Spherical Arcs using Quaternion Division

Kurt Nalty

January 11, 2014

Abstract

Quaternion division is commonly illustrated using spherical arcs on a sphere. This note turns things around, to show how to draw spherical arcs using quaternion division. We start with standard quaternion definitions, then show the math and standard c code for drawing great arcs on spheres. While illustrated using three-space, a very similar extension applies to four-space.

Standard Quaternion Definitions

Quaternions are a combination of a scalar and three-vector, resulting in an effective four-vector. I use a tilde overmark over capital letters to identify quaternions, and then use a lower case letter to identify the scalar (time) component, and a capital letter with vector overmark to identify the vector component. As an example, $\tilde{Q} = q + \vec{Q}$. While I will commonly use matched lower and upper case letters, this is not always the case. For example, with electromagnetic potential, I will often use $\tilde{\Phi} = \phi + \vec{A}$.

Given the symbology above, the right-handed quaternion product is

$$\begin{aligned} \tilde{A}\tilde{B} &= (a + \vec{A})(b + \vec{B}) \\ &= (ab - \vec{A} \cdot \vec{B}) + (a\vec{B} + b\vec{A} + \vec{A} \times \vec{B}) \end{aligned}$$

This product is associative, but not commutative. I mention in passing that there also exists a left-handed quaternion product, where the sign of the cross product is changed. This left handed product is also associative, but not commutative.

We define the magnitude of a quaternion as

$$|\tilde{A}| = \sqrt{a^2 + \vec{A} \cdot \vec{A}}$$

We define the dot product of two quaternions as

$$\tilde{A} \cdot \tilde{B} = ab + \vec{A} \cdot \vec{B}$$

The product has the fine identity that

$$|\tilde{A}\tilde{B}| = |\tilde{A}|\,|\tilde{B}|$$

We define conjugation as

$$\tilde{A}^* = a - \vec{A}$$

Combining the previous two equation, we can define unit division as

$$\frac{1}{\tilde{A}} = \frac{\tilde{A}^*}{|\tilde{A}|^2}$$

We have two forms of division: pre-division and post-division.

$$\frac{1}{\tilde{B}}\tilde{A} = \frac{\tilde{B}^*\tilde{A}}{|\tilde{B}|^2}$$

$$\tilde{A}\frac{1}{\tilde{B}} = \frac{\tilde{A}\tilde{B}^*}{|\tilde{B}|^2}$$

We can explicitly separate the tensor and versor forms of the quaternion product.

$$\begin{aligned}
\tilde{A} &= \sqrt{a^2 + \vec{A} \cdot \vec{A}}\frac{a + \vec{A}}{\sqrt{a^2 + \vec{A} \cdot \vec{A}}} \\
&= \sqrt{a^2 + \vec{A} \cdot \vec{A}}\,(\cos\theta + \vec{u}\sin\theta) \\
\vec{u} &= \frac{\vec{A}}{\sqrt{\vec{A} \cdot \vec{A}}} \\
\sin\theta &= \frac{|\vec{A}|}{\sqrt{a^2 + \vec{A} \cdot \vec{A}}} \\
\cos\theta &= \frac{a}{\sqrt{a^2 + \vec{A} \cdot \vec{A}}}
\end{aligned}$$

The generic form for a unit quaternion is $\cos\theta + \vec{u}\sin\theta$.

Quaternion Division and Spherical Arcs

I start with the identity

$$\tilde{B} \;=\; \tilde{A} * \left(\frac{1}{\tilde{A}} \tilde{B} \right)$$

The first two terms on the right hand side cancel, leading to our identity. This identity allows the interpretation of the quaternion pre-division as generating a planar operator rotating $\tilde{A}$ to $\tilde{B}$.

Let's look at the details of this rotation operator. Convert the rotator to polar format.

$$\begin{aligned}
\left(\frac{1}{\tilde{A}} \tilde{B} \right) &= \tilde{r} = r + \vec{R} \\
&= \sqrt{r^2 + \vec{R} \cdot \vec{R}} \left(\frac{r}{\sqrt{r^2 + \vec{R} \cdot \vec{R}}} + \frac{\vec{R}}{\sqrt{\vec{R} \cdot \vec{R}}} \frac{\sqrt{\vec{R} \cdot \vec{R}}}{\sqrt{r^2 + \vec{R} \cdot \vec{R}}} \right) \\
&= \sqrt{r^2 + \vec{R} \cdot \vec{R}} \left(\cos\psi + \frac{\vec{R}}{\sqrt{\vec{R} \cdot \vec{R}}} \sin\psi \right) \\
&= \sqrt{r^2 + \vec{R} \cdot \vec{R}} \left(\cos\psi + \vec{u} \sin\psi \right)
\end{aligned}$$

The scale factor $\sqrt{r^2 + \vec{R} \cdot \vec{R}} = 1$ for the case of the unit sphere. We have the axis of rotation $\vec{u}$, and the angle ψ.

To trace out the arc, we subdivide our angle ψ by the number of steps we wish to use. We create intermediate rotators $\tilde{r}(m, n)$ and associated positions $\tilde{p}(m, n)$.

$$\begin{aligned}
\tilde{r}(m, n) &= \cos\left(\frac{m\psi}{n} \right) + \vec{u} \sin\left(\frac{m\psi}{n} \right) \\
\tilde{p}(m, n) &= \tilde{A} * \tilde{r}(m, n)
\end{aligned}$$

To illustrate this process, let's draw three great arcs making a spherical triangle on a unit sphere, as illustrated in Figure 1 below.

Figure 1: Great Arcs A → B → C → A

Great Triangles on a Sphere

$$\tilde{A} \;=\; \tilde{A} * \left(\frac{1}{\tilde{A}}\tilde{B}\right) * \left(\frac{1}{\tilde{B}}\tilde{C}\right) * \left(\frac{1}{\tilde{C}}\tilde{A}\right)$$

This tautology illustrates a set of three great circles on a unit sphere, starting at $\tilde{A}$, proceding to $\tilde{B}$, then to $\tilde{C}$, and ending back up again at $\tilde{A}$.

Here is the process we use to generate the great arcs. We first do the pre-division of the position vectors, which results in another unit quaternion. We convert this quaternion to polar format, identifying the angle and unit vector associated with the rotation. We subdivide the arc into as many steps as we like, dividing the angle by the number of steps. We then calculate the intermediate quaternion factors using the scaled angle. We multiply this quaternion against the original position to find the new destination.

Standard C Code for Illustration

Code for the file below is at `http://www.kurtnalty.com/Unit_Sphere.c` Source code for the included file 'quaternion.c' is at `http://www.kurtnalty.com/quaternion.c` Source code for the verification of the quaterion routines is at `http://www.kurtnalty.com/Quaternion_Test.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "quaternion.c"

int main(void)
{
FILE* Output;
int i,j,k;
double x,y,z,t;
double r, theta, phi, pi;
double Mag, psi;
double angle;
struct quaternion a,b,c,d,e,f;
struct vector u;
```

```
pi = M_PI; // gcc predefined PI
Output = fopen("Sphere.xyz","w");

r = 1.0; // unit sphere
/*
for (j=1;j<18;j++) { // 10 degree increments
phi = j*pi/18.0; // equator
for (i=0;i<51;i++) { // approximate equatorial circle with 100 segments
theta = i*pi/25.0; // cover 2 pi
x = r*sin(phi)*cos(theta);
y = r*sin(phi)*sin(theta);
z = r*cos(phi);
if(i==0) fprintf(Output,"%g  %g  %g  %d \n",x,y,z,0);
else fprintf(Output,"%g  %g  %g  %d \n",x,y,z,1);
}
}

for (j=0;j<18;j++) {
theta = j*pi/18.0;
for (i=0;i<51;i++) { // approximate equatorial circle with 100 segments
phi = i*pi/25.0; // cover 2 pi
x = r*sin(phi)*cos(theta);
y = r*sin(phi)*sin(theta);
z = r*cos(phi);
if(i==0) fprintf(Output,"%g  %g  %g  %d \n",x,y,z,0);
else fprintf(Output,"%g  %g  %g  %d \n",x,y,z,80);
}
}
*/
// define and plot points a, b, and c

theta = 60.0*pi/180.0;
phi = 60.0*pi/180.0;
a.t = 0.0;
a.x = r*sin(phi)*cos(theta);
a.y = r*sin(phi)*sin(theta);
a.z = r*cos(phi);
fprintf(Output,"%g  %g  %g  %d \n",0.0,0.0,0.0,0); // origin
```

```
fprintf(Output,"%g  %g  %g  %d \n",a.x,a.y,a.z,40);

theta = -20.0*pi/180.0;
phi = 30.0*pi/180.0;
b.t = 0.0;
b.x = r*sin(phi)*cos(theta);
b.y = r*sin(phi)*sin(theta);
b.z = r*cos(phi);
fprintf(Output,"%g  %g  %g  %d \n",0.0,0.0,0.0,0); // origin
fprintf(Output,"%g  %g  %g  %d \n",b.x,b.y,b.z,120);

theta = 30.0*pi/180.0;
phi = -45.0*pi/180.0;
c.t = 0.0;
c.x = r*sin(phi)*cos(theta);
c.y = r*sin(phi)*sin(theta);
c.z = r*cos(phi);
fprintf(Output,"%g  %g  %g  %d \n",0.0,0.0,0.0,0); // origin
fprintf(Output,"%g  %g  %g  %d \n",c.x,c.y,c.z,220);

// define quaternion d as (1/a)b, which takes us from a to b via postmultiplicatio

d = QuaternionMult(QuaternionInv(a),b);

// convert this to polar form to show vector u and angle psi

Q2P(d, &u, &psi, &Mag);
printf("(1/a)b = "); QuaternionPrint(d); printf("  vector u = ");
VectorPrint(u);  printf(" angle = %g    Mag = %g  \n", psi*180.0/pi, Mag);

// draw this first arc in 10 segments

fprintf(Output,"%g  %g  %g  %d \n",a.x,a.y,a.z,0);  // moveto
for (i=1;i<11;i++) {
angle = psi*i/10.0;
e.t = cos(angle);
e.x = sin(angle)*u.x;
e.y = sin(angle)*u.y;
```

```
e.z = sin(angle)*u.z;
f = QuaternionMult(a,e);
fprintf(Output,"%g  %g  %g  %d \n",f.x,f.y,f.z,40);  // color 40
}

// draw the second arc in 10 segments

d = QuaternionMult(QuaternionInv(b),c);
Q2P(d, &u, &psi, &Mag);
fprintf(Output,"%g  %g  %g  %d \n",b.x,b.y,b.z,0);  // moveto
for (i=1;i<11;i++) {
angle = psi*i/10.0;
e.t = cos(angle);
e.x = sin(angle)*u.x;
e.y = sin(angle)*u.y;
e.z = sin(angle)*u.z;
f = QuaternionMult(b,e);
fprintf(Output,"%g  %g  %g  %d \n",f.x,f.y,f.z,120);  // color 120
}

// draw this third arc in 10 segments

d = QuaternionMult(QuaternionInv(c),a);
Q2P(d, &u, &psi, &Mag);
fprintf(Output,"%g  %g  %g  %d \n",c.x,c.y,c.z,0);  // moveto
for (i=1;i<11;i++) {
angle = psi*i/10.0;
e.t = cos(angle);
e.x = sin(angle)*u.x;
e.y = sin(angle)*u.y;
e.z = sin(angle)*u.z;
f = QuaternionMult(c,e);
fprintf(Output,"%g  %g  %g  %d \n",f.x,f.y,f.z,220);  // color 220
}

fclose(Output);
}
```