Conversion from Direction Vector to Quaternion

Karljohan Lundin*

13th February 2002

The following is an derivation of how to convert a direction vector $\vec{a}$ in 3d space to a quaternion q, that describes its orientation, or more exactly its rotation from the $\vec{c}$ vector. Effort has been made to avoid trigonometric function calls, and thus the result can be found unnecessary complex.

$$
\begin{aligned}
q &= [w, \vec{v}]\,, \text{where } w = \cos\frac{\phi}{2} \text{ and } \vec{v} = \hat{r} \cdot \sin\frac{\phi}{2} \\
\hat{r} &= \frac{\vec{a} \times \vec{c}}{|\vec{a} \times \vec{c}|} \\
\vec{a} \cdot \vec{c} &= |\vec{a}| \cdot |\vec{c}| \cdot \cos\phi \Rightarrow \cos\phi = \frac{\vec{a} \cdot \vec{c}}{|\vec{a}| \cdot |\vec{c}|} \\
w = \cos\frac{\phi}{2} &= \sqrt{\frac{1 + \cos\phi}{2}} \\
\sin\frac{\phi}{2} &= \sqrt{\frac{1 - \cos\phi}{2}} \\
\left.\begin{aligned}
\vec{v} = \hat{r} \cdot \sin\tfrac{\phi}{2} &= \tfrac{\vec{a}\times\vec{c}}{|\vec{a}\times\vec{c}|} \cdot \sqrt{\tfrac{1}{2} - \tfrac{\vec{a}\cdot\vec{c}}{2\cdot|\vec{a}|\cdot|\vec{c}|}} \\
w = \cos\tfrac{\phi}{2} &= \sqrt{\tfrac{1}{2} + \tfrac{\vec{a}\cdot\vec{c}}{2\cdot|\vec{a}|\cdot|\vec{c}|}}
\end{aligned}\right\} &\Rightarrow q = \left[\sqrt{\frac{1}{2} + \frac{\vec{a}\cdot\vec{c}}{2\cdot|\vec{a}|\cdot|\vec{c}|}},\ \frac{\vec{a}\times\vec{c}}{|\vec{a}\times\vec{c}|} \cdot \sqrt{\frac{1}{2} - \frac{\vec{a}\cdot\vec{c}}{2\cdot|\vec{a}|\cdot|\vec{c}|}}\right]
\end{aligned}
$$

To avoid division by zero, make sure that the direction vector $\vec{a}$ is not of zero length and that it does not coincide with the $\vec{c}$ vector. However please note that the result has not been verified and can thus not be guaranteed to be neither correct or working. If there are any questions or if there is something found to be wrong in the derivation, please contact me on this email address: karlu850@student.liu.se

Finding the rotation from the x axis using C++ and vector and quaternion types, the code become something like this:

```
inline Quaternion dir2quat( Vec3f d ){
  double dl = d.length();
  Vec3f dx = ( d % Vec3f(1,0,0) );
  double dsx = d * Vec3f(1,0,0);
  double dxl = dx.length();

  if( dxl < epsilon ||
      dl  < epsilon     )
    return Quaternion( Vec3f(1,0,0), 0 );

  return Quaternion( dx * sqrt( (dl-dsx) / (2*dl) ) / dxl,
                          sqrt( (dl+dsx) / (2*dl) )        );
}
```

*karlu850@student.liu.se