第十章　几何变换

柏拉图说“上帝总在使世界几何化”，应用于事物位置、方向和大小的变换，都可以称为几何变换．对计算机建模来说，几何变换可以用来拼接模型，描述模型的层级结构，而动画中能用几何变换描述对象的运动，从数学的角度描述动画．

本章节从二维几何变换开始，从数学的角度描述几何变换中的几种基本变换形式，并拓展到三维空间．然后简单介绍比较复杂的几何变换．

10.1　二维几何变换

本节介绍的基础二维几何变换包括平移、旋转，缩放，反射和剪切变换．

10.1.1　平移

对一个点 (x, y) 增加一个偏移量 (d_x, d_y)，将其移动到新的位置 (x', y')，即实现了平移 (translation) 操作，

$$x' = x + d_x, y' = y + d_y. \tag{10.1}$$

将平移变换表示成矩阵形式，设原始图形为 $\mathbf{P}$，偏移量表示为 $\mathbf{D}$，偏移后的图形为 $\mathbf{P}'$，有

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \mathbf{D} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{10.2}$$

从而平移可以抽象为 $\mathbf{P}' = \mathbf{P} + \mathbf{D}$.

平移是移动对象但不改变其形状的刚体变换．线段的平移可以通过移动端点，再重新绘制端点间的部分，多边形移动可以通过移动顶点，再重新进行连线，平移是最常见的改变物体位置的变换．

10.1.2　缩放

缩放 (scaling) 可以用来改变物体的大小，缩放的比例称为缩放系数 (s_x, s_y)，

$$x' = x \cdot s_x, y' = y \cdot s_y. \tag{10.3}$$

或表示成矩阵形式 $\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$，其中

$$\mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}. \tag{10.4}$$

当 $s_x = s_y$ 时，缩放后的对象与原对象比例不变，称为一致缩放，不等时则称为差值缩放，$0 <$ 缩放系数 < 1，对象变小，缩放系数 > 1，对象变大．当缩放系数为负数时，会将变

换对象进行翻转，特别的，产生镜像的变换称为反射 (reflection) 变换，关于 x 轴反射，关于 y 轴反射，关于原点对称的反射矩阵分别为，

$$\mathbf{S}_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \mathbf{S}_y = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{S}_o = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{10.5}$$

10.1.3　旋转

旋转 (rotation) 变换需要指定旋转轴和旋转角度，将对象的所有顶点绕旋转轴旋转指定角度后，对象完成旋转变换.

对二维图形来说，旋转轴通常垂直图形所在平面，投影到平面上成为旋转点. 以旋转点为原点为例，点 (x, y) 绕原点逆时针旋转角度 θ 可以表示为,

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta, \\ y' &= x\sin\theta + y\cos\theta. \end{aligned} \tag{10.6}$$

用矩阵形式表示 $\mathbf{P}' = \mathbf{R} \cdot \mathbf{P}$，其中

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{10.7}$$

我们可以很容易地验证 $\mathbf{R}$ 矩阵是一个正交矩阵，满足 $\mathbf{R}\mathbf{R}^{\mathbf{T}} = \mathbf{R}^T\mathbf{R} = \mathbf{I}$. 但并不是所有的正交矩阵都是旋转矩阵. 举一个简单的例子，公式10.5中的反射矩阵 $\mathbf{S}_x$ 和 $\mathbf{S}_y$ 是正交矩阵，但表示的不是旋转；而关于原点的反射矩阵 $\mathbf{S}_o$ 是一个旋转矩阵，对应着旋转 180 度. 事实上，我们可以验证所有行列式为 1 的正交矩阵都是旋转矩阵，而所有行列式为 -1 的正交矩阵可以写为旋转矩阵乘以反射矩阵. 形式化地，所有特征值为 1 的正交矩阵构成了一个群，称为特殊正交群（*Special Orthogonal Group*），在二维情况下记为 $SO(2)$. 抽象地来说，二维旋转变换与 $SO(2)$ 群里的元素一一对应，旋转角 θ 和旋转矩阵 R 只是 $SO(2)$ 群的两种表示方法. 在三维情况中我们会进一步讨论这个关系.

10.1.4　剪切

剪切 (shear) 是使对象形状发生变化的变换，经过剪切的对象看起来像滑动内部夹层进行的变换，常见的剪切是沿 x 轴和沿 y 轴方向进行剪切.

相对于 x 轴的 x 方向剪切变换可表示成下列矩阵：

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \tag{10.8}$$

可以验证这个矩阵不会改变 y 坐标的值，变换的结果如10.1所示.

一个有意思的结果是，二维的旋转可以分解为两个类似于剪切的变换的复合：

$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ \tan\theta & \sec\theta \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \sec\theta & -\tan\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & \cos\theta \end{bmatrix} \tag{10.9}$$

分解中的每个变换可以看成缩放变换与剪切变换的复合. 并且这些类似剪切变换都有一个特征，就是会固定一个维度不动，而对另外维度进行缩放与平移，比如最左边的矩阵对应的变换为

$$x' = x, y' = \tan\theta x + \sec\theta y \tag{10.10}$$

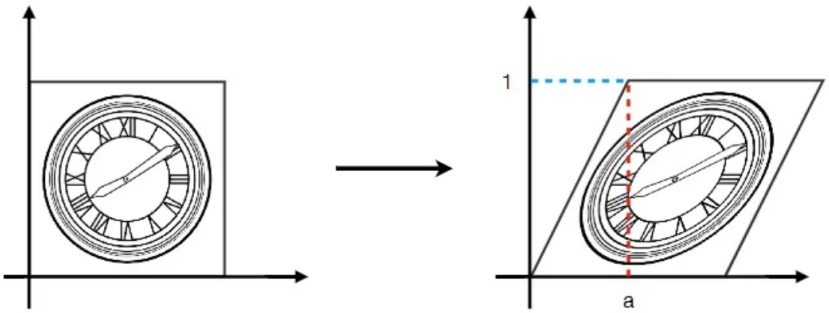

图 10.1: 剪切变换

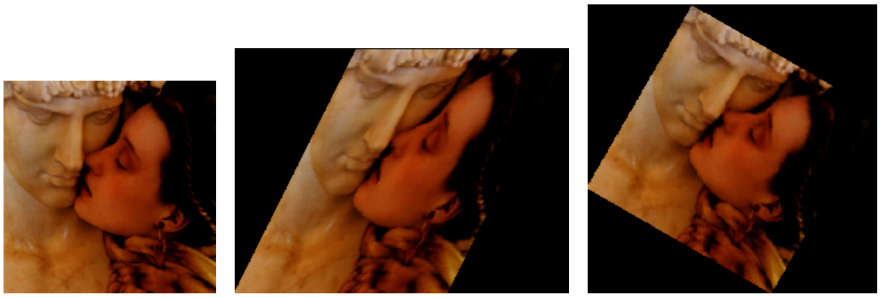

图 10.2: 两次类似剪切变换等价于旋转变换．左：原图像，中：剪切一次，右：剪切两次．

这样分解的目的是什么？注意到分解得到的类似剪切变换是非常好实现的．如果我们想旋转一张图片，以公式10.10为例，我们可以逐行绘制像素，就像之前介绍的扫描线算法，并且各行之间是完全独立的，因此可以并行实现．因此这样的分解提供了图片旋转的快速实现方法 [1]，如图10.2所示．

10.1.5　齐次坐标

二维空间下的齐次坐标 (homogeneous coordinate)，将二维坐标表示 (x, y) 扩充为 (x_w, y_w, w)，其中 w 被称为齐次参数，笛卡尔空间和齐次坐标的转换可以表示为，

$$(\frac{x}{w}, \frac{y}{w}) \Leftrightarrow (x, y, w) \tag{10.11}$$

齐次坐标表示有很多好处：

齐次坐标的引入能够描述透视空间的特征　在欧式空间中，同一平面的两条平行线不能相交．然而，在透视空间里面，两条平行线可以相交，例如：火车轨道随着我们的视线越来越窄，最后两条平行线在无穷远处交于一点．如果一个点在无穷远处，这个点的坐标将会 (∞, ∞)，在欧氏空间，这变得没有意义．而齐次坐标具有规模不变性，即对于所有的 (wx, wy, w)，它们都对应欧氏空间中同一个点的 (x, y)．从而求解欧氏空间中的平行线相交方程：

$$\begin{cases} Ax + By + C = 0, \\ Ax + By + D = 0. \end{cases} \tag{10.12}$$

在笛卡尔坐标系中，$C \neq D$ 意味着方程无解，但如果变换到齐次坐标系下，

$$\begin{cases} Ax + By + Cw = 0, \\ Ax + By + Dw = 0. \end{cases} \tag{10.13}$$

则存在无穷远处的解 $(x_w, y_w, 0)$. 在后面的三维透视投影中我们能看到齐次坐标发挥更大的作用.

齐次坐标可以用来区分点和向量　当 $w \neq 0$ 时，(x, y, w) 唯一对应笛卡尔坐标系下的点，而当 $w = 0$ 时，(x, y, w) 则可以表示向量.

齐次坐标很适合表示线性变换　注意到前面提到的平移操作，变换矩阵 $\mathbf{D}$ 是 2×1 的矩阵，而旋转矩阵和缩放矩阵都是 2×2 的矩阵，而在齐次坐标下，这些矩阵都能转变成 3×3 的矩阵，且不再区分加法和乘法，所有的操作均变成了矩阵乘法.

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{P} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10.14}$$

10.1.6　复合变换

很多时候，几何变换操作不是简单的几何变换，而是由多种变换复合而成，如绕特定点 (a, b) 进行旋转，对于特定直线的轴对称，等等. 可以将这些复杂的变换拆分成简单的变换，组合起来即得到复杂变换的变换矩阵.

举例来说，“绕特定点 (a, b) 进行旋转 θ 角”可以转变成“平移 $(-a, -b)$”+“绕原点旋转 θ 角”+“平移 (a, b)”，对应的变换矩阵就是，

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -a \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, . \tag{10.15}$$

矩阵乘法满足结合律，但不满足交换律，“先平移后旋转”和“先旋转后平移”对应是完全不同的几何变换.

10.1.7　仿射变换

如果坐标变换满足

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} & b_x \\ a_{yx} & a_{yy} & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{10.16}$$

则称该变换为仿射变换，变换后的坐标是原坐标的线性函数，且参数是常数. 仿射变换有普遍的性质，平行线变换到平行线，有限点变换到有限点.

平移、旋转、缩放、反射和剪切是仿射变换的五种特例，任何仿射变换均可以表示成这五种特例的组合，仅包括平移、旋转和反射的仿射变换保持角度和长度以及线条间平行性不变.

10.2　三维几何变换

三维空间的几何变换，在二维的基础上增加了 z 轴，三维空间中的旋转和缩放很容易从二维扩展过来，

$$\mathbf{D}=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix},\mathbf{P}=\begin{bmatrix}s_x&0&0&0\\0&s_y&0&0\\0&0&s_z&0\\0&0&0&1\end{bmatrix}. \tag{10.17}$$

但是，三维旋转却要复杂不少．因为二维空间上的旋转，仅仅是围绕垂直于 xy 平面的轴旋转，而三维空间的旋转可以围绕任意一根轴进行旋转．本节只介绍两种最基础的旋转表示方法，欧拉角和旋转向量，并简要分析它们的优缺点．

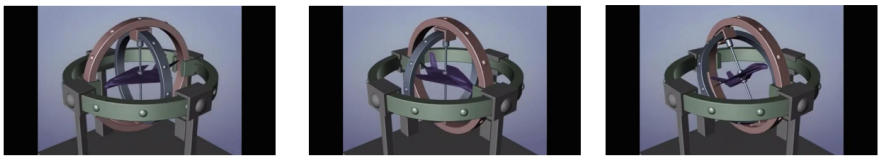

图 10.3: 欧拉角的示意图，从左至右依次是旋转绿色圈的偏航角 (yaw)，旋转红色圈的俯仰角 (pitch)，旋转蓝色圈的翻滚角 (roll)．

10.2.1　欧拉角

欧拉角是用来描述三维空间中刚体的转动角度的三个独立角度，最早由欧拉于 1776 年提出来，所以也被称为欧拉角．

三维旋转包含三个自由度，欧拉角将三个自由度分别给了三个确定的轴，即用绕三个确定轴的旋转来表示任意旋转．欧拉角的表示方法很多，以常见的航空领域的用法为例．

1. 先绕全局的 Z 轴旋转（偏航角），这时候目标自己的坐标系也发生了旋转；
2. 再绕自己的 Y 轴旋转（俯仰角）；
3. 最后绕自己的 X 轴旋转（滚转角）．

这里的旋转顺序是 ZYX，实际上，欧拉角的选取有很多种，除了用三个不同的轴，也有用两个轴（如 ZXZ）的表示方法，以及绕固定轴（全局）和运动轴（局部）的区别．这里有一个很有意思但不直观的结论，留给读者整明：三次绕固定轴旋转的最终姿态和以相反顺序三次绕运动轴旋转的最终姿态相同．

欧拉角有一个很常见的问题——*万向锁*（*Gimbal Lock*）．万向锁是指在动态表示（ZYX）下，当俯仰角等于 ±90 度时，第三次旋转和第一次旋转效果等价，丢失了一个表示维度的问题．直观上理解，当俯仰角等于 ±90 度时，局部坐标系下的 X 轴，被旋转到了全局坐标系下的 Z 轴上，与最开始绕 Z 轴旋转也就具有相同的旋转效果．对于欧拉角来说，万向锁是无法避免的问题，为了尽可能减少万向锁的影响，不同的专业领域会选择不同的旋转轴和转序．

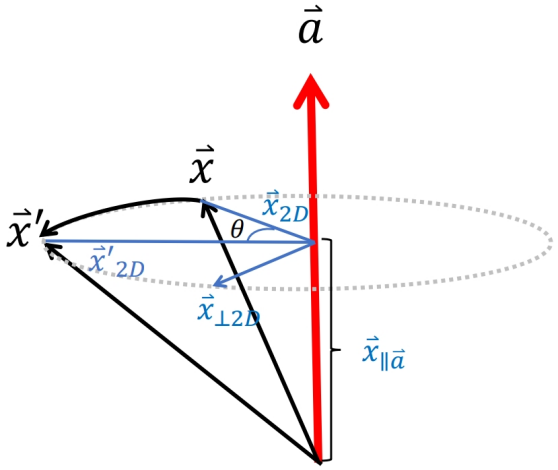

图 10.4: 轴角表示法.

10.2.2　轴角表示法

欧拉角虽然很常用，但它依赖于坐标轴的选定，且存在万向锁的问题. 轴角表示法是更普遍的旋转表示：假设有一根经过原点的旋转轴 $\vec{a}$（如果不经过原点，可以先平移到原点，旋转完，再平移回来），我们希望将向量 $\vec{x}$ 绕旋转角旋转 θ 度（从旋转轴正方向看去，进行逆时针旋转），变换到 $\vec{x}'$. 对于任意旋转轴，可以用罗德里格斯公式求解旋转矩阵.

罗德里格斯公式的推导，采取拆分的思想，将旋转向量 $\vec{x}$ 拆分成与旋转轴 $\vec{a}$ 平行的分量 $\vec{x}_{//}$ 和垂直分量 $\vec{x}_{2D}$，其中，$\vec{x}_{//}$ 旋转前后不变，只需要考虑 $\vec{x}_{2D}$ 的变化，如图 10.4所示：

$$\begin{aligned}\vec{x}' &= \vec{x}'_{2D} + \vec{x}_{//} \\ &= \vec{x}_{2D}\cos\theta + \vec{x}_{\perp}\sin\theta + \vec{x}_{//} \\ &= [(\vec{x} - (\vec{x}\cdot\vec{a})\vec{a})\cos\theta] + (\vec{a}\times\vec{x})\sin\theta + (\vec{x}\cdot\vec{a})\vec{a} \\ &= \cos\theta\vec{x} + (1-\cos\theta)(\vec{x}\cdot\vec{a})\vec{a} + \sin\theta(\vec{a}\times\vec{x})\end{aligned}$$

我们可以直接推出轴角定义下的旋转矩阵：

$$\mathbf{R} = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{a}\mathbf{a}^T + \sin\theta\mathbf{a}^*,\ \mathbf{a}^* = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{10.18}$$

轴角表示法非常直观且紧凑，但有个很大的缺点是很难进行旋转的插值. 假设我们有两个旋转 $(\vec{a}_0, \theta_0)$ 和 $(\vec{a}_1, \theta_1)$，我们希望平滑地从 $(\vec{a}_0, \theta_0)$ 运动到 $(\vec{a}_1, \theta_1)$，应该如何实现？如果我们采用的是欧拉角的表示，我们可以对三个旋转角度进行线性插值. 但是对于轴角表示，旋转轴的插值并不直接.

10.3　坐标变换

计算机图形应用经常需要在场景处理时将对象的描述从一个坐标系变换到另一个坐标系，比如在建模时，单个物体对象常在各自的局部笛卡尔坐标系下设计，但当将它们放置在大场景中时，就需要变换到以场景作为参考系或是世界坐标系下. 同时，也有不少对象表示在非笛卡尔坐标系（如极坐标，球面坐标系下），需要将其转化到统一的笛卡尔坐标系里.

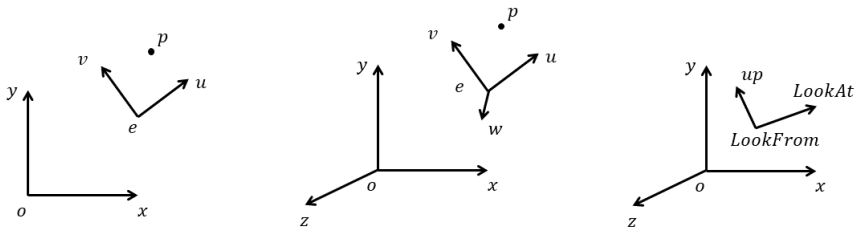

图 10.5: 坐标变换示意图.

10.3.1　二维坐标系变换

如图 10.5左边所示，点 p 在坐标 uv 和坐标 xy 下的坐标分别为

$$(p_u, p_v) = \vec{e} + p_u\vec{u} + p_v\vec{v}, \tag{10.19}$$

$$(p_x, p_y) = \vec{o} + p_x\vec{x} + p_y\vec{y}. \tag{10.20}$$

这两个坐标表示的是同一个点，从而可以得到从 uv 坐标到 xy 坐标的转换关系，从矩阵形式也可以看出来，二维坐标变换的过程，也可以看作是坐标系的变换，先将当前坐标系进行旋转，再对坐标系进行平移：

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & 0 \\ u_y & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix}. \tag{10.21}$$

10.3.2　三维坐标变换

类比二维坐标变换，对于图 10.5中间的三维空间中点的表示 (p_x, p_y, p_z) 和 (u_x, u_y, u_z)，有如下转换矩阵：

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ p_w \\ 1 \end{bmatrix}. \tag{10.22}$$

在计算机图形里，有一个很重要的三维变换，视角变换. 比如通过相机观察了一个场景中的物体，需要推测物体在三维场景中的位置，就需要从相机观察的视角下变换到世界坐标下.

通常，观察视角由三个信息组成：观察点 (LookFrom)，观察方向 (LookAt)，正上方 (up)，三个方向唯一确定了观测坐标系. 从观测坐标系到全局坐标系可以类似三维坐标变换那样将观测坐标系当作三维坐标系处理，也可以将 LookFrom 平移到原点，再将 LookAt 旋转到 z 方向，最后将 up 旋转到 y 方向.

10.4　铰接模型

如图10.6所示，我们可以只用长方体搭建起一个人体的结构．首先将长方体拉伸成对应部分的形状，然后通过旋转、平移将其移动到合适的位置．我们想改变人体的姿势，应该如何操作？如果直接修改手臂部分的旋转，我们就会发现手臂与身体其他部分脱节了．人体的各个关节限制了相连部分运动的范围，这样组成的整体称为铰接模型（*Articulated Model*）．

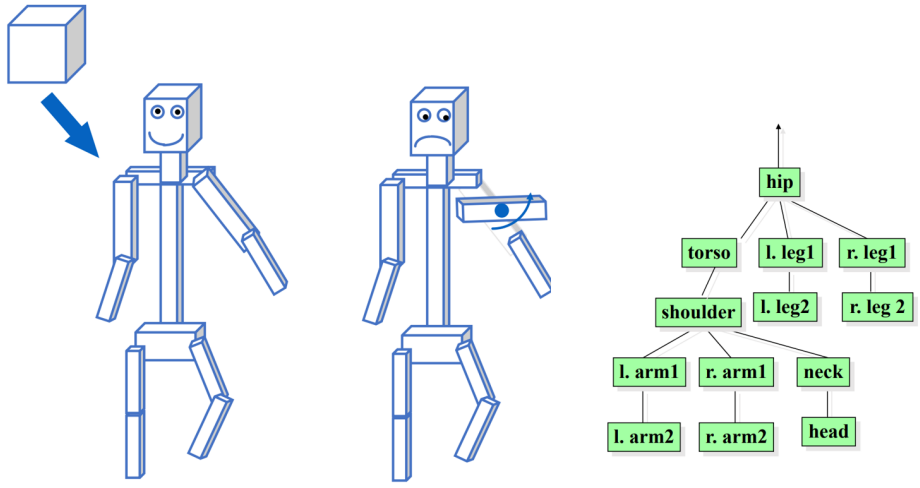

图 10.6: 铰接模型

如何在运动过程中手臂关节不会脱臼？我们需要保证手臂是围绕着肩关节旋转．用矩阵形式写出来，手臂相对于肩的坐标变换是 TR，T 表示平移，，R 表示旋转．在运动过程中，我们只能改变旋转矩阵，而不能改变平移矩阵，这样就能保证肩关节不会脱臼．但是只有这一点还不够，如果我们只移动了大臂而小臂保持不动，那么手臂还是会脱臼．因此，对于一条手臂上的多个部分，我们需要按顺序依次对各个部分进行变换．每个部分的变换相对于上一个部分变换之后的坐标进行．因此，假设大臂相对肩关节的坐标变换为 T_1R_1，小臂相对于大臂的变换为 T_2R_2，那么小臂相对于肩的变换矩阵就是 $T_2R_2T_1R_1$．

到这里我们可以发现，如果我们想追踪整个身体的运动，我们实际需要构建出一棵树，如图10.6右所示．树的根节点是身体，根据身体关节的连接关系一直延申到身体末端．树上的每个节点都记录了当前节点相对于父节点的平移和旋转．当想要计算叶子节点相对于世界坐标的变换矩阵时，我们就需要从根节点开始依次乘上各个节点的变换矩阵．这个树的结构非常重要，在之后的人体动画中我们能看到这个结构发挥重要作用．

在实现上，我们可以使用深度优先的方法遍历整棵树，用栈来维护整个过程中需要的变换矩阵．事实上，OpenGL 等图形 API 提供了 glPushMatrix()，glPopMatrix() 这样的接口来进行铰接模型的遍历．

10.5　投影变换

前面讨论的是在二维空间和三维空间内部，如何对对象进行几何变换，而对图形学来说，一个很重要的问题就是三维物体的成像，即如何把三维对象，投影到二维平面．

10.5.1　三维观察模型

三维观察分两步走，第一步是求出世界坐标系下的对象在观察坐标系中的表示，第二步是从观察坐标系中将对象投影到投影平面上.

观察坐标系由三维空间中的位置 **P**、观察平面法向量 **N**，向上向量 **V** 决定. 在观察坐标系中，**N** 通常是 z 轴正方向，**V** 是 y 轴正方向，x 轴方向根据左右手坐标系来决定. 观察坐标系确定之后，观察对象可以先通过坐标系变换从全局坐标系变换到观察坐标系下，然后在局部坐标系下进行投影.

常见的投影方法分为两种：正交投影 (orthographic projection) 和透视投影 (perspective projection). 正交投影坐标位置沿平行线变换到观察平面上，平行投影保持对象的比例关系不变，平行线在平行投影中也是平行的. 透视投影对象位置沿汇聚到观察平面后一点的直线变换到投影坐标系，透视投影不保持对象的比例关系，但真实感较好，透视投影满足近大远小的效果.

10.5.2　正交投影

连接对象点与投影点的值线称为投影线，所有投影线平行的投影称为平行投影，正交投影属于平行投影的一种，投影线均与投影平面垂直，投影线不与投影平面垂直的称为斜投影. 正交投影常被用来生成对象的三视图（前、侧、顶），工程和建筑测绘常用正交投影，因为可以精确绘制长度和角度，并能从图中测量出来.

通常情况下，成像平面大小是有限的，所以对应的三维观察空间也是有限的，这个空间称为正投影观察体，观察体的上下沿和两侧，由与投影平面边框垂直的平面组成，而沿投影平面法向 **N**，也是 z 轴方向的边缘，通过选取平行于投影平面的两个平面决定，这两个平面称为近裁剪平面 z_{near} 和远裁剪平面 z_{far}.

对于正投影来说，从观察坐标到观察平面的变换很简单，任意一点 (x, y, z) 的投影点就是 (x, y). 为了使观察处理独立于输出设备，图形系统通常将对象描述转换到规范化坐标系，规范化坐标系坐标范围为 $[-1, 1]$（也有一次为 $[0, 1]$），所以，(x, y, z) 最后需要投影到边长为 2，范围从 $(-1, -1, -1)$ 到 $(1, 1, 1)$ 的立方体内，设观察体在 x 和 y 上的范围分别为 $[x_{min}, x_{max}]$ 和 $[y_{min}, y_{max}]$，则正交投影的变换矩阵为，

$$\mathbf{M}_{ortho} = \begin{bmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & -\frac{x_{max}+x_{min}}{x_{max}-x_{min}} \\ 0 & \frac{2}{y_{max}-y_{min}} & 0 & -\frac{y_{max}+y_{min}}{y_{max}-y_{min}} \\ 0 & 0 & \frac{-2}{z_{near}-z_{far}} & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10.23}$$

10.5.3　透视投影

尽管正交投影容易生成，且可以保持对象的比例不变，但它的成像缺发真实感. 人眼观察和相机拍摄到的图像，通常是符合透视投影的规律. 透视投影下，投影线汇聚于投影中心，投影中心到投影平面的距离称为焦距 f. 一般相机的成像原理是小孔成像，小孔成像会导致图片倒过来，为了方便，将投影平面放置在拍摄对象同侧.

和正交投影类似，变换需要将观察体映射到规范化坐标系. 此时的观察体是棱台形状，近剪切面 z_{near} 小，远剪切面 z_{far} 大. 考虑棱台内任意一点 (x, y, z)，根据相似原理，它在平面上映射的位置满足，

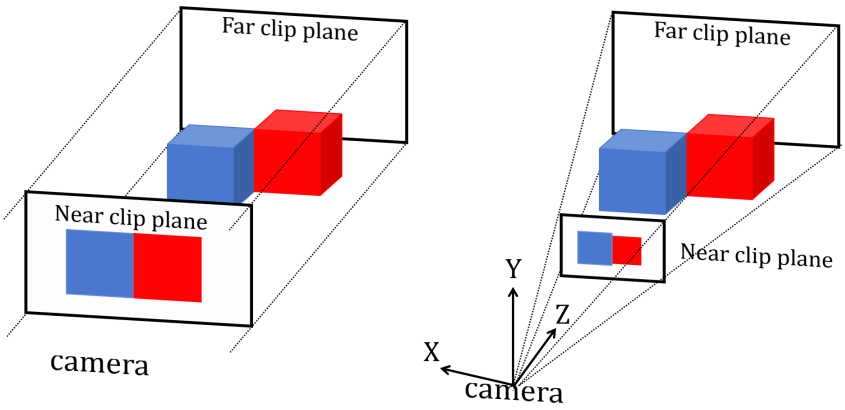

图 10.7: 正交投影和透视投影示意图.

$$\frac{x'}{f}=\frac{x}{z},\frac{y'}{f}=\frac{y}{z} \tag{10.24}$$

这里 $x'=x\frac{f}{z}$，表达式与 z 有关，并不适合用矩阵表示，但在齐次坐标下，可以将 z 移到最后一位，得到变换，

$$\begin{bmatrix} fx \\ fy \\ ? \\ z \end{bmatrix}=\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ A & B & C & D \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{10.25}$$

现在，目标变成了求解矩阵中的未知数，由于 z 的变换与 x,y 无关，所以 $A=B=0$，然后，再取两个特殊点 z_{near} 和 z_{far}，联立方程得，

$$\begin{cases} z_{near}C+D=z_{near}^2, \\ z_{far}C+D=z_{far}^2. \end{cases} \tag{10.26}$$

解得，$C=z_{near}+z_{far},D=-z_{near}z_{far}$，再应用正交矩阵的投影公式，

$$\begin{aligned} \mathbf{M}_{persp}&=\mathbf{M}_{ortho}\mathbf{M}_{persp\to ortho} \\ &=\begin{bmatrix} \frac{2}{x_{max}-x_{min}} & 0 & 0 & -\frac{x_{max}+x_{min}}{x_{max}-x_{min}} \\ 0 & \frac{2}{y_{max}-y_{min}} & 0 & -\frac{y_{max}+y_{min}}{y_{max}-y_{min}} \\ 0 & 0 & \frac{-2}{z_{near}-z_{far}} & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & z_{near}+z_{far} & -z_{near}z_{far} \\ 0 & 0 & 1 & 0 \end{bmatrix}. \end{aligned} \tag{10.27}$$

在图形软件中，相机参数通常被表示成视场角 fov，宽高比 $aspect$，近平面 $near$，远平面 far，上述的变换矩阵均可以用这些变量表示.