$$

and maps the viewing frustum $F_1F_2F_3F_4$ to the rectangular box $\bar{F}_1\bar{F}_2\bar{F}_3\bar{F}_4$ in the following way:

$$\begin{array}{lll} F_1 & = E - k + k_\perp & \to o + k_\perp = \bar{F}_1 \\ F_2 & = E - 2k + 2k_\perp & \to o + k_\perp - \frac{1}{2}k = \bar{F}_2 \\ F_3 & = E - 2k - 2k_\perp & \to o - k_\perp - \frac{1}{2}k = \bar{F}_3 \\ F_4 & = E - k - k_\perp & \to o - k_\perp = \bar{F}_4 \end{array}$$

where $k_\perp$ is a vector in the $xy-$plane (Figure 11).

5.2.2 Pseudo-perspective in 3-Dimensions by Scissor Shear in 4-Dimensions

Theorem 5.24 *Let $\mathbf{n}$ be a unit vector in 3-dimensions, and let $n = (\mathbf{n}, 0)$. For $\theta \neq 0$, the 4×4 matrix*

$$SShear(n,o,\theta) = \begin{bmatrix} I + (\cosh(\theta)-1)\mathbf{n}^\mathbf{T}\mathbf{n} & \sinh(\theta)\mathbf{n}^\mathbf{T} \\ \sinh(\theta)\mathbf{n} & \cosh(\theta) \end{bmatrix}$$

represents a scissor shear that shears vectors in the no-plane in 4-dimensions. In 3-dimensions, $SShear(n,o,\theta)$ represents pseudo-perspective in the following way: let n be the unit normal vector to the projection plane and place the eye point at $E = o - \coth(\theta)n$. Then the matrix $SShear(n,o,\theta)$ maps the eye from the point E to a vector parallel to the vector n and maps a viewing frustum to a rectangular box.

Proof. By Theorem 5.9, the matrix $SShear(n,o,\theta)$ represents a scissors shear that shears vectors in the *no*-plane by the angle θ in 4-dimensions. The rest of Theorem 5.24 concerning the action of $SShear(n,o,\theta)$ in 3-dimensions follows from Lemma 5.20 with $\alpha = \cosh(\theta)$, $\beta = \sinh(\theta)$, $\gamma = \sinh(\theta)$, and $\delta = \cosh(\theta)$.

$\diamondsuit$

Example 5.25 *Let $n = (0,0,1,0) = k$ be the unit normal vector to the projection plane. Let $\theta = \text{arcsinh}(1)$ and let $E = o - \coth(\theta)k = o - \sqrt{2}k$ be the eye point. Then the 4×4 matrix*

$$SShear(k,o,\text{arcsinh}(1)) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 1 \\ 0 & 0 & 1 & \sqrt{2} \end{bmatrix}$$

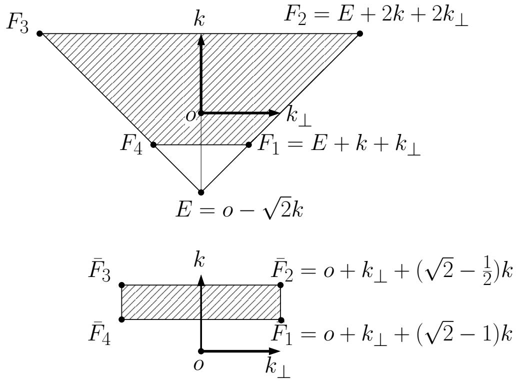

F_3
k
$F_2 = E + 2k + 2k_\perp$
o
$k_\perp$
F_4
$F_1 = E + k + k_\perp$
$E = o - \sqrt{2}k$
$\bar{F}_3$
k
$\bar{F}_2 = o + k_\perp + (\sqrt{2} - \frac{1}{2})k$
$\bar{F}_4$
$\bar{F}_1 = o + k_\perp + (\sqrt{2} - 1)k$
o
$k_\perp$

Figure 12: Pseudo-perspective in 3-dimensions by scissor shear in 4-dimensions with $n = k$ and $\theta = \text{arcsinh}(1)$ (see Example 5.25)

is a scissor shear in the $zw-$plane by the angle $\text{arcsinh}(1)$ in 4-dimensions. In 3-dimensions, $SShear(k,o,\text{arcsinh}(1))$ is a pseudo-perspective that maps the eye from the point E to the vector $-k$:

$$\begin{aligned} E * SShear(k,o,\text{arcsinh}(1)) &= (o - \sqrt{2}k) * SShear(n,o,\text{arcsinh}(1)) \\ &= (0,0,-\sqrt{2},1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 1 \\ 0 & 0 & 1 & \sqrt{2} \end{bmatrix} \\ &= (0,0,-1,0) = -k \end{aligned}$$

and maps the viewing frustum $F_1F_2F_3F_4$ to the rectangular box $\bar{F}_1\bar{F}_2\bar{F}_3\bar{F}_4$ in the following way:

$$\begin{array}{lcll} F_1 & = & E + k + k_\perp & \to o + k_\perp + (\sqrt{2}-1)k = \bar{F}_1 \\ F_2 & = & E + 2k + 2k_\perp & \to o + k_\perp + (\sqrt{2}-\frac{1}{2})k = \bar{F}_2 \\ F_3 & = & E + 2k - 2k_\perp & \to o - k_\perp + (\sqrt{2}-\frac{1}{2})k = \bar{F}_3 \\ F_4 & = & E + k - k_\perp & \to o - k_\perp + (\sqrt{2}-1)k = \bar{F}_4 \end{array},$$

where $k_\perp$ is a vector in the $xy-$plane (Figure 12).

Note that in Example 5.25 the near plane is translated by the pseudo-perspective mapping, while distances within the near plane are not rescaled. If we set $F_1 = E + (\coth(\text{arcsinh}(1))+1)k = E + (\sqrt{2}+1)k = o+k$, then the near plane is invariant under this pseudo-perspective transformation, but distances within the near plane are scaled. Contrast this example with Example 5.23, where the near plane is invariant under the pseudo-perspective transformation and distances within the near plane are not scaled.

5.2.3 Pseudo-perspective in 3-Dimensions by Rotation in 4-Dimensions

Theorem 5.26 *Let $\mathbf{n}$ be a unit vector in 3-dimensions, and let $n = (\mathbf{n}, 0)$. For θ not an integer multiple of $180°$,*

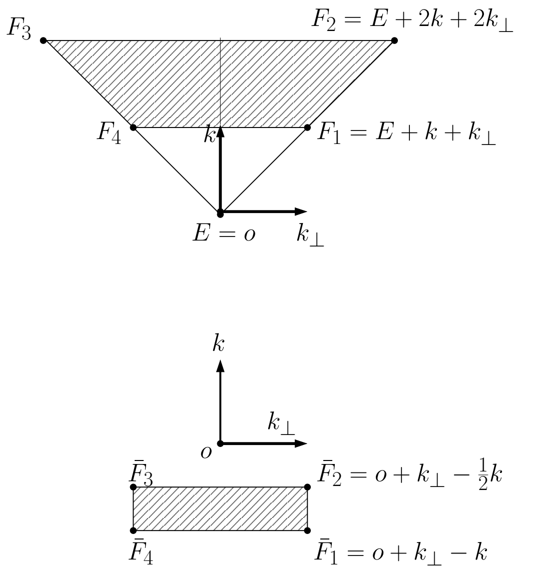

F_3
$F_2 = E + 2k + 2k_\perp$
F_4
k
$F_1 = E + k + k_\perp$
$E = o$
$k_\perp$
k
$k_\perp$
o
$\bar{F}_3$
$\bar{F}_2 = o + k_\perp - \frac{1}{2}k$
$\bar{F}_4$
$\bar{F}_1 = o + k_\perp - k$

Figure 13: Pseudo-perspective in 3-dimensions by rotation in 4-dimensions with $n = k$ and $\theta = 90°$ (see Example 5.27)

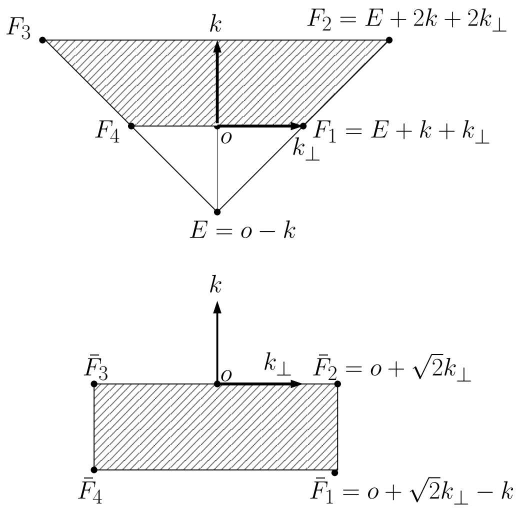


Figure 14: Pseudo-perspective in 3-dimensions by rotation in 4-dimensions with $n = k$ and $\theta = 45°$ (see Example 5.28)

the 4×4 *matrix*

$$Rot(n, o, \theta) = \left[\begin{array}{cc} I + (\cos(\theta) - 1)\mathbf{n^T n} & \sin(\theta)\mathbf{n^T} \\ -\sin(\theta)\mathbf{n} & \cos(\theta) \end{array} \right]$$

represents a rotation that rotates vectors in the no-plane by the angle θ in 4-dimensions. In 3-dimensions, $Rot(n, o, \theta)$ represents pseudo-perspective in the following way: let n be the unit normal vector to the projection plane and place the eye point at $E = o - \cot(\theta)n$. Then $Rot(n, o, \theta)$ maps the eye from the point E to a vector parallel to the vector n and maps a viewing frustum to a rectangular box.

The proof of Theorem 5.26 is analogous to the proof of Theorem 5.24 with hyperbolic functions replaced by trigonometric functions and with the hyperbolic identity $\cosh^2(\theta) - \sinh^2(\theta) = 1$ replaced by the trigonometric identity $\cos^2(\theta) + \sin^2(\theta) = 1$.

Example 5.27 *Let $n = (0, 0, 1, 0) = k$ be the unit normal vector to the projection plane. Let $\theta = 90°$ and let $E = o$ be the eye point. Then the 4×4 matrix*

$$Rot(k, o, 90°) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

is a rotation in the $zw-$plane by the angle $90°$ in 4-dimensions. In 3-dimensions, $Rot(k, o, 90°)$ is a pseudo-perspective that maps the eye from the point E to the vector $-k$

$$\begin{aligned} E * Rot(k, o, 90°) &= o * Rot(k, o, 90°) \\ &= (0, 0, 0, 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \\ &= (0, 0, -1, 0) = -k \end{aligned}$$

and maps the viewing frustum $F_1F_2F_3F_4$ to the rectangular box $\bar{F}_1\bar{F}_2\bar{F}_3\bar{F}_4$ in the following way:

$$\begin{array}{lll} F_1 & = E + k + k_\perp & \to o + k_\perp - k = \bar{F}_1 \\ F_2 & = E + 2k + 2k_\perp & \to o + k_\perp - \frac{1}{2}k = \bar{F}_2 \\ F_3 & = E + 2k - 2k_\perp & \to o - k_\perp - \frac{1}{2}k = \bar{F}_3 \\ F_4 & = E + k - k_\perp & \to o - k_\perp - k = \bar{F}_4 \end{array}$$

where $k_\perp$ is a vector in the $xy-$plane (Figure 13).

For classical shear, scissor shear, and rotation, different values of θ result in different pseudo-perspective transformations. In particular, for arbitrary θ, these pseudo-perspective transformations will scale distances in the near plane. To illustrate this phenomenon, we give a second example of pseudo-perspective with rotation.

Example 5.28 *Let $n = (0, 0, 1, 0) = k$ be the unit normal vector to the projection plane. Let $\theta = 45°$ and let $E = o - \cot(45°)k = o - k$ be the eye point. Then the 4×4 matrix*

$$Rot(k, o, 45°) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

is a rotation in the $zw-$plane by the angle 45° in 4-dimensions. In 3-dimensions, $Rot(k, o, 45^\circ)$ is a pseudo-perspective that maps the eye from the point E to a vector parallel to $-k$

$$\begin{aligned} E * Rot(k, o, 45^\circ) &= (o - k) * Rot(k, o, 45^\circ) \\ &= (0, 0, -1, 1) * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \\ &= (0, 0, -\sqrt{2}, 0) = -\sqrt{2}k \end{aligned}$$

and maps the viewing frustum $F_1F_2F_3F_4$ to the rectangular box $\bar{F}_1\bar{F}_2\bar{F}_3\bar{F}_4$ in the following way:

$$\begin{aligned} F_1 &= E + k + k_\perp & &\rightarrow o + \sqrt{2}k_\perp - k = \bar{F}_1 \\ F_2 &= E + 2k + 2k_\perp & &\rightarrow o + \sqrt{2}k_\perp = \bar{F}_2 \\ F_3 &= E + 2k - 2k_\perp & &\rightarrow o - \sqrt{2}k_\perp = \bar{F}_3 \\ F_4 &= E + k - k_\perp & &\rightarrow o - \sqrt{2}k_\perp - k = \bar{F}_4 \end{aligned}$$

where $k_\perp$ is a vector in the xy-plane (Figure 14).

5.2.4 Reflections on Pseudo-Perspective

We have presented three different techniques for computing pseudo-perspective projections in 3-dimensions: classical shears, scissors shears, or rotations in 4-dimensions. Each of these approaches is the composite of a linear transformation with a stereographic projection, and thus the final transformation is a projective transformation.

In computer graphics, a change of coordinates is usually performed to place the eye point at the origin in a local coordinate frame. While the eye point in some of our methods is not located at the origin, one could either apply a different change of coordinates to map to the canonical eye point used by one of our methods, or translate to and from the canonical eye point as we did in Section 5.1 for perspective projection to relocate the eye point to wherever it is desired.

Also in computer graphics, a viewing angle is used to specify the field of view; with our methods, the field of view is controlled by the length of the vector $n_\perp$, which is perpendicular to the viewing direction n. After mapping the viewing frustum to a box, this box can be scaled and translated to an arbitrary size and location.

Note that in Examples 5.23, 5.25, and 5.27, the choice of θ and the coefficient of $n = k$ leads to a nice scaling of $n_\perp = k_\perp$ (i.e., the dimensions of the near plane of the viewing frustum are equal to the dimensions of the top of the rectangular box to which the viewing frustum is mapped). In general the near plane and the top of the rectangular box will not have the same dimensions; for arbitrary choices of θ and F_1 and F_2, the viewing frustum will not have the same dimensions as the top of the rectangular box—see Example 5.28. Thus, if an arbitrary θ is used, an additional uniform scale will likely be in order when we transform to screen coordinates.

One common approach for pseudo-perspective used in computer graphics is essentially a classical shear; indeed, applying a translation to the matrix given in Example 5.23 ($Trans(k) * CShear(o, k, 135^\circ)$) yields almost the same matrix as the matrix for pseudo-perspective in Hearn and Baker [10] (the difference is due to non-perspective mappings incorporated in the Hearn and Baker matrix, including transforming to device coordinates, integrating the near and far clipping planes into the matrix, and mapping to a non-square window). The approaches to pseudo-perspective using scissor shear or rotation are new and are presented here for the first time.

Finally, at some point, the 3D pseudo-perspective coordinates will need to be converted to 2D coordinates. This conversion can be done using orthographic projection $Ortho(n)$—see Section 4.4—after the pseudo-perspective transformation.

5.3 Quaternions

Rotations in 4-dimensions can be represented by unit quaternions [2, 6]. This observation together with the connection between rotations in 4-dimensions and perspective and pseudo-perspective in 3-dimensions discussed in Section 5.1.3 and Section 5.2.3 allows us to use sandwiching with unit quaternions to represent both perspective and pseudo-perspective.

Let n be a unit vector and consider the unit quaternion

$$q = q(n, \theta) = \cos(\theta/2)o + \sin(\theta/2)n,$$

where the origin o in 3-dimensions now represents the identity for quaternion multiplication in 4-dimensions and the unit vector n is represented as a linear combination of the pure quaternions i, j, k. It is well-known [1, 6] that rotation by the angle θ in the plane perpendicular to n in 3-dimensions is given by the map

$$v \mapsto qvq^* = qvq^{-1},$$

where $q^* = \cos(\theta/2)o - \sin(\theta/2)n = q^{-1}$ is the quaternion conjgate of q, and the multiplication is quaternion multiplication. Moreover it can be shown [6] that rotation by the angle θ in the *on*-plane in 4-dimensions is given by the map

$$p \mapsto qpq,$$

where once again the multiplication is quaternion multiplication. (To get rotation by the angle θ in the *no*-plane, simply replace θ by $-\theta$.) So our insight that orthogonal, perspective and pseudo-perspective projections can be represented by rotations in the *no*-plane in 4-dimensions allows us to use sandwiching with unit quaternions to represent these transformations in 3-dimensions. In particular, Theorem 5.14 for perspective projection and Theroem 5.26 for pseudo-perspective remain valid if we replace the map $p \mapsto p * Rot(n, o, \theta)$ by the map $p \mapsto q(n, -\theta)pq(n, -\theta)$.

Example 5.29 *Similar to Example 5.28, let $n = k$, $\theta = 45°$, and $E = o - k$. Now set $q = q(k, -45°) = \cos(45°/2)o - \sin(45°/2)k$. To show that the map $p \mapsto qpq$ computes the same pseudo-perspective maps as in Example 5.28, let us compute this map on $o, k, k_\perp$, where $k_\perp$ is any vector perpendicular to k. To simplify our notation, let $\tilde{c} = \cos(45°/2)$, $\tilde{s} = \sin(45°/2)$, $c = \cos(45°) = \sqrt{2}/2$, $s = \cos(45°) = \sqrt{2}/2$; then $q = \tilde{c}o - \tilde{s}k$. Now recall that $\tilde{c}^2 - \tilde{s}^2 = c$ and $2\tilde{s}\tilde{c} = s$. Therefore, since $k^2 = -1$ and $ok = k = ko$,*

$$qoq = q^2 = (\tilde{c}^2 - \tilde{s}^2)o - (2\tilde{s}\tilde{c})k = co - sk = \frac{\sqrt{2}}{2}(o - k) \tag{5.1}$$

$$qkq = q^2k = \Big(\frac{\sqrt{2}}{2}(o - k)\Big)k = \frac{\sqrt{2}}{2}(o + k). \tag{5.2}$$

To compute $qk_\perp q$, recall that for any two vectors u, v in R^3, the quaternion product is given by $uv = -(u \cdot v) + u \times v$. But $k_\perp \cdot k = 0$, so $k_\perp k = k_\perp \times k = -k \times k_\perp$. Hence $k_\perp q = q^* k_\perp = q^{-1} k_\perp$. Thus

$$qk_\perp q = qq^{-1}k_\perp = k_\perp. \tag{5.3}$$

Now using Equations 5.1, 5.2, 5.3, a straightforward calculation shows that the transformation $p \mapsto qpq$ maps $E = o - k$ to $-\sqrt{2}k$ and maps F_1, F_2, F_3, F_4 as in Example 5.28 .

6 Conclusion

We have investigated 4×4 matrices that represent simple linear transformations from R^4 to R^4, which map vectors from a plane in R^4 to the same plane in R^4 and leave vectors perpendicular to this plane unchanged. Such matrices give us insight into how to represent affine transformations such as translation, uniform scaling, and reflection in 3-dimensions by simple linear transformations in 4-dimensions. In particular, we showed that the standard 4×4 matrices for computing translations in 3-dimensions represent classical shears in 4-dimensions.

These special 4×4 matrices also allows us to perform perspective projection and pseudo-perspective projection in 3-dimensions by shears or rotations in 4-dimensions. The 3-dimensional projections using scissor shears and rotations in 4-dimensions are new and are presented here for the first time. Moreover, the insight that rotations in 4-dimensions can be used to compute perspective and pseudo-perspective in 3-dimensions allows us to compute these projections using quaternions.

We have implemented and tested in Octave all the transformations discussed in this paper including the quaternions for perspective projections and pseudo-perspective projections, and we have verified that all these transformations work just as we claim.

References

[1] S. Altman. *Rotations, Quaternions, and Double Groups.* Clarendon Press, 1986.

[2] John Conway and Derek Smith. *On Quaternions and Octonions.* AK Peters, 2003.

[3] Juan Du, Ron Goldman, and Stephen Mann. Modeling 3D geometry in $R(4,4)$. *Advances in Applied Clifford Algebras*, to appear.

[4] James D. Foley, Andries van Dam, Stephen K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Welsey, 2nd edition, 1990.

[5] Ron Goldman. *An Integrated Introduction to Computer Graphics and Geometric Modeling.* CRC Press, 2009.

[6] Ron Goldman. *Rethinking Quaternions.* Morgan-Claypool, 2010.

[7] Ron Goldman. Modeling perspective projections in 3-dimensions by rotations in 4-dimensions. *Graphical Models*, 75:41–55, 2013. DOI 10.1016/j.gmod.2012.10.002.

[8] Ron Goldman and Stephen Mann. R(4,4) as a computational framework for 3-dimensional computer graphics. *Advances in Applied Clifford Algebras*, 25(1):113–149, 2015. DOI 10.1007/s00006-014-0480-2.

[9] Ron Goldman, Stephen Mann, and Xiaohong Jia. Computing perspective projections in 3-dimensions using rotors in the homogeneous and conformal models of Clifford algebra. *Advances in Applied Clifford Algebras*, 2014. DOI 10.1007/s00006-014-0439-3.

[10] Donald Hearn, M. Pauline Baker, and Warren Carithers. *Computer Graphics with OpenGL (4th edition).* Pearson, 2011.

[11] J. M. McCarthy. *Introduction to Theoretical Kinematics.* MIT Press, 1990.

[12] David Salomon. *Computer Graphics and Geometric Modeling.* Springer, 1999. DOI 10.1007/978-1-4612-1504-2.