4.1 基础变换

本节介绍最基本的变换，例如平移、旋转、缩放、剪切、变换级联、刚体变换、法线（normal）变换（不太normal）和逆计算。对于有经验的读者，它可以作为简单变换的参考手册，对于新手，它可以作为对该主题的介绍。这些材料是本章其余部分和本书其他章节的必要背景。我们从最简单的变换开始——平移。

4.1.1 平移

从一个位置到另一个位置的变化由平移矩阵$\mathbf{T}$表示。该矩阵通过向量$\mathbf{t} = (\mathrm{t_x}, \mathrm{t_y}, \mathrm{t_z})$来平移一个实体。 $\mathbf{T}$由下面的公式4.3给出：

$$\mathbf{T}(\mathbf{t}) = \mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

平移变换的效果示例如图4.1所示。很容易证明，点$\mathbf{p} = (p_x, p_y, p_z, 1)$与$\mathbf{T}(\mathbf{t})$相乘产生一个新点$\mathbf{p}\prime = (p_x + t_x, p_y + t_y, p_z + t_z, 1)$，这显然是一个平移操作。请注意，向量$\mathbf{v} = (v_x, v_y, v_z, 0)$不受乘以$\mathbf{T}$的影响，因为方向向量无法平移。相比之下，点和向量都受到其余仿射变换的影响。平移矩阵的逆是$\mathbf{T}^{-1}(\mathbf{t}) = \mathbf{T}(-\mathbf{t})$，即向量$\mathbf{t}$的反。

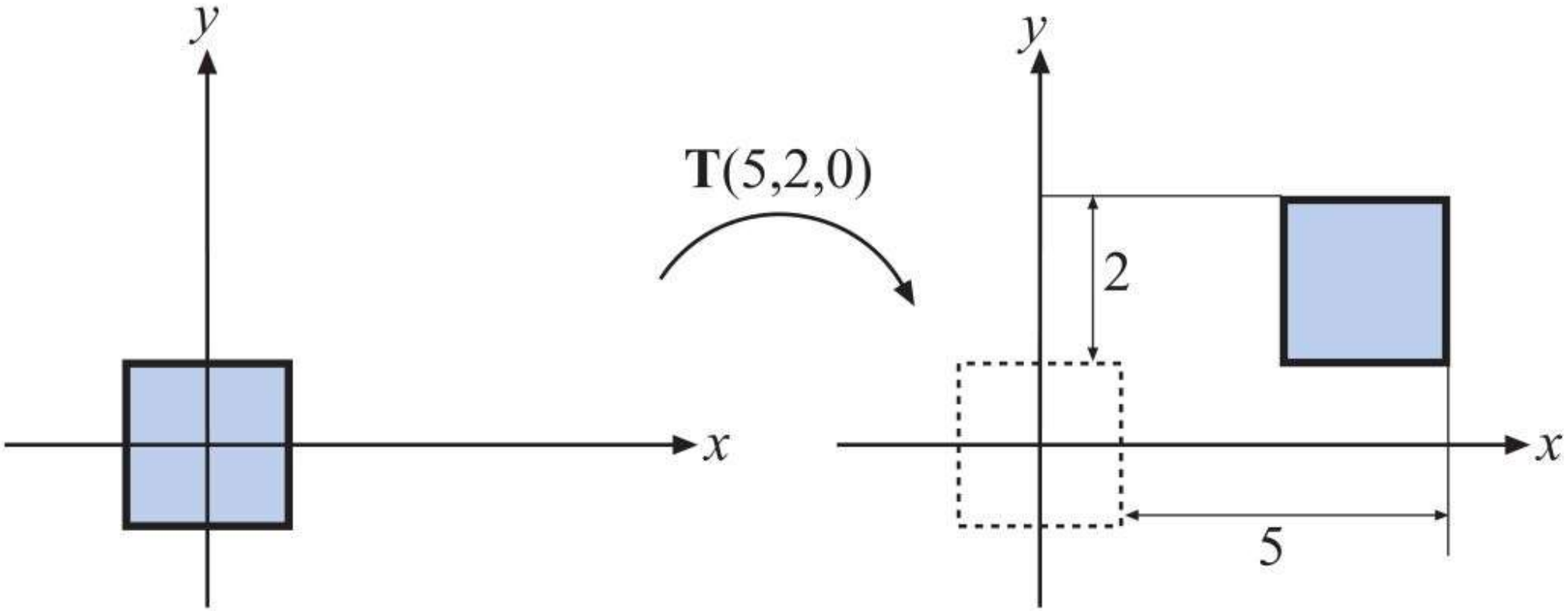

图4.1. 左边的正方形用平移矩阵$\mathbf{T}(5,2,0)$进行变换，由此正方形向右移动5个距离单位，向上移动2个距离单位。

在这一点上我们应该提到，有时在计算机图形中看到的另一种有效的符号方案:使用底行具有平移向量的矩阵。例如，DirectX使用这种形式。在这个方案中，矩阵的顺序将被颠倒，即应用程序的顺序将从左到右读取。这种表示法中的向量和矩阵被称为行优先形式，因为向量是行。在本书中，我们使用列优先形式。无论使用哪种方式，这纯粹是符号上的差异。当矩阵存储在内存中时，十六进制的最后四个值是三个平移值，后跟一个1。

4.1.2 旋转

旋转变换将向量（位置或方向）围绕通过原点的给定轴旋转给定角度。像平移矩阵一样，它是一个刚体变换，即它保留了变换点之间的距离，并保留了偏手性（即，它永远不会导致左右交换边）。这两种类型的变换在计算机图形学中对于定位和定向对象显然很有用。方向矩阵是与相机视图或对象相关联的旋转矩阵，它定义了它在空间中的方向，即它的向上和向前的方向。

在二维中，旋转矩阵很容易推导。假设我们有一个向量$\mathbf{v} = (v_x, v_y)$，我们将其参数化为$\mathbf{v} = (v_x, v_y) = (rcos\theta, rsin\theta)$。如果我们将该向量旋转$\phi$弧度（逆时针），那么我们将得到$\mathbf{u} = (rcos(\theta + \phi), rsin(\theta + \phi))$。这可以重写为:

$$\begin{aligned}\mathbf{u} &= \begin{pmatrix} rcos(\theta + \phi) \\ rsin(\theta + \phi) \end{pmatrix} = \begin{pmatrix} r(cos\theta cos\phi - sin\theta sin\phi) \\ r(sin\theta cos\phi + cos\theta sin\phi) \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix}}_{\mathbf{R}(\phi)} \underbrace{\begin{pmatrix} rcos\theta \\ rsin\theta \end{pmatrix}}_{\mathbf{v}} = \mathbf{R}(\phi)\mathbf{v} \end{aligned} \tag{4.4}$$

其中我们使用角度和关系来扩展$cos(\theta + \phi)$和$sin(\theta + \phi)$。在三个维度上，常用的旋转矩阵有$\mathbf{R}_x(\phi)$、$\mathbf{R}_y(\phi)$和$\mathbf{R}_z(\phi)$，它们分别围绕x轴、y轴和z轴旋转一个实体ϕ弧度。它们由公式4.5–4.7给出：

$$\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\phi & -sin\phi & 0 \\ 0 & sin\phi & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

$$\mathbf{R}_y(\phi) = \begin{pmatrix} cos\phi & 0 & sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -sin\phi & 0 & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

$$\mathbf{R}_z(\phi) = \begin{pmatrix} cos\phi & -sin\phi & 0 & 0 \\ sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

如果从4×4矩阵中删除最底行和最右列，则得到3×3矩阵。对于每个3×3旋转矩阵$\mathbf{R}$，它围绕任何轴旋转ϕ弧度，其迹（即矩阵中对角线元素的总和）是独立于轴的常数，并计算为[997]：

$$tr(\mathbf{R}) = 1 + 2cos\phi \tag{4.8}$$

旋转矩阵的效果可以在第65页的图4.4中看到。旋转矩阵$\mathbf{R}_i(\phi)$的特征除了它绕轴i旋转ϕ弧度这一事实之外，它还使所有留在旋转轴i上的点不变。请注意，$\mathbf{R}$也将用于表示围绕任何轴旋转的旋转矩阵。上面给出的轴旋转矩阵可用于一系列三个变换以执行任意轴旋转。此过程在第4.2.1节中讨论。4.2.4节介绍了直接绕任意轴旋转。

所有旋转矩阵的行列式都是1并且是正交的。这也适用于任意数量的这些变换的级联。旋转矩阵还有另一种求逆的方法：$\mathbf{R}_i^{-1}(\phi) = \mathbf{R}_i(-\phi)$，即绕同一轴向相反方向旋转。

示例：围绕一个点旋转。假设我们要围绕z轴将对象旋转ϕ弧度，旋转中心是某个点$\mathbf{p}$。这个变换是什么？图4.2描述了这种情况。由于围绕点的旋转的特性在于点本身不受旋转的影响，因此变换从平移对象开始，使$\mathbf{p}$与原点重合，这是通过$\mathbf{T}(-\mathbf{p})$完成的。此后跟随实际旋转：$\mathbf{R}_z(\phi)$。最后，必须使用$\mathbf{T}(\mathbf{p})$将对象平移回其原始位置。得到的变换$\mathbf{X}$ 由下面式子给出：

$$\mathbf{X} = \mathbf{T}(\mathbf{p})\mathbf{R}_z(\phi)\mathbf{T}(-\mathbf{p}) \tag{4.9}$$

注意上面矩阵的顺序。

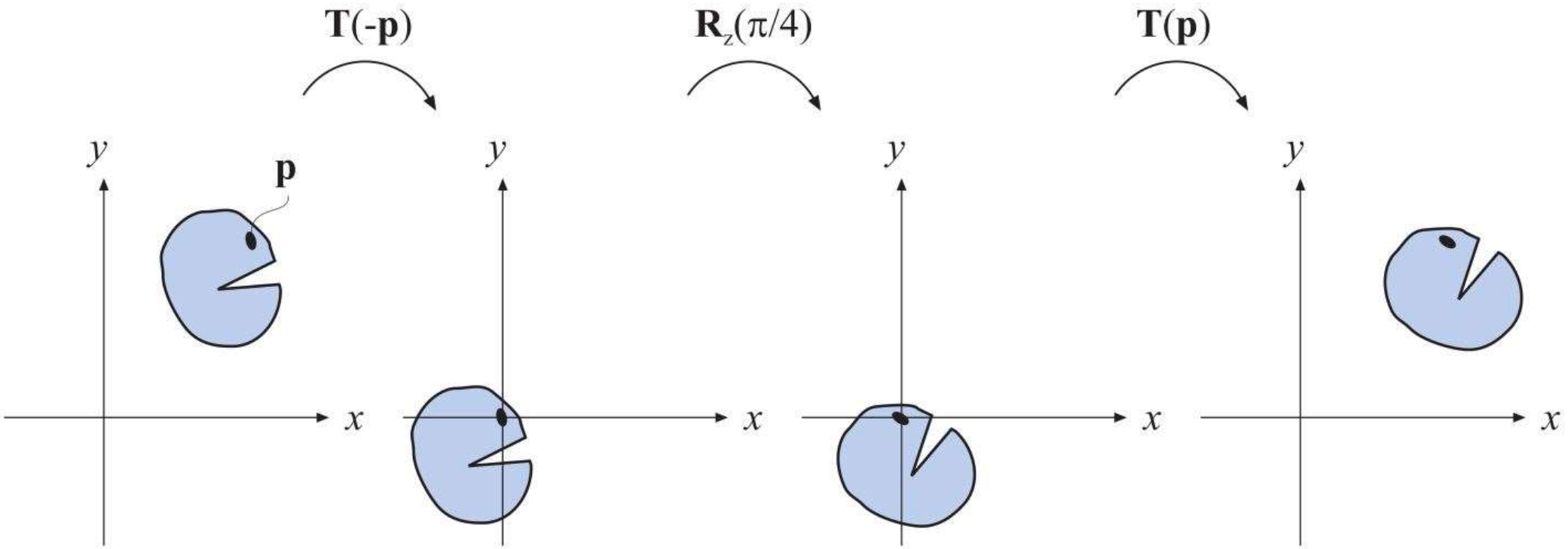

图4.2. 围绕特定点p旋转的示例。

4.1.3 缩放

缩放矩阵$\mathbf{S}(\mathbf{s}) = \mathbf{S}(s_x, s_y, s_z)$分别沿x、y和z方向使用因子$s_x$、$s_y$和$s_z$缩放实体。这意味着缩放矩阵可用于放大或缩小对象。其中的$s_i, i \in x, y, z$越大，缩放的实体在该方向上就越大。 将$\mathbf{s}$的任何分量设置为1自然会避免在该方向上缩放的变化。公式4.10显示了$\mathbf{S}$:

$$\mathbf{S}(\mathbf{s}) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.10}$$

第65页的图4.4说明了缩放矩阵的效果。如果$s_x = s_y = s_z$，则缩放操作称为均匀缩放，否则称为不均匀缩放。有时使用术语各向同性和各向异性缩放代替均匀和非均匀。其逆为$\mathbf{S}^{-1}(\mathbf{s}) = S(1/s_x, 1/s_y, 1/s_z)$。

使用齐次坐标，另一种创建均匀缩放矩阵的有效方法是操作位置$(3,3)$处的矩阵元素，即右下角的元素。该值影响齐次坐标的w分量，因此缩放由矩阵变换的点（不是方向向量）的每个坐标。例如，要均匀缩放5倍，缩放矩阵中$(0,0)$、$(1,1)$和$(2,2)$处的元素可以设置为5，或者$(3,3)$可以设置为1/5。执行此操作的两个不同矩阵如下所示：

$$\mathbf{S} = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathbf{S}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/5 \end{pmatrix} \tag{4.11}$$

与使用$\mathbf{S}$进行均匀缩放相反，使用$\mathbf{S}'$必须始终遵循齐次性。这可能是低效的，因为它涉及齐次化过程中的除法；如果右下角（位置$(3,3)$）的元素为1，则不需要除法。当然，如果系统总是做这个除法而不测试为1，那么就没有额外的成本。

$\mathbf{s}$的一个或三个分量的负值给出了一种反射矩阵，也称为镜像矩阵。如果只有两个比例因子是-1，那么我们将旋转π弧度。需要说明的是，与反射矩阵级联的旋转矩阵也是反射矩阵。因此，以下是一个反射矩阵：

$$\underbrace{\begin{pmatrix} cos(\pi/2) & sin(\pi/2) \\ -sin(\pi/2) & cos(\pi/2) \end{pmatrix}}_{\text{rotation}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}}_{\text{reflection}} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \tag{4.12}$$

反射矩阵在检测时通常需要特殊处理。例如，顶点按逆时针顺序排列的三角形在通过反射矩阵变换时将得到顺时针顺序。这种顺序更改可能会导致不正确的照明和背面剔除发生。要检测给定矩阵是否以某种方式反射，请计算矩阵左上角3×3元素的行列式。如果值为负，则矩阵是反射的。例如，方程4.12中矩阵的行列式是$0 \cdot 0 - (-1) \cdot (-1) = -1$。

示例：在某个方向上缩放。缩放矩阵$\mathbf{S}$仅沿x、y和z轴缩放。如果要在其他方向进行缩放，则需要进行复合变换。假设应该沿着正规化的、右向坐标系下的$\mathbf{f}^x$、$\mathbf{f}^y$ 和$\mathbf{f}^z$的轴进行缩放。首先构造矩阵F，改变基，如下所示：

$$\mathbf{F} = \begin{pmatrix} \mathbf{f}^x & \mathbf{f}^y & \mathbf{f}^z & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.13}$$

思路是让三个轴给定的坐标系与标准轴重合，然后使用标准缩放矩阵，再变换回来。第一步是乘以转置，即$\mathbf{F}$的逆。然后进行实际的缩放，跟着进行反向变换。变换如公式4.14所示：

$$\mathbf{X} = \mathbf{FS(s)F}^T \tag{4.14}$$

4.1.4 剪切

另一类变换是剪切矩阵的集合。例如，这些可以在游戏中用于扭曲整个场景，以产生迷幻效果；或以其他方式扭曲模型的外观。有六个基本剪切矩阵，它们表示为$\mathbf{H}_{xy}(s)$、$\mathbf{H}_{xz}(s)$、$\mathbf{H}_{yx}(s)$、$\mathbf{H}_{yz}(s)$、$\mathbf{H}_{zx}(s)$和$\mathbf{H}_{zy}(s)$。第一个下标用于表示剪切矩阵正在改变哪个坐标，而第二个下标表示进行剪切的坐标。剪切矩阵$\mathbf{H}_{xz}(s)$的示例如公式4.15所示。观察下标可以用来求参数s在下面矩阵中的位置；x(其数字索引为0)标识第0行，z(其数字索引为2)标识第二列，因此s位置如下所示：

$$\mathbf{H}_{xz}(s) = \begin{pmatrix} 1 & 0 & s & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.15}$$

将此矩阵与点$\mathbf{p}$相乘的效果是产生一个点：$(p_x + sp_z, p_y, p_z)T$ 。图形上，这在图4.3中显示为单位正方形。$\mathbf{H}_{ij}(s)$(相对于第j个坐标剪切第i个坐标，其中$i \neq j$)的逆是通过反向剪切产生的，即$\mathbf{H}_{ij}^{-1}(s) = \mathbf{H}_{ij}(-s)$。

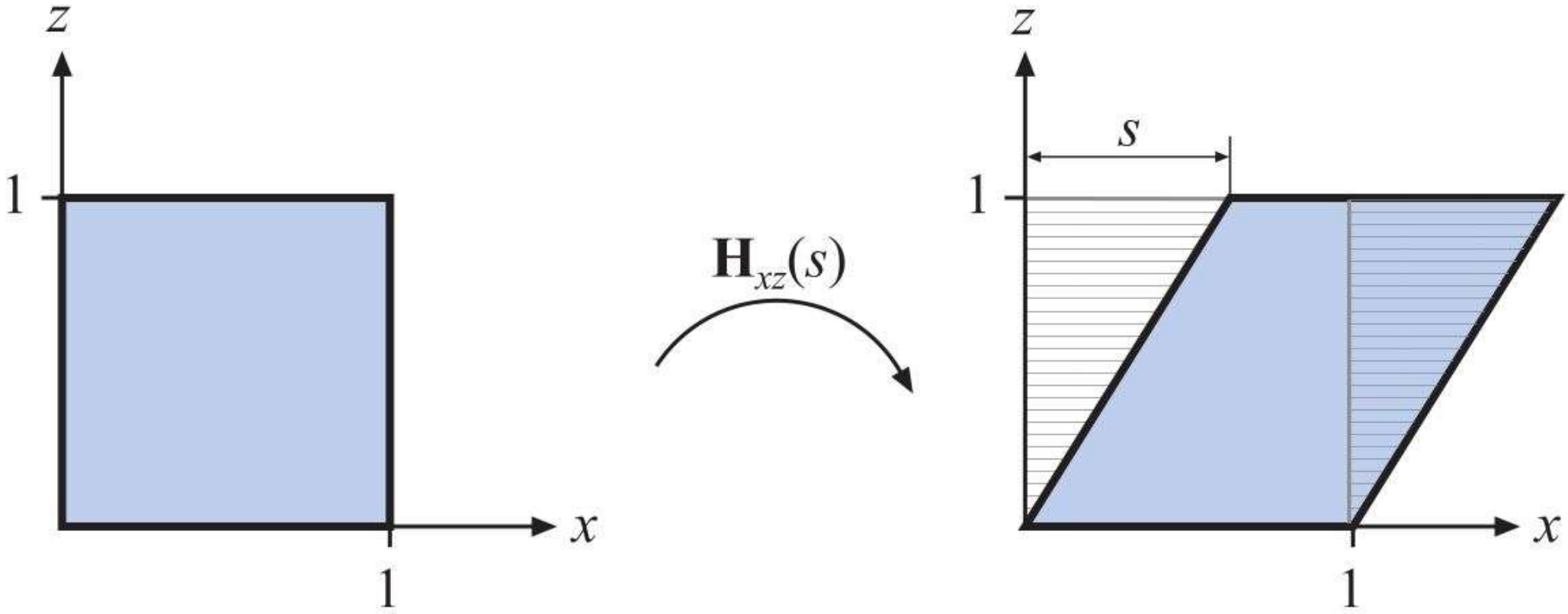

图4.3. 用$\mathbf{H}_{xz}(s)$剪切单位正方形的效果。y值和z值都不受变换的影响，而x值是旧x值和s乘以z值的总和，从而导致正方形倾斜。这种变换是保面积的，可以看出虚线区域是相同的。

你还可以使用稍微不同的剪切矩阵：

$$\mathbf{H}'_{xy}(s,t) = \begin{pmatrix} 1 & 0 & s & 1 \\ 0 & 1 & t & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.16}$$

然而，在这里，两个下标都用于表示这些坐标将被第三个坐标剪切。这两种不同类型的描述之间的联系是$\mathbf{H}'_{ij}(s,t) = \mathbf{H}_{ik}(s)\mathbf{H}_{jk}(t)$，其中$k$用作第三坐标的索引。使用正确的矩阵是一个品味问题。最后，应该注意的是，由于任何剪切矩阵的行列式$|H| = 1$，这是一个保体积的变换，如图4.3所

示。

4.1.5 变换的级联

由于矩阵乘法运算的不可交换性，矩阵出现的顺序很重要。因此，变换的级联被认为是顺序相关的。

作为顺序相关性的示例，请考虑两个矩阵$\mathbf{S}$和$\mathbf{R}$。$\mathbf{S}(2, 0.5, 1)$将x分量按因子2缩放，将y分量按因子0.5缩放。$\mathbf{R}_z(\pi/6)$绕z轴（在右手坐标系中，从本书的页面向外指向）逆时针旋转$\pi/6$弧度。这些矩阵可以通过两种方式相乘，结果完全不同。这两种情况如图4.4所示。

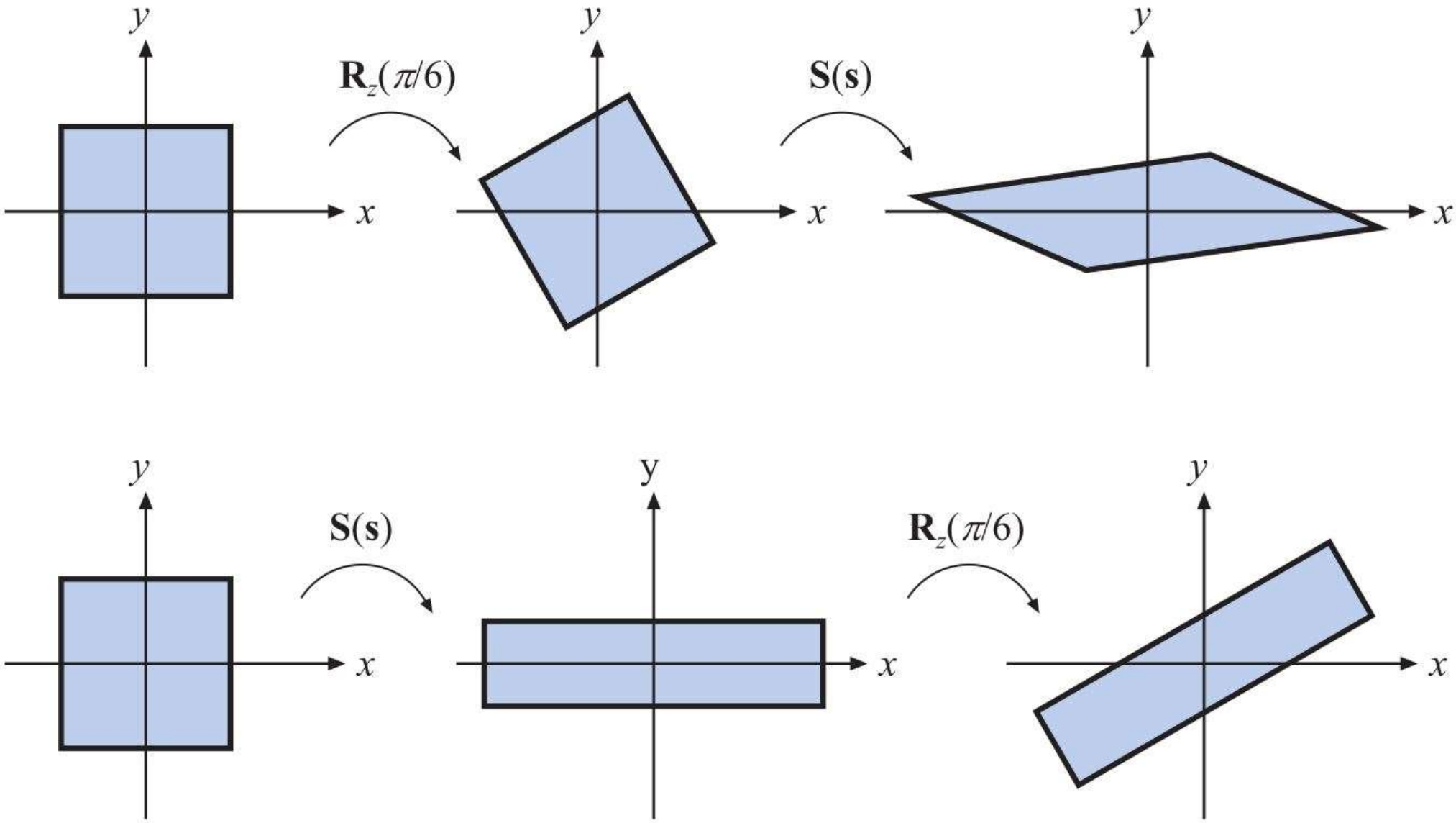

图4.4. 这说明了矩阵相乘时的顺序依赖性。在顶行，应用旋转矩阵$\mathbf{R}_z(\pi/6)$，然后进行缩放，$\mathbf{S}(\mathbf{s})$，其中$\mathbf{s} = (2, 0.5, 1)$。复合矩阵则为$\mathbf{S}(\mathbf{s})\mathbf{R}_z(\pi/6)$。在底行，矩阵以相反的顺序应用，产生$\mathbf{R}_z(\pi/6)\mathbf{S}(\mathbf{s})$。结果明显不同。对于任意矩阵$M$和$N$，通常认为$MN \neq NM$。

将一系列矩阵连接成一个矩阵的明显原因是为了提高效率。例如，假设你有一个具有数百万个顶点的游戏场景，并且场景中的所有对象都必须进行缩放、旋转和最终平移。现在，不是将所有顶点与三个矩阵中的每一个相乘，而是将三个矩阵连接成一个矩阵。然后将此单个矩阵应用于顶点。这个复合矩阵是$\mathbf{C} = \mathbf{TRS}$。注意这里的顺序。缩放矩阵$\mathbf{S}$应首先应用于顶点，因此出现在合成中的右侧。这种排序意味着$\mathbf{TRSp} = (\mathbf{T}(\mathbf{R}(\mathbf{Sp})))$，其中$\mathbf{p}$是要转换的点。顺便说一下，$\mathbf{TRS}$是场景图系统常用的顺序。

值得注意的是，虽然矩阵级联是顺序相关的，但矩阵可以根据需要进行分组。例如，假设你希望使用$\mathbf{TRSp}$计算一次刚体运动变换$\mathbf{TR}$。将这两个矩阵组合在一起，$(\mathbf{TR})(\mathbf{Sp})$，并替换为中间结果是有效的。因此，矩阵级联满足结合律。

4.1.6 刚体变换

当一个人抓住一个固体物体，比如从桌子上拿一支笔，把它移到另一个位置，也许是衬衫口袋，只有物体的方向和位置发生了变化，而物体的形状通常不受影响。这种仅由平移和旋转级联组成的变换称为刚体变换。它具有保留长度、角度和偏手性的特性。

任何刚体矩阵$\mathbf{X}$都可以写成平移矩阵$\mathbf{T}(\mathbf{t})$和旋转矩阵$\mathbf{R}$的串联。因此，$\mathbf{X}$具有方程4.17中矩阵的外观：

$$\mathbf{X} = \mathbf{T}(\mathbf{t})\mathbf{R} = \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.17}$$

$\mathbf{X}$的逆计算为$\mathbf{X}^{-1} = (\mathbf{T}(\mathbf{t})\mathbf{R})^{-1} = \mathbf{R}^{-1}\mathbf{T}(\mathbf{t})^{-1} = \mathbf{R}^T\mathbf{T}(-\mathbf{t})$。因此，要计算逆，左上角3×3$\mathbf{R}$的矩阵被转置，T的平移值改变符号。这两个新矩阵以相反的顺序相乘以获得逆矩阵。计算$\mathbf{X}$的逆的另一种方法是在以下符号中考虑$\mathbf{R}$（使$\mathbf{R}$显示为3×3矩阵）和$\mathbf{X}$（第6页上的符号用公式1.2描述）：

$$\overline{\mathbf{R}} = (\mathbf{r}_{,0}\ \mathbf{r}_{,1}\ \mathbf{r}_{,2}) = \begin{pmatrix} \mathbf{r}_{0,}^T \\ \mathbf{r}_{1,}^T \\ \mathbf{r}_{2,}^T \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} \overline{\overline{\mathbf{R}}} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{4.18}$$

其中$\mathbf{r}_{,0}$表示旋转矩阵的第一列（即，逗号表示0到2之间的任何值，而第二个下标为0），而$\mathbf{r}_{0,}^T$是列矩阵的第一行。请注意，$\mathbf{0}$是一个填充了零的3×1列向量。一些计算得出公式4.19所示表达式的逆：

$$\mathbf{X}^{-1} = \begin{pmatrix} \mathbf{r}_{0,} & \mathbf{r}_{1,} & \mathbf{r}_{2,} & -\overline{\mathbf{R}}^T\mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.19}$$

示例：定向相机。图形中的一个常见任务是调整相机的方向，使其看向某个位置。在这里，我们将介绍gluLookAt()（来自OpenGL实用程序库，简称GLU）的作用。尽管现在这个函数调用本身并不常用，但这个任务仍然很常见。假设相机位于$\mathbf{c}$处，我们希望相机观察目标$\mathbf{l}$，并且相机的给定方向是$\mathbf{u}\prime$，如图4.5所示。我们要计算由三个向量$\{\mathbf{r}, \mathbf{u}, \mathbf{v}\}$组成的基。我们首先将观察向量计算为$\mathbf{v} = (\mathbf{c} - \mathbf{l})/||\mathbf{c} - \mathbf{l}||$，即从目标到相机位置的归一化向量。向右看的向量可以计算为$\mathbf{r} = -(\mathbf{v} \times \mathbf{u}\prime)/||\mathbf{v} \times \mathbf{u}\prime||$。$\mathbf{u}\prime$ 向量通常不能保证准确向上，因此最终向上向量是另一个叉积，$\mathbf{u} = \mathbf{v} \times \mathbf{r}$，它保证被归一化，因为$\mathbf{v}$和$\mathbf{r}$都被归一化并且通过构造垂直。在我们将构建的相机变换矩阵$\mathbf{M}$中，其想法是首先平移所有内容，使相机位置位于原点$(0, 0, 0)$，然后更改基，使$\mathbf{r}$与$(1, 0, 0)$对齐，$\mathbf{u}$与$(0, 1, 0)$对齐，$\mathbf{v}$与$(0, 0, 1)$对齐。如下公式所示：

$$\mathbf{M} = \underbrace{\begin{pmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{change of basis}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{translation}} = \begin{pmatrix} r_x & r_y & r_z & -\mathbf{t} \cdot \mathbf{r} \\ u_x & u_y & u_z & -\mathbf{t} \cdot \mathbf{u} \\ v_x & v_y & v_z & -\mathbf{t} \cdot \mathbf{v} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.20}$$

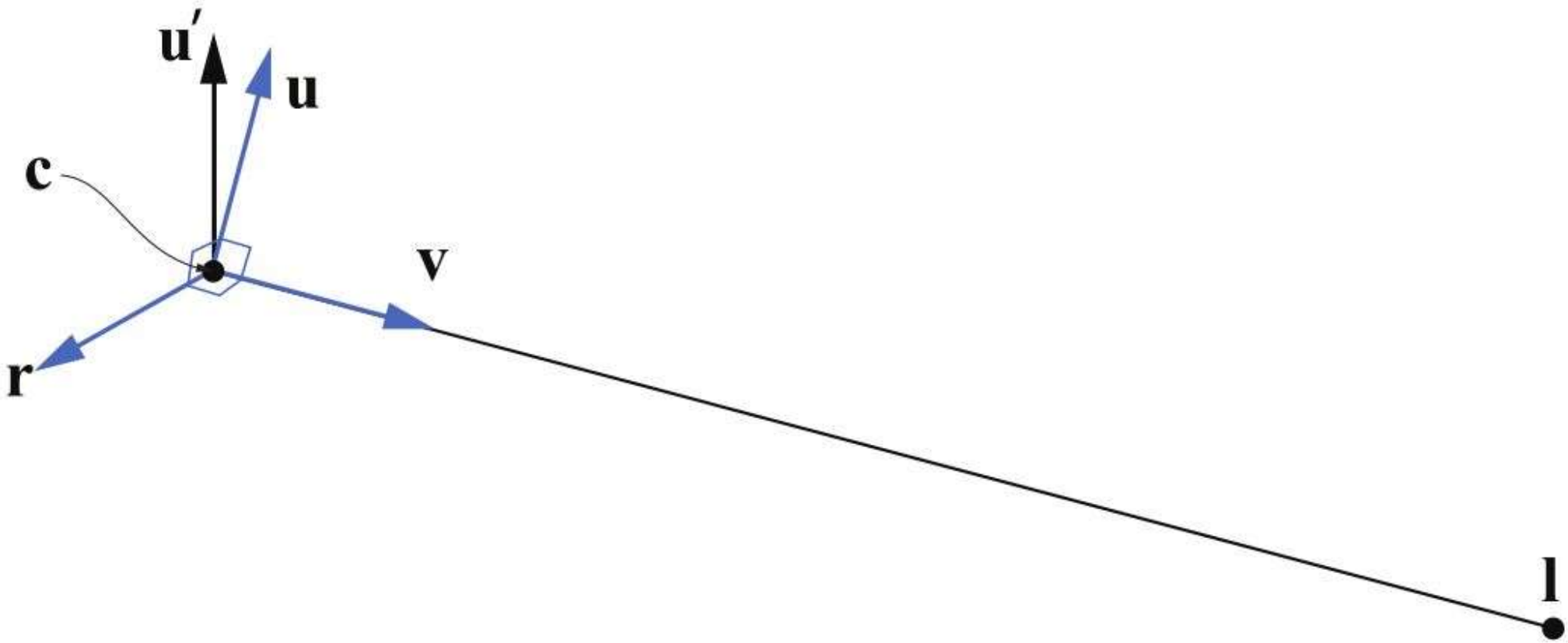

图4.5. 计算变换的几何图形，该变换将相机定向在$\mathbf{c}$处，向上向量$\mathbf{u}'$，观察点$\mathbf{l}$。为此，我们需要计算$\mathbf{r}$、$\mathbf{u}$和$\mathbf{v}$。

请注意，当将平移矩阵与基矩阵的变化级联起来时，平移$-\mathbf{t}$在右边，因为它应该首先应用。记住将$\mathbf{r}$、$\mathbf{u}$和$\mathbf{v}$的分量放在哪里的一种方法如下。我们想让r变成$(1,0,0)$，所以当基矩阵的变化乘以$(1,0,0)$时，我们可以看到矩阵的第一行一定是$\mathbf{r}$的元素，因为$\mathbf{r}\cdot\mathbf{r}=1$。此外，第二行和第三行必须由垂直于$\mathbf{r}$的向量组成，即$\mathbf{r}\cdot\mathbf{x}=0$。当对$\mathbf{u}$和$\mathbf{v}$也应用相同的想法时，我们得出基矩阵的变化如上。

4.1.7 法向量变换

单个矩阵可用于一致地变换点、线、三角形和其他几何图形。相同的矩阵也可以变换沿着这些线或三角形表面上的切向量。然而，这个矩阵不能总是用于变换一个重要的几何属性，即表面法线（和顶点照明法线）。图4.6显示了如果使用相同的矩阵会发生什么。

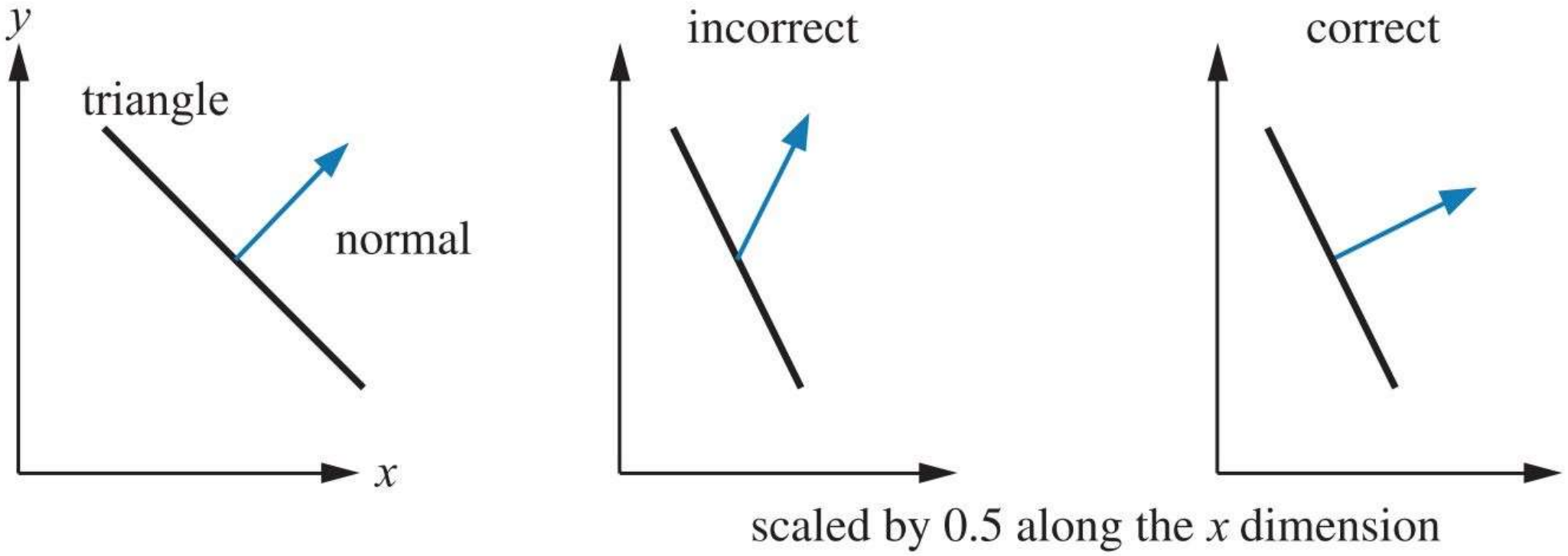

图4.6. 左边是原始几何图形，一个三角形及其从侧面显示的法线。中间的插图显示了如果模型沿x轴缩放0.5并且法线使用相同的矩阵会发生什么。右图显示了法线的正确变换。

正确的方法是使用矩阵的伴随[227]的转置，而不是乘以矩阵本身。伴随式的计算在我们的在线线性代数附录中进行了描述。伴随总是保证存在。法线在转换后不能保证是单位长度，因此通常需要进行归一化。

转换法线的传统答案是计算逆的转置[1794]。这种方法通常有效。然而，完整的逆不是必需的，并且有时无法创建。逆是伴随矩阵除以原始矩阵的行列式。如果该行列式为零，则矩阵为奇异矩阵，逆矩阵不存在。

即使只计算一个完整的4×4矩阵的伴随矩阵，其代价也可能很昂贵，而且通常没有必要。由于法线是一个向量，平移不会影响它。此外，大多数建模变换都是仿射的。它们不会改变传入的齐次坐标的w分量，即它们不执行投影。在这些（常见）情况下，正常变换所需的只是计算左上角3×3分量的伴随。

通常甚至不需要这种伴随计算。假设我们知道变换矩阵完全由平移、旋转和均匀缩放操作（没有拉伸或挤压）的级联组成。平移不会影响法线。均匀缩放只会改变法线的长度。剩下的是一系列旋转，它总是产生某种顺序的旋转组合，仅此而已。逆的转置可用于变换法线。旋转矩阵的定义是它的转置是它的逆矩阵。代入法线变换，两个转置（或两个逆）给出原始旋转矩阵。综上所述，在这些情况下，原始变换本身也可以直接用于变换法线。

最后，完全重新规范化产生的法线并不总是必要的。如果仅将平移和旋转级联在一起，则法线在矩阵转换时不会改变长度，因此不需要重新归一化。如果还级联了均匀缩放，则可以使用整体比例因子（假设已知或者参看第4.2.3节）直接对生成的法线进行归一化。例如，如果我们知道应用了一系列缩放使对象变大5.2倍，那么由该矩阵直接变换的法线将通过除以5.2重新归一化。或者，要创建一个可以产生归一化结果的正常变换矩阵，可以将原始矩阵的3×3左上角除以这个比例因子一次。

请注意，在变换后，表面法线从三角形导出的系统中，法线变换不是问题（例如，使用三角形边线的叉积）。切线向量本质上不同于法线，并且总是由原始矩阵直接变换。

4.1.8 逆计算

许多情况下都需要逆，例如，在坐标系之间来回更改时。根据有关变换的可用信息，可以使用以下三种计算矩阵逆的方法之一：

- 如果矩阵是单个变换或具有给定参数的简单变换序列，则可以通过“反转参数”和矩阵顺序轻松计算矩阵。例如，如果$\mathbf{M} = \mathbf{T}(\mathbf{t})\mathbf{R}(\phi)$，则$\mathbf{M}^{-1} = \mathbf{R}(-\phi)\mathbf{T}(-\mathbf{t})$。这很简单，并保持了变换的准确性，这在渲染巨大世界时很重要[1381]。
- 如果已知矩阵是正交的，则$\mathbf{M}^{-1} = \mathbf{M}^T$，即转置是逆矩阵。任何旋转的序列都是旋转，因此是正交的。
- 如果什么都不知道，则可以使用伴随方法、克莱姆法则、LU分解或高斯消元来计算逆。克莱姆法则和伴随方法通常更可取，因为它们的分支操作较少；在现代架构上避免“if”测试是很好的。有关如何使用伴随来反转变换法线，请参见第4.1.7节。

优化时也可以考虑逆向计算的目的。例如，如果逆是用于变换向量，那么通常只需要在矩阵的3×3左上部分（见上一节）求逆。