Rotation: A C++ Class for 3D Rotations
Reference Sheet

Description	*Mathematical Notation*	*Computer Code*
Definition[a]	Let R be a rotation.	`Rotation R;`
	Let R be a rotation specified by yaw, pitch, and roll.[b]	`Rotation R(yaw, pitch, roll, ZYX);`
	Let R be a rotation specified by three angles, ϕ_1, ϕ_2, ϕ_3 applied in the order x-y-z.[b]	`Rotation R(phi1, phi2, phi3, XYZ);`
	Let $R_{\mathbf{a}}(\alpha)$ be the rotation about the vector $\mathbf{a}$ through the angle α.	`Vector a;` `Rotation R( a, alpha );`
	Let R be the rotation about the direction specified by the angles (θ, ϕ) through the angle α.	`sphericalCoord s( theta, phi );` `Rotation R( s, alpha );`
	Let R be the rotation specified by the vector cross product $\mathbf{a} \times \mathbf{b}$.	`Vector a, b;` `Rotation R( a, b );`
	Let R be the rotation that maps the set of basis vectors $\mathbf{a}_i$ to the set $\mathbf{b}_i$, $i = 1, 2, 3$.	`Vector a1, a2, a3, b1, b2, b3;` `Rotation R( a1,a2,a3, b1,b2,b3 );`
	Let R be the rotation specified by the (unit) quaternion q.[c]	`quaternion q;` `Rotation R( q );`
	Let R be the rotation specified by the 3×3 rotation matrix A_{ij}.[d]	`matrix A;` `Rotation R( A );`
	Let R be a random rotation, designed to randomly orient any vector uniformly over the unit sphere.	`Random rv;` `Rotation R( rv );`
Input a rotation R	*NA*	`cin >> R;`
Output the rotation R	*NA*	`cout << R;`
Assign one rotation to another	Let $R_2 = R_1$ *or* $R_2 \Leftarrow R_1$	`R2 = R1;` *or* `R2( R1 );`
Product of two successive rotations[e]	$R_2 R_1$	`R2 * R1;`
Rotation of a vector $\mathbf{a}$	$R\,\mathbf{a}$	`R * a;`
Inverse rotation	R^{-1}	`inverse( R );` *or* `-R;`
Convert a rotation to a quaternion	If $R_{\mathbf{u}}(\theta)$ is the rotation, then $q = \cos(\theta/2) + \mathbf{u}\sin(\theta/2)$.	`to_quaternion( R );`
Convert a rotation to a 3×3 matrix	*This space is too small to describe it.*	`to_matrix( R );`
Factor a rotation into a yaw, pitch and roll sequence	*This space is too small to describe it.*	`sequence s = factor( R, ZYX );`[f]

Rotation: A C++ Class for 3D Rotations
Reference Sheet (Continued)

Description	*Mathematical Notation*	*Computer Code*
Unit vector along the axis of rotation (in standard form where rotation angle is counterclockwise)[g]	Unit vector $\mathbf{u}$ in the rotation $R_{\mathbf{u}}(\theta)$.	`Vector( R );` *or* `vec( R );`
Angle of rotation, which in standard form is nonnegative and counterclockwise[g]	Angle of rotation θ in the rotation $R_{\mathbf{u}}(\theta)$.	`double( R );` *or* `ang( R );`
Check for equality	Is $R_2 = R_1$?	`R2 == R1;`
Check for inequality	Is $R_2 \neq R_1$?	`R2 != R1;`

a A rotation is represented here by the pair $(\mathbf{u}, \theta)$, where $\mathbf{u}$ is the unit vector along the axis of rotation, and θ is the counterclockwise rotation angle.

b The order is significant: first yaw is applied as a counterclockwise rotation about the z–axis, then pitch is applied as a counterclockwise rotation about the y'–axis, and finally, roll is applied as a counterclockwise rotation about the x''–axis. The coordinate system is constructed from the local tangent plane in which the z–axis points toward earth center, the x–axis points along the direction of travel, and the y–axis points to the right, in order to form a right-handed coordinate system. This particular order is specified by using `ZYX`. There are a total of twelve possible orderings available to the user, six of them have distinct principal rotation axes: `ZYX`, `XYZ`, `XZY`, `YZX`, `YXZ`, `ZYX`; and six have repeated principal rotation axes: `ZYZ`, `ZXZ`, `YZY`, `YXY`, `XYX`, `XZX`.

c A quaternion is defined in the Rotation class as follows:

```
struct quaternion {
   double w;   // scalar part
   Vector v;   // vector part
};
```

A *unit* quaternion requires that $w^2 + |\mathbf{v}|^2 = 1$.

d A matrix is defined in the Rotation class as follows:

```
struct matrix {
   double a11, a12, a13,   // 1st row
          a21, a22, a23,   // 2nd row
          a31, a32, a33;   // 3rd row
};
```

In order to qualify as a rotation, the 3×3 matrix A must satisfy the two conditions: $A^\dagger = A^{-1}$ and $\det A = 1$.

e In general, rotations do not commute, i.e., $R_1 R_2 \neq R_2 R_1$, so the order is significant and goes from right to left.

f A (rotation) sequence is defined in the Rotation class as simply a set of three angles (in radians):

```
struct sequence {
   double first,    // first rotation (rad) to apply to body axis
          second,   // second rotation (rad) to apply to body axis
          third;    // third rotation (rad) to apply to body axis
};
```

The order these are appplied is always left to right, `first`, `second`, `third`. *How* they get applied is specified by using one of the following, which is also applied left to right: `ZYX`, `XYZ`, `XZY`, `YZX`, `YXZ`, `ZYX`, `ZYZ`, `ZXZ`, `YZY`, `YXY`, `XYX`, `XZX`.
For example, the order `XYZ` would apply `first` to rotation about the x-axis, `second` to rotation about the y-axis, and `third` to rotation about the z-axis.

g We make use of the fact that $R_{\mathbf{u}}(\theta)$ and $R_{-\mathbf{u}}(2\pi - \theta)$ represent the same rotation to always store the couterclockwise rotation with $0 \leq \theta \leq \pi$.