实时渲染 第4版

第4章 Transforms 转换

A transform is an operation that takes entities such as points, vectors, or colors and

converts them in some way. For the computer graphics practitioner, it is extremely

important to master transforms. With them, you can position, reshape, and animate

objects, lights, and cameras. You can also ensure that all computations are carried

out in the same coordinate system, and project objects onto a plane in different

ways. These are only a few of the operations that can be performed with transforms,

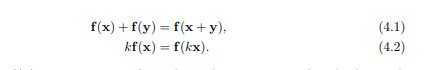
but they are sufficient to demonstrate the importance of the transform’s role in real-

time graphics, or, for that matter, in any kind of computer graphics.

转化是一种操作，将比如点、向量、或者颜色等实体以某种方式转换。对于计算机图形专业人员来讲，精通转换是非常重要的。通过转换，你可以定位、重塑和动画，物体、灯光和摄像机。你也可以确保所有的计算是在同一坐标系统，以不同的方式将物体投影在同一平面。这些知识通过转换执行的一小部分操作，但是他们足够证明转换在实时渲染中重要性，或者，就此而言，任何一种计算机图形。

A linear transform is one that preserves vector addition and scalar multiplication.

Specifically，



线性变换是一种保留向量加法和标量乘法的变换，具体如公式。

As an example, f(x) = 5x is a transform that takes a vector and multiplies each

element by five. To prove that this is linear, the two conditions (Equations 4.1 and

4.2) need to be fulfilled. The first condition holds since any two vectors multiplied by

five and then added will be the same as adding the vectors and then multiplying.

The scalar multiplication condition (Equation 4.2) is clearly fulfilled. This function is

called a scaling transform, as it changes the scale (size) of an object. The rotation

transform is another linear transform that rotates a vector about the origin. Scaling

and rotation transforms, in fact all linear transforms for three-element vectors, can

be represented using a 3 × 3 matrix.

举个例子，f(x) = 5x是一种转换将一个向量每个元素乘以5。为了证明这个线性的，两个条件需要满足。第一个条件成立因为任意两个向量乘以5之后相加与相加后再乘以5是相同的。标量乘法条件（方程4.2）是很容易证明的。这个功能称为缩放转换，由于他改变了一个物体的尺寸。旋转转换是另一种线性转换。缩放和旋转变换，实际上所有的三维向量的线性变换，可以使用3x3矩阵来表示。

However, this size of matrix is usually not large enough. A function for a three

element vector x such as f(x) = x + (7, 3, 2) is not linear. Performing this function

on two separate vectors will add each value of (7, 3, 2) twice to form the result.

Adding a fixed vector to another vector performs a translation, e.g., it moves all

locations by the same amount. This is a useful type of transform, and we would like

to combine various transforms, e.g., scale an object to be half as large, then move it

to a different location. Keeping functions in the simple forms used so far makes it

difficult to easily combine them.

然而，矩阵的大小通常是不是足够大的。一个三维向量x的函数比如：f(x) = x + (7, 3, 2)并不是线性的。在两个独立的向量上执行这个函数将各自加上(7, 3, 2)两次形成结果。在另一个向量上增加一个固定的向量执行了一个平移，比如，他以相同的数量移动了所有的位置。这是常用的一种转换类型，并且我们经常串联各种转换，比如，缩放一个物体一半大小，然后移动到一个不同的位置。以简单形式的保持函数到目前为止很难容易的串联起他们。

Combining linear transforms and translations can be done using an affine transform,

typically stored as a 4 × 4 matrix. An affine transform is one that performs a linear

transform and then a translation. To represent four-element vectors we use

homogeneous notation, denoting points and directions in the same way (using bold

lowercase letters). A direction vector is represented as v = (vx vy vz 0)T and a point

as v = (vx vy vz 1)T . Throughout the chapter, we will make extensive use of the

terminology and operations explained in the downloadable linear algebra appendix,

found on realtimerendering.com.

串联线性变换和平移可以使用一个仿射变换来完成，以4x4拒转表示。一个仿射变换是执行了一次线性变换然后进行平移。为了表示4维向量我们使用齐次符号，用同样的方法表示点和方向（使用粗体小写字母）。方向向量被表示为v= (vx vy vz 0)T以及点表示为v=(vx vy vz 1)T。这一章节自始至终，我们将广泛使用这些术语和操作，可以在realtimerendering.com上下载到有关这方面解释的线性代数。

All translation, rotation, scaling, reflection, and shearing matrices are affine. The

main characteristic of an affine matrix is that it preserves the parallelism of lines, but

not necessarily lengths and angles. An affine transform may also be any sequence of

concatenations of individual affine transforms.

所有的平移、旋转、缩放、投影和剪切矩阵都是仿射的。仿射矩阵最主要的特点是保持了线的平行性，但并不是长度和角度。仿射变换也可以是任意序列的独立仿射变换的串联。

This chapter will begin with the most essential, basic affine transforms. This section

can be seen as a “reference manual” for simple transforms. More specialized

matrices are then described, followed by a discussion and description of quaternions,

a powerful transform tool. Then follows vertex blending and morphing, which are

two simple but effective ways of expressing animations of meshes. Finally, projection

matrices are described. Most of these transforms, their notations, functions, and

properties are summarized in Table 4.1, where an orthogonal matrix is one whose

inverse is the transpose.

这一章将从最基本：基础仿射变换开始讲起。这个部分可以看成简单变换的引用手册。更多专业矩阵将在后面介绍，紧接着将讨论和描述四元数，一种强有力的转换工具。接着是顶点混合和变形，两种简单但是有效方式可以表示网格的动画。最后描述的是投影矩阵。这其中大部分变换，他们的方程式，函数以及属性在表4.1中总结，其中正交矩阵其逆就是他的转置。

Transforms are a basic tool for manipulating geometry. Most graphics application

programming interfaces let the user set arbitrary matrices, and sometimes a library

may be used with matrix operations that implement many of the transforms

discussed in this chapter. However, it is still worthwhile to understand the real

matrices and their interaction behind the function calls. Knowing what the matrix

does after such a function call is a start, but understanding the properties of the

matrix itself will take you further. For example, such an understanding enables you

to discern when you are dealing with an orthogonal matrix, whose inverse is its

transpose, making for faster matrix inversions. Knowledge like this can lead to

accelerated code.

转换是一种基础工具对于操作几何体。大部分图形应用程序接口使我们使用任意矩阵，并且有时可能使用矩阵操作的一个库执行在这章描述的多种转换。然而，对于理解真实矩阵和函数调用背后的矩阵交互是很重要的。了解在一个函数调用后矩阵将做如何操作只是一个开始，理解矩阵自身的属性将会使你走的更远。举个例子，当你在解决正交矩阵时，理解他的逆就是他的转置将比直接求矩阵逆要有更快的速度。像这样的知识可以有助于加速代码。

4.1 Basic Transforms 基础转换

This section describes the most basic transforms, such as translation, rotation,

scaling, shearing, transform concatenation, the rigid-body transform, normal

transform (which is not so normal), and computation of inverses. For the

experienced reader, this can be used as a reference manual for simple transforms,

and for the novice, it can serve as an introduction to the subject. This material is

necessary background for the rest of this chapter and for other chapters in this book.

We start with the simplest of transforms—the translation.

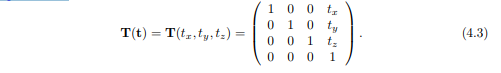
这一部分描述了大部分的基础转换，比如平移、旋转、缩放、裁减，转换串联，刚体转换，法线转换（并不想其他一样普通），以及相反的计算。对于老手，这可以作为简单转换的一个参考，但是对于新手，这可以作为一个主题的介绍。这些材料是章节剩余部分以及本书其他章节的必要基础。我们以最简单的转换——平移开始。

4.1.1 Translation 平移

A change from one location to another is represented by a translation matrix, T. This

matrix translates an entity by a vector t = (tx, ty, tz). T is given below by Equation

4.3:



从一个位置到另一个位置的变化由一个平移矩阵T表示。这个矩阵转换使一个实体被向量t = (tx, ty, tz)变换。T在4.3方程给出。

An example of the effect of the translation transform is shown in Figure 4.1. It is

easily shown that the multiplication of a point p = (px, py, pz, 1) with T(t) yields a

new point p ′ = (px+tx, py +ty, pz +tz, 1), which is clearly a translation. Notice that

a vector v = (vx, vy, vz, 0) is left unaffected by a multiplication by T, because a

direction vector cannot be translated. In contrast, both points and vectors are

affected by the rest of the affine transforms. The inverse of a translation matrix is

T−1 (t) = T(−t), that is, the vector t is negated.

如图4.1所示平移矩阵的作用。很明显的显示出一个点p = (px, py, pz, 1)乘以矩阵T(t)，将会变成一个新的点p ′ = (px+tx, py +ty, pz +tz, 1)，这是一个清晰的变换，注意向量v = (vx, vy, vz, 0)在左侧乘以T，并不会受到影响，因为方向向量不能被平移。对比着，点和向量都能余下的仿射变换所影响。平移矩阵的相反矩阵时T−1 (t) = T(−t)，那是，向量t的反方向。

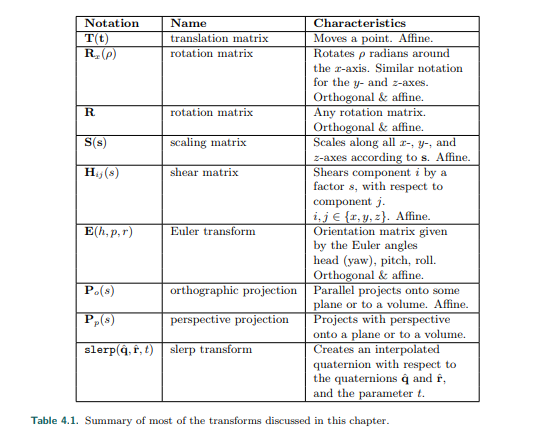


表4.1 本章讨论的大部分转换的总结

We should mention at this point that another valid notational scheme sometimes

seen in computer graphics uses matrices with translation vectors in the bottom row.

For example, DirectX uses this form. In this scheme, the order of matrices would be

reversed, i.e., the order of application would read from left to right. Vectors and

matrices in this notation are said to be in row-major form since the vectors are rows.

In this book, we use column-major form. Whichever is used, this is purely a

notational difference. When the matrix is stored in memory, the last four values of

the sixteen are the three translation values followed by a one.

在这个点上我们已经提到过，另外一种有效的符号形式有时候可以在计算机图形书上看到，转换矩阵的转换向量是在底部行。比如，DirectX就是使用这种形式。在这种形式下，矩阵的顺序是颠倒的，即，应用顺序将从左到右读取。这种形式的向量和矩阵被称为行主要形式，由于向量是行向量。在本书中，我们使用列主要形式。无论使用哪一种，纯粹是符号的差异。当一个矩阵存储在内存中，这16个值得最后4个值是3个位平移值，后面跟着一个1。

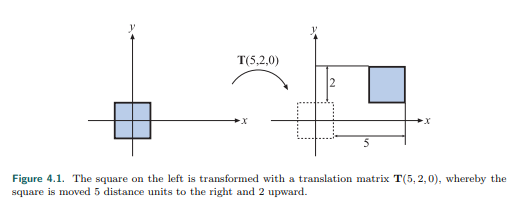


图4.1 左侧的方块被一个平移矩阵（5,2,0）所转换。最终，方块被移动了右侧5个距离以及上方2个距离。

4.1.2 Rotation 旋转

A rotation transform rotates a vector (position or direction) by a given angle around

a given axis passing through the origin. Like a translation matrix, it is a rigid-body

transform, i.e., it preserves the distances between points transformed, and preserves

handedness (i.e., it never causes left and right to swap sides). These two types of

transforms are clearly useful in computer graphics for positioning and orienting

objects. An orientation matrix is a rotation matrix associated with a camera view or

object that defines its orientation in space, i.e., its directions for up and forward.

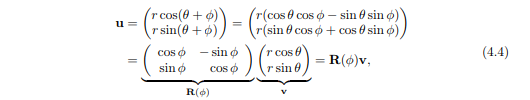
旋转转换可以通过给定一个角度以及给定一个围绕的通过原点的轴旋转一个向量（位置或方向）。像平移矩阵，他是一个刚体转换，即他保留了点转换后的距离，以及保留了利手性（即，他从来不会导致左右两边交换）。这两种转换类型的转换在计算机图形中对于定位和定向对象显然是有用的。方向矩阵时一个旋转矩阵与摄像机视角或者物体联系一起，定义了其在空间中的方向，即它的方向是向上和向前。

In two dimensions, the rotation matrix is simple to derive. Assume that we have a

vector, v = (vx, vy), which we parameterize as v = (vx, vy) = (r cos θ, r sin θ). If we

were to rotate that vector by φ radians (counterclockwise), then we would get u = (r

cos(θ + φ), r sin(θ + φ)). This can be rewritten as



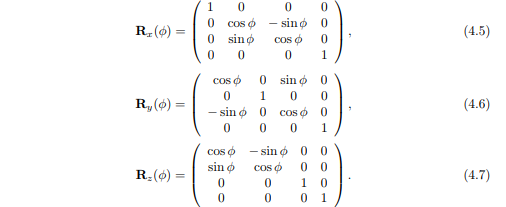
在二维空间，旋转矩阵可以很容易导出。假定我们有一个向量，v = (vx, vy)，我们参数化为v = (vx, vy) = (r cos θ, r sin θ)。如果我们以φ（逆时针）旋转这个向量，我们可以得到u = (r cos(θ + φ), r sin(θ + φ))。这个可以写为下图。

where we used the angle sum relation to expand cos(θ + φ) and sin(θ + φ). In

three dimensions, commonly used rotation matrices are Rx(φ), Ry(φ), and Rz(φ),

which rotate an entity φ radians around the x-, y-, and z-axes, respectively. They are

given by Equations 4.5–4.7:



我们使用角度总和拓展cos(θ + φ) 和 sin(θ + φ)。在三维空间，通常使用旋转矩阵是Rx(φ), Ry(φ), 和 Rz(φ)，将以φ弧度值绕着x-，y-，z-轴旋转一个实体。如方程4.5-4.7所示。

If the bottom row and rightmost column are deleted from a 4 × 4 matrix, a 3 × 3

matrix is obtained. For every 3 × 3 rotation matrix, R, that rotates φ radians around

any axis, the trace (which is the sum of the diagonal elements in a matrix) is

constant independent of the axis, and is computed as [997]:



如果4x4矩阵底部行和最右侧列删除，将获得一个3x3。对于每个3x3旋转矩阵R，绕任意轴旋转φ弧度值，追踪（这是一个矩阵对角线元素之和）是恒定的轴的独立以及计算如下所示[997]。

The effect of a rotation matrix may be seen in Figure 4.4 on page 65. What

characterizes a rotation matrix, Ri(φ), besides the fact that it rotates φ radians

around axis i, is that it leaves all points on the rotation axis, i, unchanged. Note that

R will also be used to denote a rotation matrix around any axis. The axis rotation

matrices given above can be used in a series of three transforms to perform any

arbitrary axis rotation. This procedure is discussed in Section 4.2.1. Performing a

rotation around an arbitrary axis directly is covered in Section 4.2.4.

旋转矩阵的效果如65页图4.4所示。描述了一个旋转矩阵Ri(φ)，实际上，他绕着轴i旋转了φ弧度，除此之外，他离开所有旋转轴i上的点，并没有改变。注意，R也用来表示绕着任意轴的旋转矩阵。上面给出的轴旋转矩阵可以用在一系列三个变换中用于执行任意轴旋转。这个程序将在章节4.2.1节中讨论。第4.2.4章节将直接围绕任意轴执行旋转。

All rotation matrices have a determinant of one and are orthogonal. This also holds

for concatenations of any number of these transforms. There is another way to

obtain the inverse: R−1 i (φ) = Ri(−φ), i.e., rotate in the opposite direction around

the same axis.

所有旋转矩阵的行列式为1并且是正交的。这也适用于任意数量的这些转换的连接。有另一种方式来获得他的逆：R−1 i (φ) = Ri(−φ)，即绕着同样轴旋转相反的方向。

Example: Rotation Around a Point. Assume that we want to rotate an object by φ

radians around the z-axis, with the center of rotation being a certain point, p. What

is the transform? This scenario is depicted in Figure 4.2. Since a rotation around a

point is characterized by the fact that the point itself is unaffected by the rotation,

the transform starts by translating the object so that p coincides with the origin,

which is done with T(−p). Thereafter follows the actual rotation: Rz(φ). Finally, the

object has to be translated back to its original position using T(p). The resulting

transform, X, is then given by



Note the order of the matrices above.

例如：绕着一个点旋转。假设我们要绕着z轴旋转φ弧度，旋转的中心变成一个中心点p。转换是什么。情景如图4.2所示。因为绕着一个点旋转的特征是这个点自身是不受旋转影响的，变换从平移物体开始，使P与原点重合，用T(-p)来表示。此后遵循实际旋转：Rz(φ)。最后，使用T(p)使物体平移回原来位置。由此产生的变换X，如下给出。

注意上面的矩阵顺序。

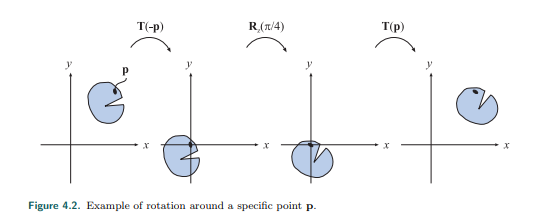


图4.2：绕特定点P旋转的例子。

4.1.3 Scaling 缩放

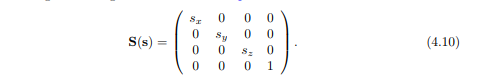
A scaling matrix, S(s) = S(sx, sy, sz), scales an entity with factors sx, sy, and sz

along the x-, y-, and z-directions, respectively. This means that a scaling matrix can

be used to enlarge or diminish an object. The larger the si , i ∈ {x, y, z}, the larger

the scaled entity gets in that direction. Setting any of the components of s to 1

naturally avoids a change in scaling in that direction. Equation 4.10 shows S:



缩放矩阵，S(s) = S(sx, sy, sz)，分别通过因子sx, sy, 和sz沿着x轴，y轴和z轴缩放一个实体。意味着一个缩放矩阵可以用来放大或者缩小物体。Si，i ∈ {x, y, z}越大，在这个方向的实体放大越大。将s的任一分量设置为1，自然可以避免在这个方向上缩放。方程4.10位S。

Figure 4.4 on page 65 illustrates the effect of a scaling matrix. The scaling operation

is called uniform if sx = sy = sz and nonuniform otherwise. Sometimes the terms

isotropic and anisotropic scaling are used instead of uniform and nonuniform. The

inverse is S −1 (s) = S(1/sx, 1/sy, 1/sz).

65页的图4.4说明的缩放矩阵的影响。如果sx = sy = sz缩放操作被称为均匀缩放，否则就是不均匀缩放。有时候用术语各项同性和各项异性缩放来代替均匀和不均匀缩放。矩阵的反方向是S −1 (s) = S(1/sx, 1/sy, 1/sz)。

Using homogeneous coordinates, another valid way to create a uniform scaling matrix is by manipulating matrix element at position (3, 3), i.e., the element at the lower right corner. This value affects the w-component of the homogeneous coordinate, and so scales every coordinate of a point (not direction vectors) transformed by the matrix. For example, to scale uniformly by a factor of 5, the elements at (0, 0), (1, 1), and (2, 2) in the scaling matrix can be set to 5, or the element at (3, 3) can be set to 1/5. The two different matrices for performing this are shown below:

