计算机图形学

变换

陶钧

taoj23@mail.sysu.edu.cn

中山大学 数据科学与计算机学院

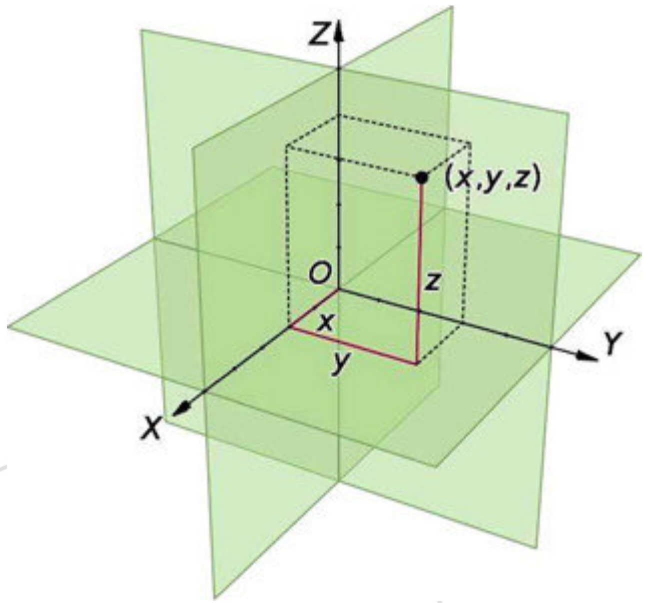
国家超级计算广州中心

课程提纲

- 基本几何概念
- 表现形式
- 变换

基本几何

- 几何研究的是n维空间中物体的表达
  - 计算机图形学中，我们主要关注的是二维和三维空间
  - 希望通过最小的几何形状集合表达复杂的物体
  - 三种基本几何元素
    - 点（point），标量（scalar），向量（vector）
  - 在初等几何中，我们使用笛卡尔（Cartisian）坐标系
    - 一个点在空间中的位置$\mathbf{P} = (x, y, z)$
    - 在这个坐标上使用代数运算进行操作
    - 这一方法并非对物理空间的模拟
      - 物理空间中点的存在不依赖于其位置
      - 大多数几何的计算也不依赖于位置
      - 例如，全等三角形



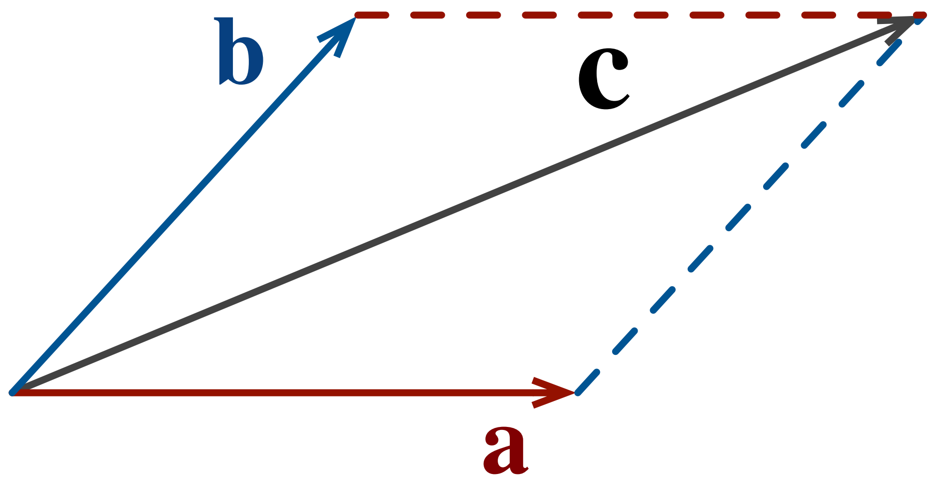
基本几何

标量（scalar）与向量（vector）

- 标量：只有数值大小，而没有方向的量
  - 关于运算+与*是闭集
    - 满足结合律与交换律，具有逆运算
  - 标量本身不具备几何性质
- 向量：既具有大小，也具有方向的量
  - 如，力，速度等
  - 常表示为带有箭头的线段
    - 箭头指向的方向表示向量方向，线段长度表示线段大小
  - 由复数（二维向量）及四元数发展而来
  - 关于运算+是闭集

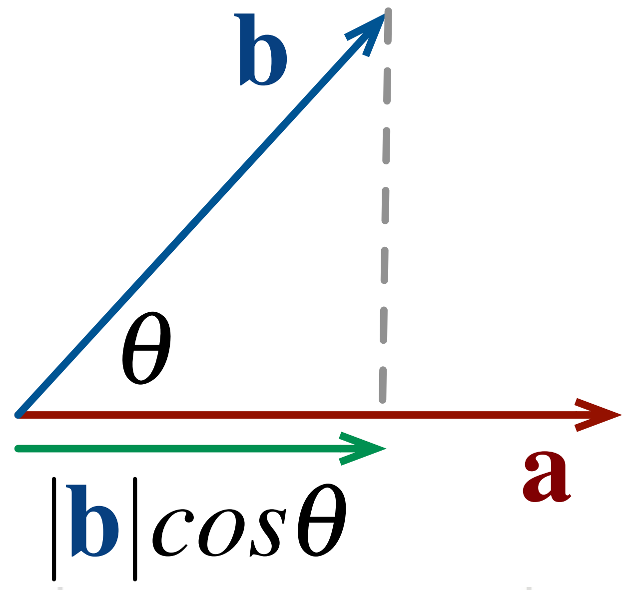
向量基本知识回顾:加法

- $\mathbf{a} + \mathbf{b} = \mathbf{c}$
- 平行四边形法则
- 各分量分别相加
  - $\mathbf{a} + \mathbf{b} = < u, v > + < p, q > = < u + p, v + q >$
- 满足交换律：$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$
- 满足结合律：$(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$
- 具有单位元0向量：$\mathbf{a} + \mathbf{0} = \mathbf{a}$
- 每个向量都具有其逆元素−a：$\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$



基本几何

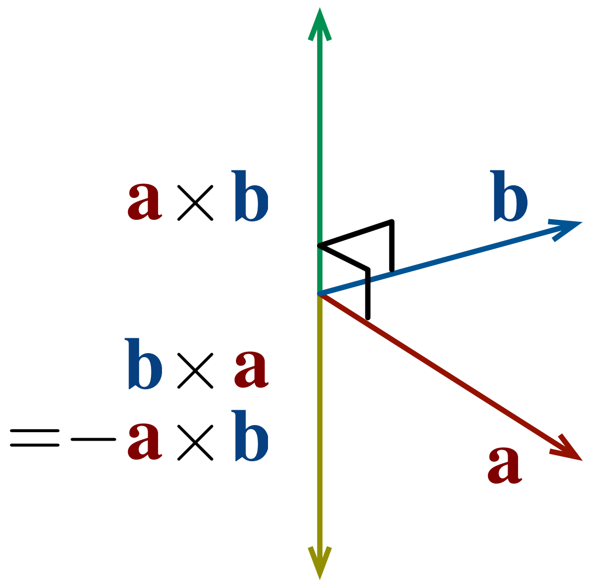
- 向量基本知识回顾:内积（数量积、点积）
  - 各分量分别相乘之和
    - $\mathbf{a}\cdot\mathbf{b} = <u,v>\cdot<p,q> = u*p+v*q$
  - 满足交换律： $\mathbf{a}\cdot\mathbf{b} = \mathbf{b}\cdot\mathbf{a}$
  - 满足分配率： $(\mathbf{a}+\mathbf{b})\cdot\mathbf{c} = \mathbf{a}\cdot\mathbf{c}+\mathbf{b}\cdot\mathbf{c}$
  - 向量a的长度表示为|a|
    - $|\mathbf{a}|^2 = \mathbf{a}\cdot\mathbf{a}$
  - $\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$
    - 常用来求向量间夹角
    - 或向量b在向量a上的投影
      - $|\mathbf{b}|\cos\theta = |\mathbf{b}|\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \frac{\mathbf{a}}{|\mathbf{a}|}\cdot\mathbf{b}$



基本几何

向量基本知识回顾:外积（向量积）

- 两个向量a与b的向量积a×b为垂直于a、b的向量
  - 只定义于三维向量上
  - 其方向由右手定则给出
  - 不满足交换律



- 表达式
  - 假设$\mathbf{a} =< a, b, c >, \mathbf{b} =< p, q, r >, \mathbf{c} =< x, y, z >$
  - 由于$\mathbf{a}\cdot\mathbf{c} = 0$及$\mathbf{b}\cdot\mathbf{c} = 0$，可知

$$\begin{aligned} ax &+ by &= -cz \\ px &+ qy &= -rz \end{aligned}$$

  - 使用Cramer’s rule求解可知得$\mathbf{c} = \left\langle \begin{vmatrix} b & c \\ q & r \end{vmatrix}, -\begin{vmatrix} a & c \\ p & r \end{vmatrix}, \begin{vmatrix} a & b \\ p & q \end{vmatrix} \right\rangle$

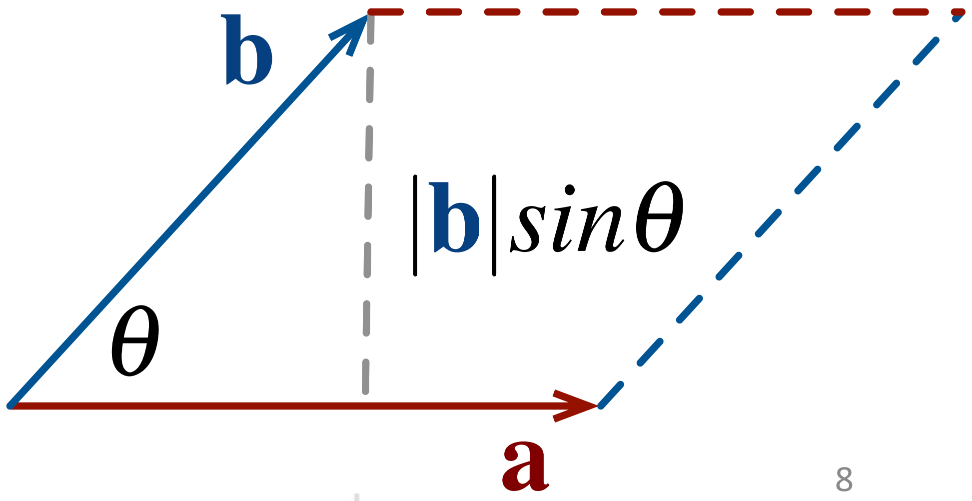
向量基本知识回顾:外积（向量积）

– 设i, j, k为表示坐标轴的单位向量，则向量积a×b可表示为

- $\mathbf{a}\times\mathbf{b}=\begin{vmatrix}\mathbf{i} & \mathbf{j} & \mathbf{k}\\ a & b & c\\ p & q & r\end{vmatrix}=\begin{vmatrix}b & c\\ q & r\end{vmatrix}\mathbf{i}+\begin{vmatrix}c & a\\ r & p\end{vmatrix}\mathbf{j}+\begin{vmatrix}a & b\\ p & q\end{vmatrix}\mathbf{k}$

– 向量a×b长度$|\mathbf{a}\times\mathbf{b}|=|\mathbf{a}||\mathbf{b}|\sin\theta$

- $|\mathbf{a}\times\mathbf{b}|^2=(a^2+b^2+c^2)(p^2+q^2+r^2)-(ap+bq+cr)^2$
  $=|\mathbf{a}|^2|\mathbf{b}|^2-(\mathbf{a}\cdot\mathbf{b})^2=(|\mathbf{a}||\mathbf{b}|\sin\theta)^2$
- 为a, b所围平行四边形面积



线性空间（linear space）

- 最为常见的数学空间：（线性）向量空间
- 两类基本几何元素
  - 标量，向量
- 运算
  - 数量向量乘：$\mathbf{b} = s\mathbf{a}$
  - 向量加法：$\mathbf{a} + \mathbf{b} = \mathbf{c}$
- 简单地说，线性空间是这样一种集合，其中任意两元素相加可构成此集合内的另一元素，任意元素与任意数（可以是实数也可以是复数，也可以是任意给定域中的元素）相乘后得到此集合内的另一元素

基本几何

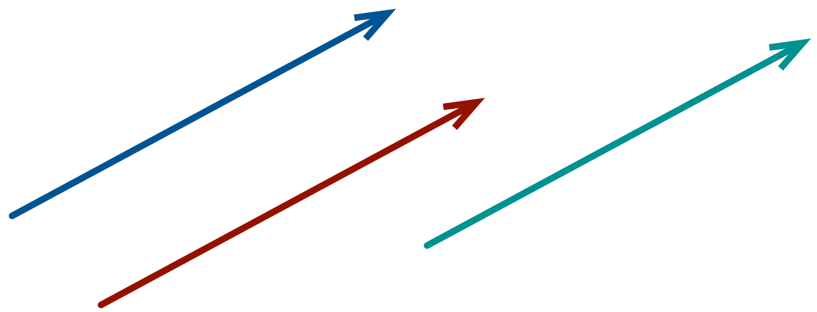
线性空间（linear space）

- 矩阵的列向量空间
  - 以矩阵的每一列向量为基组成的线性空间
  - 矩阵与向量的乘法可视为对向量的一个线性变换
    - 例如，以下例子将直角坐标（x，y，z）变换到以（1，0，1），（0，1，0）及（0，1，1）为轴的坐标系中
  - 同理，矩阵A、B相乘可视作映射的复合
    - 将矩阵B的列向量空间的基向量分别置于矩阵A的列向量空间中

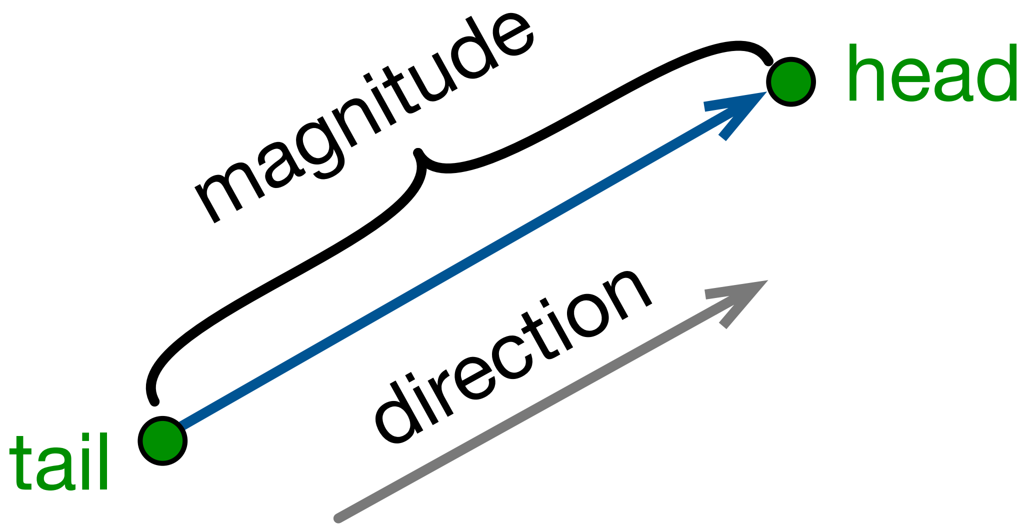
$$\begin{bmatrix}1&0&0\\0&1&1\\1&0&1\end{bmatrix}\times\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}1\\0\\1\end{bmatrix}x+\begin{bmatrix}0\\1\\0\end{bmatrix}y+\begin{bmatrix}0\\1\\1\end{bmatrix}z=\begin{bmatrix}x\\y+z\\x+z\end{bmatrix}$$

点（point）与向量（vector）

- 向量与位置无关
  - 所有长度相等，方向一致的向量都相等

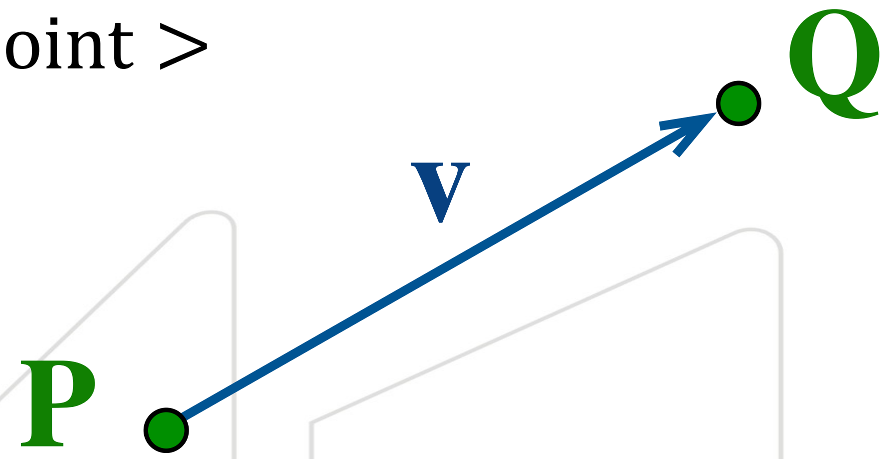


- 因此，在几何中，只有向量空间是不够的
  - 我们还需要表示位置：“点”



点（point）与向量（vector）

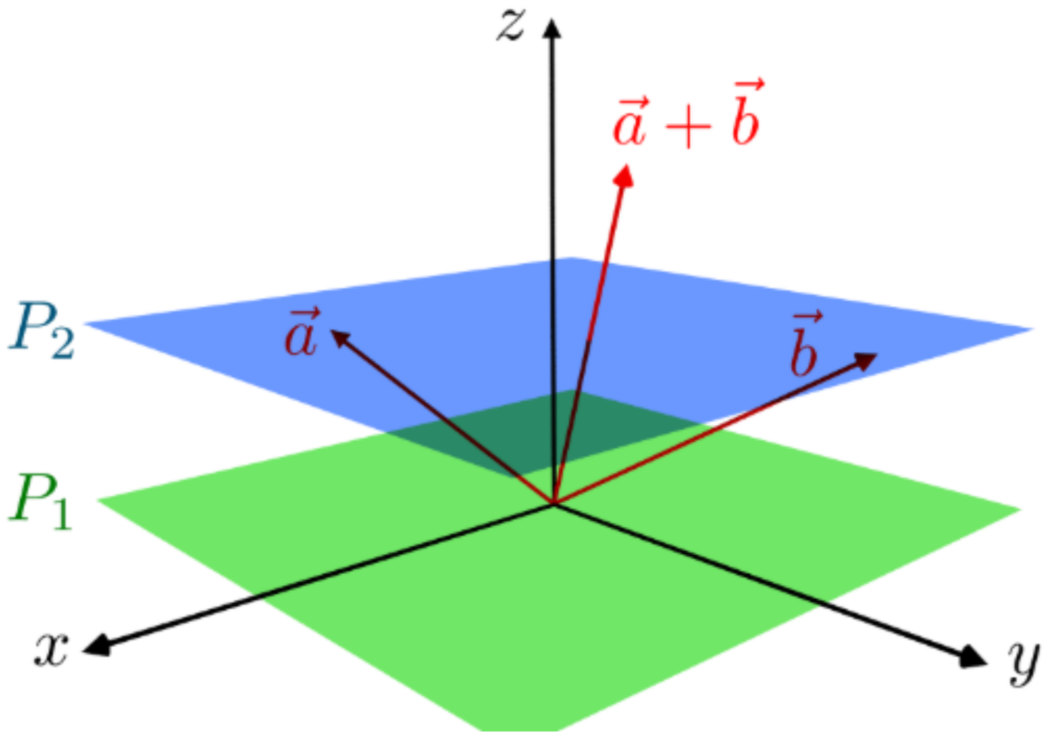
- 点表示空间中的一个位置
  - 一个点：原点+一个向量偏移
  - 与坐标系统相关
- 点与向量运算
  - 加法：$<\text{point}>=<\text{point}>+<\text{vector}>$
    - $\mathbf{Q}=\mathbf{P}+\mathbf{v}$
  - 减法：$<\text{vector}>=<\text{point}>-<\text{point}>$
    - $\mathbf{v}=\mathbf{P}-\mathbf{Q}$



基本几何

仿射空间（affine space）

- 由标量、点和向量组成的空间
- 运算
  - < vector >+< vector >=< vector >
  - < scalar >×< vector >=< vector >
  - < point >+< vector >=< point >
  - < scalar >+< scalar >=< scalar >



- 从基本数学概念上来说，一个坐标系对应了一个仿射空间 (affine space)，当向量从一个坐标系变换到另一个坐标系时要进行线性变换 (linear transformation)

基本几何

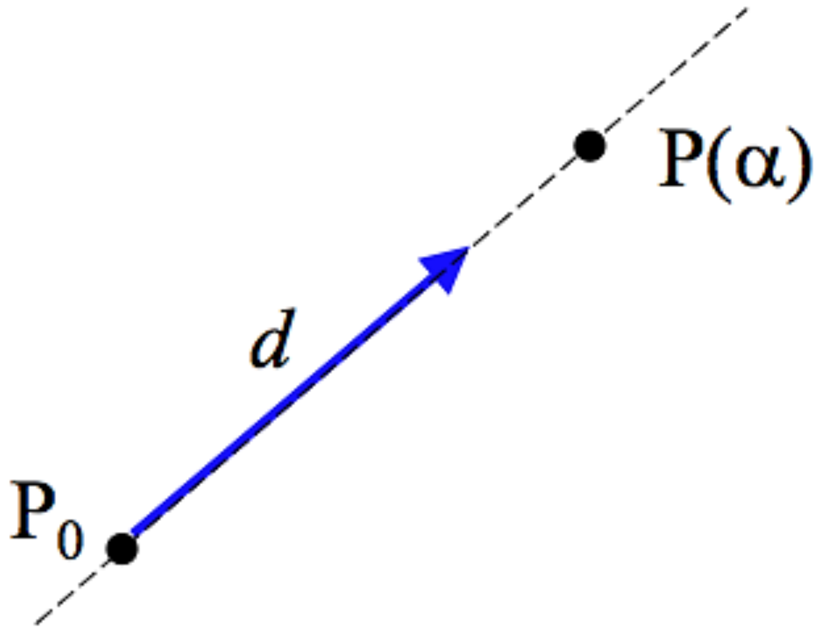
线性组合（linear combination）

- 给定一组向量$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathrm{n}$与一组标量$s_1, s_2, \ldots, s_n$，其线性组合$\mathbf{v} = s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_\mathrm{n}\mathbf{v}_\mathrm{n}$仍然为向量
  - 与坐标系无关
- 给定一个坐标系中的两个点$\mathbf{P}_1$与$\mathbf{P}_2$
  - 若$\mathbf{P}_1$为原点，则$\mathbf{P}_1 + \mathbf{P}_2 = \mathbf{P}_2$
  - 若$\mathbf{P}_1$与$\mathbf{P}_2$关于原点对称，则$\mathbf{P}_1 + \mathbf{P}_2 =$ 原点
  - 坐标系相关
- 一组点的线性组合$\mathbf{P} = s_1\mathbf{P}_1 + s_2\mathbf{P}_2 + \cdots + s_\mathrm{n}\mathbf{P}_\mathrm{n}$为点$\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_\mathrm{n}$所围成的凸包中的一点（$s_1 + s_2 + \cdots + s_n = 1$）
  - 参考上节课所提到的barycentric coordinate

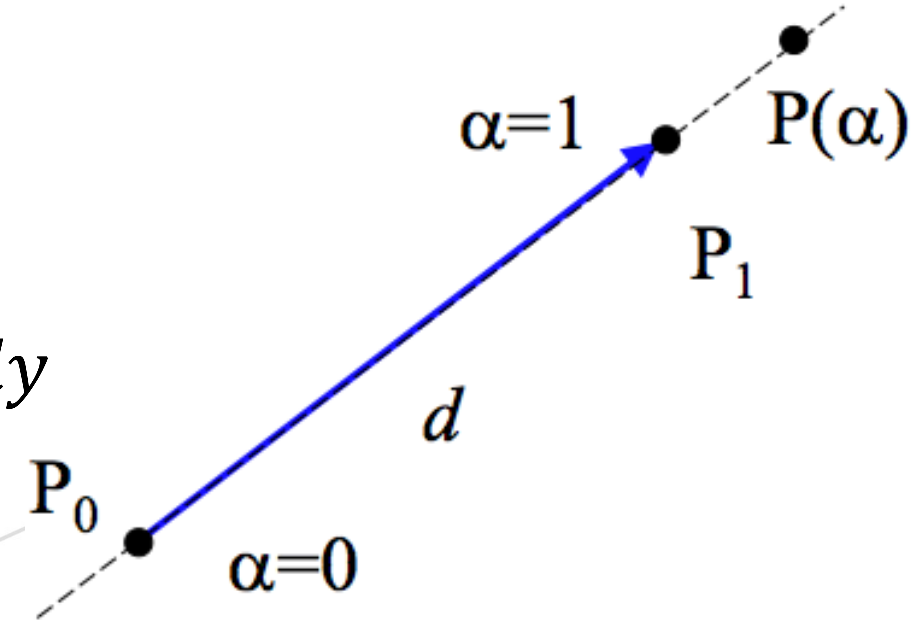
基本几何

线性组合（linear combination）

- 一条直线可表示为
  - $\mathbf{P}(\alpha) = \mathbf{P}_0 + \alpha \cdot \mathbf{d}$
  - 直线的参数化表现形式
    - 更为通用
    - 可应用于曲线及曲面



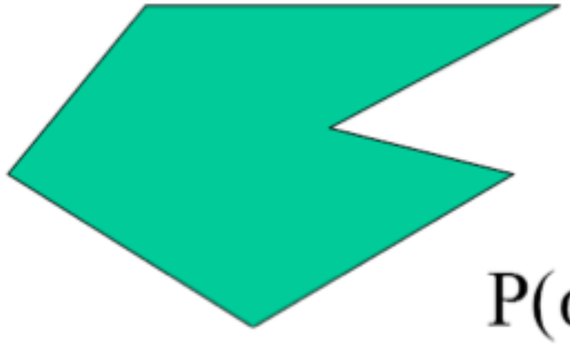
- 二维直线表现形式对比
  - 显示：$y = mx + b$
  - 隐式：$ax + by + c = 0$
  - 参数化：$x(\alpha) = x_0 + \alpha \cdot dx, y(\alpha) = y_0 + \alpha \cdot dy$
- 使用线性组合，一个线段可表示为
  - $\mathbf{P}(\alpha) = \mathbf{P}_0 + \alpha(\mathbf{P}_0 - \mathbf{P}_1) = (1-\alpha)\mathbf{P}_0 + \alpha\mathbf{P}_1$



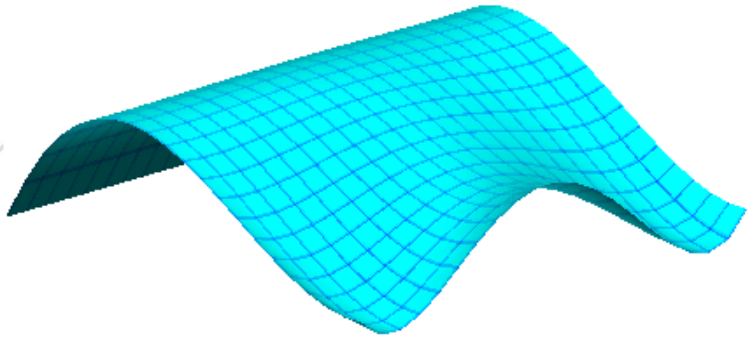
基本几何

- 线性组合（linear combination）
  – 曲线与曲面
    - 曲线是由一个参数所定义的几何体
      – 非线性函数$\mathbf{P}(\alpha)$
    - 曲面是由两个参数所定义的几何体
      – 非线性函数$\mathbf{P}(\alpha,\beta)$
  – 线性函数只能用于表达平面或多边形

P(α)



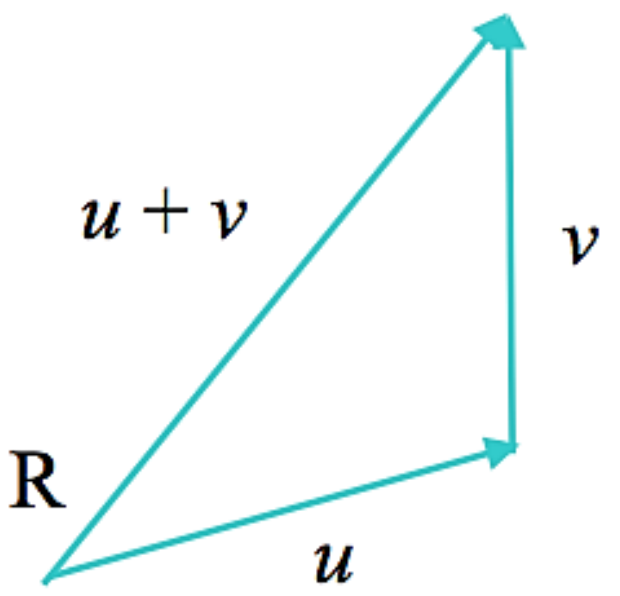
P(α, β)

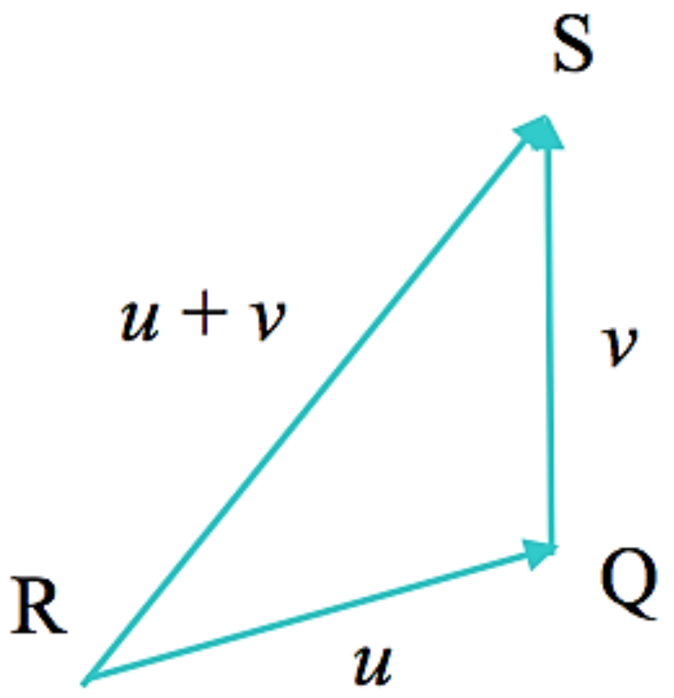


基本几何

线性组合（linear combination）

– 一个平面由一个点与两个向量（或三个点）决定





$P(\alpha, \beta) = R + \alpha u + \beta v$

$P(\alpha, \beta) = R + \alpha(Q - R) + \beta(S - Q)$

课程提纲

- 基本几何概念
- 表现形式
- 变换

表现形式

- 标架（frame）
  - 至此，我们讨论了几何物体，而没使用任何标架
  - 然而，在讨论实际物理世界时，我们需要一个参照点（reference point）和标架（frame）
    - 如，图形学中的世界坐标系
  - n维向量空间由n个线性无关的基向量表示$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$
    - 空间中的任意向量可以由这组基向量的线性组合表示：
      - $\mathbf{v} = s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_n\mathbf{v}_n$
    - 线性组合中所使用的系数$\mathbf{s} = [s_1, s_2, \ldots, s_n]^T$即为向量 $\mathbf{v}$ 在此坐标系下的坐标
      - 如，$\mathbf{v} = 2\mathbf{v}_1 + 3\mathbf{v}_2 - 4\mathbf{v}_3$，则其坐标为$[2, 3, -4]^T$
      - OpenGL中存在多个坐标系统（通过矩阵乘法进行转换）

表现形式

- 标架（frame）
  - 然而，向量空间无法表示点，我们还需要一个参照点（原点）
  - 标架由原点+向量空间定义：$(\mathbf{O}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathrm{n})$
  - 在标架下，一个向量为
    - $\mathbf{v} = s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_\mathrm{n}\mathbf{v}_\mathrm{n}$
  - 而一个点为
    - $\mathbf{P} = \mathbf{O} + s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_\mathrm{n}\mathbf{v}_\mathrm{n}$
  - 在n维坐标系下，无法区分n维点和向量：
    - $\mathbf{v} = [s_1, s_2, \ldots, s_n]^\mathrm{T}$
    - $\mathbf{P} = [s_1, s_2, \ldots, s_n]^\mathrm{T}$
    - 无法表示n维仿射空间

表现形式

- 齐次坐标（homogeneous coordinate）
  - 统一点和向量的表现形式
  - $\mathbf{v} = s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_n\mathbf{v}_n$

    $= [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \mathbf{O}][s_1, s_2, \ldots, s_n, 0]^{\mathrm{T}}$
  - $\mathbf{P} = \mathbf{O} + s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \cdots + s_n\mathbf{v}_n$

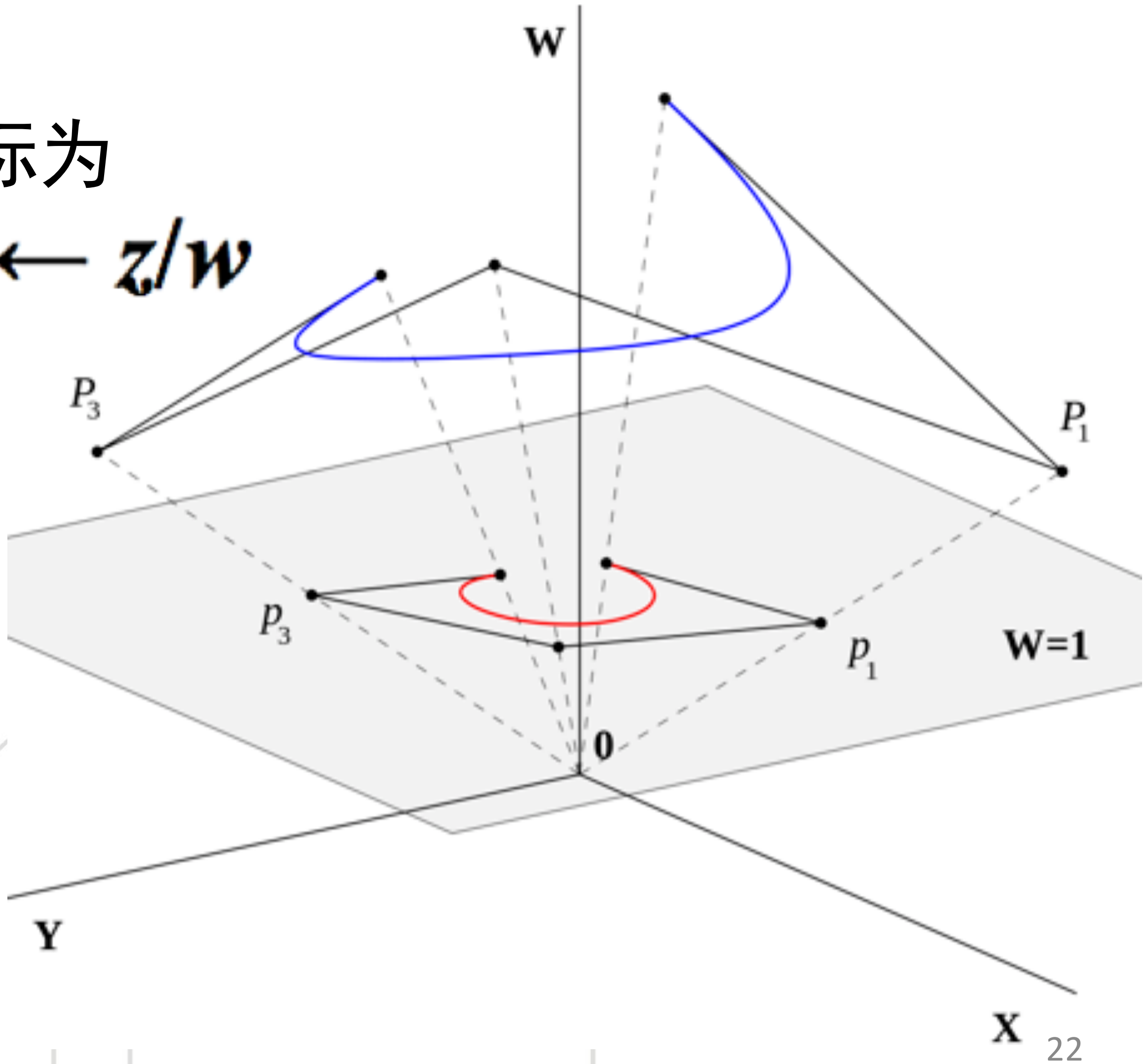
    $= [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \mathbf{O}][s_1, s_2, \ldots, s_n, 1]^{\mathrm{T}}$
  - 因此，n维仿射空间，可以由n+1维齐次坐标表示
    - 另一个作用为表示无穷远点

表现形式

- 齐次坐标（homogeneous coordinate）
  - OpenGL使用4维齐次坐标： $\mathbf{P} = [x, y, z, w]^{\mathrm{T}}$
    - 当w为0时，**P**表示一个向量
    - 当w不为0时，**P**表示一个点，其三维坐标为

$$x \leftarrow x/w, \quad y \leftarrow y/w, \quad z \leftarrow z/w$$

    - 在齐次坐标系中，穿过原点的一条直线对应的是三维空间中的一个点



表现形式

- 齐次坐标（homogeneous coordinate）
  - 齐次坐标是所有计算机图形学的关键
    - 所有标准转换（旋转，平移，缩放）都可以被表示为4x4的矩阵乘法
    - 硬件渲染管线可直接作用于四维表现形式
    - 对于orthographic投影，向量w=0，点w=1
    - 对于perspective投影，齐次坐标可用于处理perspective division

$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

表现形式

坐标系变换

– 考虑一下两组基向量：$\boldsymbol{a} = [\alpha_1, \alpha_2, \alpha_3]^T$

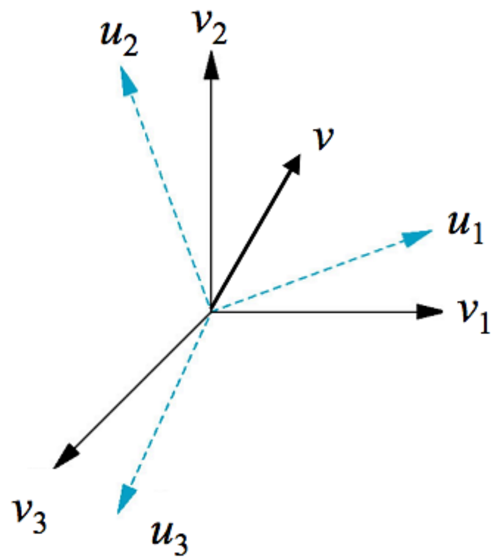
$$\boldsymbol{b} = [\beta_1, \beta_2, \beta_3]^T$$

– 一个向量使用这两组基向量表示为

$$\begin{aligned} \boldsymbol{v} &= \alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + \alpha_3 \boldsymbol{v}_3 = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3]\,[\alpha_1, \alpha_2, \alpha_3]^T \\ &= \beta_1 \boldsymbol{u}_1 + \beta_2 \boldsymbol{u}_2 + \beta_3 \boldsymbol{u}_3 = [\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3]\,[\beta_1, \beta_2, \beta_3]^T \end{aligned}$$

– 若使用一组向量基表示另一组：

$$\begin{aligned} \boldsymbol{u}_1 &= \gamma_{11} \boldsymbol{v}_1 + \gamma_{12} \boldsymbol{v}_2 + \gamma_{13} \boldsymbol{v}_3 \\ \boldsymbol{u}_2 &= \gamma_{21} \boldsymbol{v}_1 + \gamma_{22} \boldsymbol{v}_2 + \gamma_{23} \boldsymbol{v}_3 \\ \boldsymbol{u}_3 &= \gamma_{31} \boldsymbol{v}_1 + \gamma_{32} \boldsymbol{v}_2 + \gamma_{33} \boldsymbol{v}_3 \end{aligned}$$

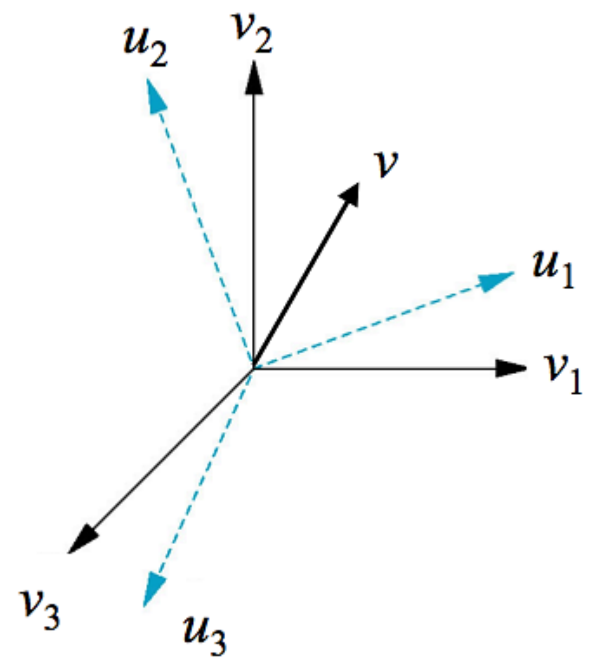


表现形式

坐标系变换

- 若使用一组向量基表示另一组：

$$u_1 = \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3$$
$$u_2 = \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3$$
$$u_3 = \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3$$



- 则3x3的系数矩阵可用于两组基向量对应的坐标系之间的变换

$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix} \qquad \mathbf{a} = M^T \mathbf{b}$$

表现形式

- 坐标系变换
  - 类似变换也可以用于齐次坐标
  - 假设有两个标架，可表示三维空间中的任意一个向量或点

$$(P_0, v_1, v_2, v_3)$$
$$(Q_0, u_1, u_2, u_3)$$

  - 使用其中一个标架表示另一个标架

$$u_1 = \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3$$
$$u_2 = \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3$$
$$u_3 = \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3$$
$$Q_0 = \gamma_{41} v_1 + \gamma_{42} v_2 + \gamma_{43} v_3 + P_0$$

表现形式

坐标系变换

– 则4x4的系数矩阵M可用于表示两个标架间的变换

$$\mathbf{M}=\begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix} \qquad \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

– 而标架下两个点或向量的齐次坐标变换也可以由矩阵M完成

$$\mathbf{b}^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ Q_0 \end{bmatrix} = \mathbf{b}^T\mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix} = \mathbf{a}^T \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ P_0 \end{bmatrix}$$

表现形式

坐标系变换

– 在OpenGL中，点/向量被表示为列向量，因此实际使用的为$\mathbf{M}$的转置矩阵$\mathbf{M}^{\mathrm{T}}$

- $\mathbf{a} = \mathbf{M}^{\mathrm{T}}\mathbf{b}$
- 由12个可变化的系数组成（另外4个系数为固定值）

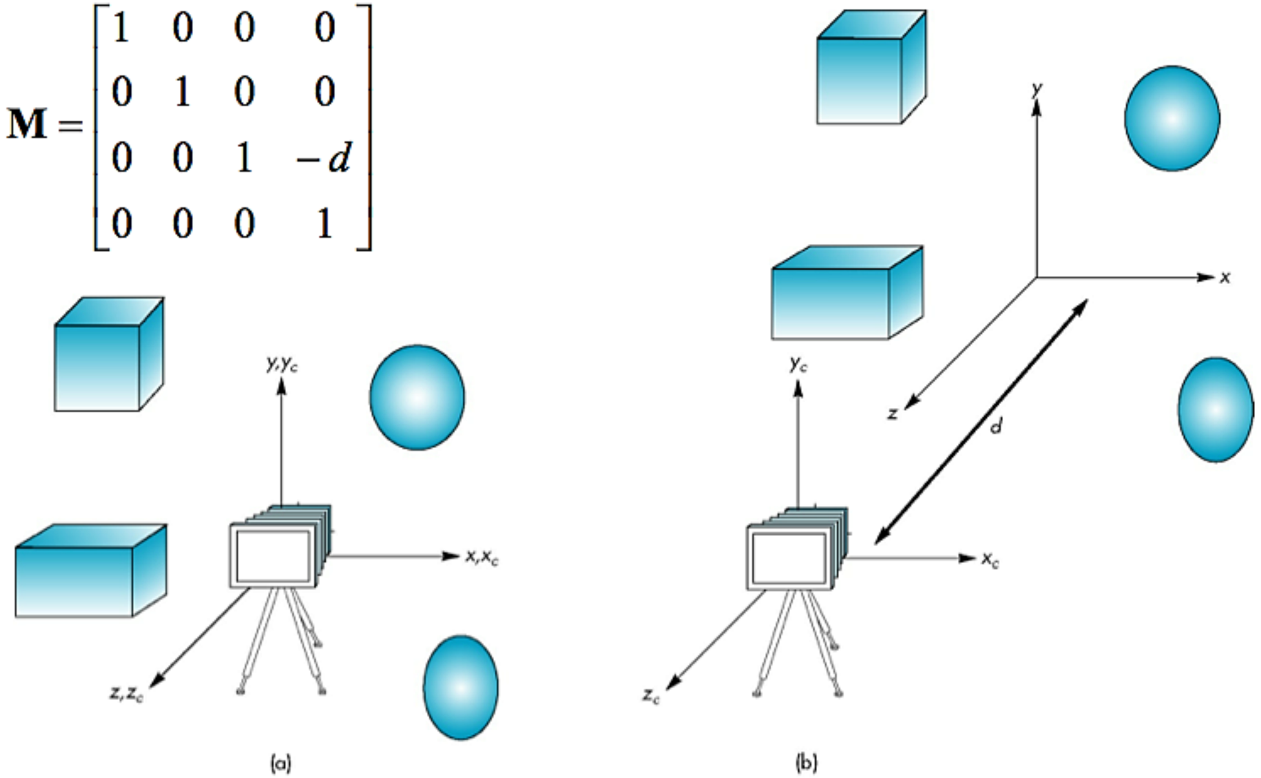
$$\mathrm{M}^T = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

表现形式

仿射变换的优点

- 所有仿射变换都能表示为齐次坐标下的矩阵乘法
  - 统一了点和向量的变换，使硬件能高效计算三维下的几乎所有重要变换
  - 后续的变换能轻易通过矩阵相乘叠加
- 如，使用矩阵表示摄像机的移动

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



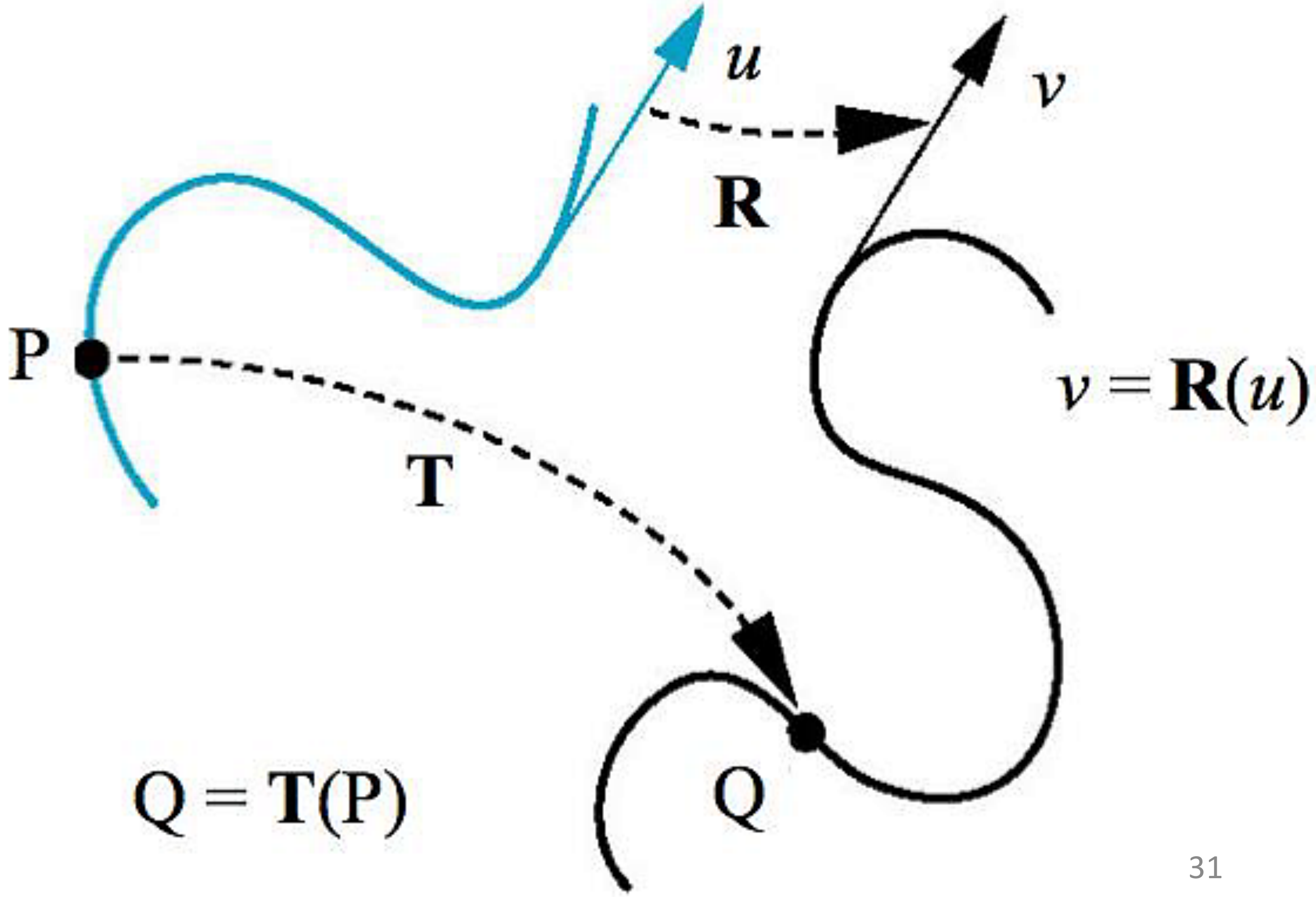
课程提纲

- 基本几何概念
- 表现形式
- 变换

变换

广义的变换

- 广义的变换指的是从点到点，或从向量到向量的映射
- 由于几何物体在计算机上都最终表示为点集，因此，了解如何对点变换就能够对物体进行变换
  - 计算物体的新位置
  - 将物体从一个坐标系统转换到另一个坐标系统



变换

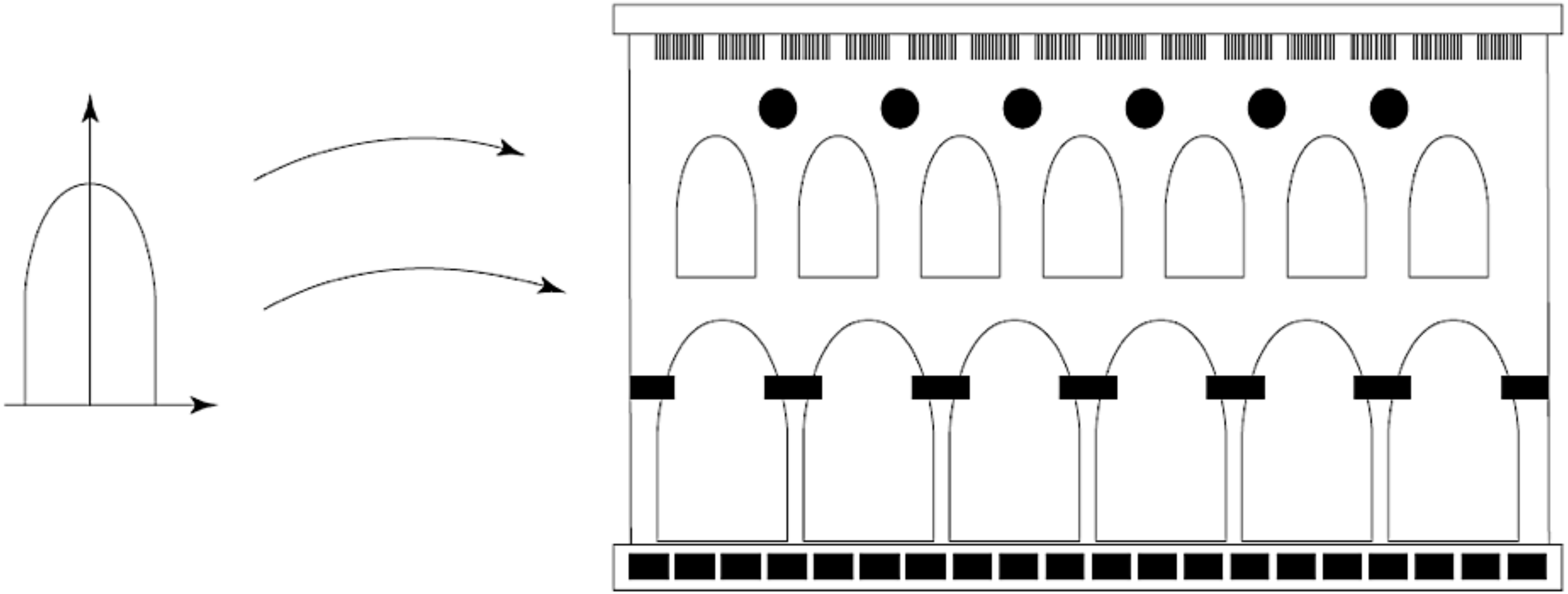
仿射变换

- 包含了多种重要的物理世界的变换
  - rotation，translation，scaling，shearing
- 我们只需要对直线的端点进行变换
  - 其他点在变换后通过重新连接端点即可得到（仿射变换具有共线性）

Type Preserves	Rigid Body: Rotation & translation	Linear: General 3x3 matrix	Affine: Linear + translation	Projective: 4x4 matrix with last row ≠(0,0,0,1)
Lengths	Yes	No	No	No
Angles	Yes	No	No	No
Parallelness	Yes	Yes	Yes	No
Straight lines	Yes	Yes	Yes	Yes

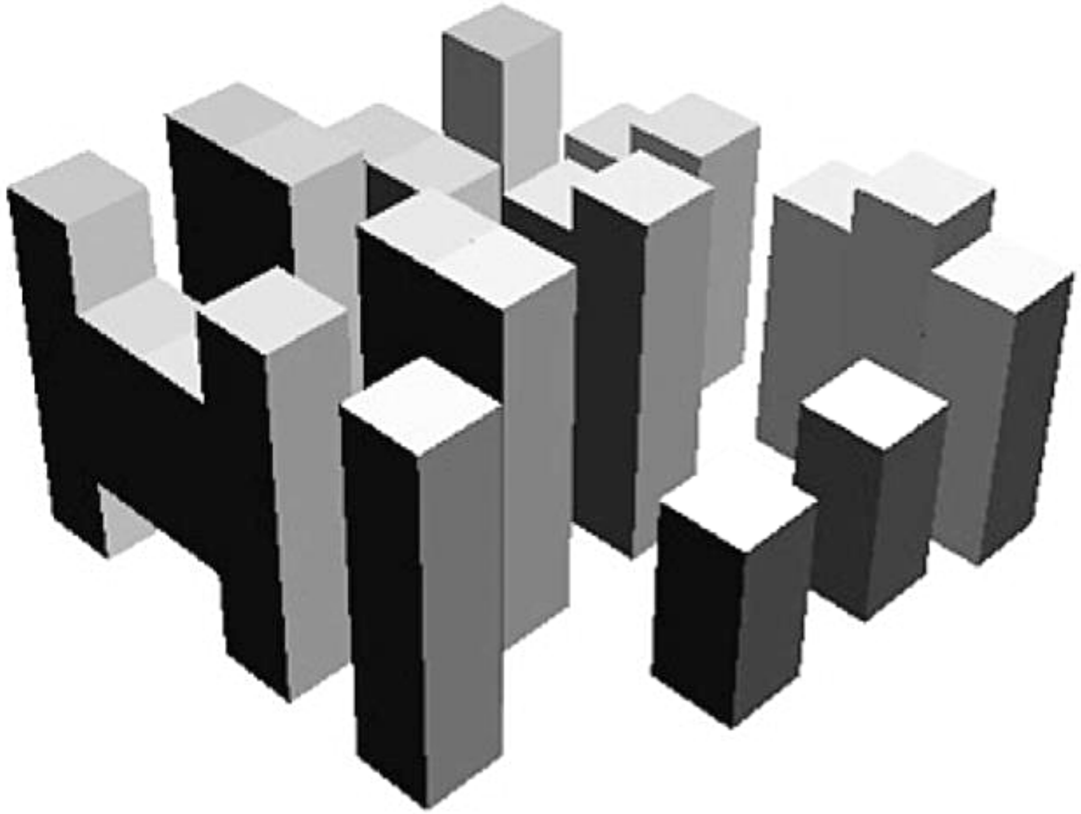
变换

- 为什么需要使用变换？
  – 构建场景



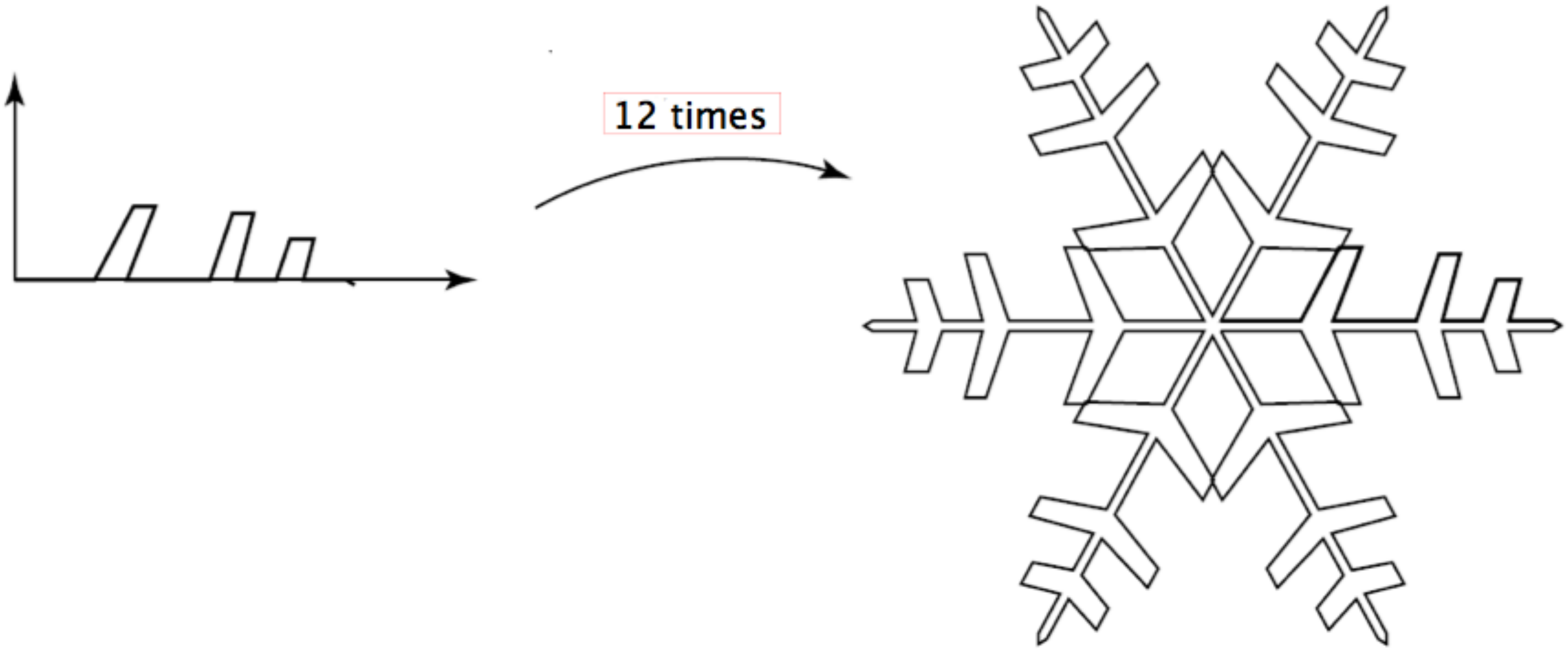
变换

- 为什么需要使用变换？
  - 构建场景

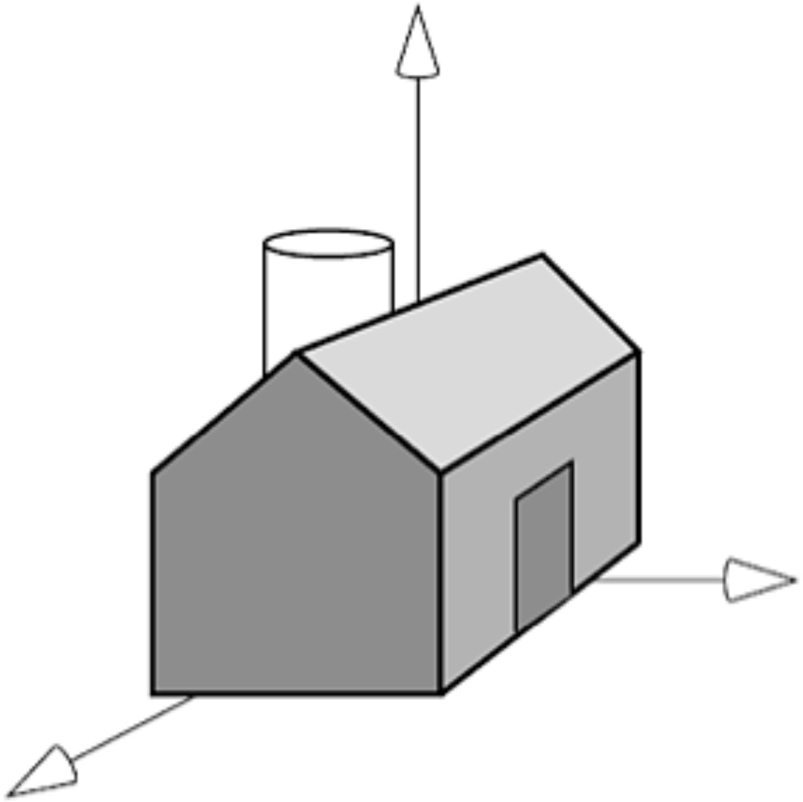


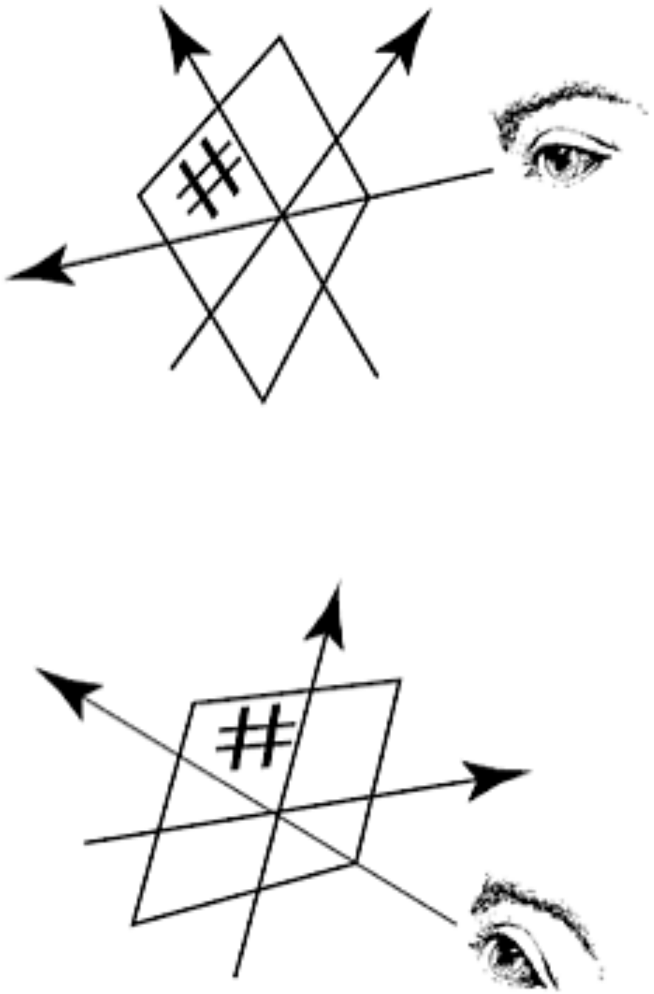
变换

- 为什么需要使用变换？
  - 构建场景



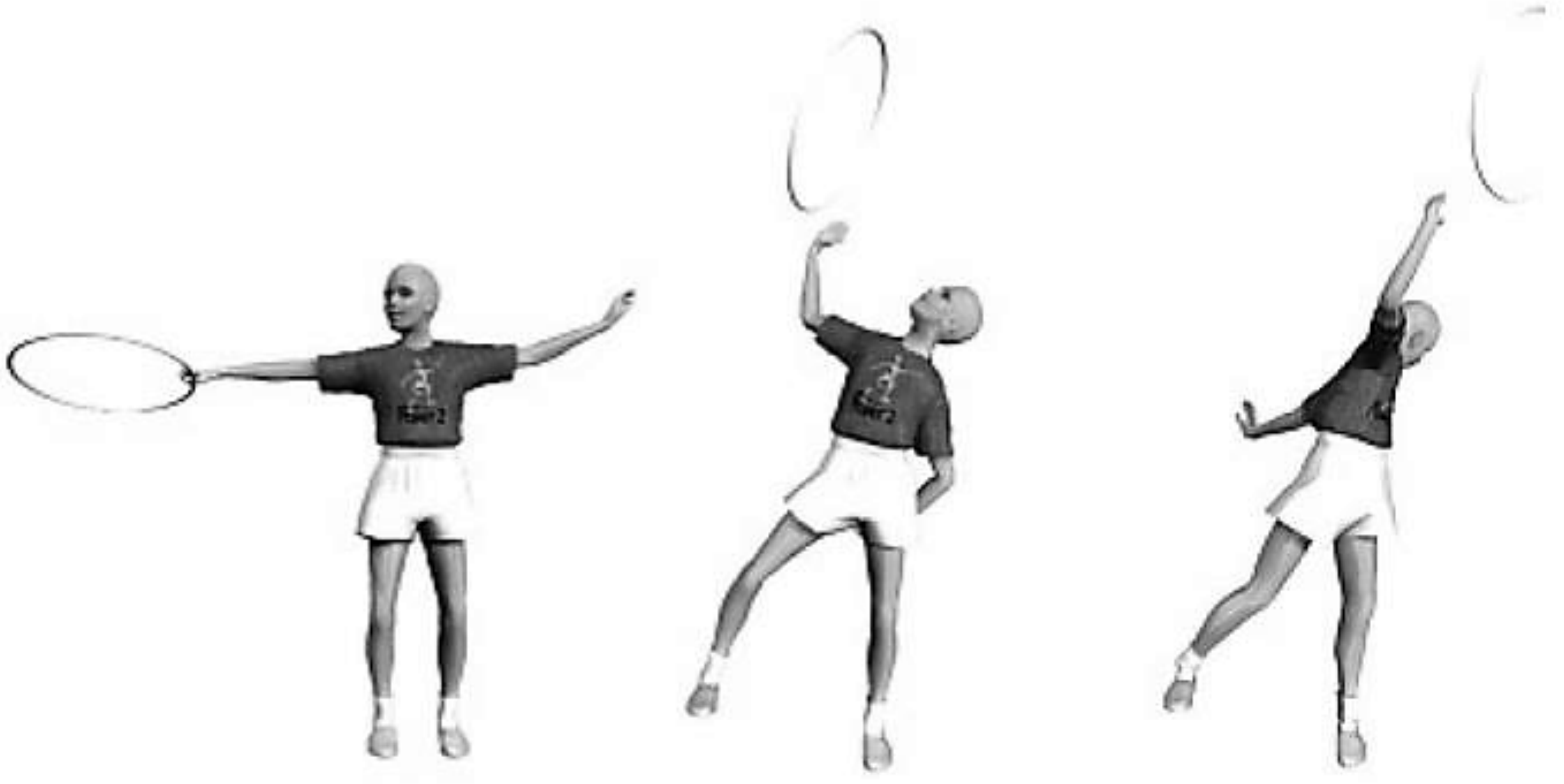
- 为什么需要使用变换？
  - 从不同角度观察物体：物体不变，摄像机位置、朝向改变





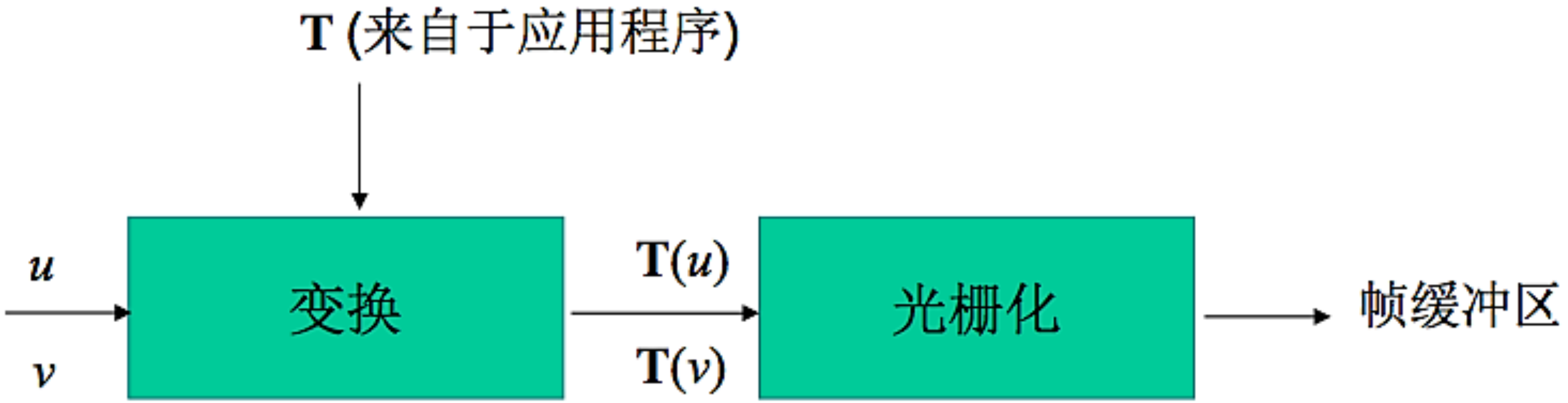
变换

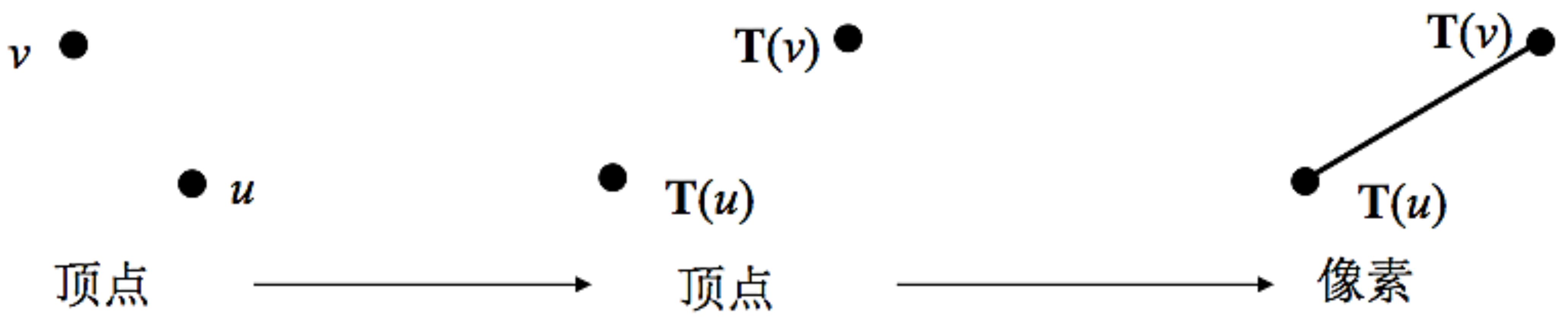
- 为什么需要使用变换？
  - 在计算机动画中，相邻帧之间，物体的相对位置发生变化
    - 这通常是由局部坐标系统的平移、旋转等变换完成的



变换

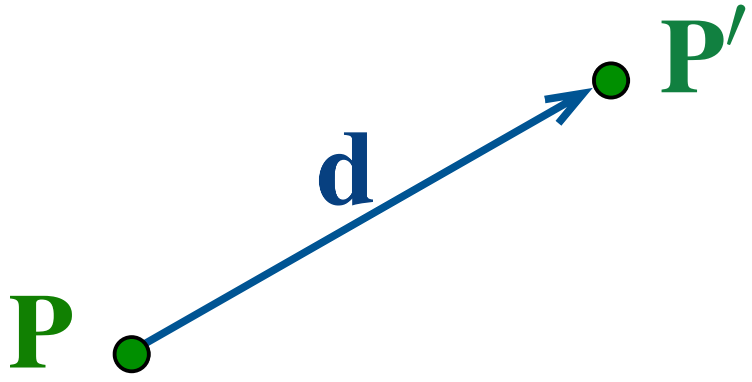
渲染管线

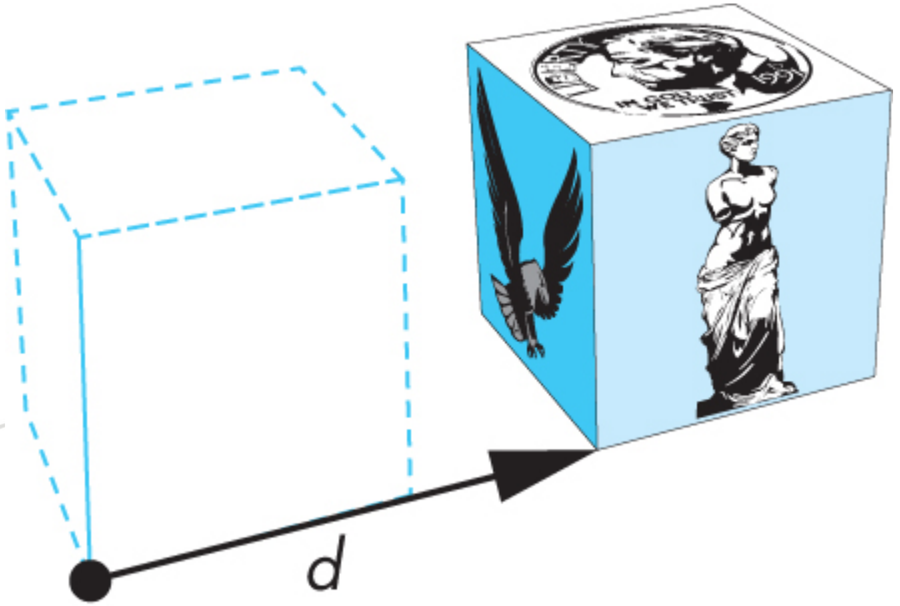




平移变换

- 将点从一个位置移动至另一个位置
  - $\mathbf{P}' = \mathbf{P} + \mathbf{d}$
  - 由向量**d**决定，具有3自由度(x, y, z)
  - 涉及点与向量的加法
  - 在齐次坐标中
    - $\mathbf{P} = [x, y, z, 1]^T \rightarrow \mathbf{P}' = [x', y', z', 1]^T$
    - $\mathbf{d} = \left[d_x, d_y, d_z, 0\right]^T$
  - 平移后可得
    - $x' = x + d_x$
    - $y' = y + d_y$
    - $z' = z + d_z$





平移变换

- 4x4齐次坐标矩阵T表示平移

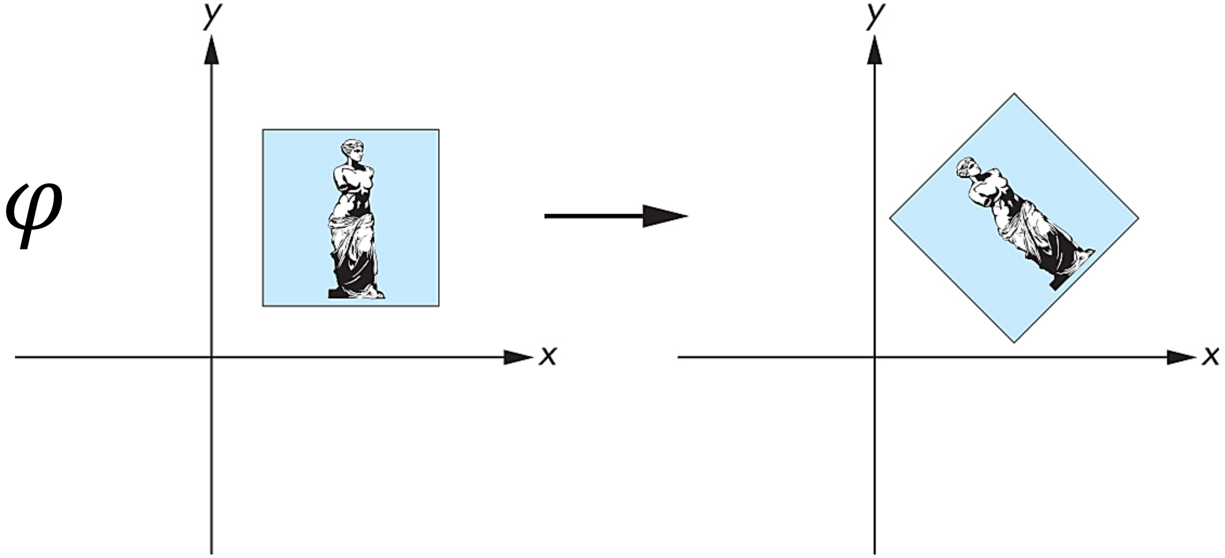
$$\mathbf{T}=\mathbf{T}(d_x,d_y,d_z)=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}$$

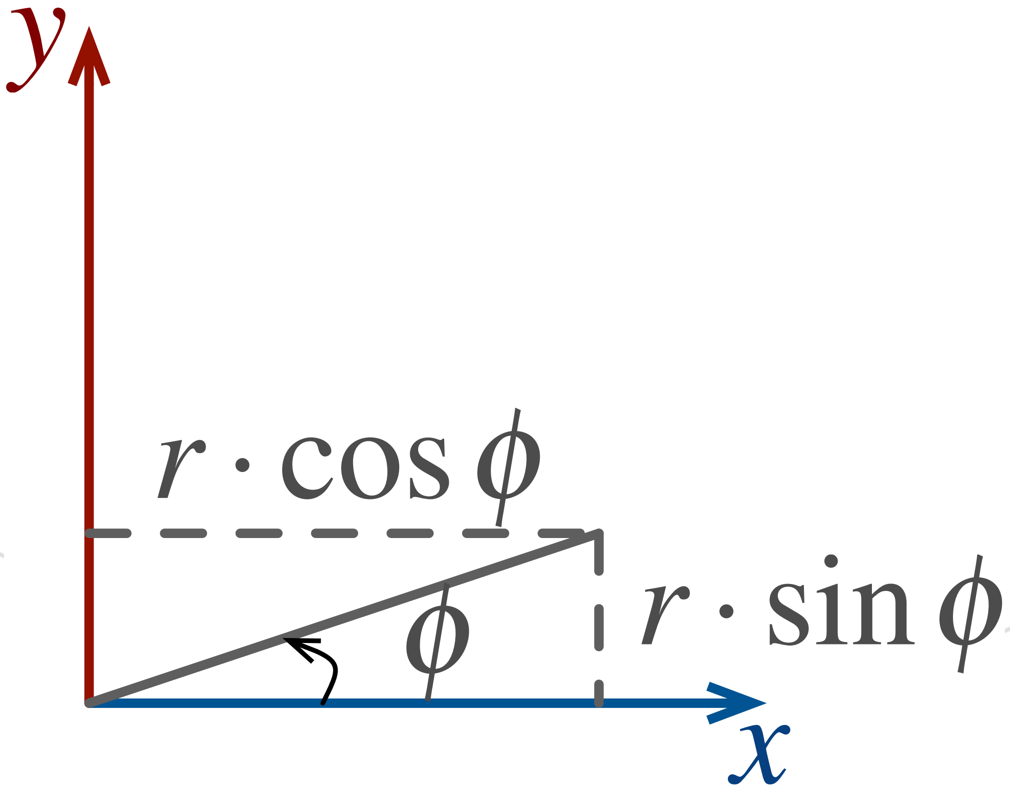
– 点的平移表示为矩阵相乘

$$\mathbf{P}'=\mathbf{TP}=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}x+\begin{bmatrix}0\\1\\0\\0\end{bmatrix}y+\begin{bmatrix}0\\0\\1\\0\end{bmatrix}z+\begin{bmatrix}d_x\\d_y\\d_z\\1\end{bmatrix}1=\begin{bmatrix}x+d_x\\y+d_y\\z+d_z\\1\end{bmatrix}$$

2D旋转

- 以原点为旋转轴，将向量逆时针旋转φ
  - 线性变换
  - 若该向量沿x轴：$\mathbf{v} = (r, 0)$
    - $\mathbf{v}' = (r \cdot \cos\varphi,\ r \cdot \sin\varphi)$





旋转变换

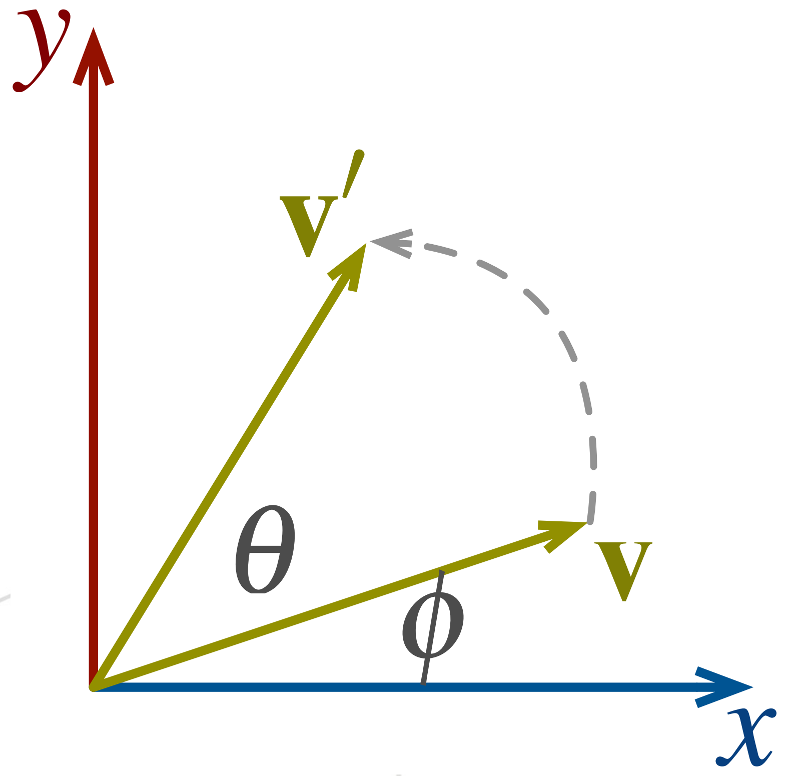
2D旋转

- 以原点为旋转轴，将向量逆时针旋转θ
  - 对任意长度为r的向量：$\mathbf{v} = (x, y) = (r \cdot \cos\varphi,\ r \cdot \sin\varphi)$
    - $\mathbf{v}' = \big(r \cdot \cos(\varphi + \theta),\ r \cdot \sin(\varphi + \theta)\big)$

$$= \big(r(\cos\varphi\cos\theta - \sin\varphi\sin\theta), r(\sin\varphi\cos\theta + \cos\varphi\sin\theta)\big)$$

$$= (x \cdot \cos\theta - y \cdot \sin\theta, x \cdot \sin\theta + y \cdot \cos\theta)$$

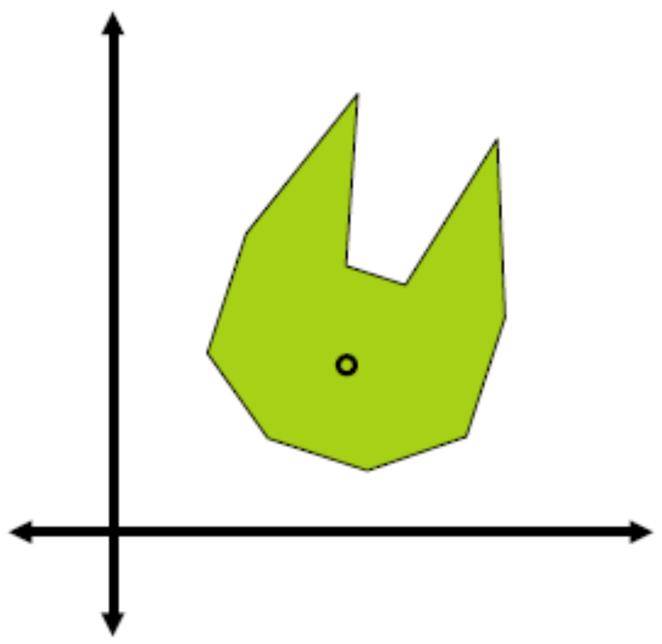
$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



旋转变换

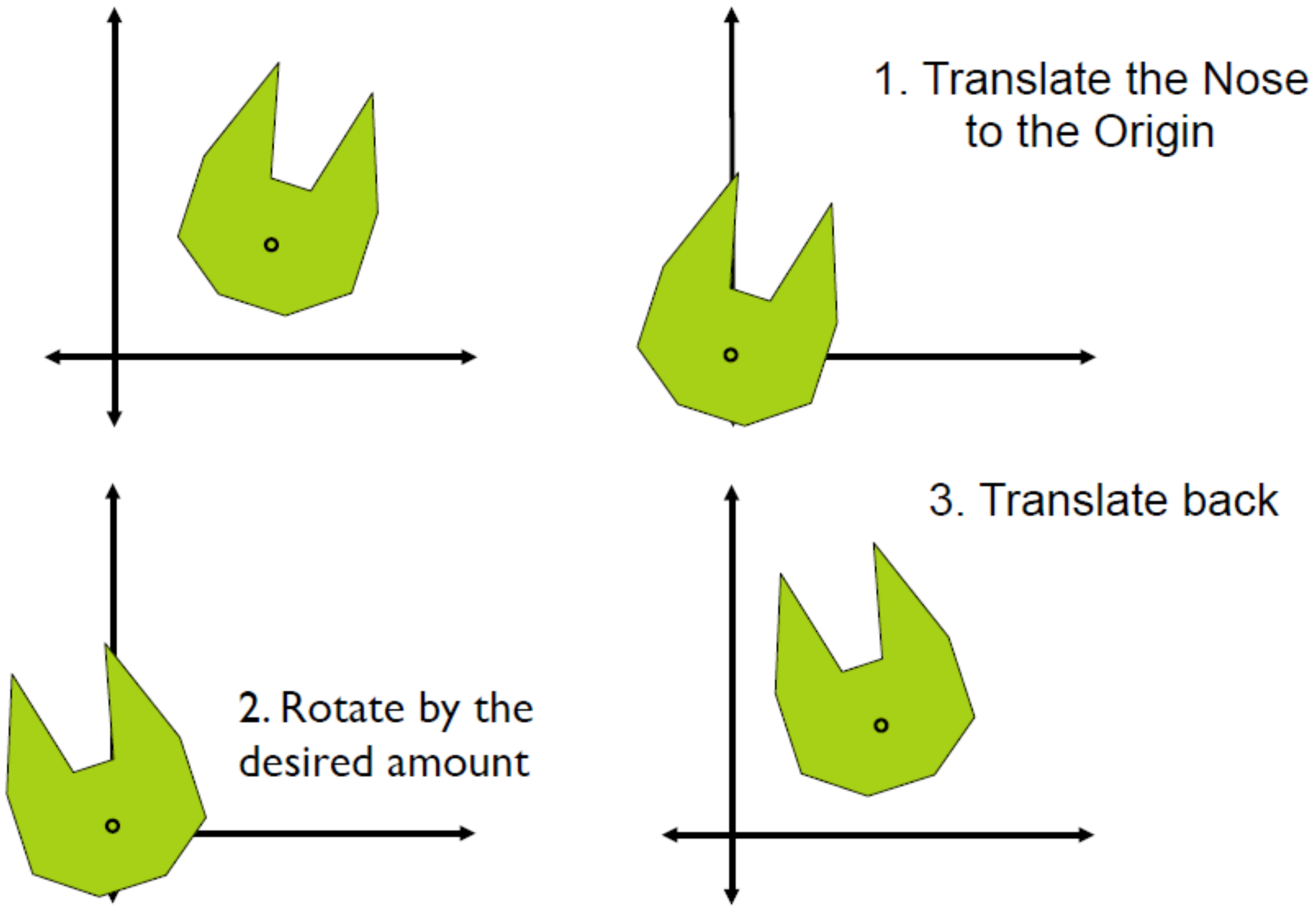
- 2D旋转
  - 以任意点为轴进行旋转

To rotate the cat's head about its nose

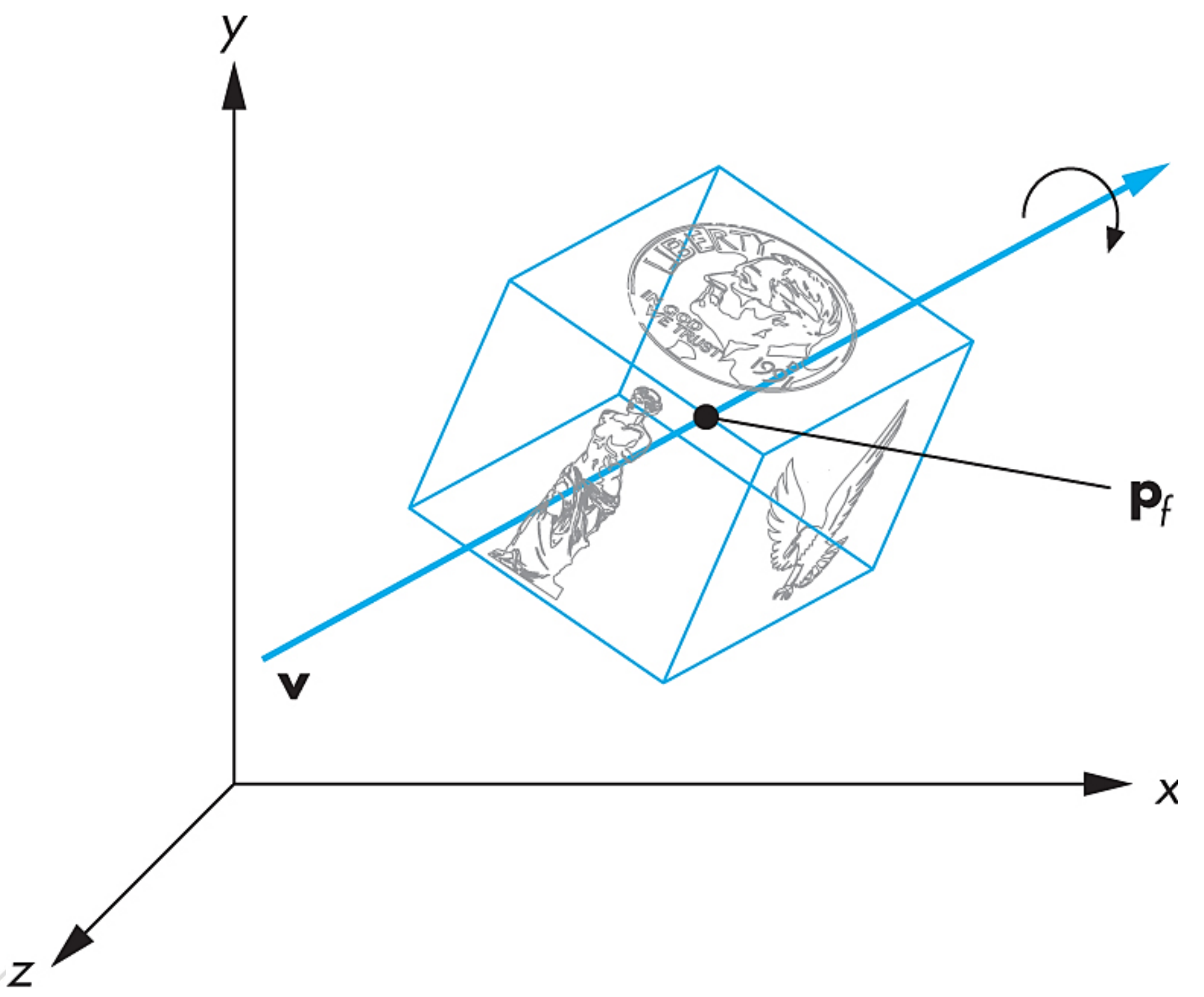


旋转变换

- 2D旋转
  - 以任意点为轴进行旋转：是否为线性变换？
    - $\mathbf{T}(-\mathbf{P})$
    - $\mathbf{R}\ (\theta)$
    - $\mathbf{T}(\mathbf{P})$



- 3D旋转
  - 分以下几种情况考虑：
    - 以x，y，z轴为旋转轴
    - 以经过原点的任意向量为旋转轴
    - 以不经过原点的任意向量为旋转轴



旋转变换

- 3D旋转
  - 以z轴为旋转轴时，z坐标在旋转过程中不发生改变
    - $x' = x \cdot \cos\theta - y \cdot \sin\theta$
    - $y' = x \cdot \sin\theta + y \cdot \cos\theta$
    - $z' = z$
  - 齐次坐标表示：$\mathbf{P}' = \mathbf{R}_z(\theta)\mathbf{P}$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

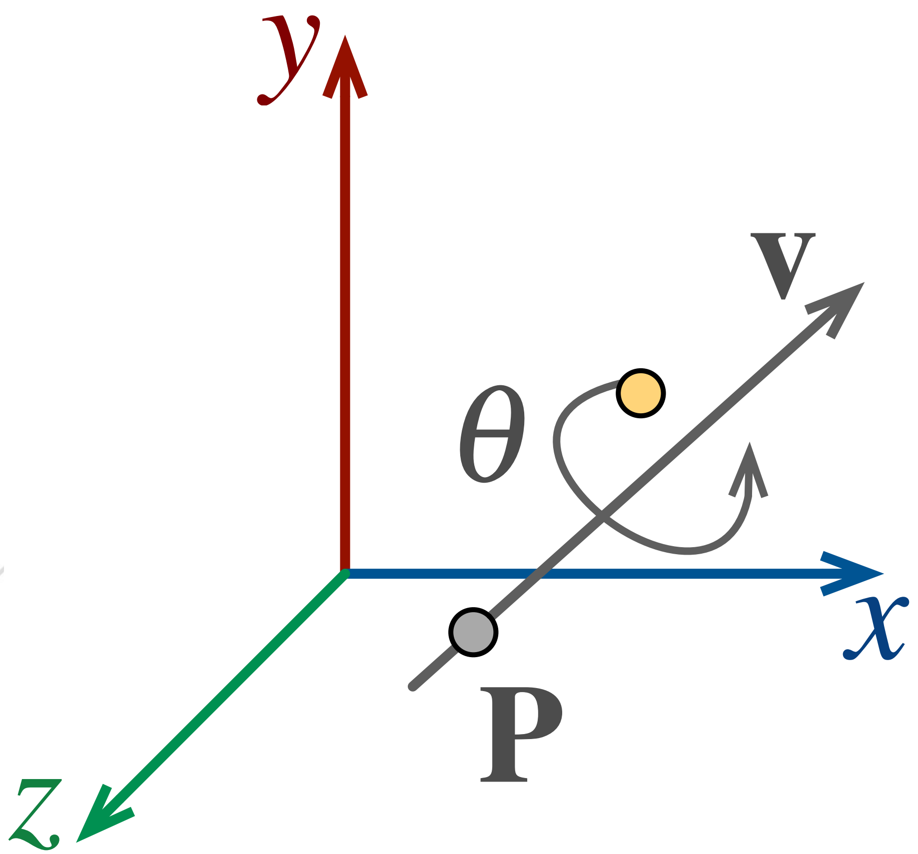
旋转变换

- 3D旋转
  - 同理，以x(y)为旋转轴的3D旋转变换时，x(y)坐标不发生改变

$$\mathbf{R}_x(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\qquad \mathbf{R}_y(\theta)=\begin{bmatrix}\cos\theta & 0 & \sin\theta & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

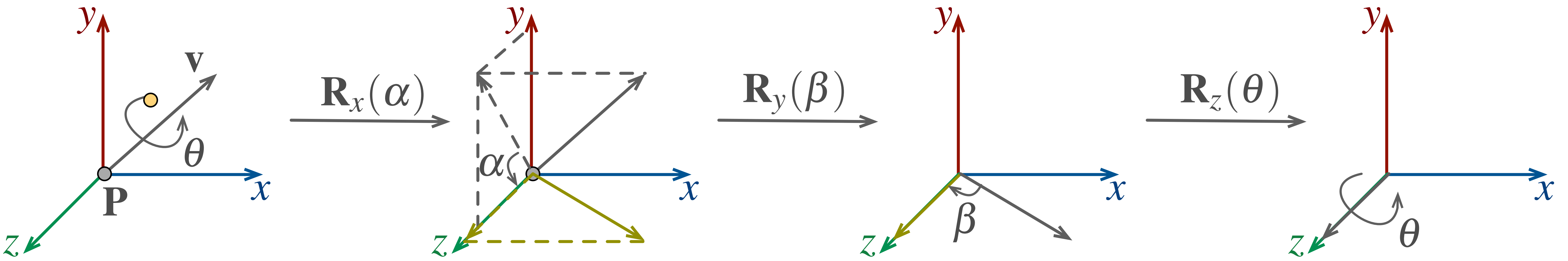
3D旋转

- 以过原点的任意向量为轴旋转
  - 不能通过三个轴上的旋转直接叠加！
  - 通过旋转将旋转轴与z轴对齐
  - 通过构建变换后的基向量
  - 使用四元数（quaternion）



旋转变换

- 以过原点的任意向量为轴旋转：通过旋转将旋转轴与z轴对齐
  - $\mathbf{R_V}(\theta)=\mathbf{R}_x(-\alpha)\mathbf{R}_y(-\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)$



以 x 为轴旋转（将旋转轴置于xOz平面）

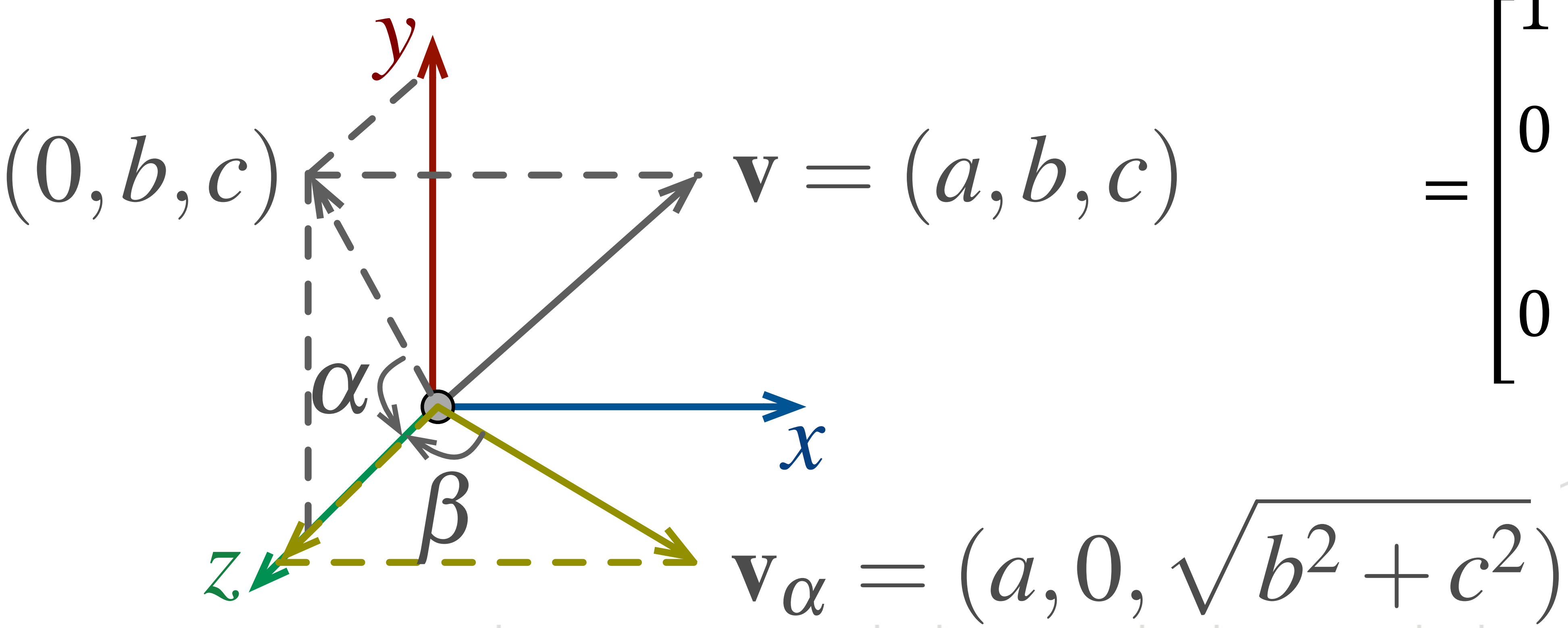
以 y 为轴旋转（旋转轴与z轴重合）

以 z 为轴旋转（实际需要执行的旋转）

旋转变换

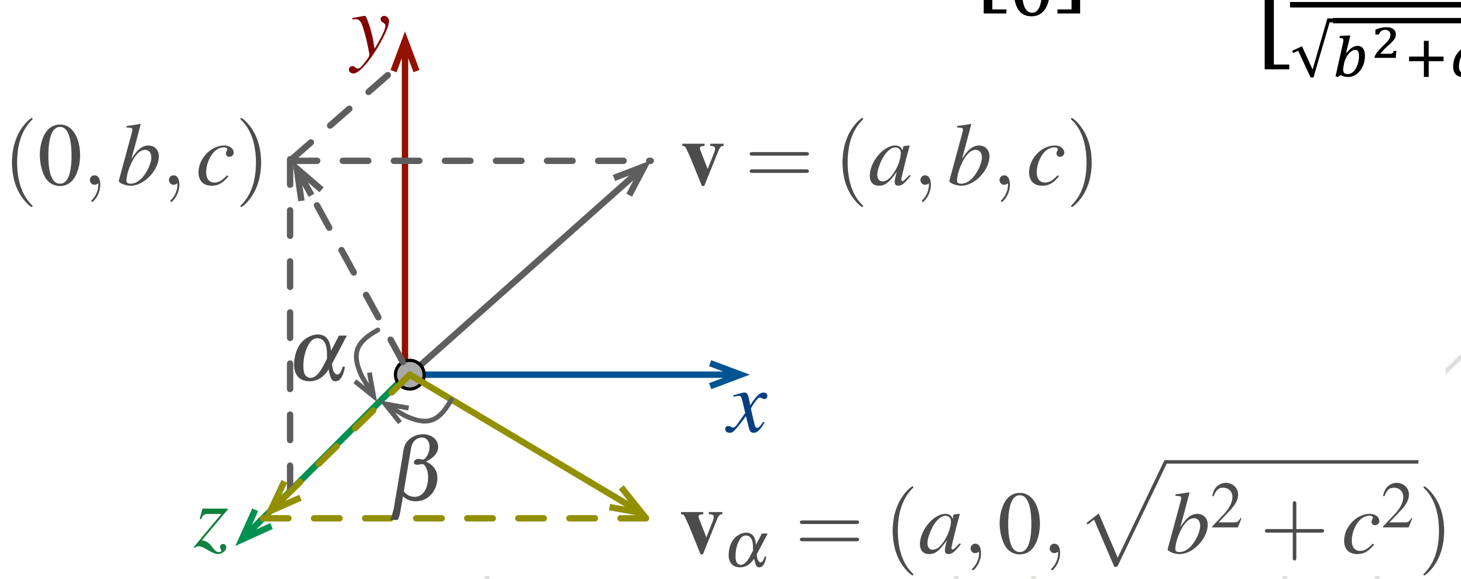
- 以过原点的任意向量为轴旋转：通过旋转将旋转轴与z轴对齐
  - $\mathbf{R}_{\mathbf{v}}(\theta) = \mathbf{R}_x(-\alpha)\mathbf{R}_y(-\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)$
  - $\sin\alpha = \frac{b}{\sqrt{b^2+c^2}}$， $\cos\alpha = \frac{c}{\sqrt{b^2+c^2}}$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \dfrac{c}{\sqrt{b^2+c^2}} & -\dfrac{b}{\sqrt{b^2+c^2}} \\ 0 & \dfrac{b}{\sqrt{b^2+c^2}} & \dfrac{c}{\sqrt{b^2+c^2}} \end{bmatrix}$$



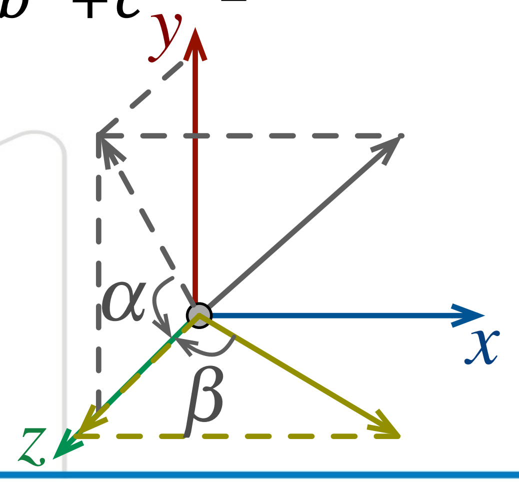
旋转变换

- 以过原点的任意向量为轴旋转：通过旋转将旋转轴与z轴对齐
  - $\mathbf{R_V}(\theta) = \mathbf{R}_x(-\alpha)\mathbf{R}_y(-\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)$
  - $\sin\alpha = \frac{b}{\sqrt{b^2+c^2}}$，$\cos\alpha = \frac{c}{\sqrt{b^2+c^2}}$
  - 验证：$\mathbf{v}_\alpha = \mathbf{R}_x(\alpha)\mathbf{v} = \begin{bmatrix}1\\0\\0\end{bmatrix} a + \begin{bmatrix}0\\ \frac{c}{\sqrt{b^2+c^2}}\\ \frac{b}{\sqrt{b^2+c^2}}\end{bmatrix} b + \begin{bmatrix}0\\ -\frac{b}{\sqrt{b^2+c^2}}\\ \frac{c}{\sqrt{b^2+c^2}}\end{bmatrix} c = \begin{bmatrix}a\\0\\ \sqrt{b^2+c^2}\end{bmatrix}$



旋转变换

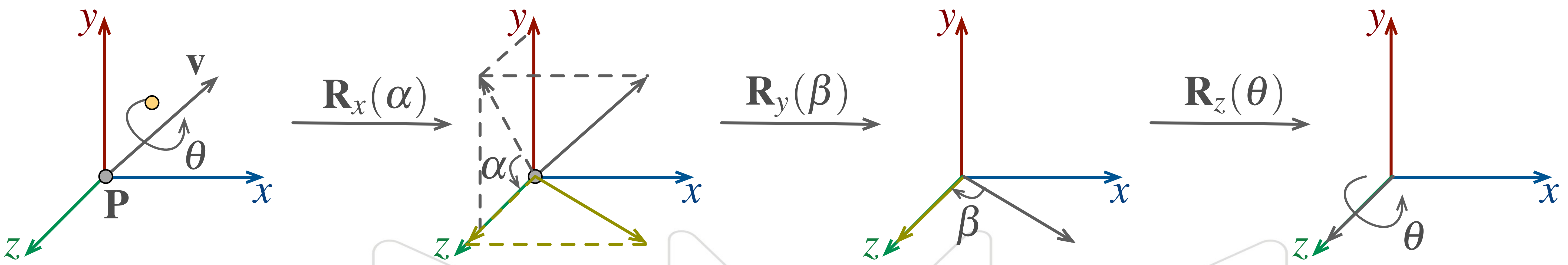
- 以过原点的任意向量为轴旋转：通过旋转将旋转轴与z轴对齐
  - $\mathbf{R}_{\mathrm{V}}(\theta) = \mathbf{R}_x(-\alpha)\mathbf{R}_y(-\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)$
  - $\sin\beta = -\frac{a}{\sqrt{a^2+b^2+c^2}}$，$\cos\beta = \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}}$
  - $\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}} & 0 & -\frac{a}{\sqrt{a^2+b^2+c^2}} \\ 0 & 1 & 0 \\ \frac{a}{\sqrt{a^2+b^2+c^2}} & 0 & \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}} \end{bmatrix}$
  - $\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$



旋转变换

- 以过原点的任意向量为轴旋转：通过旋转将旋转轴与z轴对齐

$$
\begin{aligned}
\mathbf{R}_{\mathbf{V}}(\theta) &= \mathbf{R}_x(-\alpha)\mathbf{R}_y(-\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha) \\
&= \begin{bmatrix} a^2(1-\cos\theta)+\cos\theta & ab(1-\cos\theta)+c\cdot\sin\theta & ac(1-\cos\theta)-b\sin\theta \\ ab(1-\cos\theta)-c\cdot\sin\theta & b^2(1-\cos\theta)+\cos\theta & bc(1-\cos\theta)+a\cdot\sin\theta \\ ac(1-\cos\theta)+b\cdot\sin\theta & bc(1-\cos\theta)-a\cdot\sin\theta & c^2(1-\cos\theta)+\cos\theta \end{bmatrix}
\end{aligned}
$$



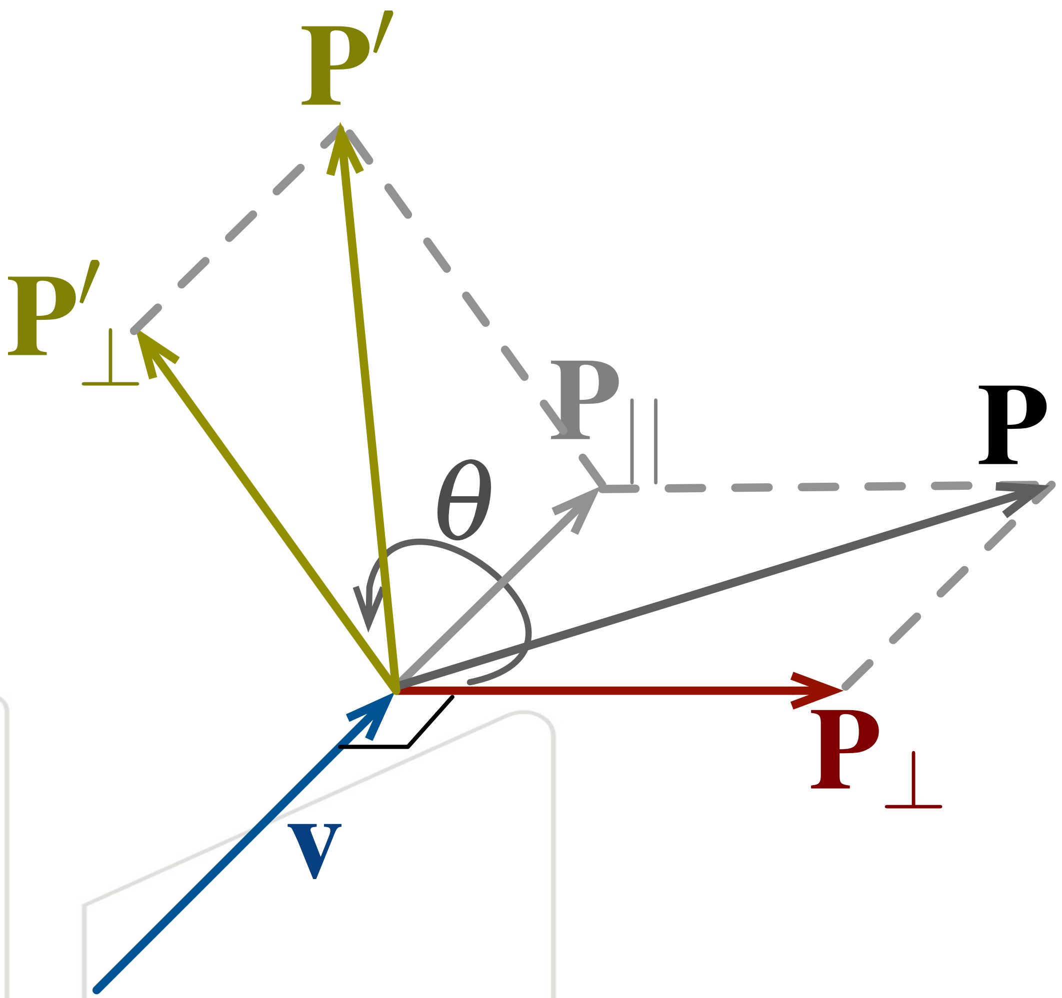
以 x 为轴旋转（将旋转轴置于 xOz 平面）

以 y 为轴旋转（旋转轴与 z 轴重合）

以 z 为轴旋转（实际需要执行的旋转）

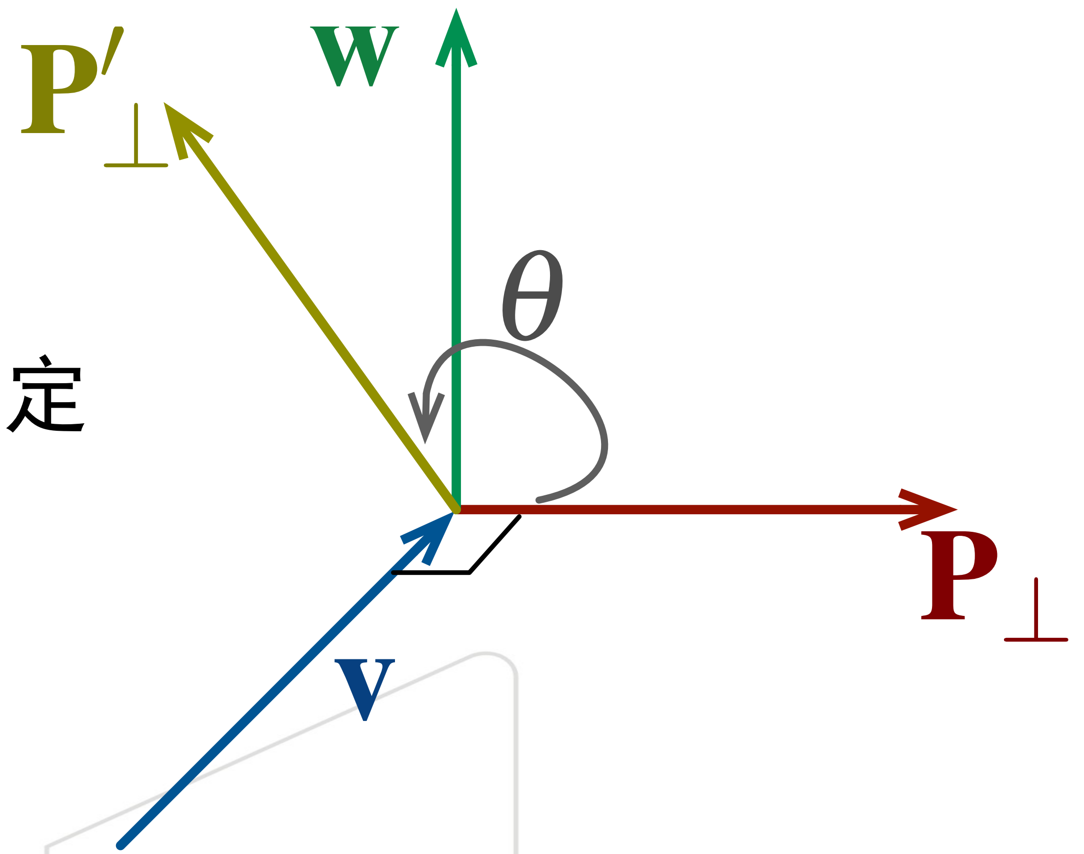
旋转变换

- 以过原点的任意向量为轴旋转：构建变换后的基向量
  - 将$\mathbf{P}$分解为平行于旋转轴$\mathbf{v}$的向量$\mathbf{P}_{\perp}$与垂直于旋转轴的向量$\mathbf{P}_{||}$
  - $\mathbf{P}_{||}$与旋转无关
  - 因此，只需将$\mathbf{P}_{\perp}$旋转θ得到$\mathbf{P}'_{\perp}$
  - 则$\mathbf{P}' = \mathbf{P}_{||} + \mathbf{P}'_{\perp}$



旋转变换

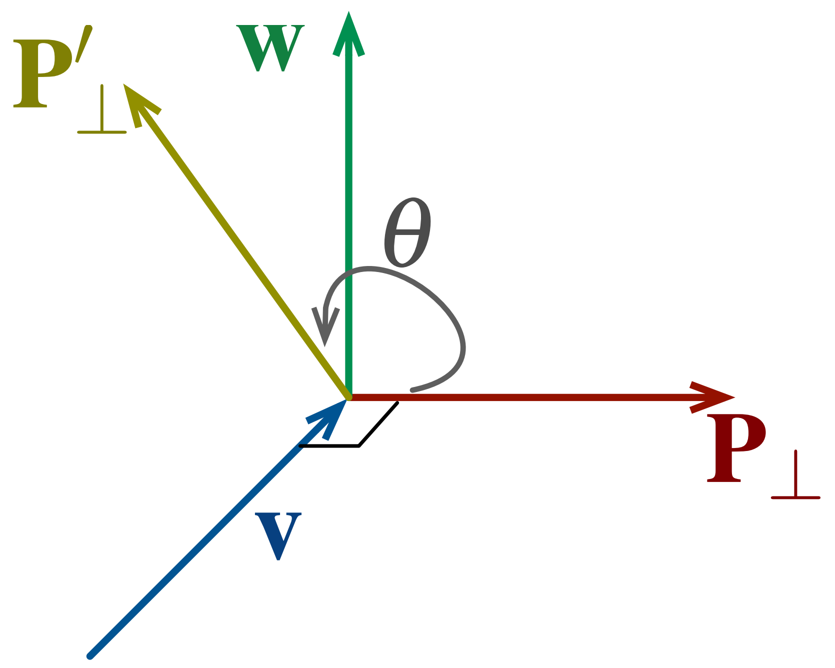
- 以过原点的任意向量为轴旋转：构建变换后的基向量
  - 将$\mathbf{P}$投影至$\mathbf{v}$可得$\mathbf{P}_{||}$
    - $\mathbf{P}_{||} = (\mathbf{P} \cdot \mathbf{v})\mathbf{v}$
  - 将$\mathbf{P}_{||}$由$\mathbf{P}$中减去可得$\mathbf{P}_{\perp}$
    - $\mathbf{P}_{\perp} = \mathbf{P} - \mathbf{P}_{||}$
  - 旋转在垂直于$\mathbf{v}$的平面中进行
  - 该平面的一组基底可由$\mathbf{P}_{\perp}$及另一向量$\mathbf{w}$确定
    - $\mathbf{w} = \mathbf{v} \times \mathbf{P}_{\perp}$
  - $\mathbf{P}'_{\perp}$可由平面旋转得到
    - $\mathbf{P}'_{\perp} = \cos\theta\, \mathbf{P}_{\perp} + \sin\theta\, \mathbf{w}$



旋转变换

以过原点的任意向量为轴旋转：构建变换后的基向量

– 令$\mathbf{P} = [1,0,0]^T, \mathbf{v} = (a,b,c)$，则



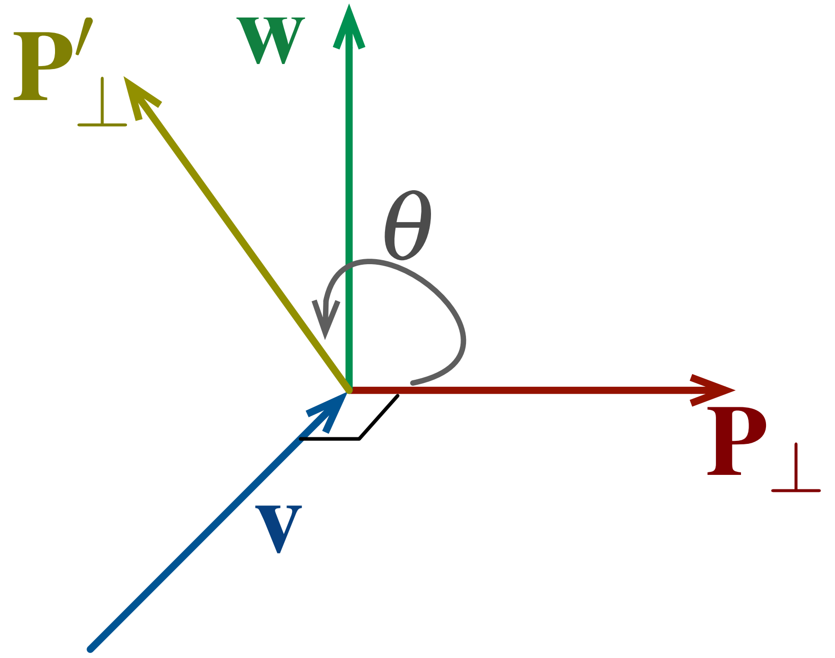
- $\mathbf{P}_{||} = (\mathbf{P}\cdot\mathbf{v})\mathbf{v} = \left(\begin{bmatrix}1\\0\\0\end{bmatrix}\cdot\begin{bmatrix}a\\b\\c\end{bmatrix}\right)\begin{bmatrix}a\\b\\c\end{bmatrix} = \begin{bmatrix}a^2\\ab\\ac\end{bmatrix}$

$$\begin{aligned}
\mathbf{P}' &= \mathbf{P}'_{\perp} + \mathbf{P}_{||}\\
&= \left(\mathbf{P}-\mathbf{P}_{||}\right)\cos\theta + (\mathbf{v}\times\mathbf{P})\sin\theta + \mathbf{P}_{||}\\
&= \left(\begin{bmatrix}1\\0\\0\end{bmatrix} - \begin{bmatrix}a^2\\ab\\ac\end{bmatrix}\right)\cos\theta + \left(\begin{bmatrix}a\\b\\c\end{bmatrix}\times\begin{bmatrix}1\\0\\0\end{bmatrix}\right)\sin\theta + \begin{bmatrix}a^2\\ab\\ac\end{bmatrix}\\
&= \begin{bmatrix}1-a^2\\-ab\\-ac\end{bmatrix}\cos\theta + \begin{bmatrix}0\\c\\-b\end{bmatrix}\sin\theta + \begin{bmatrix}a^2\\ab\\ac\end{bmatrix} = \begin{bmatrix}a^2(1-\cos\theta)+\cos\theta\\ ab(1-\cos\theta)+c\cdot\sin\theta\\ ac(1-\cos\theta)-b\cdot\sin\theta\end{bmatrix}
\end{aligned}$$

旋转变换

- 以过原点的任意向量为轴旋转：构建变换后的基向量
  - 同样，可得 $\begin{bmatrix}0\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}ab\,(1-\cos\theta)-c\cdot\sin\theta\\ b^2(1-\cos\theta)+\cos\theta\\ bc(1-\cos\theta)+a\cdot\sin\theta\end{bmatrix}$

$$\begin{bmatrix}0\\0\\1\end{bmatrix} \rightarrow \begin{bmatrix}ac\,(1-\cos\theta)+b\cdot\sin\theta\\ bc(1-\cos\theta)-a\cdot\sin\theta\\ c^2(1-\cos\theta)+\cos\theta\end{bmatrix}$$



$$\mathbf{R}_{\mathbf{V}}(\theta)=\begin{array}{ccc} [1,0,0]^T & [0,1,0]^T & [0,0,1]^T \end{array}$$

$$\mathbf{R}_{\mathbf{V}}(\theta)=\begin{bmatrix} a^2(1-\cos\theta)+\cos\theta & ab(1-\cos\theta)+c\cdot\sin\theta & ac(1-\cos\theta)-b\sin\theta\\ ab(1-\cos\theta)-c\cdot\sin\theta & b^2(1-\cos\theta)+\cos\theta & bc(1-\cos\theta)+a\cdot\sin\theta\\ ac(1-\cos\theta)+b\cdot\sin\theta & bc(1-\cos\theta)-a\cdot\sin\theta & c^2(1-\cos\theta)+\cos\theta \end{bmatrix}$$

旋转变换

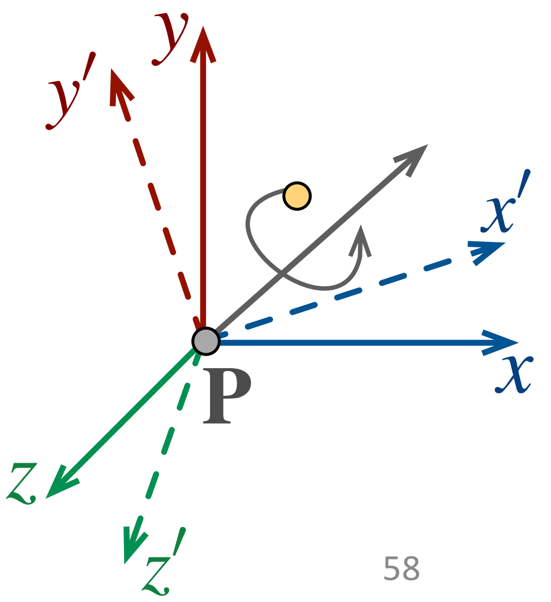
- 以过原点的任意向量为轴旋转：构建变换后的基向量

$$\begin{array}{ccc} [1,0,0]^T & [0,1,0]^T & [0,0,1]^T \end{array}$$

$$\mathbf{R_V}(\theta) = \begin{bmatrix} a^2(1-\cos\theta)+\cos\theta & ab(1-\cos\theta)+c\cdot\sin\theta & ac(1-\cos\theta)-b\sin\theta \\ ab(1-\cos\theta)-c\cdot\sin\theta & b^2(1-\cos\theta)+\cos\theta & bc(1-\cos\theta)+a\cdot\sin\theta \\ ac(1-\cos\theta)+b\cdot\sin\theta & bc(1-\cos\theta)-a\cdot\sin\theta & c^2(1-\cos\theta)+\cos\theta \end{bmatrix}$$

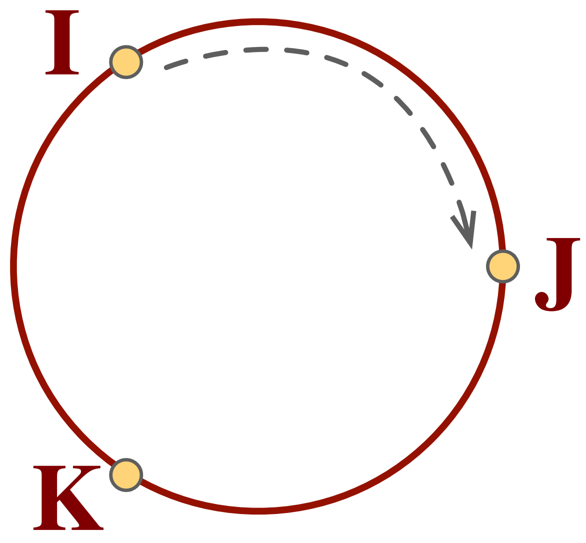
$$\mathbf{R_V}(\theta)\mathbf{P} = \begin{bmatrix} a^2(1-\cos\theta)+\cos\theta \\ ab(1-\cos\theta)-c\cdot\sin\theta \\ ac(1-\cos\theta)+b\cdot\sin\theta \end{bmatrix} x + \begin{bmatrix} ab(1-\cos\theta)+c\cdot\sin\theta \\ b^2(1-\cos\theta)+\cos\theta \\ bc(1-\cos\theta)-a\cdot\sin\theta \end{bmatrix} y + \begin{bmatrix} ac(1-\cos\theta)-b\sin\theta \\ bc(1-\cos\theta)+a\cdot\sin\theta \\ c^2(1-\cos\theta)+\cos\theta \end{bmatrix} z$$

– 将对点的旋转转为在旋转后的坐标系统下画点的过程
  - 旋转后的坐标系统由x, y, z轴分别旋转得到
  - 每个拆分为平行于旋转轴的分量与垂直于旋转轴的分量
  - 对垂直于旋转轴的分量进行二维旋转



旋转变换

- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - 通常被认为是旋转的“最佳”表现形式
  - 对旋转的稳定插值方法
  - 冗余信息少（4个实数）
    - 3D旋转自由度为3（旋转轴自由度2，旋转角度自由度1）
    - 3x3的矩阵需要9个实数描述3D旋转！
  - 定义：$\boldsymbol{Q} = x\boldsymbol{I} + y\boldsymbol{J} + z\boldsymbol{K} + w$，其中$x, y, z, w$为实数， $\boldsymbol{I}, \boldsymbol{J}, \boldsymbol{K}$为quaternion units，满足
    - 1. $\boldsymbol{I}^2 = \boldsymbol{J}^2 = \boldsymbol{K}^2 = -1$
    - 2. $\boldsymbol{I} * \boldsymbol{J} = \boldsymbol{K}$，$\boldsymbol{J} * \boldsymbol{K} = \boldsymbol{I}$，$\boldsymbol{K} * \boldsymbol{I} = \boldsymbol{J}$
    - 3. $\boldsymbol{I} * \boldsymbol{K} = -\boldsymbol{J}$，$\boldsymbol{K} * \boldsymbol{J} = -\boldsymbol{I}$，$\boldsymbol{J} * \boldsymbol{I} = -\boldsymbol{K}$
    - 四元数也可写作$\boldsymbol{Q} = [\mathbf{v}, w]$，其中$\mathbf{v} = (x, y, z)$



旋转变换

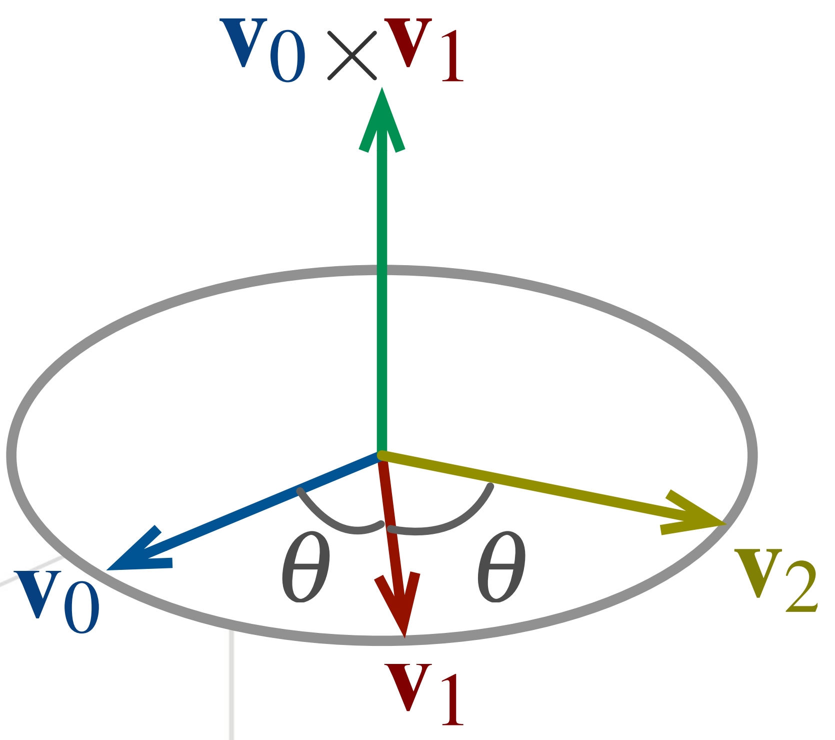
- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - 加法： $\boldsymbol{Q}_a + \boldsymbol{Q}_b = [\mathbf{v}_a, w_a] + [\mathbf{v}_b, w_b] = [\mathbf{v}_a + \mathbf{v}_b, w_a + w_b]$
  - 乘法：$\boldsymbol{Q}_a\boldsymbol{Q}_b = [\mathbf{v}_a, w_a][\mathbf{v}_b, w_b]$
    $= [\mathbf{v}_a \times \mathbf{v}_b + w_a\mathbf{v}_b + w_b\mathbf{v}_a, w_a w_b - \mathbf{v}_a \cdot \mathbf{v}_b]$
    - 乘法不满足交换律，但满足结合律
  - 共轭：$\boldsymbol{Q} = x\boldsymbol{I} + y\boldsymbol{J} + z\boldsymbol{K} + w = [\mathbf{v}, w]$的共轭值为
    - $\boldsymbol{Q}^* = -x\boldsymbol{I} - y\boldsymbol{J} - z\boldsymbol{K} + w = [-\mathbf{v}, w]$
  - 长度：$|\boldsymbol{Q}|^2 = \boldsymbol{Q}\boldsymbol{Q}^*$
  - 逆元素：$\boldsymbol{Q}^{-1} = \boldsymbol{Q}^*/|\boldsymbol{Q}|^2$（由于$\boldsymbol{Q}(\boldsymbol{Q}^*/|\boldsymbol{Q}|^2) = 1$）
    - 当$|\boldsymbol{Q}| = 1$（单位四元数）时， $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^*$
  - 纯四元数： $\boldsymbol{Q} = x\boldsymbol{I} + y\boldsymbol{J} + z\boldsymbol{K} = [\mathbf{v}, 0]$
    - 当$|\mathbf{v}| = 1$时，$\mathbf{v}\mathbf{v} = [\mathbf{0}, -1] = -1$

旋转变换

- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - $\boldsymbol{Q} = [\mathbf{n}\sin\theta, \cos\theta]$将任意向量沿单位向量$\mathbf{n}$旋转$2\theta$
    - 如图，假设$\mathbf{v}_0, \mathbf{v}_1$为单位向量其夹角为$\theta$
    - 令四元数$\boldsymbol{Q} = \mathbf{v}_1\mathbf{v}_0^* = [\mathbf{v}_0\times\mathbf{v}_1, \mathbf{v}_0\cdot\mathbf{v}_1] = [\mathbf{n}\sin\theta, \cos\theta]$
      - 其中$\mathbf{n} = \mathbf{v}_0\times\mathbf{v}_1/|\mathbf{v}_0\times\mathbf{v}_1|$
      - $\boldsymbol{Q}^* = \mathbf{v}_0\mathbf{v}_1^*$
    - 令向量$\mathbf{v}_2 = \boldsymbol{Q}\mathbf{v}_0\boldsymbol{Q}^*$，则
      - $\mathbf{v}_2\mathbf{v}_1^* = \boldsymbol{Q}\mathbf{v}_0\boldsymbol{Q}^*\mathbf{v}_1^* = \boldsymbol{Q}\mathbf{v}_0(\mathbf{v}_0\mathbf{v}_1^*)\mathbf{v}_1^*$

        $= \boldsymbol{Q}(\mathbf{v}_0\mathbf{v}_0)(\mathbf{v}_1^*\mathbf{v}_1^*) = \boldsymbol{Q}(-1)(-1)$

        $= \boldsymbol{Q} = [\mathbf{n}\sin\theta, \cos\theta]$
      - $\mathbf{v}_2$与$\mathbf{v}_0, \mathbf{v}_1$共面，且$\mathbf{v}_2$与$\mathbf{v}_1$,间夹角为$\theta$
      - $\boldsymbol{Q}\mathbf{v}_0\boldsymbol{Q}^*$以$\mathbf{n}$为轴将$\mathbf{v}_0$旋转$2\theta$



旋转变换

- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - 单位四元数$\boldsymbol{Q} = x\boldsymbol{I} + y\boldsymbol{J} + z\boldsymbol{K} + w$所表示的旋转矩阵为
    - $\begin{bmatrix} 1-2y^2-2z^2 & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2x^2-2z^2 & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2x^2-2y^2 \end{bmatrix}$
  - 基于四元数的旋转运算举例：以$[1,1,0]$为轴将向量$[1,0,0]$旋转90度
    - 将旋转轴归一化： $[\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},0]$
    - 旋转90度意味着$\theta = 45$度， $\cos\theta = \sin\theta = \frac{\sqrt{2}}{2}$
    - $\boldsymbol{Q} = \left[\left[\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},0\right]\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right] = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{J} + \frac{\sqrt{2}}{2}$

旋转变换

- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - 基于四元数的旋转运算举例：以$[1,1,0]$为轴将向量$[1,0,0]$旋转90度
    - $\boldsymbol{Q}=\left[\left[\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},0\right]\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right]=\frac{1}{2}\boldsymbol{I}+\frac{1}{2}\boldsymbol{J}+\frac{\sqrt{2}}{2}$
    - 向量$[1,0,0]=\boldsymbol{I}$
    - $\boldsymbol{QI}=\left(\frac{1}{2}\boldsymbol{I}+\frac{1}{2}\boldsymbol{J}+\frac{\sqrt{2}}{2}\right)\boldsymbol{I}=\frac{1}{2}\boldsymbol{I}*\boldsymbol{I}+\frac{1}{2}\boldsymbol{J}*\boldsymbol{I}+\frac{\sqrt{2}}{2}\boldsymbol{I}=-\frac{1}{2}-\frac{1}{2}\boldsymbol{K}+\frac{\sqrt{2}}{2}\boldsymbol{I}$
    - $\boldsymbol{QIQ}^*=\left(\frac{\sqrt{2}}{2}\boldsymbol{I}-\frac{1}{2}\boldsymbol{K}-\frac{1}{2}\right)\left(-\frac{1}{2}\boldsymbol{I}-\frac{1}{2}\boldsymbol{J}+\frac{\sqrt{2}}{2}\right)=\frac{1}{2}\boldsymbol{I}+\frac{1}{2}\boldsymbol{J}-\frac{\sqrt{2}}{2}\boldsymbol{K}$
    - 旋转后所得向量为$[\frac{1}{2},\frac{1}{2},-\frac{\sqrt{2}}{2}]$

以过原点的任意向量为轴旋转：四元数（quaternion）

– 基于四元数的旋转运算举例：以$[1,1,0]$为轴将向量$[1,0,0]$旋转90度

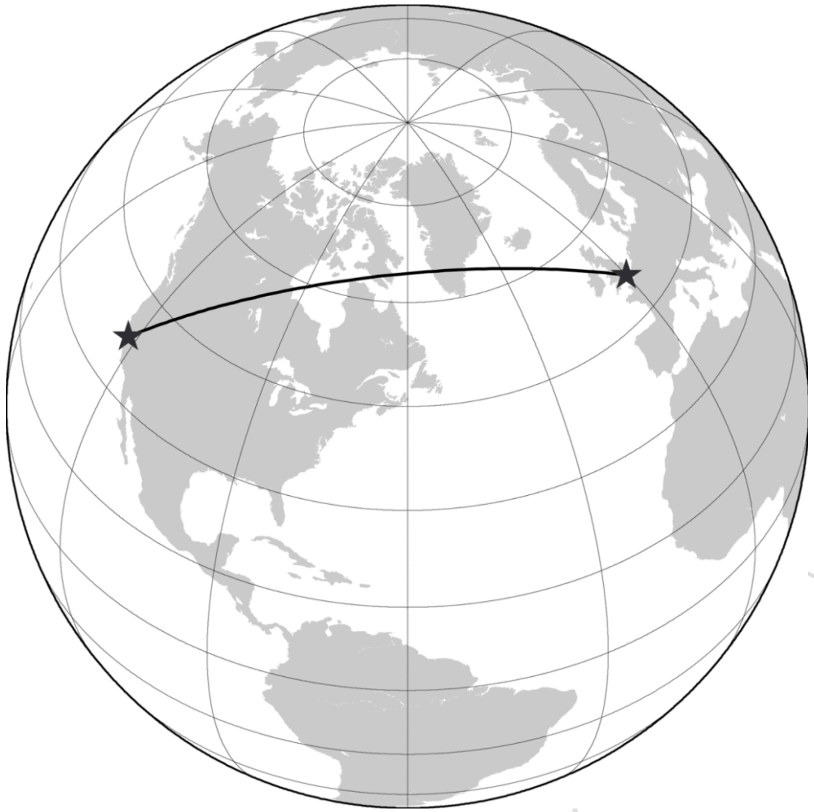
- $\boldsymbol{Q} = \left[\left[\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},0\right]\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right] = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{J} + \frac{\sqrt{2}}{2}$
- $\boldsymbol{QIQ}^* = \left(\frac{\sqrt{2}}{2}\boldsymbol{I} - \frac{1}{2}\boldsymbol{K} - \frac{1}{2}\right)\left(-\frac{1}{2}\boldsymbol{I} - \frac{1}{2}\boldsymbol{J} + \frac{\sqrt{2}}{2}\right) = \frac{1}{2}\boldsymbol{I} + \frac{1}{2}\boldsymbol{J} - \frac{\sqrt{2}}{2}\boldsymbol{K}$
- 旋转后所得向量为$[\frac{1}{2},\frac{1}{2},-\frac{\sqrt{2}}{2}]$

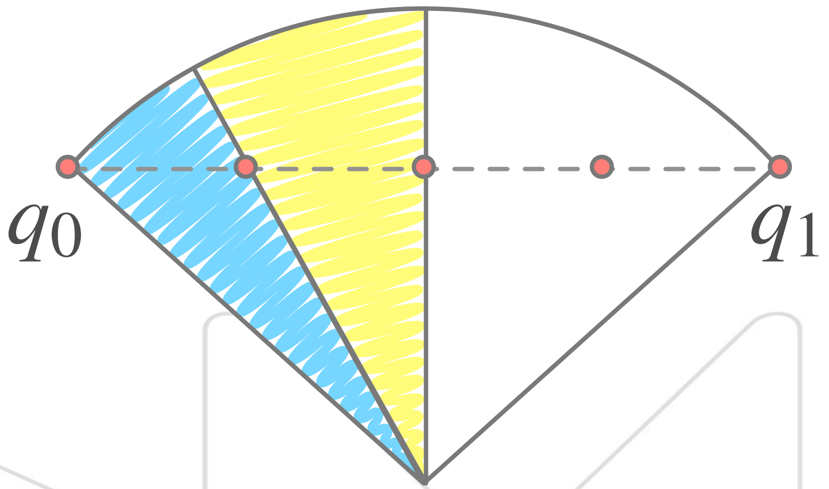
– 矩阵验算:

- $$\begin{bmatrix} 1-2y^2-2z^2 & \dots & \dots \\ 2(xy+wz) & \dots & \dots \\ 2(xz-wy) & \dots & \dots \end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix} 1-2y^2-2z^2 \\ 2(xy+wz) \\ 2(xz-wy) \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \\ -\sqrt{2}/2 \end{bmatrix}$$

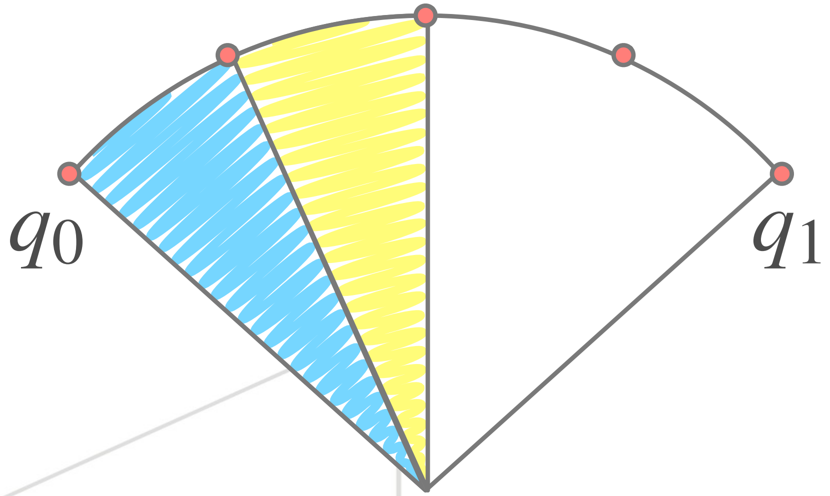
旋转变换

- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - 线性插值：$\mathrm{LERP}(x_0, x_1, t) = (1-t)\cdot x_0 + t\cdot x_1$
  - 球面线性插值（spherical linear interpolation, SLERP）
    - 角速度均匀变化
    - 过圆心与球面上两点的平面与球面相交的弧中较短的一段





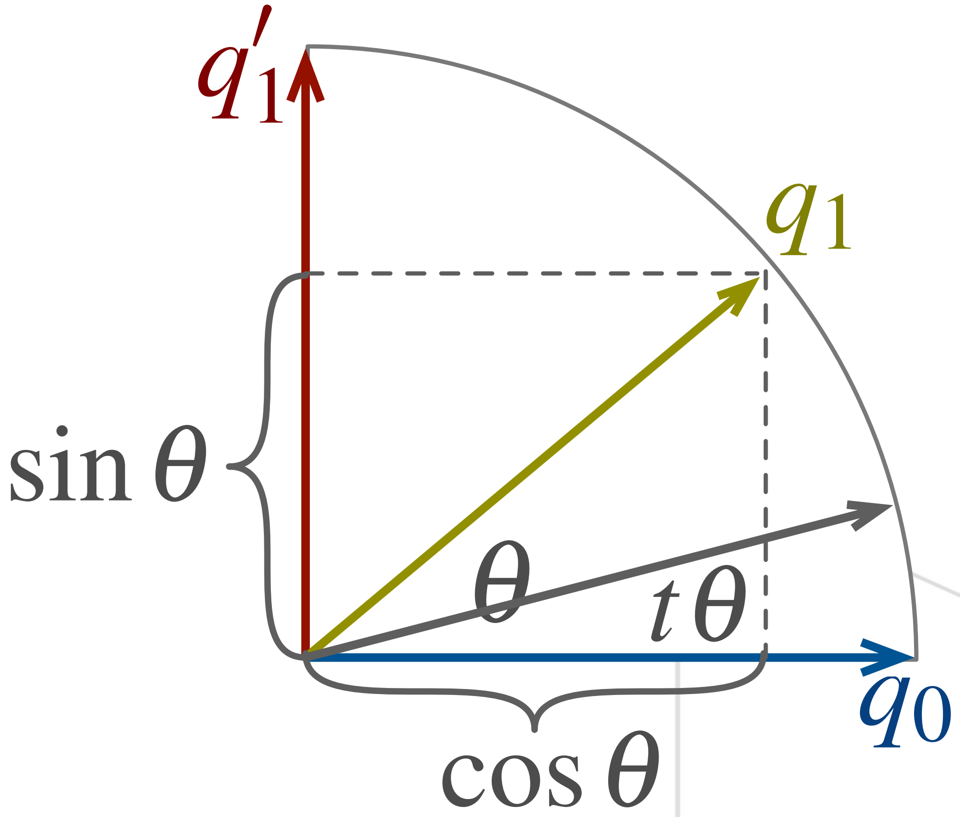
线性插值

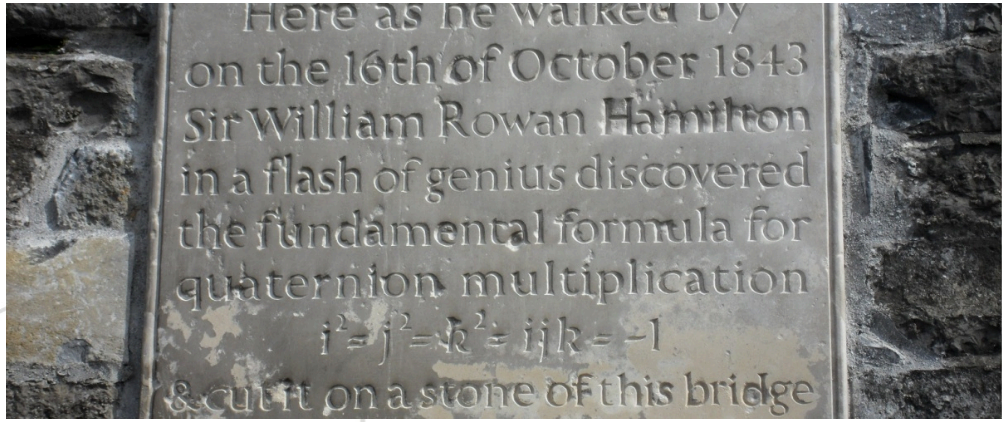


球面线性插值

旋转变换

- 以过原点的任意向量为轴旋转：四元数（quaternion）
  - 线性插值：$\mathrm{LERP}(x_0, x_1, t) = (1-t)\cdot x_0 + t \cdot x_1$
  - 球面线性插值（spherical linear interpolation, SLERP）
    - $$\begin{aligned}\mathrm{SLERP}(\boldsymbol{q}_0, \boldsymbol{q}_1, t) &= \boldsymbol{q}_0 \cos t\theta + \boldsymbol{q}_1' \sin t\theta \\ &= \boldsymbol{q}_0 \cos t\theta + \frac{\boldsymbol{q}_1 - \boldsymbol{q}_0 \cos\theta}{\sin\theta} \sin t\theta \\ &= \boldsymbol{q}_0 \frac{\cos t\theta \sin t\theta - \sin t\theta \cos\theta}{\sin\theta} + \boldsymbol{q}_1 \frac{\sin t\theta}{\sin\theta} \\ &= \boldsymbol{q}_0 \frac{\sin(1-t)\theta}{\sin\theta} + \boldsymbol{q}_1 \frac{\sin t\theta}{\sin\theta}\end{aligned}$$



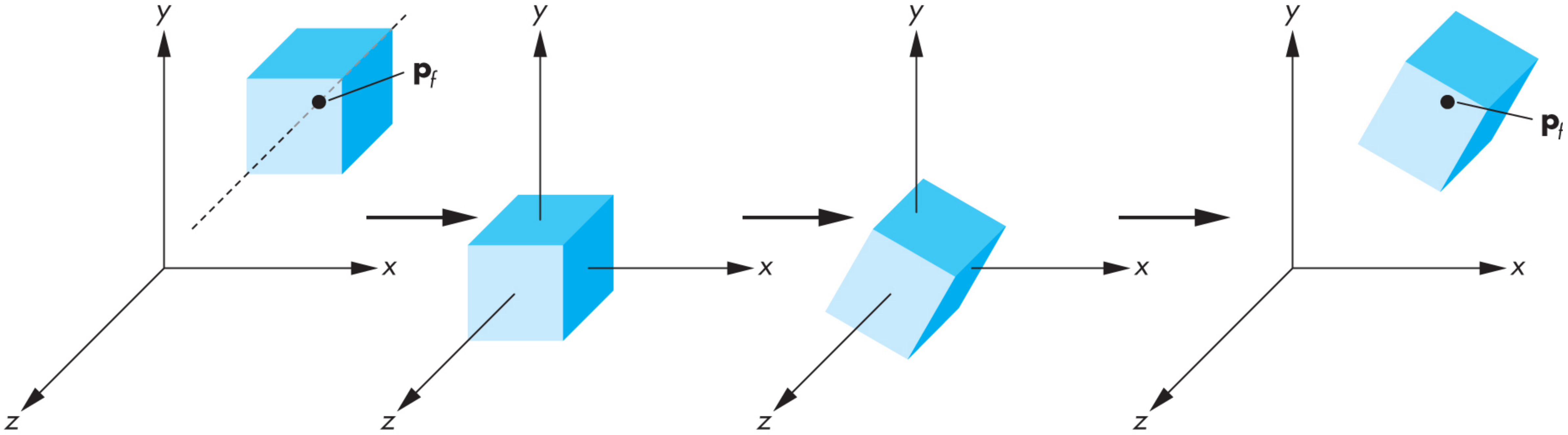


旋转变换

3D旋转

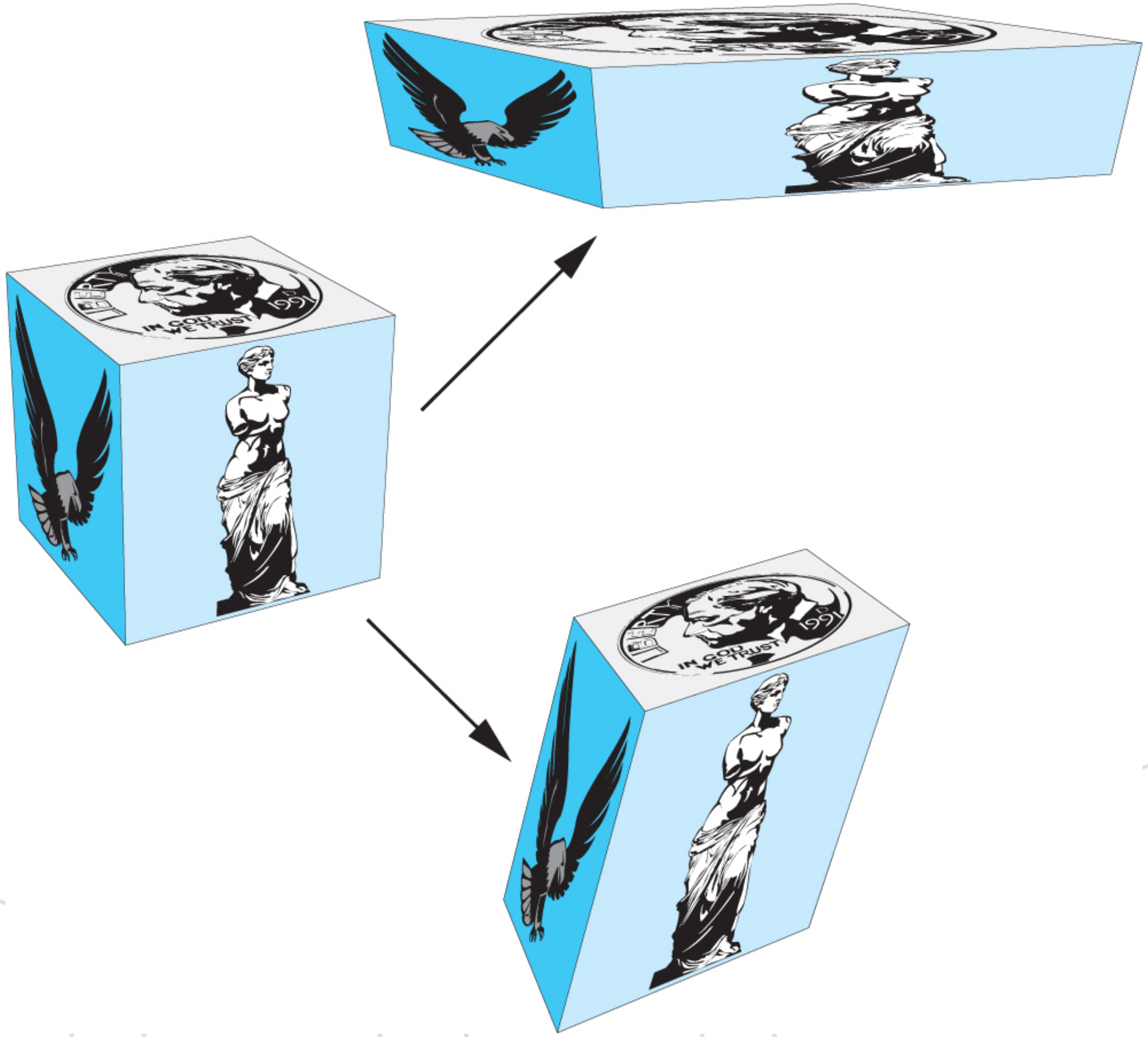
– 以不经过原点的任意向量为轴旋转

- 平移坐标系，使旋转轴经过原点
- 旋转
- 平移坐标系回到原始位置
- $\mathbf{M} = \mathbf{T}(\mathbf{P}_{\mathrm{f}})\mathbf{R}(\theta)\mathbf{T}(-\mathbf{P}_f)$



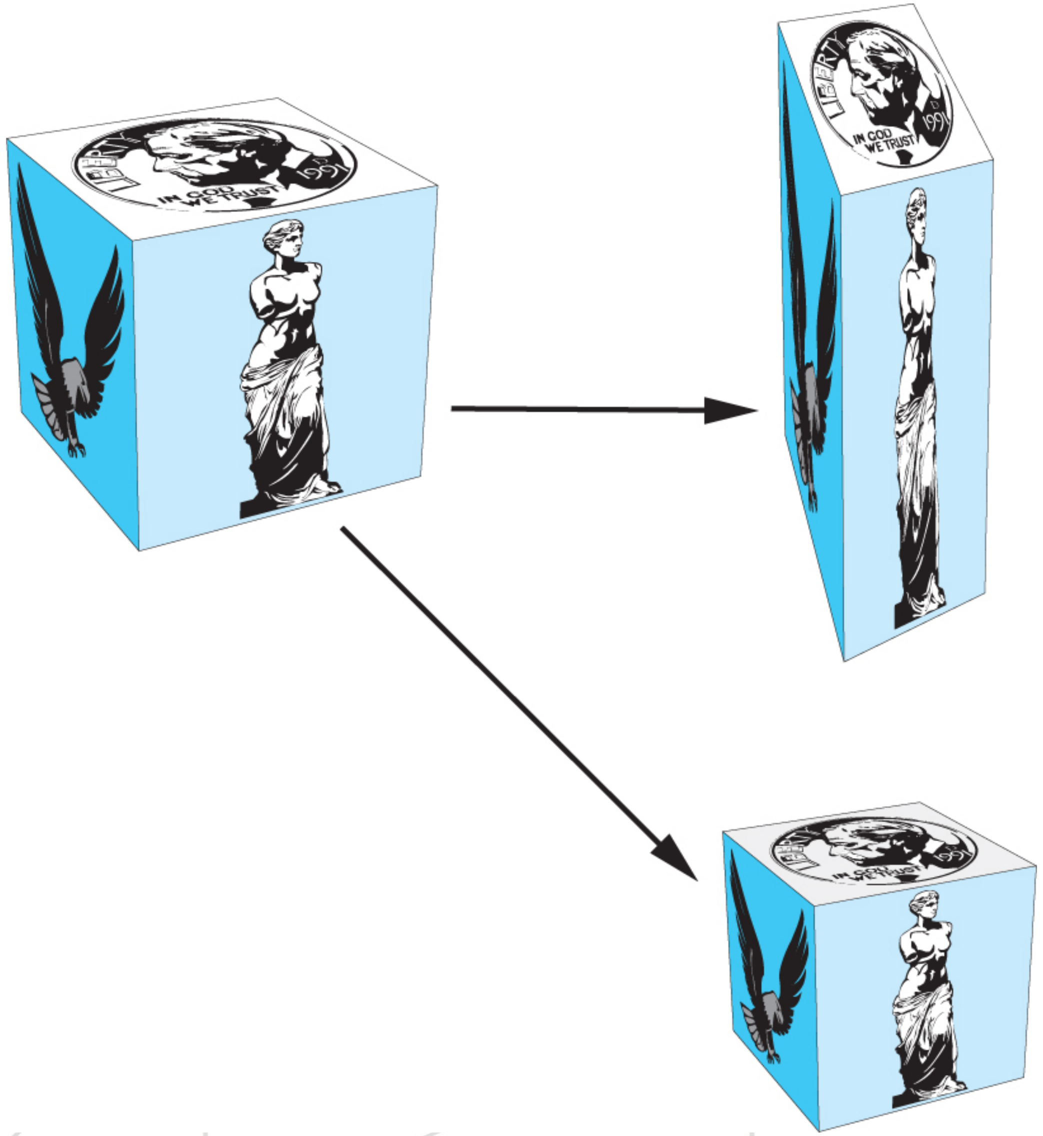
非刚体变换

- 平移与旋转均为刚体变换（rigid transformation）
  - 仿射变换中的另外两种变换为非刚体变换
    - 改变物体的形状



非刚体变换

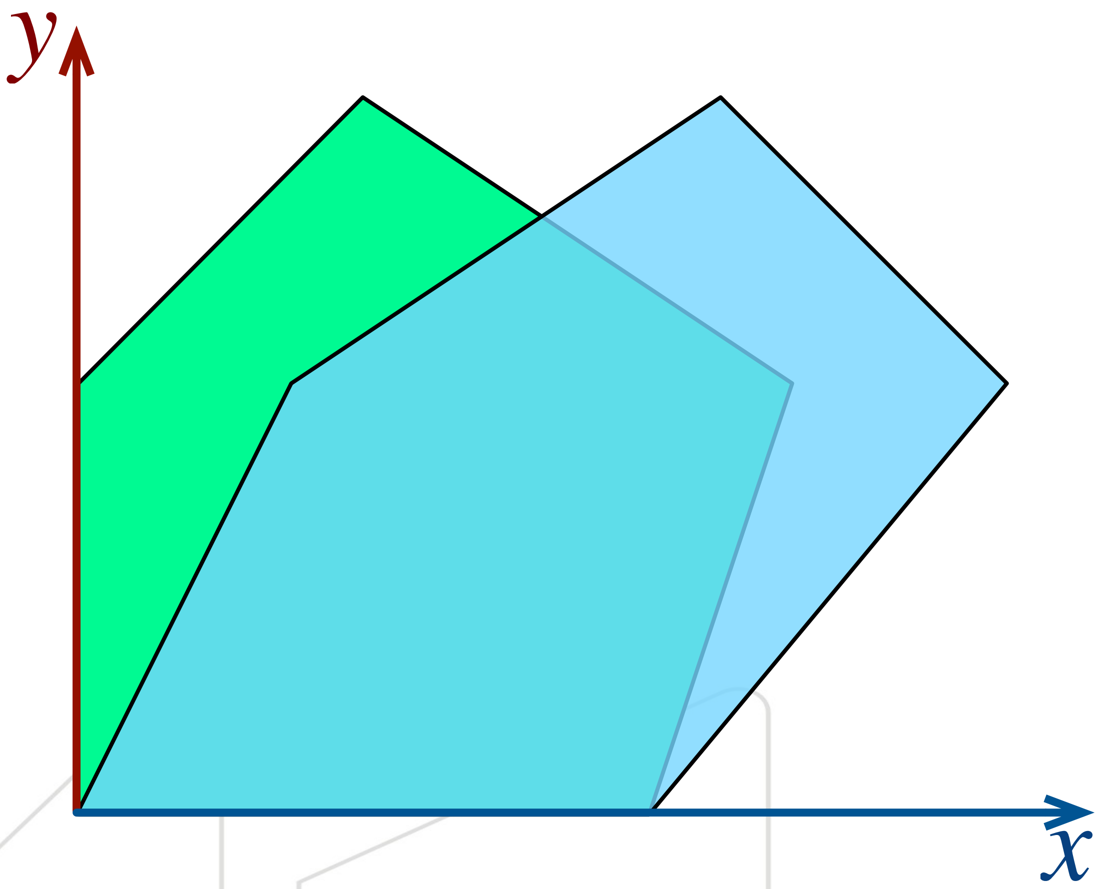
- 缩放（scaling）
  - $\mathbf{P}' = \mathbf{SP}$
  - $x' = S_x x$
  - $y' = S_y y$
  - $z' = S_z z$
  - $\mathbf{S}(S_x, S_y, S_z) = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$



非刚体变换

错切（shearing）

- 产生形状的变形
- 举例：沿x轴的错切
  - $x' = x + a \cdot y$
  - $y' = y$
  - $z' = z$
  - 是否为线性变换？
  - 矩阵表示？



变换的矩阵表示

- 所有点及向量均表示为4D列向量
- 变换表示为一个4x4矩阵
  - 变换由矩阵与向量的乘法完成
  - 多个变换可由矩阵乘法合成

• Translation

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 根据其几何意义，变换的逆运算也可由矩阵定义

Translation: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathrm{T}(-d_x, -d_y, -d_z)$

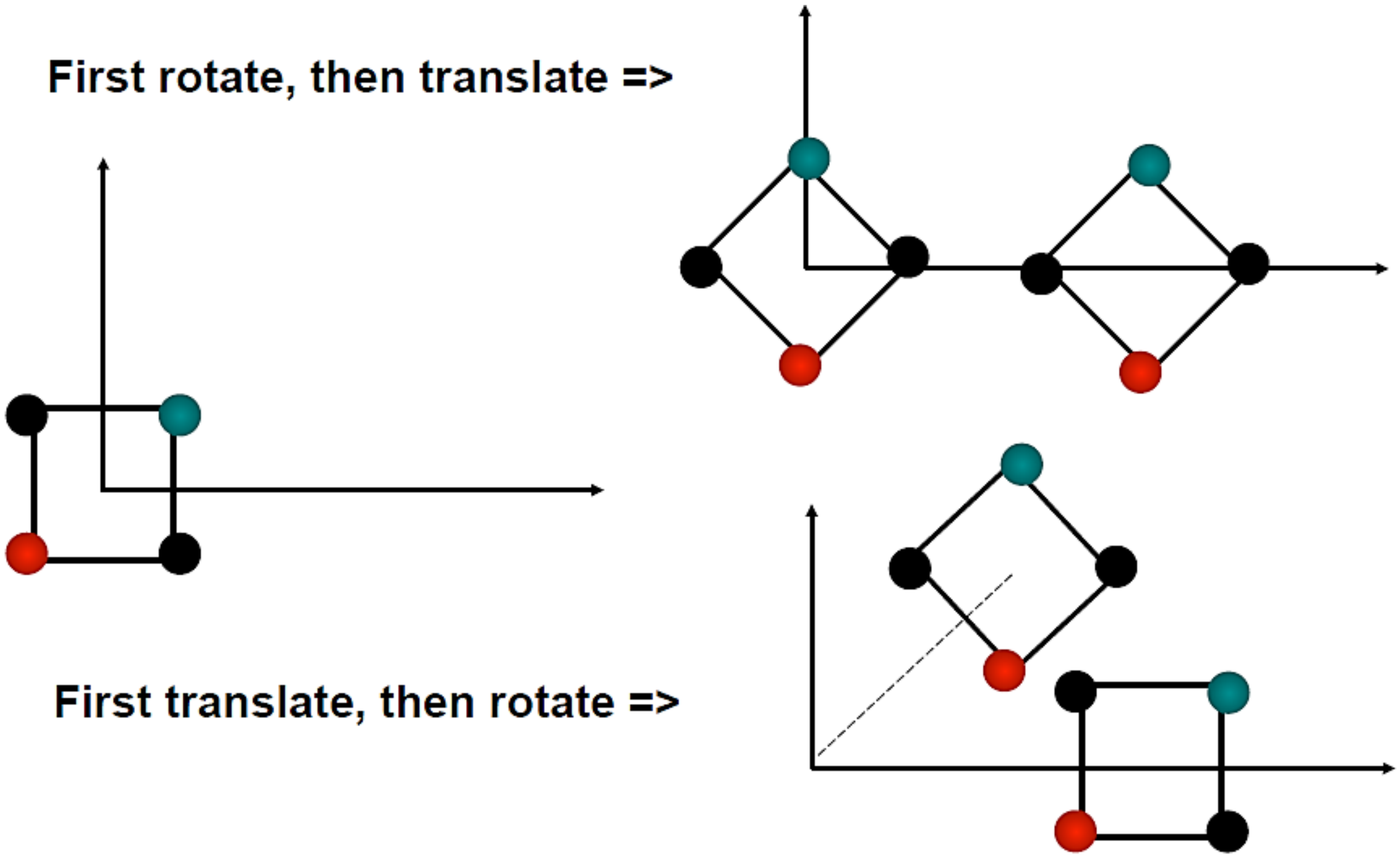
Rotation: $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$

- For any rotation matrix
- Note: $\cos(-\theta) = \cos\theta$, $\sin(-\theta) = -\sin\theta$., so $\mathbf{R}^{-1}(\theta) = \mathbf{R}^{T}(\theta)$

Scale: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$

变换的矩阵表示

- 矩阵乘法不满足交换律
  - 因此，变换也不满足交换律



课程提纲

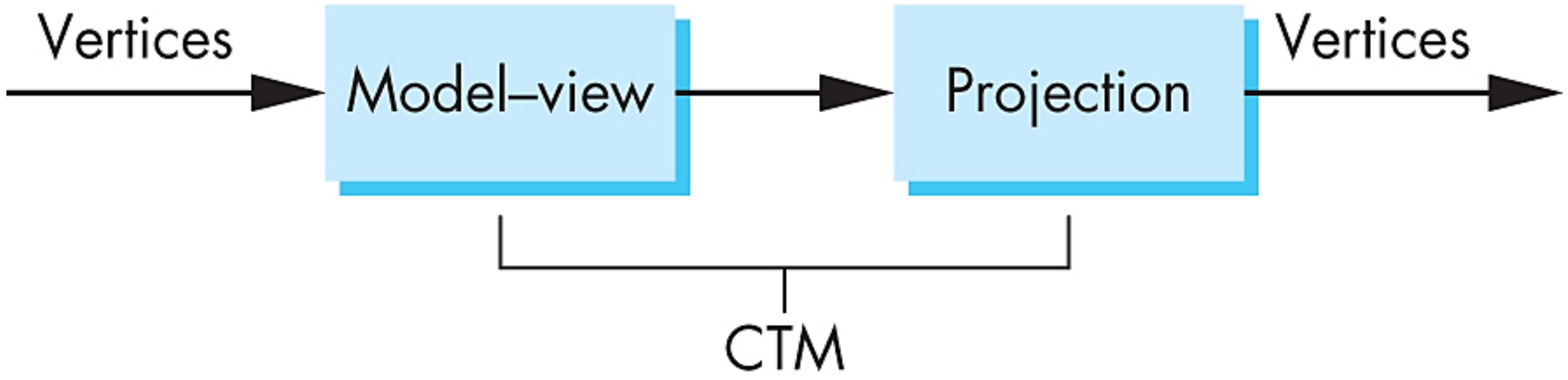
- 基本几何概念
  - 三种基本元素：点、标量、向量
  - 线性组合
  - 仿射空间
- 表现形式
  - 齐次坐标
- 变换
- OpenGL中的变换

OpenGL中的变换

- OpenGL中所有变换都是由矩阵完成的
  - OpenGL是状态机，而变换矩阵是状态的一部分
  - 变换矩阵必须在绘制顶点前设置以达到变换的效果
  - 在建模过程中，物体通常是在物体坐标空间中定义的，因此需要使用变换将物体从其坐标空间中“移动”至场景中
  - OpenGL提供多个堆栈以保存不同类型的变换矩阵
    - Modelview
    - Projection
    - Texture

OpenGL中的变换

- Current Transformation Matrix (CTM)
  - CTM是一个4x4的齐次坐标矩阵
  - 可由一系列函数更改，并作用于渲染管线中其后定义的顶点上
  - CTM为modelview及projection两个矩阵堆栈栈顶矩阵的乘积



OpenGL中的变换

- 更改CTM
  - 指定CTM模式：`glMatrixMode(mode);`
    - mode取值为GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE
  - 载入CTM：`glLoadIdentity(void);` `glLoadMatrix{fd}(*m);`
    - m为指针，指向长度为16的数组（column major）
  - 乘CTM：`glMultMatrix{fd}(*m);`
  - 构建变换矩阵并修改CTM：（CTM乘相应变换矩阵）
    - `glTranslate{fd}(x, y, z);`
    - `glScale{fd}(x, y, z);`
    - `glRotate{fd}(scale, x, y, z);`

OpenGL中的变换

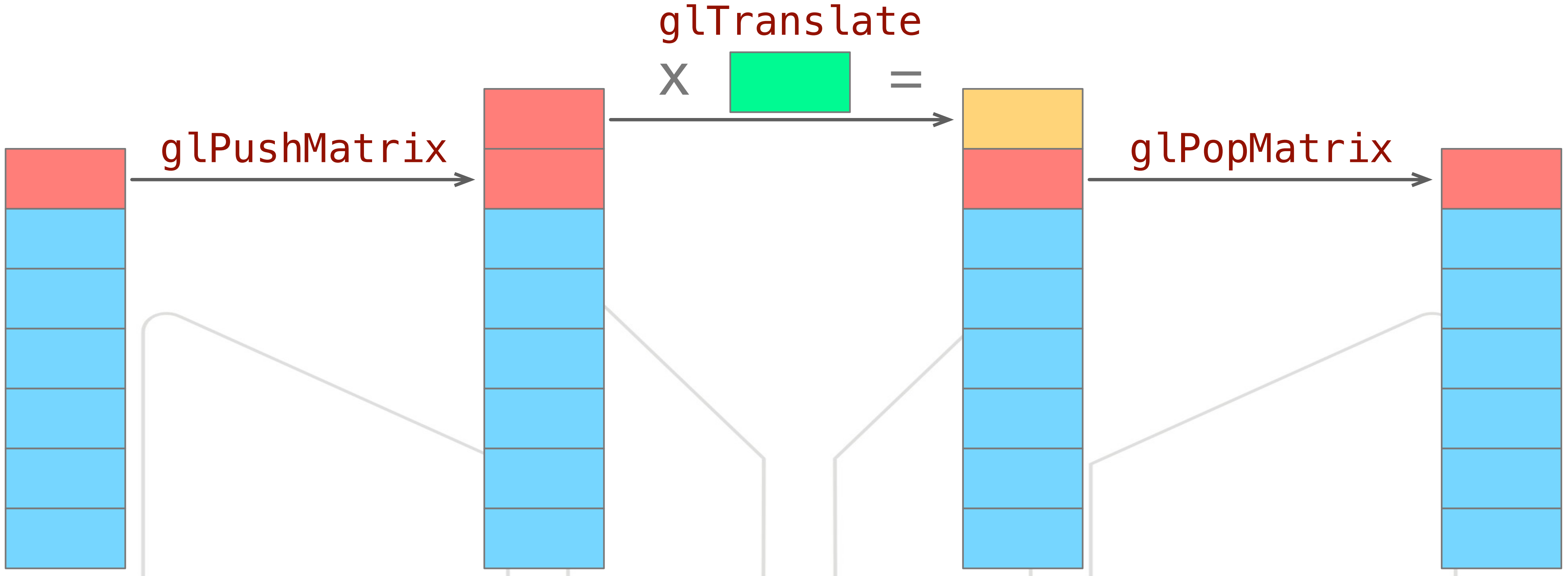
- 变换举例：沿不经过原点的任意轴旋转
  - $\mathbf{T}(\mathbf{P})\mathbf{R}(\theta)\mathbf{T}(-\mathbf{P})$
  - 如，沿[4,3,2]到[5,6,7]的轴旋转45度
    - 注意变换调用顺序

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

glTranslatef(4.0f, 3.0f, 2.0f);
glRotatef(45.0f, 1.0f, 3.0f, 5.0f);
glTranslatef(-4.0f, -3.0f, -2.0f);

glBegin(...)
glVertex(...)
...
glEnd(...)
```

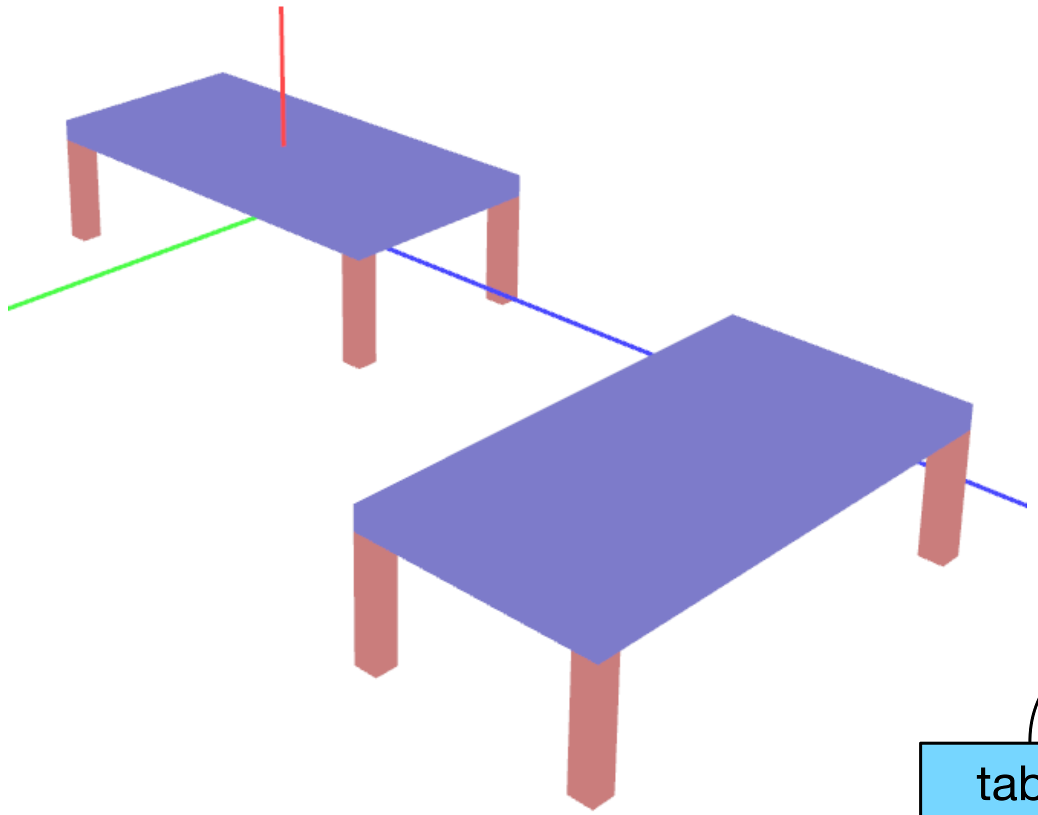
OpenGL中的变换

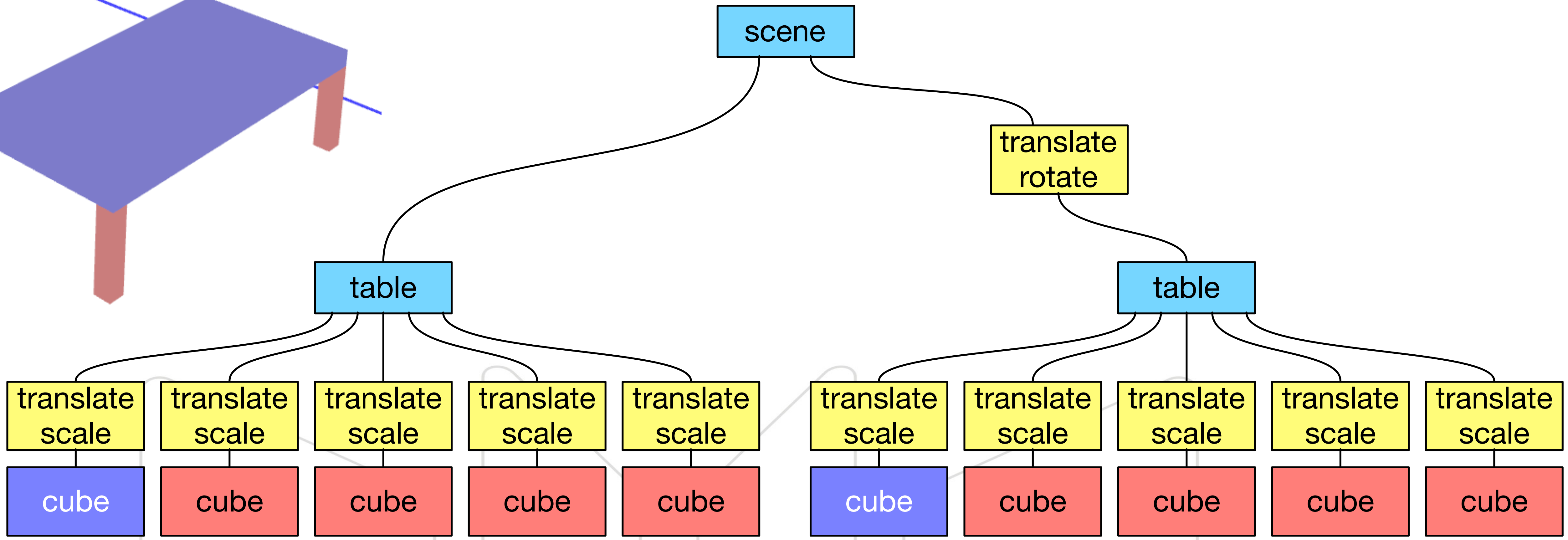
矩阵堆栈

- 使用以下函数修改堆栈
  - `glPushMatrix(void);`
  - `glPopMatrix(void);`
- 不同类型矩阵拥有各自堆栈
  - 当前操作修改的堆栈由matrix mode决定



- 矩阵应用举例：如何绘制图中场景？

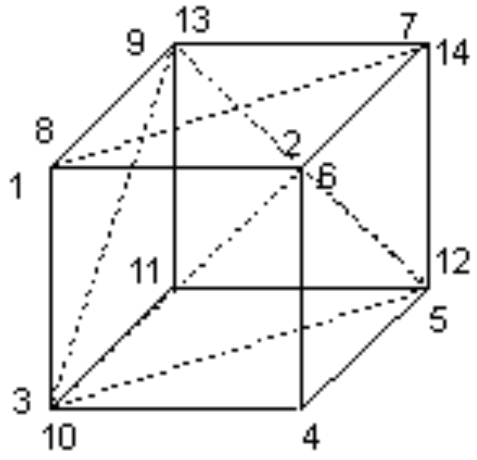


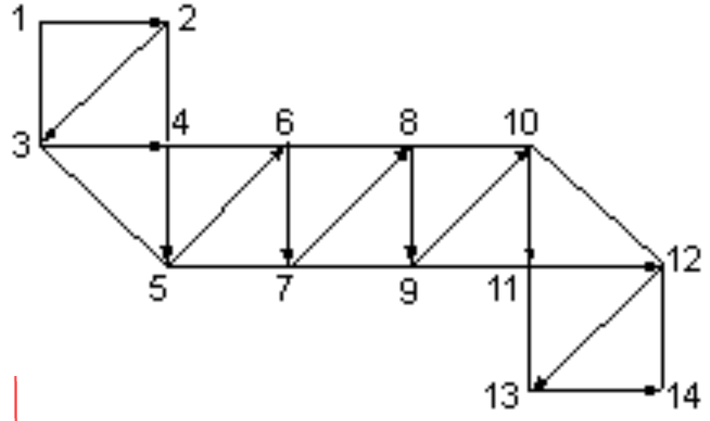


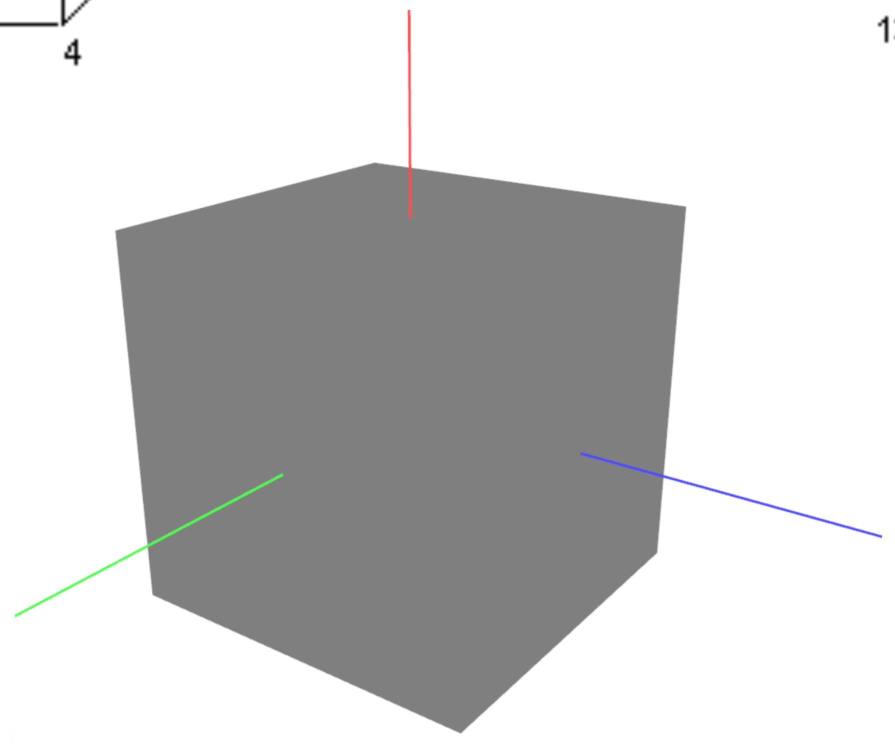
OpenGL中的变换

- 矩阵应用举例：如何绘制图中场景？
  - 首先，绘制正方体
    - 使用一个triangle strip进行绘制

```
void drawUnitBox() {
    glBegin(GL_TRIANGLE_STRIP);
    glVertex3f(-0.5f, 0.5f, -0.5f);
    glVertex3f(0.5f, 0.5f, -0.5f);
    glVertex3f(-0.5f, -0.5f, -0.5f);
    glVertex3f(0.5f, -0.5f, -0.5f);
    glVertex3f(0.5f, -0.5f, 0.5f);
    glVertex3f(0.5f, 0.5f, -0.5f);
    glVertex3f(0.5f, 0.5f, 0.5f);
    glVertex3f(-0.5f, 0.5f, -0.5f);
    glVertex3f(-0.5f, 0.5f, 0.5f);
    glVertex3f(-0.5f, -0.5f, -0.5f);
    glVertex3f(-0.5f, -0.5f, 0.5f);
    glVertex3f(0.5f, -0.5f, 0.5f);
    glVertex3f(-0.5f, 0.5f, 0.5f);
    glVertex3f(0.5f, 0.5f, 0.5f);
    glEnd();
}
```





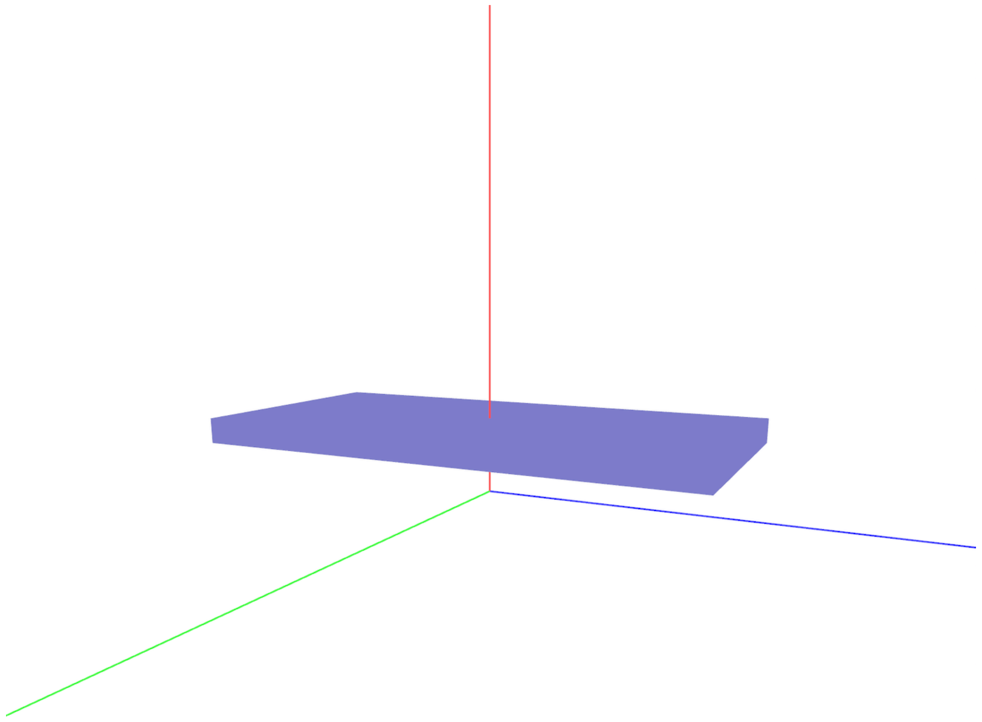


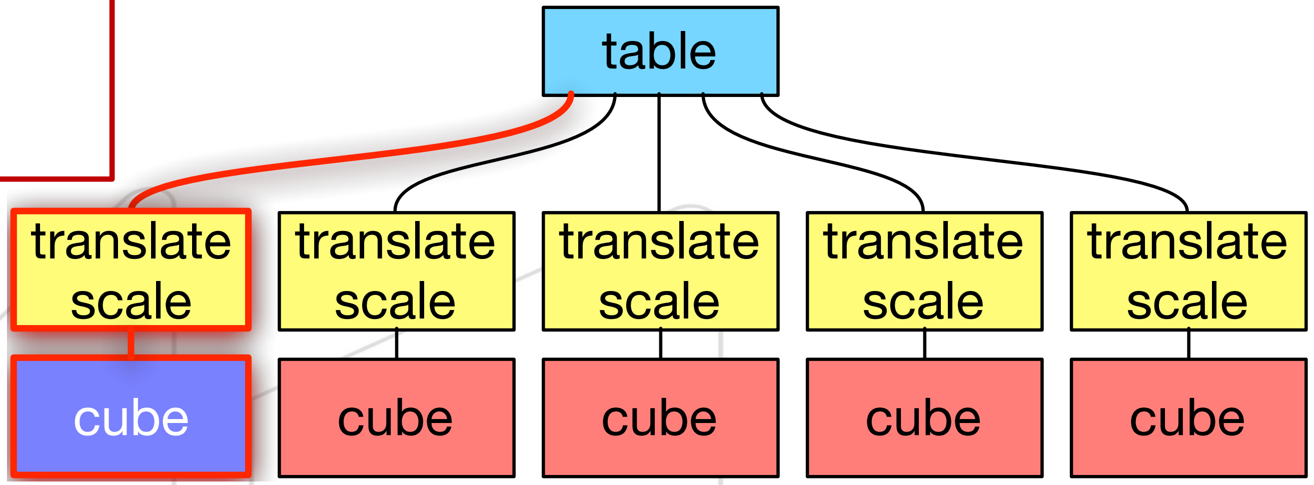
OpenGL中的变换

- 矩阵应用举例：如何绘制图中场景？
  - 绘制桌面：拉伸、平移正方体

```
void drawTable()
{
    glPushMatrix();
    glTranslatef(0.0f, 2.5f, 0.0f);
    glScalef(20.0f, 1.0f, 10.0f);
    glColor4f(0.5f, 0.5f, 0.8f, 0.5f);
    drawUnitBox();
    glPopMatrix();
    ...
}
```

为什么需要push/pop？

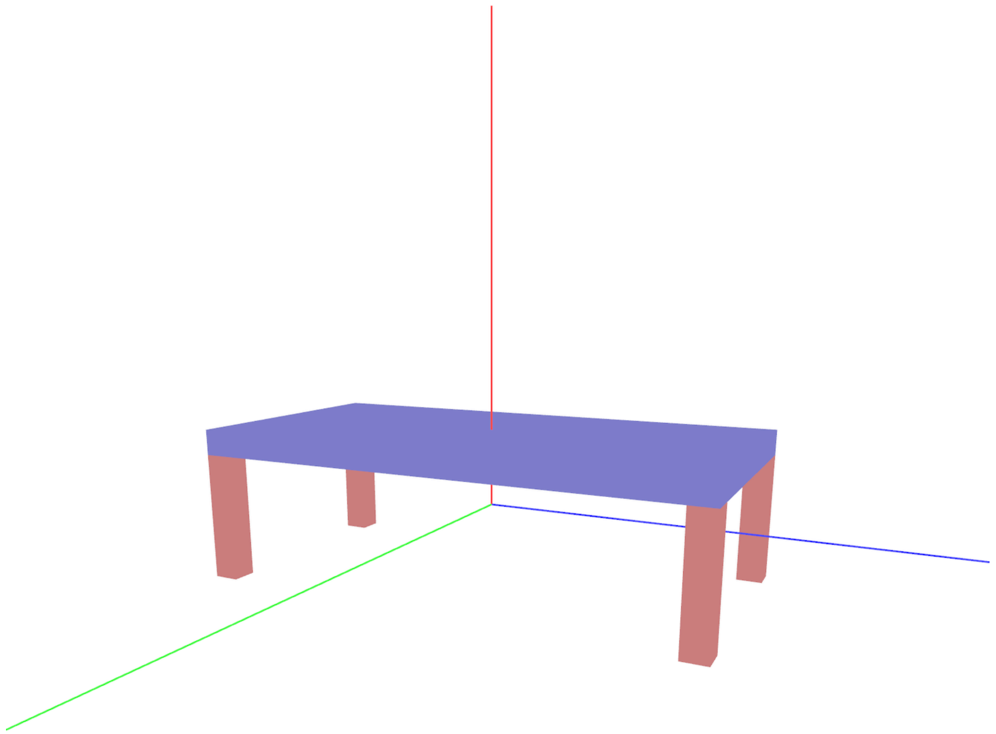


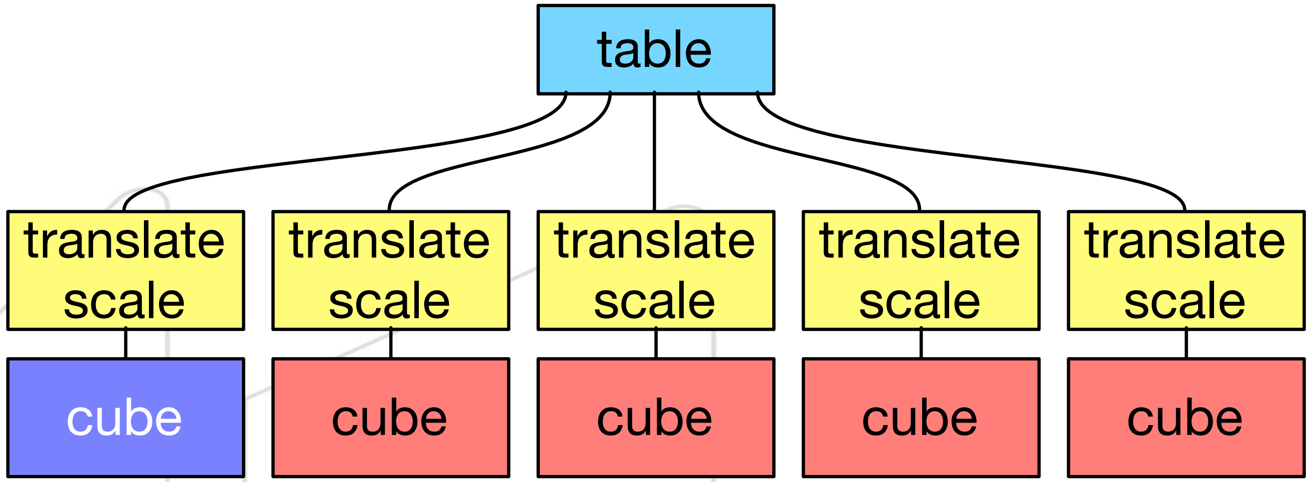


OpenGL中的变换

- 矩阵应用举例：如何绘制图中场景？
  - 绘制桌脚：拉伸、平移正方体

```
void drawTable() {
    ...
    glPushMatrix();
    glTranslatef(9.5f, -0.5f, 4.5f);
    glScalef(1.0f, 5.0f, 1.0f);
    glColor4f(0.8f, 0.5f, 0.5f, 0.5f);
    drawUnitBox();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-9.5f, -0.5f, 4.5f);
    glScalef(1.0f, 5.0f, 1.0f);
    drawUnitBox();
    glPopMatrix();
    ...
}
```

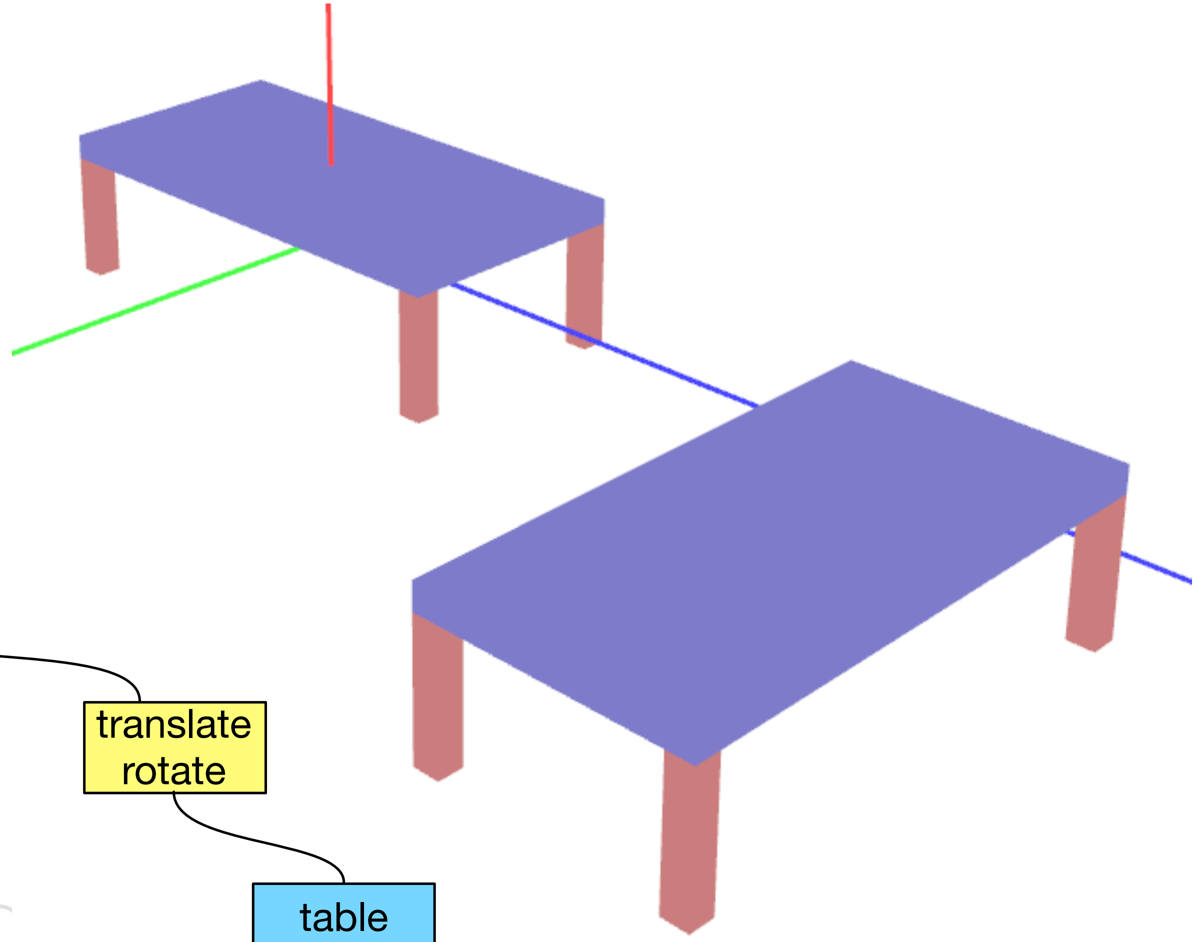


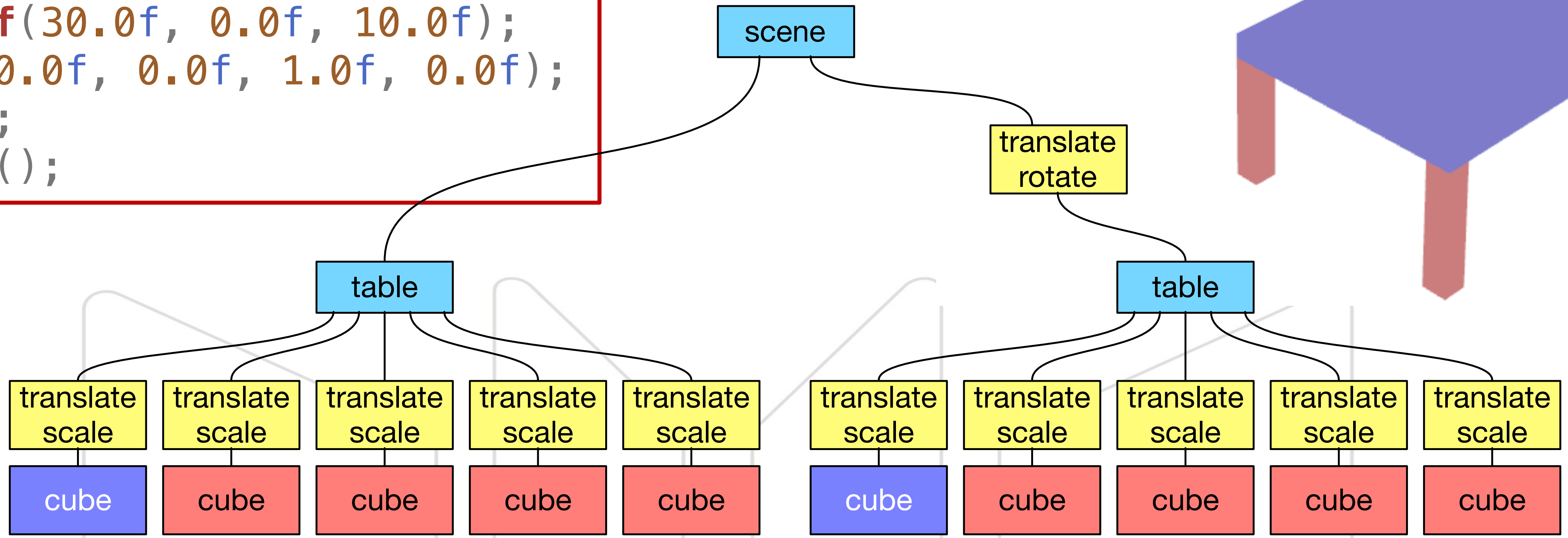


OpenGL中的变换

- 矩阵应用举例：如何绘制图中场景？
  - 绘制两张桌子

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

drawTable();

glPushMatrix();
glTranslatef(30.0f, 0.0f, 10.0f);
glRotatef(90.0f, 0.0f, 1.0f, 0.0f);
drawTable();
glPopMatrix();
```





小结

- 基本概念
  - 基本元素：点、标量、向量
  - 线性空间与仿射空间
    - 齐次坐标
- 变换
  - 平移、旋转、缩放、错切
  - 注意多个变换合成时的顺序
- OpenGL实现
  - OpenGL为状态机，通过函数修改当前变换矩阵
    - `glMatrixMode(mode); glLoadIdentity(void); glLoadMatrix{fd}(*m); glMultMatrix{fd}(*m); glTranslate{fd}(x, y, z); glScale{fd}(x, y, z); glRotate{fd}(scale, x, y, z);`

Questions?