[2023.11.27]点、向量、矩阵

前言

于是，在了解了Shader的基础构成之后，你终于来到了第一份挑战，我们要了解Shader所需的数学知识，包括线性代数和高等数学(可能有)；

> ~~虽然凡乎问过的TA都说不需要多少数学知识，就那么几个公式，但当初我问的时候他们的百度页面都是各种光照数学模型，各种高等数学公式，所以TA的话不可信！~~

点和向量

点是空间中的一个位置，一个坐标，在三维空间中它可以用P=(x,y,z)来表示；

向量也叫矢量，它包含了标量和方向两个信息，比如空间中的一个点，它表示在原点坐标系下，这个点所处的位置，而把原点和这个点连接起来的这条带有方向信息的箭头，就叫做向量；

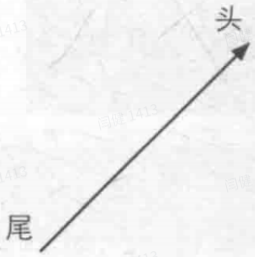
在三维空间下，它可以用v=(x,y,z)表示三维向量，在四维空间下，它可以用v=(x,y,z,w)表示四维向量；

向量的模指的是这个矢量的长度，也可以叫标量，标量可以是任意的非负数；

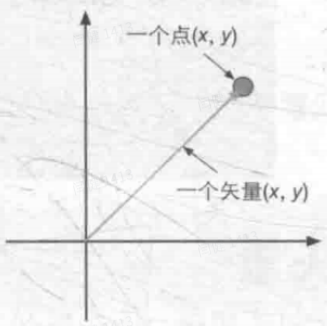
向量的方向则描述了这个向量在空间中的指向，也就是尾到头；

举个实际生活中的例子，假设我们的生活中存在一个原点，那么生活在这个坐标系中，我们来描述一下家和公司：

1. 家、公司都是坐标系中的一个点Point；
2. 家到公司的直线距离是1km，这表示家和公司之间存在一个长度为1km的标量；
3. 公司在你家的正南方1km，这表示公司在你家正南方的1km远处，这是一个向量，此时你家就成为了这个向量的原点，这个向量表示家和公司的相对位置；



▲图 4.15　一个二维向量以及它的头和尾



▲图 4.16　点和矢量之间的关系

向量运算

向量和标量的乘/除法

> 假设别人跟你问路，你告诉别人从这里往南走1km，然后再走1km；
>
> 别人问你，往哪里再走1km?
>
> 你说反正再走1km！你看，会被打的；

标量是只有模没有方向的，所以矢量和标量无法做加/减运算；但是你可以对他们进行乘除，得到一个不同长度且方向可能相反的新向量；

对于乘法来说，只需要把向量的每个分量和标量相乘即可：k*v = (kx,ky,kz)；

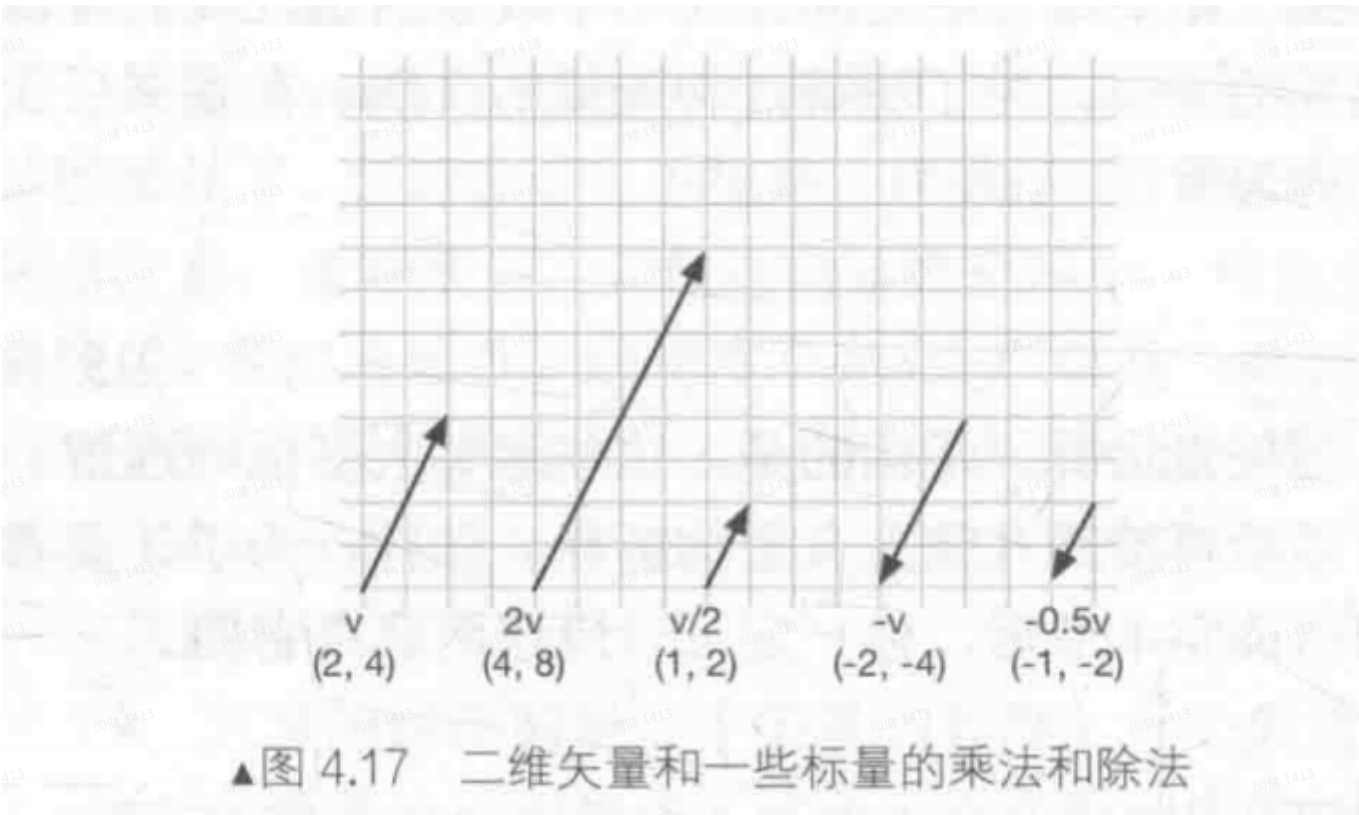
对于除法来说，向量的每个分量都可以被一个非零的标量除：v/k = (x,y,z)/k = (x/k,y/k,z/k)；

比如：

1. 2 * (1,2,3) = (2,4,6)
2. (1,2,3)/2 = (0.5,1,1.5)

对于乘法来说，是k * v 还是 v * k是没差别的；但对于除法来说，**向量只能被标量除**，我完全想象不到标量被向量除是在表达什么意义；

从几何意义上来说，一个向量v和一个标量k相乘，意味着对向量进行一个大小为|k|的缩放；当k<0时，向量的方向也会取反；



▲图 4.17　二维矢量和一些标量的乘法和除法

向量和向量的加/减法

相同维度的两个向量可以做加减法，其结果是一个相同维度的新向量；运算过程只需要把向量的各个分量进行相加或相减即可；

$$\mathbf{a}+\mathbf{b}=(a_x+b_x, a_y+b_y, a_z+b_z)$$
$$\mathbf{a}-\mathbf{b}=(a_x-b_x, a_y-b_y, a_z-b_z)$$

例如：

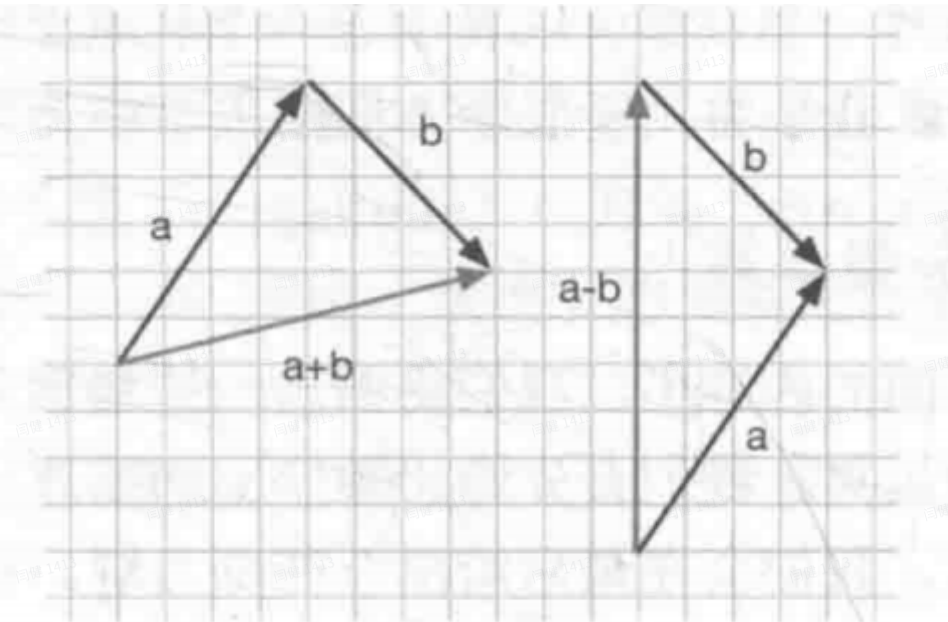
1. (1,2,3) + (4,5,6) = (5,7,9)
2. (5,2,7) - (3,8,4) = (2,-6,3)

向量不可以和标量或者不同纬度的向量做加减法这一点我们之前已经提及了；

从几何意义上看：

1. 加法，相当于把向量a的头连接到向量b的尾，然后a尾b头两点会形成一条新的向量，即为二者相加的结果；
2. 减法，相当于把向量a的头链接到向量b的头，然后a尾b尾两点会形成一条新的向量，即为二者相减的结果；

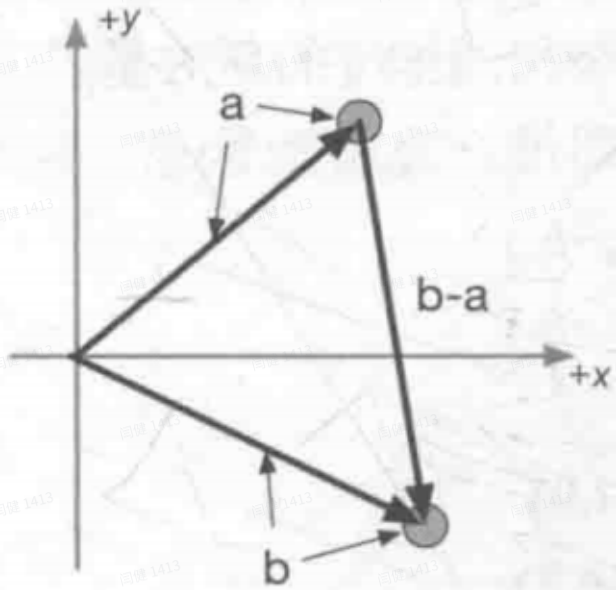
也就是说，如果我们从一个起点开始进行了一个位置偏移a，然后有进行了一个位置偏移b，那么就等同于进行了一个a+b或a+(-b)的位置偏移，这被称为向量加法的**三角形定则**；



▲图 4.18　二维矢量的加法和减法

在图形学中，向量通常用于描述位置偏移(位移)，所以我们可以利用向量的加法和减法来计算一点相对于另一点的位移；

假设坐标中有a和b两点，则向量va和vb表示它们相对于原点的位移；如果要计算b点相对于a点的位移，则可以通过vb-va来得到；



▲图 4.19　使用矢量减法来计算从点 a 到点 b 的位移

向量的模

向量本身包含了长度和方向信息，而向量的模(标量)就是向量在空间中的长度|v|；

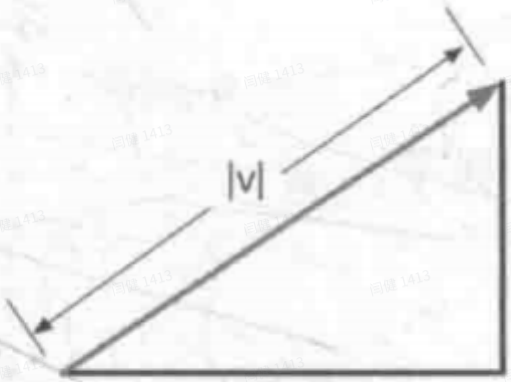
$$|\mathbf{v}|=\sqrt{v_x^2+v_y^2+v_z^2}$$

其他维度的矢量的模计算类似，都是对每个分量的平方相加后再开根号得到。
下面给出一些例子：

$$|(1,2,3)|=\sqrt{1^2+2^2+3^2}=\sqrt{1+4+9}=\sqrt{14}\approx 3.742$$

$$|(3,4)|=\sqrt{3^2+4^2}=\sqrt{9+16}=\sqrt{25}=5$$

对于二维向量来说，实际上就是勾股定理，向量的模就是直角三角形的斜边；



▲图 4.20　矢量的模

单位向量

单位向量的模是1，它是确定的，也就是说，我们只关心单位向量的方向，而不关心长度；比如在计算光照模型时，往往只关心顶点的法线方向和光源方向，至于这些向量有多长，who care，所以这时候就需要用到单位向量；

将任何非零向量转化为单位向量的过程称为**归一化**，所以单位向量就是被归一化的向量；

给定任意非零矢量 $\mathbf{v}$，我们可以计算和 $\mathbf{v}$ 方向相同的单位矢量。在本书中，我们通过在一个矢量的头上添加一个戴帽符号来表示单位矢量，例如 $\hat{\mathbf{v}}$。为了对矢量进行归一化，我们可以用矢量的模除以该矢量来得到。公式如下：

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}, \mathbf{v} \text{ 是非零矢量}$$

下面给出一些例子：

$$\frac{(3,-4)}{|(3,-4)|} = \frac{(3,4)}{\sqrt{3^2+(-4)^2}} = \frac{(3,-4)}{\sqrt{25}} = \frac{(3,-4)}{5} = \left(\frac{3}{5}, \frac{-4}{5}\right) = (0.6,-0.8)$$

注意归一化指的是，将向量的模变成1，而不是将向量的分量变成1；

零向量是无法归一化的；

向量的点积

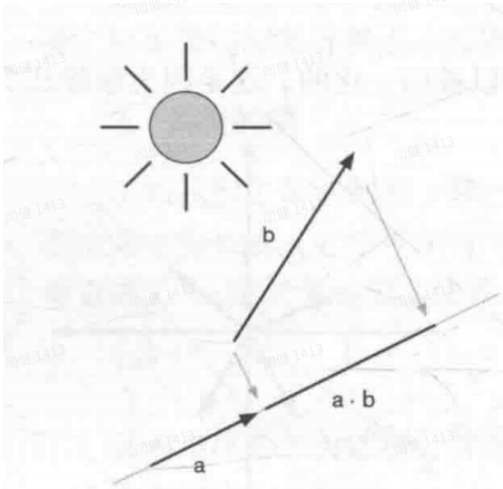
首先，两个向量的点积，其公式为：`a·b`；点积的公式有两种形式，第一种形式就是把两个向量，各自对应的分量相乘，最后再相加，结果就是向量的点积；

所以向量的点积结果是一个标量；

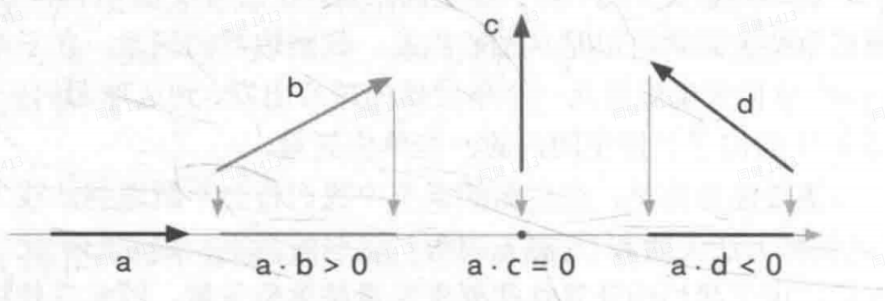
公式一：

$$\mathbf{a}\cdot\mathbf{b}=(a_x, a_y, a_z)\cdot(b_x, b_y, b_z)= a_xb_x+ a_y b_y+a_z b_z$$

向量的点积满足交换律，即`a·b = b·a`；点积的几何意义很重要，其几乎应用到了图形学的各个方面，其中一个几个意义就是**投影**；投影的值可能是负数，正负号结果与向量a和b的方向相关；



▲图 4.22　矢量 **b** 在单位矢量 **a** 方向上的投影



▲图 4.23　点积的符号

点积有三种性质，分别是：

1. 点积可结合标量乘法：点积的操组数之一可以是另一个运算的结果，`(ka)·b=a·(kb)=k(a·b)`，也可以认为对点积的其中一个向量进行缩放，等于对最后的结果进行缩放；
2. 点积可以结合矢量加法和减法，以加法来举例，`a·(b+c)=a·b+a·c`；
3. 一个向量和自身进行点积，其结果是该向量的模的平方；这意味着，比如要对比模的大小，可以直接使用点积来进行，而无需求向量真正的模，相较于求平方根的消耗，还是乘法来的更香一些；

$$\mathbf{v}\cdot\mathbf{v}=v_xv_x+v_yv_y+v_zv_z=|\mathbf{v}|^2$$

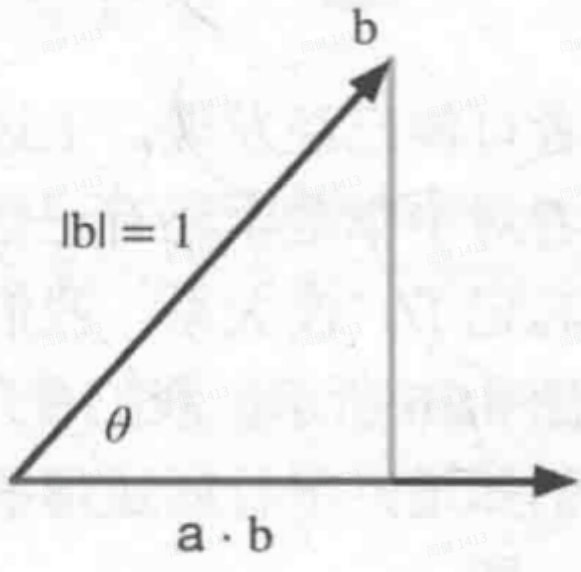
之前说了，向量点积的几何意义就是求投影，那为什么是求投影呢，另一个公式二能更直观地解释：

公式二：

$$\mathbf{a}\cdot\mathbf{b}=|\mathbf{a}||\mathbf{b}|\cos\theta$$

两个向量的模相乘的结果再乘以夹角的余弦值就是一个向量在另一个向量上的投影，高中数学上就讲过的知识；现在有两个向量a和b；将他们归一化，得到单位向量a^和b^，二者的值此时都为1；

将二者进行点积，点积的结果就是b^在a^上的投影，也是|b|cosΘ(1*cosΘ)：



▲图 4.24　两个单位矢量进行点积

所以a^·b^的结果就是cosΘ，也就是说，两个单位向量的点积是他们之间夹角的余弦值；

> 如果你觉得这是用投影来证明投影的话，你也可以这么认为：
>
> 假设单位向量a是(1,0)，单位向量b是(0.6, 0.8)，a·b=(1*0.6) + (0*0.8) = 0.6，而斜边(1)*cosΘ = 直角边(0.6)，所以cosΘ = 直角边(0.6)/斜边(1) = a·b；

再利用性质一和单位向量的归一化公式结合，就可以得到公式二：

$$\mathbf{a}\cdot\mathbf{b}=(|\mathbf{a}|\hat{\mathbf{a}})\cdot(|\mathbf{b}|\hat{\mathbf{b}})=|\mathbf{a}||\mathbf{b}|(\hat{\mathbf{a}}\cdot\hat{\mathbf{b}})=|\mathbf{a}||\mathbf{b}|\cos\theta$$

这样推导下来，就更能直观感受到，点积是求一个向量在另一个向量上投影的概念了；同时，由于夹角Θ的不同，cosΘ的值存在正负号，所以投影的方向也存在正负号；

最后，利用这个公式还可以求两个向量间的夹角：Θ=arcos(a^·b^)，arcos是反余弦操作；

此外，向量点积可以用矩阵相乘来计算。

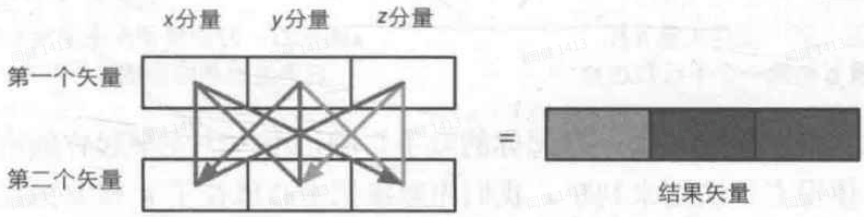
向量的叉积

另一个重要的向量运算是叉积，叉积的结果**仍是一个向量**，而非标量；

叉积的运算符号是a×b，叉积的公式为：`a(x, y, z) × b(i, j, k) = (yk - zj, zi - xk, xj - yi)`

$$\mathbf{a}\times\mathbf{b}=(a_x, a_y, a_z)\times(b_x, b_y, b_z)=(a_yb_z-a_zb_y, a_zb_x-a_xb_z, a_xb_y-a_yb_x)$$

上面的公式看起来很复杂，但其实是有一定规律的。图 4.25 给出了这样的规律图示。



▲图 4.25　三维矢量叉积的计算规律。不同颜色的线表示了计算结果矢量中对应颜色的分量的计算路径。以红色为例，即结果矢量的第一个分量，它是从第一个矢量的 y 分量出发乘以第二个矢量的 z 分量，再减去第一个矢量的 z 分量和第二矢量的 y 分量的乘积

需要注意的是，叉积不满足交换律，`a×b ≠ b×a`(叉积满足的是反交换律，即`a×b = -(b×a)`，叉积也不满足结合律，`(a×b)×c ≠ a×(b×c)`；

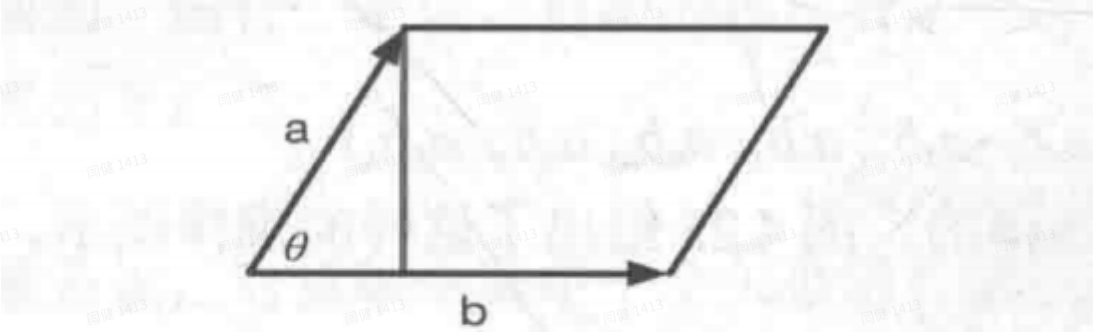
从几何的意义出发，对两个向量进行叉积的结果会是一个同时垂直于这两个向量的新向量；我们先来看它的模，axb的长度等于a和b的模的乘积再乘以它们之间夹角的余弦值；

$$|\mathbf{a}\times\mathbf{b}|=|\mathbf{a}||\mathbf{b}|\sin\theta$$

这个公式和计算平行四边形面积的公式是一样的：

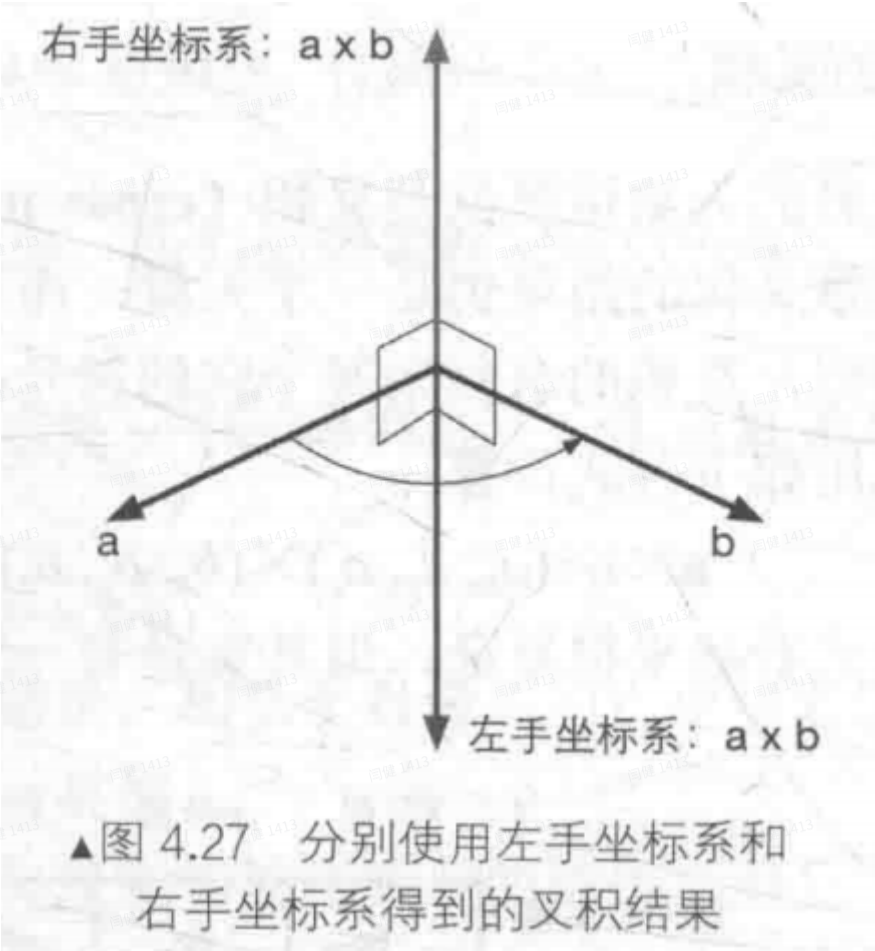
我们知道，平行四边形的面积可以使用$|\mathbf{b}|h$来得到，即底乘以高。而 h 又可以使用$|\mathbf{a}|$和夹角θ来得到，即

$$A=|\mathbf{b}|h=|\mathbf{b}|(|\mathbf{a}|\sin\theta)=|\mathbf{a}||\mathbf{b}|\sin\theta=|\mathbf{a}\times\mathbf{b}|$$



▲图 4.26　使用矢量 a 和矢量 b 构建一个平行四边形

叉积结果的方向，在左手坐标系和右手坐标系下，会有不同；



▲图 4.27　分别使用左手坐标系和右手坐标系得到的叉积结果

矩阵

又到了日常后悔没学好线性代数的时间了；这一节截图的比重会上升，因为天杀的矩阵不太好用排版来展现；

矩阵是一个网格结构，它看起来是一个由m*n个标量组成的长方形数组，这个数组中有**行(row)**和**列(rolumn)**之分；一个3*3的矩阵可以写成：

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

m_{ij} 表明了这个元素在矩阵 M 的第 i 行、第 j 列。

矩阵和向量

向量其实也是一个数组，矩阵也是数组，显而易见，我们可以用矩阵来表示向量；实际上，向量可以看做n*1的**列矩阵**或1*n的**行矩阵**，其中n表示向量的维度，例如向量v=(3,8,6)可以写成行矩阵或列矩阵：

$$\begin{bmatrix} 3 & 8 & 6 \end{bmatrix} \qquad \begin{bmatrix} 3 \\ 8 \\ 6 \end{bmatrix}$$

当我们把向量和矩阵联系在一起后，他们两个就可以参与运算了，这在空间变换中会非常有用(**天杀的矩阵**)；在进行矩阵运算时，向量使用行矩阵还是列矩阵会对计算结果产生影响；

矩阵运算

矩阵和标量的乘法

和向量相似，矩阵也可以和标量相乘，它的结果仍然是一个相同维度的矩阵；他们的乘法就是矩阵的每个元素都和该标量相乘，以3*3的矩阵为例：

$$kM = Mk = k\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} km_{11} & km_{12} & km_{13} \\ km_{21} & km_{22} & km_{23} \\ km_{31} & km_{32} & km_{33} \end{bmatrix}$$

矩阵和矩阵的乘法

两个矩阵相乘，它们的结果会是一个新的矩阵，且该矩阵的维度和两个原矩阵都有关系；一个**r***n的矩阵A和一个n***c**的矩阵B相乘，结果会是一个**r*c**大小的矩阵；第**一**个矩阵的**列数**必须和第**二**个矩阵的**行数**相同，结果中，行数是**第一个矩阵的行数**，列数是**第二个矩阵的列数**；

比如[4*3]*[3*6]的结果就是[4*6]，如果两个矩阵不满足这样的要求，则他们无法相乘；

我们先给出看起来很复杂难懂（当给出直观的表达式后读者会发现其实它没那么难懂）的数学表达式：设有 $r\times n$ 的矩阵 $\boldsymbol{A}$ 和一个 $n\times c$ 的矩阵 $\boldsymbol{B}$，它们相乘会得到一个 $r\times c$ 的矩阵 $\boldsymbol{C}=\boldsymbol{AB}$。那么，$\boldsymbol{C}$ 中的每一个元素 c_{ij} 等于 $\boldsymbol{A}$ 的第 i 行所对应的矢量和 $\boldsymbol{B}$ 的第 j 列所对应的矢量进行矢量点乘的结果，即

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{in}b_{nj}=\sum_{k=1}^{n}a_{ik}b_{kj}$$

一种更直观的方式如图 4.30 所示。假设 $\boldsymbol{A}$ 的大小是 4×2，$\boldsymbol{B}$ 的大小是 2×4，那么如果要计算 $\boldsymbol{C}$ 的元素 c_{23} 的话，先找到对应的行矩阵和列矩阵，即 $\boldsymbol{A}$ 中的第 2 行和 $\boldsymbol{B}$ 中的第 3 列，把它们进行矢量点积后就可以得到结果值。因此，$c_{23}=a_{21}b_{13}+a_{22}b_{23}$。

$$\begin{bmatrix} b_{11} & b_{12} & \boxed{b_{13}} & b_{14} \\ b_{21} & b_{22} & \boxed{b_{23}} & b_{24} \end{bmatrix}$$

$$\begin{bmatrix} b_{11} & b_{12} & \boxed{b_{13}} & b_{14} \\ b_{21} & b_{22} & \boxed{b_{23}} & b_{24} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ \boxed{a_{21}} & \boxed{a_{22}} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \quad \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & \boxed{c_{23}} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

▲图 4.30　计算 c_{23} 的过程

在Shader的计算中，我们更多地是使用4*4的矩阵来运算的，矩阵乘法有一些性质：

1. 矩阵乘法不满足交换律，即AB≠BA；
2. 矩阵乘法满足结合律，即(AB)C=A(BC)；
    a. 更多的矩阵相乘也可以扩展到：ABCDE = ((A(BC))D)E = (AB)(CD)E；

特殊矩阵

有一些特殊的矩阵类型在Shader中经常见到，这些特殊矩阵往往具有一些重要的性质；

方块矩阵

方块矩阵简称方针，指那些行和列数目相等的矩阵，三维渲染里最常使用的就是3*3和4*4的方阵；某些运算和性质是方阵独有的，比如对角元素；对角元素指的是行号和列号相等的元素，比如m11,m22,m33等；如果把方阵看成一个正方形的话，这些元素排列在正方形的对角线上，这也是对角元素名字的由来；

如果一个矩阵除了对角元素外，所有的元素都为0，那么这个矩阵就叫做对角矩阵：

下面就是一个 4×4 的对角矩阵：

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

单位矩阵

单位矩阵是特殊的对角矩阵，用In来表示，一个3*3的单位矩阵如下：

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

任何一个矩阵和单位矩阵相乘的结果还是原来的矩阵，MI = IM = M，单位矩阵类似于标量1；

转置矩阵

转置矩阵实际是对原矩阵的一种运算，转置运算；给定一个r*c的矩阵M，它的转置可以表示成MT，这是一个c*r的矩阵；转置矩阵的计算就是把原矩阵翻转一下，原本的第i行变成了第i列，第j列变为了第j行；

$$M_{ij}^{\mathrm{T}} = M_{ji}$$

例如，

$$\begin{bmatrix} 6 & 2 & 10 & 3 \\ 7 & 5 & 4 & 9 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 6 & 7 \\ 2 & 5 \\ 10 & 4 \\ 3 & 9 \end{bmatrix}$$

对于行矩阵和列矩阵来说，我们可以使用转置操作来转换行列矩阵：

$$\begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x & y & z \end{bmatrix}$$

转置矩阵也有一些常用性质：

1. 转置矩阵的转置等于原矩阵，相当于什么都没做：(MT)T = M；
2. 矩阵串接的转置，等于**反向串接**各个矩阵的**转置**：(AB)T = BT*AT；该性质同样可以扩展到更多矩阵相乘的情况；

逆矩阵

逆矩阵的前提是，该矩阵必须是一个方阵；给定一个方阵M，它的逆矩阵用M-1来表示。逆矩阵最重要的性质是，如果把一个矩阵和它的逆矩阵相乘，其结果会是一个单位矩阵：

$$MM^{-1} = M^{-1}M = I$$

并非所有矩阵都有逆矩阵，比如零矩阵，它的所有单位都是零，任何矩阵和它相乘都会得到零矩阵；

如果一个矩阵有对应的逆矩阵，则这个矩阵是可逆的，否则是不可逆的；

逆矩阵有很多重要的性质：

1. 逆矩阵的逆矩阵是原矩阵本身，假设矩阵M是可逆的，那么：$(M^{-1})^{-1}=M$
2. 单位矩阵的逆矩阵是它本身：$I^{-1}=I$
3. (一个)转置矩阵的逆矩阵是(这个矩阵的)逆矩阵的转置：$(M^{T})^{-1}=(M^{-1})^{T}$
4. 矩阵串接相乘后的逆矩阵等于反向串接各个矩阵的逆矩阵：$(AB)^{-1}=B^{-1}A^{-1}$
   a. 这个性质也可以拓展到更多的矩阵连乘：$(ABCD)^{-1}=D^{-1}C^{-1}B^{-1}A^{-1}$

逆矩阵是有几何意义的，矩阵可以表示一个变换，而逆矩阵允许我们还原这个变换，或者说计算这个变换的反向变换；当我们使用变换矩阵M对向量v进行了一次变换，然后再使用它的逆矩阵(M)-1进行另一次变换，那么我们会得到原来的向量；

$$M^{-1}(Mv)=(M^{-1}M)v=Iv=v$$

正交矩阵

另一个特殊的方阵是正交矩阵；正交矩阵是矩阵的一种属性，如果一个方阵M和它的转置矩阵的乘积是单位矩阵的话，则这个矩阵是正交的，反向串联也是成立的，也就是说，矩阵M正交等价于：

$$MM^{T}=M^{T}M=I$$

这个公式和逆矩阵的公式是非常相似的，所以把两个公式结合起来，我们得到了正交矩阵的重要性质是，如果一个矩阵是正交的，那么它的**转置矩阵和逆矩阵是相同的**，也就是说，矩阵M正交等价于：

$$M^{T}=M^{-1}$$

这个式子的意义很重大，比如在三维变换中，我们可能会需要用逆矩阵来求反向的变换，但逆矩阵求解的计算量很大，但转置矩阵确比较容易；

因此，我们需要结合正交矩阵的几何意义来尝试判断一个矩阵是否是正交的；

以3*3的正交矩阵为例，根据正交矩阵的定义，我们有：

$$
\begin{aligned}
\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M} &= \begin{bmatrix} - & \mathbf{c}_1 & - \\ - & \mathbf{c}_2 & - \\ - & \mathbf{c}_3 & - \end{bmatrix} \begin{bmatrix} | & | & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ | & | & | \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{c}_1 \cdot \mathbf{c}_1 & \mathbf{c}_1 \cdot \mathbf{c}_2 & \mathbf{c}_1 \cdot \mathbf{c}_3 \\ \mathbf{c}_2 \cdot \mathbf{c}_1 & \mathbf{c}_2 \cdot \mathbf{c}_2 & \mathbf{c}_2 \cdot \mathbf{c}_3 \\ \mathbf{c}_3 \cdot \mathbf{c}_1 & \mathbf{c}_3 \cdot \mathbf{c}_2 & \mathbf{c}_3 \cdot \mathbf{c}_3 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{I}
\end{aligned}
$$

这样，我们就有了9个等式：

$$
\begin{gathered}
\mathbf{c}_1 \cdot \mathbf{c}_1 = 1, \quad \mathbf{c}_1 \cdot \mathbf{c}_2 = 0, \quad \mathbf{c}_1 \cdot \mathbf{c}_3 = 0 \\
\mathbf{c}_2 \cdot \mathbf{c}_1 = 0, \quad \mathbf{c}_2 \cdot \mathbf{c}_2 = 1, \quad \mathbf{c}_2 \cdot \mathbf{c}_3 = 0 \\
\mathbf{c}_3 \cdot \mathbf{c}_1 = 0, \quad \mathbf{c}_3 \cdot \mathbf{c}_2 = 0, \quad \mathbf{c}_3 \cdot \mathbf{c}_3 = 1
\end{gathered}
$$

同时可以得到以下结论：

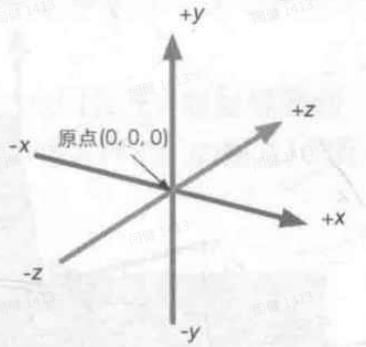
- 矩阵的每一行，即 $\mathbf{c}_1$、$\mathbf{c}_2$ 和 $\mathbf{c}_3$ 是单位矢量，因为只有这样它们与自己的点积才能是 1；
- 矩阵的每一行，即 $\mathbf{c}_1$、$\mathbf{c}_2$ 和 $\mathbf{c}_3$ 之间互相垂直，因为只有这样它们之间的点积才能是 0。
- 上述两条结论对矩阵的每一列同样适用，因为如果 $\boldsymbol{M}$ 是正交矩阵的话，$\boldsymbol{M}^{\mathrm{T}}$ 也会是正交矩阵。

也就是说，如果一个矩阵满足上面的条件，那么它就是一个正交矩阵；一组标准正交基可以满足上述条件，如果使用**标准正交基**来构建用于空间变换的矩阵，就可以直接使用转置矩阵来求得该变换的逆变换；

度就大一些；对于更高维度的空间（如四维空间），理解难度就更大了。

在三维笛卡儿坐标系中，我们需要定义 3 个坐标轴和一个原点。图 4.6 显示了一个三维笛卡儿坐标系。

这 3 个坐标轴也被称为是该坐标系的**基矢量（basis vector）**。通常情况下，这 3 个坐标轴之间是互相垂直的，且长度为 1，这样的基矢量被称为**标准正交基（orthonormal basis）**，但这并不是必须的。例如，在一些坐标系中坐标轴之间互相垂直但长度不为 1，这样的基矢量被称为**正交基（orthogonal basis）**。如非特殊说明，本书默认情况下使用的坐标轴指的都是标准正交基。



▲图 4.6　一个三维笛卡儿坐标系

行矩阵还是列矩阵

在前面的章节中我们知道，可以把一个向量转换成一个行矩阵或列矩阵，他们本身没有区别，但当我们需要把它和另一个矩阵相乘时，就会出现一些差异；

假设有一个向量v=(x,y,z)，我们可以把它转换成行矩阵或列矩阵，现在有另一个矩阵M：

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

当我们和行矩阵相乘时，需要把行矩阵放在M的左边，(3*1)*(3*3)；

$$vM = [xm_{11} + ym_{21} + zm_{31} \quad xm_{12} + ym_{22} + zm_{32} \quad xm_{13} + ym_{23} + zm_{33}]$$

而当我们和列矩阵相乘时，需要把列矩阵放在M的右边，(3*3)*(1*3)；

$$Mv = \begin{bmatrix} xm_{11} + ym_{12} + zm_{13} \\ xm_{21} + ym_{22} + m_{23} \\ xm_{31} + ym_{32} + zm_{33} \end{bmatrix}$$

你会发现，除了矩阵行列的区别外，里面的元素也是不一样的；这意味着在矩阵相乘时，选择行矩阵还是列矩阵来表示向量是很重要的，因为这决定了矩阵乘法的书写次序和结果；

在Unity中，常规做法是把向量放在矩阵的右侧，即把向量转换成**列矩阵**来进行运算，所以我们的矩阵乘法通常都是右乘；

$$CBAv = (C(B(Av)))$$

使用列向量的结果是，我们的阅读顺序是从右到左，即先对 v 使用 A 进行变换，再使用 B 进行变换，最后使用 C 进行变换。

上面的计算等价于下面的行矩阵运算：

$$vA^{\mathrm{T}}B^{\mathrm{T}}C^{\mathrm{T}} = (((vA^{\mathrm{T}})B^{\mathrm{T}})C^{\mathrm{T}})$$