An Introduction to Quaternions and their Applications to Rotations in Computer Graphics

Jace Miller

December 11, 2006

Abstract

In the realm of computer graphics, three dimensional scenes are created by describing geometric objects with vertices and specifying vectors that determine how a camera is to view the scene. Traditionally, these vertices and vectors have been manipulated using matrix operations. Once introduced to the mathematical concepts behind quaternions, the reader will be able to recognize advantages they have over Euler angle representation in describing rotations.

1 Introduction

Programming computer graphics requires a good grasp of mathematics. A background in linear algebra is crucial to understanding modern 3D computer graphics. Historically, computer graphics systems such as DirectX and OpenGL have been based on schemes utilizing Euler angles to perform rotation-related calculations. Euler angles assign one angle to each of three orthogonal axes of a coordinate system in three dimensional space. Any orientation can be described by a combination of these three angles. However, systems based on Euler angles have a few problems that have prompted programmers to look to other sources for solutions.

Although not as well known as Euler angles, quaternions provide a mathematical framework for handling rotation-related calculations. A quaternion is a four-dimensional vector with certain operations defined. The quaternion is a mathematical object that has strong ties to rotations in three dimensions. This property is being exploited in a variety of applications in the field of computer graphics.

2 Historical Context

The discovery of quaternions is attributed to Sir William Rowan Hamilton. Hamilton had been searching for a way to extend complex numbers in a way that would be applicable to three dimensions. At first, he thought that adding another imaginary dimension to complex

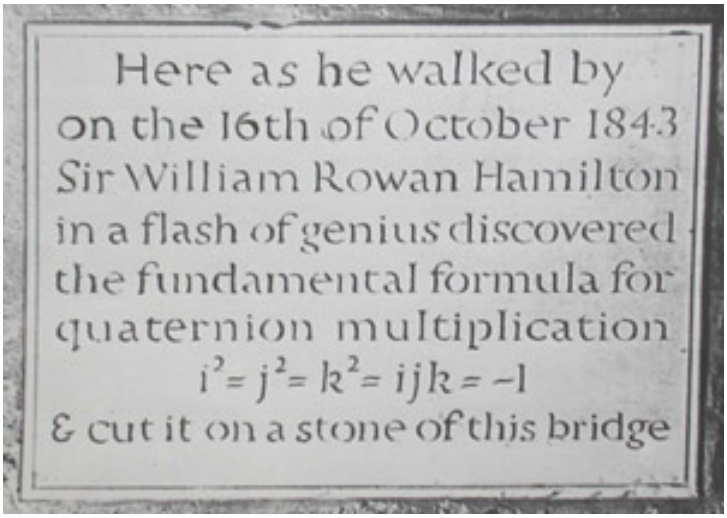

Figure 1: Broome Bridge plaque honoring Hamilton's discovery

numbers would be sufficient. However, he was unable to come up with an algebraic system with one real part and two imaginary parts in which division made sense.

On October 16, 1843, while walking with his wife past the Broome Bridge, Hamilton made a breakthrough in his quest for extending complex numbers with the concept of a system that contained one real and *three* imaginary parts. In the following excerpt from a letter to his son, Hamilton describes the moment of inspiration [2].

> " . . . it is not too much to say that I felt at once the importance. An electric circuit seemed to close; and a spark flashed forth, the herald of many long years to come of definitely directed thought and work, by myself if spared, and at all events on the part of others, if I should even be allowed to live long enough distinctly to communicate the discovery. Nor could I resist the impulse – unphilosophical as it may have been – to cut with a knife on a stone of Brougham Bridge, as we passed it, the fundamental formula with the symbols, i, j, k; namely,
>
> $$i^2 = j^2 = k^2 = ijk = -1$$
>
> which contains the Solution to the Problem . . . "

Hamilton's excitement at the discovery prompted him to carve the critical equation into a nearby bridge as insurance against the possibility that he might die before he told someone else of his breakthrough. A plaque is now located at Broome Bridge in Dublin to commemorate the event.

3 Quaternions

3.1 A Complex Connection

Since quaternions were discovered by seeking an extension to complex numbers, it is not surprising that multiple connections exist between the two. The most significant connection comes from the fact that they are both division algebras. A division algebra is a ring in which every nonzero element has a multiplicative inverse, but multiplication is not necessarily commutative [5]. Complex numbers and quaternions are two of only four division algebras that preserve the Euclidean norm. The other two are real numbers and octonians, an eight-component extension of quaternions. The multiplication rule for quaternions contains three subrules that parallel ordinary complex multiplication. Just as rotations in two dimensions can be represented using complex multiplication, rotations in three dimensions can be represented using quaternion multiplication. When unit complex numbers, those whose magnitude is one, are multiplied, the result is a unit complex number. Likewise, multiplication of unit quaternions results in a unit quaternion. This fact becomes useful when interpolating between orientations - a common problem faced by graphics programmers.

3.2 Notation, Definition of Operations, and Properties

Quaternions have four components, one real and three imaginary. They can be written as a sum of these components

$$q = q_0 + iq_1 + jq_2 + kq_3$$

or as four-dimensional vectors

$$q = (q_0, q_1, q_2, q_3) = (q_0, \vec{q}).$$

The notation indicating that the last three components can be interpreted as a vector is intentional, and its significance in relation to rotations will be explained later.

The dot product of two quaternions is similar to that of three-dimensional vectors, except with an added dimension.

$$\begin{aligned} p \cdot q &= (p_0, p_1, p_2, p_3) \cdot (q_0, q_1, q_2, q_3) \\ &= p_0q_0 + p_1q_1 + p_2q_2 + p_3q_3 \\ &= p_0q_0 + \vec{p} \cdot \vec{q} \end{aligned}$$

Quaternion addition is performed component-wise.

$$\begin{aligned} p + q &= (p_0, p_1, p_2, p_3) + (q_0, q_1, q_2, q_3) \\ &= (p_0 + q_0, p_1 + q_1, p_2 + q_2, p_3 + q_3) \\ &= (p_0 + q_0, \vec{p} + \vec{q}) \end{aligned}$$

Quaternion multiplication, denoted by the $\star$ operator, is the key to performing rotations. It is the operation we will be primarily concerned with for the remainder of this paper. The rule for multiplying quaternions is given below.

$$\begin{aligned} p \star q &= (p_0, p_1, p_2, p_3) \star (q_0, q_1, q_2, q_3) \\ &= \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_1q_0 + p_0q_1 + p_2q_3 - p_3q_2 \\ p_2q_0 + p_0q_2 + p_3q_1 - p_1q_3 \\ p_3q_0 + p_0q_3 + p_1q_2 - p_2q_1 \end{bmatrix} \\ &= (p_0q_0 - \mathbf{p} \cdot \mathbf{q}, p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}) \end{aligned}$$

Similar to the conjugate of a complex number, the conjugate of a quaternion is defined as

$$\bar{q} = (q_0, -q_1, -q_2, -q_3) = (q_0, -\vec{q}),$$

and was constructed so that $q \star \bar{q} = (q \cdot q, 0, 0, 0)$. The inverse of a quaternion is defined as

$$q^{-1} = \frac{(q_0, -q_1, -q_2, -q_3)}{q_0^2 + q_1^2 + q_2^2 + q_3^2} = \frac{\bar{q}}{|q|^2},$$

and was constructed so that $qq^{-1} = q^{-1}q = (1, 0, 0, 0)$ [3]. Notice that the inverse is just the conjugate in the unit-length case. When dealing with rotation, restricting the quaternions to unit length simplifies some calculations. Unit length quaternions obey the following restriction:

$$\begin{aligned} q \cdot q &= (q_0)^2 + (q_1)^2 + (q_2)^2 + (q_3)^2 \\ &= (q_0)^2 + \vec{q} \cdot \vec{q} = 1. \end{aligned}$$

This equation also describes the three-sphere, a four-dimensional object denoted by $\mathbf{S}^3$. The three-sphere consists of all the points located distance one away from a fixed point in four-dimensional Euclidean space [6].

The multiplication rule can also be written in matrix form.

$$\begin{aligned} p \star q &= \mathbf{P} \cdot \mathbf{Q} \\ &= \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \cdot \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \\ &= (p_0q_0 - \mathbf{p} \cdot \mathbf{q}, p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}) \end{aligned} \tag{1}$$

Now we have the tools to confirm that quaternion multiplication preserves membership in $\mathbf{S}^3$. Suppose $p \cdot p = 1$ and $q \cdot q = 1$, then

$$(p \star q) \cdot (p \star q) = q^T\mathbf{P}^T\mathbf{P}q = q^Tq = q \cdot q = 1.$$

4 Euler Angles

4.1 Computer Graphics Overview

Computer graphics allow virtual three-dimensional worlds to be described and rendered to a two-dimensional screen. Objects in the virtual world are made up of polygons which are defined by a collection of vertices. These three-dimensional objects, or meshes, are manipulated by performing vector operations on each of the vertices that make up the mesh. Basic manipulations include the transformations of scaling, translation, and rotation.

Scenes are rendered to the screen based on the position and orientation of a virtual camera. The camera position and orientation is described using three vectors. The *eye* vector represents a translation from the origin of the world; it describes the point in space where the camera is looking from. The other two vectors are referred to as the *look-at* and *up* vectors; these help describe the orientation of the camera. The third vector needed for a complete orientation frame, the *right* vector, can be calculated as the cross product of the *look-at* and *up* vectors. An orientation frame is a set of three orthogonal axes that contain all the information needed to describe an orientation in three dimensions. When associated with the camera, interpolating between orientation frames can produce smooth camera movement. Orientation frames can also be associated with meshes, or subsets of meshes, to perform animation.

4.2 Matrix Transformations

The meshes and camera that describe a virtual scene are manipulated using matrix multiplication. A 4x4 transformation matrix T is constructed as follows and multiplied by each vector the programmer wishes to transform.

$$T = \begin{bmatrix} R & t \\ s^T & u \end{bmatrix}$$

The R entry represents a 3x3 matrix containing information about rotation. The s^T and t entries correspond to 1x3 and 3x1 vectors related to scaling and translation, respectively. Multiplying such a matrix by a vector produces the desired transformed vector.

Euler angles represent rotations by assigning one angle to each of three orthogonal axes of a coordinate system in three-dimensional space. Euler angles can be used to construct matrices that, when multiplied by a vector, rotate the vector around each of the given axes. Matrices R_x, R_y, and R_z rotate a vector α, β, and γ degrees about the x, y, and z axes, respectively.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \quad R_z(\gamma) = \begin{bmatrix} cos(\gamma) & sin(\gamma) & 0 \\ -sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

All three axis-rotation matrices can be combined into a single rotation matrix R, which looks to be computationally intense.

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} cos(\beta)cos(\gamma) & -cos(\alpha)sin(\gamma) - sin(\alpha)sin(\beta)cos(\gamma) & sin(\alpha)sin(\gamma) - cos(\alpha)sin(\beta)cos(\gamma) \\ cos(\beta)sin(\gamma) & cos(\alpha)cos(\gamma) - sin(\alpha)sin(\beta)sin(\gamma) & -sin(\alpha)cos(\gamma) - cos(\alpha)sin(\beta)sin(\gamma) \\ sin(\beta) & sin(\alpha)cos(\beta) & cos(\alpha)cos(\beta) \end{bmatrix}$$

4.3 Multiple Representation

Although Euler angles are the traditional way to represent rotations in computer graphics, they cause a few problems. One problem occurs because a single orientation can be represented in multiple ways. An example will help illustrate how multiple representations can occur.

Imagine a book lying on a table so that if it were opened, it could be read by a person sitting at the table. Call this the 'home' orientation of the book. Assign a coordinate system so that the origin is at the center of the book. The x-axis passes left to right through the book, the y-axis is normal to the table and passes up and down through the book, and the z-axis is parallel to the spine of the book. If the book were rotated 180° about the z-axis, it would appear that the book had been read and closed. Call this the 'finish' orientation of the book.

Besides the z-axis rotation, there is another combination of rotations that produce the same result. Beginning at the 'home' position, rotate the book 180° about the y-axis. Now, rotate it 180° about the x-axis. The book is now in the 'finish' position, without rotating about the z-axis! The composition of rotations about two perpendicular axes can result in an orientation equivalent to a rotation about a third axis. Taking into consideration the fact that a 180° rotation can be achieved whether the book is rotated clockwise or counterclockwise in each step, there are six Euler angle representations that produce the same rotation. Stated in the form of the composite rotation matrix,

$$R(0,0,180) = R(0,0,-180) = R(180,180,0) = \\ R(180,-180,0) = R(-180,180,0) = R(-180,-180,0)$$

This 'over-representation' of rotations indicates that the Euler angle representation contains excess information. Converting an orientation to Euler angles can be ambiguous since there may be more than one option.

4.4 Gimbal Lock and Singularities

Another problem that occurs when using Euler angles is the possibility of mathematical singularities. This problem has a physical analog called gimbal lock. Understanding gimbal lock, which is easy to visualize, should help explain mathematical singularities.

A simplified drawing of a guidance system taken from the NASA Apollo 15 Flight Journal, Figure 2 illustrates the problem of gimbal lock [8]. The central rectangle depicts a gyroscopically-stabilized platform. A spinning gyroscope resists changes in orientation because of its rotational inertia. The outer frame represents the spaceship, which is free to rotate around any of the axes labeled 1 through 3. The rings which spin on the axes are called gimbals.

By default, the apparatus is oriented as in part a. Rotating 90° about axis 2 causes axes 1 and 3 to line up, as shown in part b. This effectively removes a degree of freedom from the system. The frame can no longer freely rotate about the original axis 1. From this position, if a torque acts in the direction around the original axis 1, as shown in part c, the central

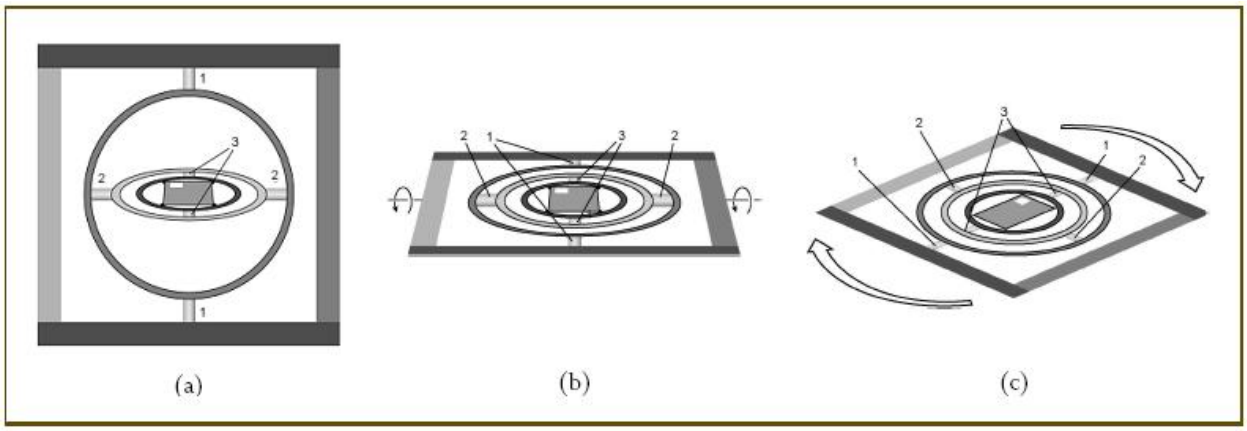

Figure 2: Demonstrating Gimbal Lock

platform is forced to move. If the system uses only one gyroscope, the gyroscope will quickly pitch 90° to conserve rotational momentum. When this happens, the guidance system will think the ship has just pitched 90° and tell the control system to fire thrusters to realign the ship, sending it in the wrong direction. If the system has multiple gyroscopes pointing in different directions, the resulting forces on the system act against each other and could cause the guidance system to self-destruct [2].

When smoothly changing between orientations using Euler angles, we encounter a similar problem. If the three rotation matrices related to Euler angles ($R_x(\alpha)$, $R_y(\beta)$, and $R_z(\gamma)$) are expressed in terms of three quaternion rotations, and the result is graphed for fixed values of β, we see a two-dimensional surface described by the parameters α and γ [2]. As β approaches 90°, the two-dimensional surface becomes a one-dimensional ring. Trying to describe a one-dimensional circle with two parameters results in problems similar to those caused by gimbal lock.

5 Quaternion Rotation

A rotation can be described with an angle of rotation and an axis about which rotation occurs. This idea is referred to as axis-angle representation and is embedded in quaternions. This kind of representation is free from the difficulties of multiple representation and gimbal lock that plague Euler angles.

Vectors can be multiplied by quaternions to produce rotations. To accommodate the four-dimensional nature of quaternions, a three-dimensional vector is cast as a quaternion. The real part of a vector written as a quaternion is zero. The three imaginary parts of the quaternion correspond to the three components of the vector. To summarize, a vector $\vec{v} = (v_0, v_1, v_2)$ can be written as a quaternion of the form $q = (0, v_0, v_1, v_2)$.

A quaternion that rotates a quaternion-cast vector θ degrees about the normalized vector

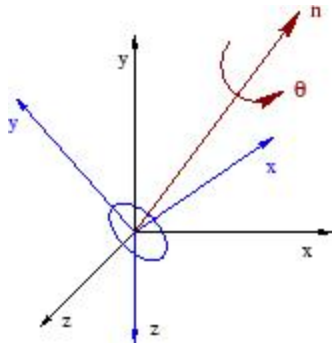

Figure 3: Axis-Angle Rotation Representation

$n = (n_0, n_1, n_2)$ is given by

$$q = (cos(\frac{\theta}{2}), n_x sin(\frac{\theta}{2}), n_y sin(\frac{\theta}{2}), n_z sin(\frac{\theta}{2}))\ [4].$$

When a vector is multiplied by a single non-zero quaternion, the result is not a vector because its real component is not zero. However, we can make the real part zero by multiplying again by the original quaternion's inverse [7]. To obtain a transformed vector $\vec{v\prime}$ from a vector $\vec{v}$ and a rotation quaternion q, multiply like this:

$$\vec{v\prime} = q \star \vec{v} \star q^{-1}$$

If unit quaternions are used, the above equation is equivalent to $v\prime = q \star v \star \bar{q}$. This conjugate shortcut is used in practice to improve efficiency. The multiplication by two quaternions explains the denominator of the $\frac{\theta}{2}$ parameter in the rotation quaternion definition. Since each multiplication rotates the vector $\vec{v}$ half of θ, two multiplications rotate $\vec{v}$ through the entire angle.

5.1 Evaluation of Efficiency

Lack of gimbal lock and multiple representation are not the only advantages quaternions have over Euler angles. When it comes to storing a representation of an orientation frame in computer memory, quaternions require less space than the 3x3 matrix required for Euler angles. Only four floating point numbers are needed to store a quaternion, compared to the nine floating point numbers required for a 3x3 matrix. From a computer science point of view, the advantages fall into a few different categories. A smaller space requirement in memory and on hard disk, and faster network transmission time are some of the specific benefits. These advantages become more pronounced when large numbers of rotations are being handled.

Measuring the number of arithmetic operations involved in common vector operations serves as a basis for comparing the computational complexity of systems based on quaternions

and Euler angles. With a comparison of computational complexity, programmers can glean an idea of how quaternion-based computer systems should be arranged. The following tables list the number of additions and multiplications required for common rotation operations.

Rotate n Vectors

	Additions	Multiplications
With rotation matrix	$6n$	$9n$
With quaternion	$12 + 6n$	$12 + 9n$

Compose Two Rotations

	Additions	Multiplications
Rotation matrix	18	27
Quaternion	12	16

The two main systems available for graphics programming, OpenGL and DirectX, inherently use matrix operations to transform vertices. Before quaternions can be used in practice by these systems, they must be converted to a matrix. The additional 12 multiplications and divisions required for rotating vectors exists because of this required conversion.

Interestingly, the only place where quaternion operations are more efficient than standard matrix operations is in the composition of multiple rotation operations [2]. Programmers should note that using quaternions to compose rotations leads to a savings in computational complexity and time, but the result should be converted back to matrix representation before doing other vector operations.

6 Conclusion

An intuitive, albeit somewhat obscure, way to describe rotations, quaternions are becoming more commonplace in computer graphics systems as awareness about them spreads. As recently as 2002, even professional 3D graphics tools like Maya and 3DS Max did not have quaternions integrated into their code [1]. Their advantages over Euler angles, such as eliminating the problem of gimbal lock and more efficient interpolation of orientation frames, are a compelling factor in their growing popularity. Computer graphics researchers have already developed advanced applications of quaternions to orientation frame interpolation. It will be exciting to see what the future holds for this field that combines computer science and mathematics.

References

[1] Kevin G. Clark. Discreet heats up siggraph 2002 with 3d animation product releases. *Business Wire*, 2002. http://investors.autodesk.com/phoenix.zhtml?c=117861&p=irol-newsArticle&ID=317501.

[2] Andrew J. Hanson. *Visualizing Quaternions*. Morgan Kaufmann, 2006.

[3] Pieter Albertus Rautenbach. Facial feature reconstruction using structure from motion. Master's thesis, University of Stellenbosch, Matieland, South Africa, April 2005.

[4] Ken Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245–254, 1985.

[5] Eric W. Weisstein. *Division Algebra*. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/DivisionAlgebra.html.

[6] Wikipedia. 3-sphere — wikipedia, the free encyclopedia, 2006. [Online; accessed 12-November-2006].

[7] Wikipedia. Quaternions and spatial rotation — wikipedia, the free encyclopedia, 2006. [Online; accessed 18-November-2006].

[8] W. D. Woods and F. O'Brien. *Apollo 15: Solo Orbital Operations - 3*. www.hq.nasa.gov/office/pao/History/ap15fj/15solo%5Fops3.htm.