4.0 变换

> 要是愤怒的航船改变了方向
> 围绕着你沉睡的脑袋，和身体
> 那就永远不必去害怕
> 穷苦世界的抽象风暴之暴行
> --罗伯特·佩恩·沃伦

变换是一种采用点、向量或颜色等实体并以某种方式转换它们的操作。对于计算机图形从业者来说，掌握变换是极其重要的。使用它们，您可以定位、重塑对象、灯光和相机并为其设置动画。您还可以确保所有计算都在同一坐标系中执行，并以不同方式将对象投影到平面上。这些只是可以使用变换执行的少数操作，但它们足以证明变换在实时图形（某种程度上是在任何类型的计算机图形)中的作用的重要性。

线性变换是一种保留向量加法和标量乘法的变换。具体来说，也就是：

$$f(x) + f(y) = f(x + y) \tag{4.1}$$

$$kf(x) = f(kx) \tag{4.2}$$

例如，$f(x) = 5x$是一种采用向量并将每个元素乘以5的变换。为了证明这是线性的，需要满足两个条件（公式4.1和4.2）。第一个条件成立，因为任何两个向量乘以5然后相加，会与向量相加然后相乘相同。显然满足标量乘法条件（公式 4.2）。此函数称为缩放变换，因为它会更改对象的缩放（大小）。旋转变换是另一种线性变换，它围绕原点旋转向量。 缩放和旋转变换，实际上所有三元素向量的线性变换，都可以用3×3矩阵表示。

然而，这个矩阵的大小通常不够大。三元素向量x的函数，例如$f(x) = x + (7, 3, 2)$不是线性的。在两个单独的向量上执行此函数会将(7,3,2)的每个值相加两次以形成结果。将固定向量与另一个向量相加会执行平移，例如，它将所有位置移动相同的量。这是一种有用的变换类型，我们希望结合各种变换，例如，将对象缩放为原来的一半，然后将其移动到不同的位置。将函数保持在迄今为止使用的简单形式中，很难轻松地将它们组合起来。

可以使用仿射变换来组合线性变换和平移，通常存储为4×4矩阵。仿射变换是先执行线性变换然后再进行平移的变换。为了表示四元素向量，我们使用齐次符号，以相同的方式表示点和方向（使用粗体小写字母）。方向向量表示为$\boldsymbol{v} = \begin{pmatrix} v_x & v_y & v_z & 0 \end{pmatrix}^T$，点表示为$\boldsymbol{v} = \begin{pmatrix} v_x & v_y & v_z & 1 \end{pmatrix}^T$。在本章中，

我们将广泛使用 realtimerendering.com 上可下载的线性代数附录中解释的术语和操作。

所有平移、旋转、缩放、反射和剪切矩阵都是仿射矩阵。仿射矩阵的主要特征是它保留了线的平行度，但不一定保留了长度和角度。仿射变换也可以是单个仿射变换的任何级联序列。

本章将从最基本的仿射变换开始。本节可以看作是简单转换的“参考手册”。然后描述更专业的矩阵，然后讨论和描述四元数，这是一种强大的变换工具。然后是顶点混合和变形，这是表达网格动画的两种简单但有效的方法。最后，描述了投影矩阵。大多数这些变换、它们的符号、函数和属性都在表4.1中进行了总结，其中，正交矩阵是这样的矩阵，其逆矩阵是转置矩阵。

符号	名称	特性
$\mathbf{T}(\mathrm{t})$	平移矩阵	移动一个点。仿射。
$\mathbf{R}_x(\rho)$	旋转矩阵	绕x轴旋转ρ弧度。绕y轴和z轴也是类似表示。 正交和仿射。
$\mathbf{R}$	旋转矩阵	任何旋转矩阵。正交和仿射。
$\mathbf{S(s)}$	缩放矩阵	根据$\boldsymbol{s}$沿所有 x、y 和 z 轴缩放。仿射。
$\mathbf{H}_{ij}(s)$	剪切矩阵	相对于分量j，以因子s剪切分量i。 $i, j \in \{x, y, z\}$。仿射。
$\mathbf{E}(h, p, r)$	欧拉变换	由欧拉角如航向/偏航(heading/yaw）、 俯仰(pitch)、滚动(roll)给出的方向矩阵。 正交和仿射。
$\mathbf{P}_o(s)$	正射投影	平行投影到某个平面或容体。仿射。
$\mathbf{P}_p(s)$	透视投影	透视投影到平面或容体上。
$\mathrm{slerp}(\hat{\mathbf{q}}, \hat{\mathbf{r}}, \mathrm{t})$	球面线性插值变换	创建一个关于四元数 $\hat{\mathbf{q}}$ 和 $\hat{\mathbf{r}}$ 以及参数t的内插四元数。

Table 4.1. 本章讨论的大多数变换的摘要。

变换是操作几何的基本工具。大多数图形应用程序编程接口允许用户设置任意矩阵，有时一个库可以用于实现本章中讨论的许多变换的矩阵运算。但是，了解函数调用背后的真实矩阵及其相互作用仍然是值得的。在这样的函数调用之后了解矩阵的作用是一个开始，但了解矩阵本身的属性会让你走得更远。例如，这样的理解使您能够辨别何时处理正交矩阵，其逆是其转置，从而加快矩阵求逆。像这样的知识可以让我们增加编程效率。