三维旋转

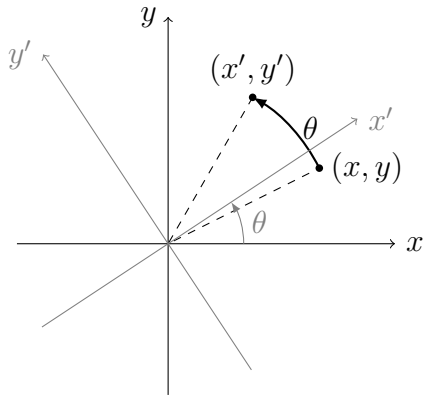
Josh-Cena

2020 年 10 月 27 日

给定过原点和点 (rx, ry, rz) 的直线 r，求点 (x, y, z) 绕 r 逆时针旋转 α（从原点看）后得到的点坐标。

前置知识：变换矩阵

在直角坐标系中，可以把任何一个仿射变换理解为将向量乘以一个变换矩阵——该变换矩阵代表了坐标系的相应变换。举个例子，对于旋转变换：



我们既可以将其理解为点 (x, y) 旋转到了 (x', y')，也可以理解成整个坐标系逆时针旋转了 θ。旋转后，原本用来定义坐标系的两个正交基

$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

变成了

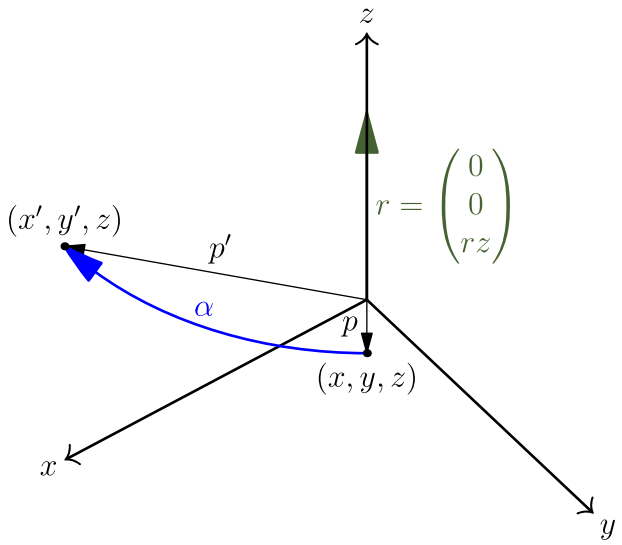
$$\mathbf{x}' = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \mathbf{y}' = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}$$

此时，任意一个点 (x, y) 都可以乘上这两个正交基组成的矩阵 $\mathbf{T}$，得到变换后的坐标。

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

三维坐标系需要三个正交基；但方法大致同理。

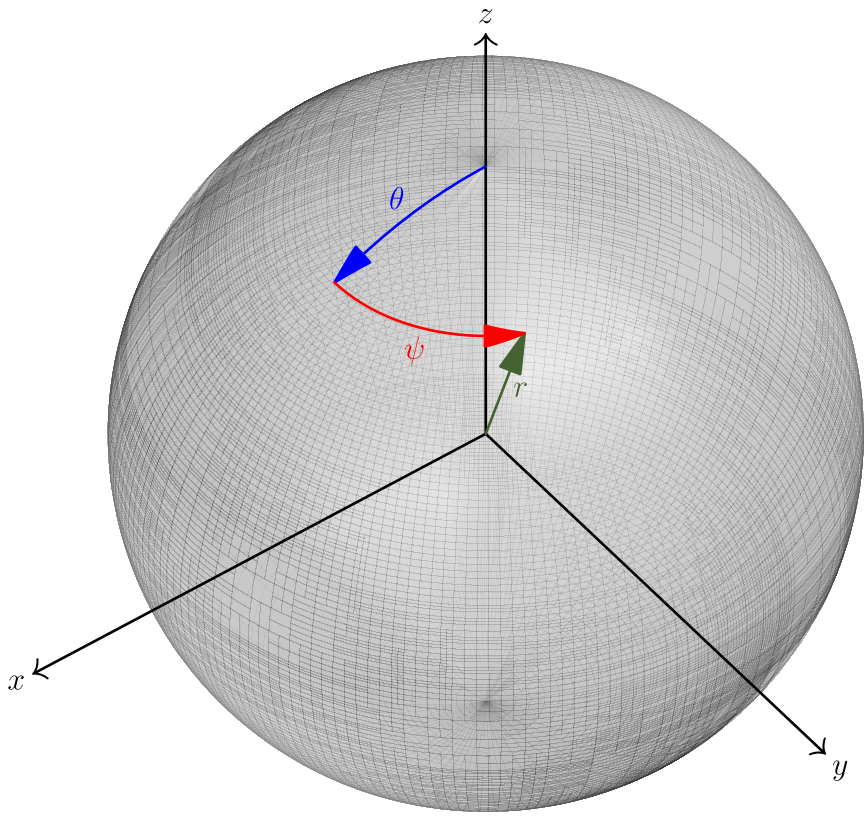
题解. 对于这种旋转问题，我们的通用做法都是先将坐标系变换到一个比较容易处理的位置，再把它转回来。在这题中，我们善于处理的自然是二维的，在 xOy 平面上的旋转：



这个变换是乘以旋转矩阵：

$$p' = \begin{pmatrix} x' \\ y' \\ z \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\alpha - y\sin\alpha \\ x\sin\alpha + y\cos\alpha \\ z \end{pmatrix}$$

那么如何把旋转轴和点转到这个位置？考虑旋转轴的方位角和仰角：



因此只要把 r 绕 z 轴逆时针转 ψ，再绕 y 轴逆时针转 θ 即可。叠加两个旋转矩阵：

$$
\begin{aligned}
r' &= \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} rx \\ ry \\ rz \end{pmatrix} \\
&= \begin{pmatrix} x\cos\theta\cos\psi - y\cos\theta\sin\psi - z\sin\theta \\ y\cos\psi + x\sin\psi \\ x\sin\theta\cos\psi - y\sin\theta\sin\psi + z\cos\theta \end{pmatrix}
\end{aligned}
$$

为了求出 ψ 和 θ 的值，

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;


int main(){
   double x, y, z, rx, ry, rz, t;
```

```
10      cin >> x >> y >> z >> rx >> ry >> rz >> t;
11      double l = sqrt(rx*rx + ry*ry + rz*rz);
12      rx /= l;
13      ry /= l;
14      rz /= l;
15      double d = sqrt(ry*ry + rz*rz);
16      double qx, qy, qz;
17      if (d != 0) {
18         qx = x;
19         qy = y * rz / d - z * ry / d;
20         qz = y * ry / d + z * rz / d;
21      } else {
22         qx = x;
23         qy = y;
24         qz = z;
25      }
26      x = qx * d - qz * rx;
27      y = qy;
28      z = qx * rx + qz * d;
29      qx = x * cos(t) - y * sin(t);
30      qy = x * sin(t) + y * cos(t);
31      qz = z;
32      x = qx * d + qz * rx;
33      y = qy;
34      z = -qx * rx + qz * d;
35      if (d != 0) {
36         qx = x;
37         qy = y * rz / d + z * ry / d;
38         qz = -y * ry / d + z * rz / d;
39      } else {
40         qx = x;
41         qy = y;
42         qz = z;
43      }
44      cout << setiosflags(ios::fixed) << setprecision(6) << qx << " " << qy << "
           " << qz << endl;
45      return 0;
46  }
```