Quaterions and their Applications to Rotation in 3D Space

Matt Gravelle

May 1, 2006

Abstract: *Quaternion algebra can be used for generating rigid-body rotations in 3-dimensional space. Rotation through quaternions are used in many applications such as virtual reality, aerospace engineering, and orbital mechanics. This paper explores the basics of quaternion algebra and how they can be used for rigid-body rotations in the context of computer graphics. The use of quaternions are advantageous over Euler angle sequences because they do not experience the phenomenon of gimbal lock if properly used. When using Euler angle sequences, Gimbal lock occurs when two axes align and a degree of freedom is lost.*

1 Introduction

To many, the applications and behavior of complex numbers are very intriguing. The very idea of "imaginary" numbers being useful in real word applications is astounding. Naturally, if one finds complex number intriguing then quaternion algebra, an extension to complex numbers, is bound to catch interest. Quaternions are commonly used in an application that is widely used by many people's professions and lives - computer game development and 3D virtual worlds. Quaternion algebra is commonly used as a method for rigid body rotation in three-dimensional space. Compared to another common method known as Euler angle sequence rotation, quaternions have a major advantage. Quaternions can easily be applied so they do not experience gimbal lock, allowing for smooth, unhindered rotation.

Before discussing quaternions, this paper will begin by giving a brief overview of Euler angles and traditional rotation matrices (Section 2) and how they are used to rotate objects in $\mathbb{R}^3$. Also in Section 2, the problem of gimbal lock is described, which introduces an important motivation for learning about rotations with quaternions. Then shifting gears to the main topic of this paper, Section 3 will give a brief overview of quaternion algebra that will be crucial to our understanding of how quaternions can rotate objects in $\mathbb{R}^3$. Moving on to Section 4, a few important links between rotations in $\mathbb{R}^3$ and quaternion algebra will be discussed. Lastly, Section 5 and Section 6 combine the previous information together to define the quaternion rotation operator and the theorem describing the use of quaternions in 3D space. After a discussion on rigid-body rotations using quaternions, the reason they do not encounter gimbal lock will present itself.

Euler angle sequences and gimbal lock can be very difficult to visualize. Provided for this paper is a visualization program named "Euler Angles" to aide in the understanding of Euler angle sequences and gimbal lock. Quaternion rotations are visualized with a similar program, "Quaternions," to show a specific usage of quaternion rotations. Both of these programs can be found at <http://cda.morris.umn.edu/∼grav0145/quaternions.html>. These programs will be referred to later in the paper, so it is suggested that they be downloaded for future reference.

2 Rigidbody Rotations with Rotation Matrices

2.1 The $SO(3)$ Group and Rotation Matrices

Before delving into the realm of Quaternions, it is a good to briefly look at an alternative method of rotation using rotation matrices and rotation sequences in $\mathbb{R}^3$. The *rotation group* $SO(3)$ is the group of all orthogonal rotations in $\mathbb{R}^3$. An element of $SO(3)$ can be represented by a quaternion rotation sequence, but more commonly are represented by a *rotation matrix.* A rotation matrix in $\mathbb{R}^3$ is a 3×3 matrix representing a rotation about a particular axis through a defined angle. An element $A \in SO(3)$ has the property such that $\det A = 1$. Also, a vector $\vec{v} \in \mathbb{R}^3$ multiplied by a rotation matrix A will result in a rotated vector $\vec{v}'$ such that $|\vec{v}| = |\vec{v}'|$. Rotations about the coordinate x, y, and z axes, written as $R_x(\psi)$, $R_y(\theta)$, and $R_z(\phi)$ respectively, are

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix} \quad (1)$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 1 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \quad (2)$$

$$R_z(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where the angle of rotation about a coordinate axis is called an Euler Angle [1].

2.2 Euler Angles

One of Leonard Euler's many contributions to mathematics was his work with rotation sequences. One of his theorems, often referred to as *Euler's Rotation Theorem* states:

Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis [1].

This implies that our three rotation matrices from Eq.(1)-(3) can be multiplied in sequence to describe *any* rotation in $\mathbb{R}^3$. Such a sequence of rotations is called an *Euler Angle Sequence* [1]. There are twelve different sequences that do not contain successive rotations about the same axis [1][2] (note that you can, however, use the same axis twice as long as it's not successive). One of the most common conventions, used in clasical mechanics, is called the "x-convention" (zxz-convention), which is a rotation first about the z-axis, then the x-axis, and then the z-axis again [2][4]. Another common convention is the "y-convention" (zyz-convention). Used in aerospace engineering and computer graphics is the xyz-convention (commonly called "roll-pitch-yaw" convention), which consists of a rotation about the z-axis, then the y-axis and last the x-axis. For the purposes of this paper, we will focus on the xyz-convention below since it is most widely used in the context of computer graphics. Each Euler angle sequence is perfectly capable of achieving the same results, but depending upon the application one convention may be advantageous over another.

Let us look deeper into the xyz Euler angle sequence. The matrix representation of the successive rotations are found by multiplying $R_xR_yR_z$, which results in

$$\begin{aligned} R_xR_yR_z &= R_{x''y'z} \\ &= \begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ \sin\psi\sin\theta\cos\phi-\cos\psi\sin\phi & \sin\psi\sin\theta\sin\phi+\cos\psi\cos\phi & \cos\theta\sin\psi \\ \cos\psi\sin\theta\cos\phi+\cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi-\sin\psi\cos\phi & \cos\theta\cos\psi \end{bmatrix}. \end{aligned} \tag{4}$$

Now we have a rotation matrix that will allow us to define any rotation in $\mathbb{R}^3$ using the xyz-convention with Euler angles. It is important to note that the axis of rotation are not static from a global point of view. After the rotation about the z-axis, the y and x-axes shift positions to make the axis y' and x'. Likewise, after the rotation about y', the x'-axis shifts once again to create x''. In short, each Euler angle rotation is highly dependent upon the previous rotation(s).

To help visualize, explore the motion of the airplane in the visualization software titled "Euler Angles" (see Appendix A for instructions and details about the software). If the aircraft is orientated such that there is no rotation about the z-axis, then the "pitch" of the plane can be acquired by a rotation about the y-axis. But if the aircarft has been rotated along the z-axis, then when the nose tilts up and down it no longer is rotating about the y-axis, but rather the y'-axis defined by the rotation about the z axis.

2.3 Gimbal Lock

Euler angle sequences are relatively easy to understand, easy to represent mathematically, and it doesn't take much effort to visualize – so why bother with quaternions? In every Euler rotation sequence, there is at least one point where we lose a degree of freedom [1]. This problem happens when two of the coordinate axis align with each other. This loss of a degree of freedom is referred to as *gimbal lock* [1]. In the xyz-convention, if an object is rotated 90 degrees about the y-axis, the z and x axes will fold onto one another and a degree of freedom will be lost. In the visualization software, simply rotate the airplane 90 degrees (or 270) from its initial position in the y-axis so that it is pointed straight up (or down). Note that the blue and green rings (called gimbals) align with one another, and the x and z axis collapse onto one another. Observe the motion of the plane when rotating about the z and x-axes – it the same rotation! This is a visual example of gimbal lock.

2.4 The Mathematics of Gimbal Lock

As discussed earlier, gimbal lock can cause problems with Euler angle systems. The mathematics of gimbal lock in the xyz Euler angle sequence can be used to show where this problem arises.

Beginning with a rotation matrix defined by the xyz-convention (as in Eq. (4)), and recalling that ψ, θ, and ϕ represent rotation about the x, y, and z axes respectively.

$$R_{xyz}(\psi,\theta,\phi) = \begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\psi \\ \cos\psi\sin\theta\cos\phi + \cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\theta\cos\psi \end{bmatrix} \tag{5}$$

As mentioned earlier, to acheive gimbal lock in the xyz sequence we must have a rotation of $\theta = \frac{\pi}{2}$ about the y-axis to collapse the z and x axis on top of each other. Then our resulting matrix is

$$R_{xyz}(\psi,\frac{\pi}{2},\phi) = \begin{bmatrix} 0 & 0 & -1 \\ \sin\psi\cos\phi - \cos\psi\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi & 0 \\ \cos\psi\cos\phi + \cos\psi\cos\phi & \cos\psi\sin\phi - \sin\psi\cos\phi & 0 \end{bmatrix} \tag{6}$$

Considering the case where $\psi = 0$ and ϕ is an arbitrary angle yields:

$$R_{xyz}(0,\frac{\pi}{2},\phi) = \begin{bmatrix} 0 & 0 & -1 \\ -\sin\phi & \cos\phi & 0 \\ \cos\phi & \sin\phi & 0 \end{bmatrix} \tag{7}$$

Another case where $\phi = 0$ and ψ is an arbitrary angle gives:

$$R_{xyz}(\psi,\frac{\pi}{2},0) = \begin{bmatrix} 0 & 0 & -1 \\ \sin\psi & \cos\psi & 0 \\ \cos\psi & -\sin\psi & 0 \end{bmatrix} \tag{8}$$

Replace ψ by $-\psi$ and our result is

$$R_{xyz}(-\psi,\frac{\pi}{2},0) = \begin{bmatrix} 0 & 0 & -1 \\ -\sin\psi & \cos\psi & 0 \\ \cos\psi & \sin\psi & 0 \end{bmatrix}. \tag{9}$$

Thus, $R_{xyz}(-\psi,\frac{\pi}{2},0) = R_{xyz}(0,\frac{\pi}{2},\phi)$ and we have a degenerate case in the xyz-convention with only two degrees of freedom. In the context of computer graphics and virtual worlds, gimbal lock creates unexpected and limited movement. The correct use of quaternions do not experience the phenomena of gimbal lock, which is one of the main reasons they are advantageous over the use of Euler angles. Let us proceed then, by talking about rotations with quaternions, and later the reason quaternions avoid gimbal lock will become apparent.

3 The Basics of Quaternion Algebra

The set of Quaternions are a non-commutative division algebra under addition and multiplication. In other words, under addition the set of quaternions is closed, associative, commutative, the identity element exists, and inverse elements exist for non-zero elements. Under multiplication the set of quaternions is closed, associative, the identity element exists, and inverse elements exist for non-zero elements. Furthermore, distributivity holds in this system. Note that quaternions are not commutative under multiplication as will be explained in the next section. These properties are very helpful in seeing how a quaternion behaves, but they do not give us a precise definition of quaternions. This section is devoted to defining the quaternion, quaternion addition, quaternion multiplication, and other important concepts. By exploring the specifics, the properties of a non-commutative division algebra will present themselves.

3.1 The Basic Definition and Representation

Following common convention, the set of quaternions in this paper will be denoted by $\mathbb{H}$. Let $q \in \mathbb{H}$, then q can be described by

$$\begin{aligned} q &= q_0 + \vec{q}, \\ &= q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3, \end{aligned} \tag{10}$$

where the components q_0, q_1, q_2, and q_3 are real-valued numbers and $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the standard orthonormal basis for $\mathbb{R}^3$. It can be seen that a quaternion has a scalar part q_0, and a vector part $\mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$. However, crucial detail in defining the quaternion remains. We need to know how the components behave and interact with each other. This behavior is described in the *fundamental formula of quaternion algebra*:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1. \tag{11}$$

Furthermore,

$$\mathbf{k} = \mathbf{ij} = -\mathbf{ji}, \tag{12}$$

$$\mathbf{i} = \mathbf{jk} = -\mathbf{kj}, \tag{13}$$

$$\mathbf{j} = \mathbf{ki} = -\mathbf{ik}, \tag{14}$$

The importance of Eq.(11)-(14) will be more apparent when we explore quaternion multiplication. These relations are often called *Hamilton's Rules*, named after William Hamilton who is credited as the inventor of quaternions. It is an interesting historical note that when Hamilton discovered these relations on his way to a meeting of the Irish Academy, he was so excited that he scratched the fundamental formula of quaternion algebra into the Brougham Bridge over the Royal Canal [3].

3.2 Equality, Addition, and Multiplication by a Scalar

Quaternions are a 4D vector space over $\mathbb{R}$, so equality, adition, and scalar multiplication are very straightforward and familiar. If $p, q \in \mathbb{H}$ and $c \in \mathbb{R}$, then equality, addition, and scalar multiplication are represented as:

$$\begin{aligned} &p = q \quad \Longleftrightarrow \quad p_0, = q_0,\ p_1 = q_1, p_2 = q_2, \text{ and } p_3 = q_3 \\ &p + q = (p_0 + q_0) + \mathbf{i}(p_1 + q_1) + \mathbf{j}(p_2 + q_2) + \mathbf{k}(p_3 + q_3) \\ &cq = cq_0 + cq_1\mathbf{i} + cq_2\mathbf{j} + cq_3\mathbf{k}. \end{aligned}$$

3.3 Multiplication of Two Quaternion

Let p, $q \in \mathbb{H}$. The product of two quaternions p and q is defined as

$$\begin{aligned} pq &= (p_0 + \mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3)(q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3) \\ &= p_0q_0 + \mathbf{i}p_0q_1 + \mathbf{j}p_0q_2 + \mathbf{k}p_0q_3 \\ &\quad +\mathbf{i}p_1q_0 + \mathbf{i}^2p_1q_1 + \mathbf{ij}p_1q_2 + \mathbf{ik}p_1q_3 \\ &\quad +\mathbf{j}p_2q_0 + \mathbf{ji}p_2q_1 + \mathbf{j}^2p_2q_2 + \mathbf{jk}p_2q_3 \\ &\quad +\mathbf{k}p_3q_0 + \mathbf{ki}p_3q_1 + \mathbf{kj}p_3q_2 + \mathbf{k}^2p_3q_3. \end{aligned}$$

This follows familiar algebraic multiplication, with special care given to maintain the order of the vector components. However, the above can be drastically simplified using Hamilton's rules Eq. (12)-(14). As a result of simplification and combining like terms, we find that

$$\begin{aligned} pq &= p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3) + p_0(\mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3) + q_0(\mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3) \\ &\quad + \mathbf{i}(p_2q_3 - p_3q_2) + \mathbf{j}(p_3q_1 - p_1q_3) + \mathbf{k}(p_1q_2 - p_2q_1). \end{aligned}$$

Now by inspecting the above equation, we are able to see a few familiar groupings of terms that will help us to simplify even further. This simplification requires that we denote the entire vector part of the quaternions p and q as $\vec{p}$ and $\vec{q}$ respectively. This simplification leads us to the first proposition.

Proposition 1: *Let $\forall p, q \in \mathbb{H}$. Then*

$$pq = p_0q_0 - (\vec{p} \cdot \vec{q}) + p_0\vec{q} + q_0\vec{p} + (\vec{p} \times \vec{q}). \tag{15}$$

Where p_0 and q_0 represent the scalar part of the quaternions and $\vec{p}$ and $\vec{q}$ reprsent the vector part of the quaternions.

It is important to note that $pq \neq qp$, which is most apparent in our simplified form because the cross-product is used to define our multiplication. Therefore quaternions are not commutative under multiplication. By using Proposition 1 it can be proved without much effort that quaternions are closed and associative under multiplication, the multiplicative identity element is $1+\mathbf{i}0+\mathbf{j}0+\mathbf{k}0$, and distribution over addition holds. Inverse under multiplication also holds, but to find the inverse of an element requires more investigation, which will be described in the next few sections.

3.4 The Conjugate, Norm, and Unit Quaternion

3.4.1 The Quaternion Conjugate

Once again, let $q \in \mathbb{H}$. The *quaternion conjugate*, denoted q^*, is defined as

$$q^* = q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3. \tag{16}$$

It is worth noting that quaternions are considered a specific example of *hypercomplex numbers* and share many parallels with that of complex numbers [5] and are sometimes referred to as *complex numbers of rank 4* [1]. This paper will not elaborate any further on these parallels; however, if you are familiar with complex algebra you may notice many properties extending to quaternion algebra.

3.4.2 The Norm

The *norm* of a quaternion q, is denoted $|q|$ and is defined as

$$|q| = \sqrt{q^*q}. \tag{17}$$

Using Eq. (17) it can be shown that the quaternion norm is multiplicative (*i.e.* $|ab| = |a||b|$). Furthermore, it is important to note that

$$\begin{aligned} |q|^2 &= q^*q \\ &= (q_0 - \vec{q})(q_0 + \vec{q}), \end{aligned}$$

and after we multiply using Eq. (15), we conclude

$$|q|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2. \tag{18}$$

Now that we have defined the conjugate and the norm, we can use these to describe the inverse of a quaternion.

3.4.3 Unit Quaternion

The last important term is to define the *unit quaternion* (also called a *normalized quaternion*). A unit quaternion, q, is a quaternion such that $|q| = 1$.

3.5 The Inverse

From here, finding the inverse is very straightforward. Let $q \in \mathbb{H}$, and define q^{-1} as the inverse of q. Then we know

$$\begin{aligned} & q^{-1}q = qq^{-1} = 1 \\ \Rightarrow\ & q^{-1}qq^* = q^*qq^{-1} = q^* \\ \Rightarrow\ & q^{-1} = \frac{q^*}{qq^*} = \frac{q^*}{|q|^2} \qquad \text{(since } qq^* \text{ is a real number)} \end{aligned}$$

so

$$q^{-1} = \frac{q^*}{|q|^2}. \tag{19}$$

4 Important Connections Between Quaternions and Rotations in $\mathbb{R}^3$

Before proving exacly how quaternions can perform rotations in $\mathbb{R}^3$, it is a good idea to first establish some connections between quaternions and rotation matrices in $\mathbb{R}^3$.

4.1 Pure Quaternions

A *pure quaternion* is a quaternion whose scalar part is zero. Let us denote the set of pure quaternions as $\mathbb{H}_0$, where set of pure quaternions is in a 1:1 correspondence to the set of vectors in $\mathbb{R}^3$. If given a vector $\vec{v} \in \mathbb{R}^3$, then the corresponding pure quaternion, $v \in \mathbb{H}_0$, is $v = 0 + \vec{v}$. Although the concept of pure quaternions is necessary to obtain rigidbody rotations in $\mathbb{R}^3$, it is not very apparent at first because the set $\mathbb{H}_0$ is not closed under multiplication. Let p,$q \in \mathbb{H}_0$. Multiply p and q by utilizing Eq. (15) we get

$$pq = p_0q_0 - (\vec{p}\cdot\vec{q}) + p_0\vec{q} + q_0\vec{p} + (\vec{p}\times\vec{q}),$$

since both p_0 and q_0 are equal to zero, a few of our terms dissapear to get

$$pq = -\vec{p}\cdot\vec{q} + \vec{p}\times\vec{q},$$

The term $-\vec{p} \cdot \vec{q}$ is not, in general, equal to zero; therefore the resulting quaternion of $pq = r$ may contain a non zero scalar part. So we can conclude pure quaternions are not closed under multiplication. However, we should not depair! By letting $v \in \mathbb{H}_0$ and $q \in \mathbb{H}$, it can be shown by using Proposition 1 that $qvq^* \in \mathbb{H}_0$ and $q^*vq \in \mathbb{H}_0$. For the purposes of this paper, however, we will focus only on the conjugation qvq^*.

Proposition 2: *If* $\forall v \in \mathbb{H}_0$ *and* $\forall q \in \mathbb{H}$, *then*

$$qvq^* = (q_0^2 - |\vec{q}|^2)\vec{v} + 2(\vec{q} \cdot \vec{v})\vec{q} + 2q_0(\vec{q} \times \vec{v}) \tag{20}$$

and $qvq^* \in \mathbb{H}_0$.

The proof of Proposition 2 can be achieved simply by utilizing Proposition 1 to multiply the triple product.

Let us put Proposition 2 into context. Given a vector $\vec{v} \in \mathbb{R}^3$, we define its corresponding pure quaternion as v. As usual, let $q \in \mathbb{H}$. We can get a pure quaternions, w from the following triple product:

$$w = qvq^*. \tag{21}$$

And then in turn, w corresponds to some vector $\vec{w} \in \mathbb{R}^3$. Thus, we can begin to see that quaternion conjugation defines a map onto 3D Euclidean space. As we move on, we will show that this map is in fact a linear tranformation, and therefore has a marix representation.

4.2 Connection to Rigidbody Rotations

We know that standard rotation matrices preserve the length of a vector, making it a rigidbody rotation (*i.e.* if $\vec{v} \in \mathbb{R}^3$ and $A \in SO(3)$, then $\|\vec{v}\| = \|A\vec{v}\|$). If we further explore Eq. (21), we will see a possible connection that shows conjugation of a pure quaternion can represent a rigidbody rotation in $\mathbb{R}^3$. Using the same triple product as in Eq. (21), we must prove that $|w| = |v|$. In order to prove this, however, q must be a unit quaternion as is apparent below

$$\begin{aligned} |w| &= |qvq^*| \\ &= |q||v||q^*| \\ &= |v|. \end{aligned}$$

It is vitally important to note that the above can only hold true if q is a unit quaternion. From this point on, as I often refer back to previous equations and propositions, q **will represent a unit quaternion.** This property leads us to believe that somehow the triple product in Eq. (21) could represent a rigidbody rotation in $\mathbb{R}^3$ when q is a unit quaternion.

4.3 Associating Quaternions with Angles

Let us finally begin to approach the goal of proving that quaternions can be used for rigidbody rotation in 3D Euclidean space by associating a quaternion q with an angle θ. Recall that

$$q = q_0 + \vec{q},$$

and since q is a unit quaternion,

$$q_0^2 + q_1^2 + q_2^2 + q_3^3 \quad = \quad 1,$$

and by using the familiar definition of norm for a vector in $\mathbb{R}^3$ and using it on the vector part of the quaternion, we can simplify to

$$q_0^2 + |\vec{q}|^2 \quad = \quad 1. \tag{22}$$

By combining the well known trigonometric identity $\cos^2\theta + \sin^2\theta = 1$ with Eq. (22),

$$q_0^2 + |\vec{q}|^2 = \cos^2\theta + \sin^2\theta.$$

Furthermore, we can conclude that there must be some angle such that

$$\cos^2\theta = q_0^2, \tag{23}$$

and

$$\sin^2\theta = |\vec{q}|^2,$$

for $-\pi < \theta < \pi$ (by giving θ a restriction, it will be unique). Now we have an angle θ associated with a quaternion q, but it is still unclear how exactly this is useful. So lets move further and define a vector $\vec{u}$ as

$$\begin{aligned} \vec{u} &= \frac{\vec{q}}{|\vec{q}|} = \frac{\vec{q}}{\sin\theta} && (24) \\ \Rightarrow \vec{q} &= \vec{u}\sin\theta && (25) \end{aligned}$$

Combining Eq. (25) with Eq. (23), we can write q as

$$q = \cos\theta + \vec{u}\sin\theta. \tag{26}$$

The usefulness of this representation is still not very apparent, but it will be a crucial aspect to Theorem 1 in Section 6.

4.3.1 A Connection to Successive Rotations in $\mathbb{R}^3$

Let both p and q be a unit quaternion, and $\vec{u}$ be an arbitrary vector in $\mathbb{R}^3$. Then according to Eq. (26) we can write

$$\begin{aligned} q &= \cos\alpha + \vec{u}\sin\alpha \\ p &= \cos\beta + \vec{u}\sin\beta \end{aligned}$$

then by Proposition 2,

$$\begin{aligned} pq &= \cos\alpha\cos\beta - \vec{u}\sin\alpha \cdot \vec{u}\sin\beta \\ &\quad + \cos\alpha\vec{u}\sin\beta + \cos\beta\vec{u}\sin\alpha \\ &\quad + \vec{u}\sin\alpha \times \vec{u}\sin\beta \\ &= \cos\alpha + \beta + \vec{u}\sin\alpha + \beta \end{aligned}$$

This is similar to rotation matrices in $\mathbb{R}^3$, since successive rotations $A(\theta)$, $A(\phi) \in SO(3)$ about the same vector $\vec{v}$ implies a rotation $A(\theta + \phi)$ about the vector $\vec{v}$.

5 The Quaternion Rotation Operator

The rotation operator, which we will call L_q, is what we will ultimately prove to be the method by which quaternions can be rotated in $\mathbb{R}^3$. We will define L_q as

$$L_q : \mathbb{R}^3 \longrightarrow \mathbb{R}^3 \quad \text{such that} \quad L_q(\vec{v}) = q\vec{v}q^* \quad \text{where } \vec{v} \in \mathbb{R}^3 \text{and } q \text{ is a unit quaternion.} \tag{27}$$

Using our newly defined rotation operator and Proposition 2, we can write

$$L_q(\vec{v}) = q\vec{v}q^* = (q_0^2 - |\vec{q}|^2)\vec{v} + 2(\vec{q} \cdot \vec{v})\vec{q} + 2q_0(\vec{q} \times \vec{v}). \tag{28}$$

Before the main proof, however, it is necessary introduce a few more propositions.

Proposition 3: L_q *is a Linear Operator.*

Proof Given $L_q(k\vec{a} + \vec{b}$ where $\forall k \in \mathbb{R}$ and $\forall \vec{a}, \vec{b} \in \mathbb{R}^3$, we are to show $L_q(k\vec{a} + \vec{b} = kL_q(\vec{a}) + L_q(\vec{b})$. By definition,

$$\begin{aligned}
L_q(k\vec{a} + \vec{b}) &= q(k\vec{a} + \vec{b})q^* \\
&= (kq\vec{a} + q\vec{b})q^* \qquad \text{(by distribution of quaternions)} \\
&= kq\vec{a}q^* + q\vec{b}q^* \\
&= qk\vec{a}q^* + q\vec{b}q^* \qquad \text{(since k is a scalar, } qk = kq) \\
&= (kq\vec{a} + \vec{b})q^* \qquad \text{(by distribution of quaternions)} \\
&= kL_q(\vec{a}) + L_q(\vec{b}) \qquad \text{(by definition of } L_q)
\end{aligned}$$

Therefore, the quaternion operator L_q is a linear operator. ■

Proposition 4: *The operator* L_q *preserves the length of a vector.*

Proof Given $\forall \vec{v} \in \mathbb{R}^3$, we are to show $\|L_q(\vec{v})| = |\vec{v}|$. By definition of L_q,

$$\begin{aligned}
|L_q(\vec{v})| &= |q\vec{v}q^*| \\
&= |q||\vec{v}||q^*| \qquad \text{(since the quaternion norm is multiplicative)} \\
&= |\vec{v}|. \qquad \text{(since } q \text{ is a unit quaternion)}
\end{aligned} \tag{29}$$

Therefore the length of a vector is preserved under the quaternion operator L_q. ■

6 The Theorem and Proof

Theorem 1: (As presented by Kuipers [1]) For any unit quaternion

$$q = q_0 + \vec{q} = \cos\theta + \vec{u}\sin\theta$$

and for any vector $\vec{v} \in \mathbb{R}$ the action of the operator

$$L_q(\vec{v}) = q\vec{v}q^*$$

on $\vec{v}$ may be interpreted geometrically as a rotation of the vector $\vec{v}$ through an angle 2θ about $\vec{q}$ as the axis of rotation.

Proof: Let q be a unit quaternion. Recall that a quaternion q can be broken into a scalar part, q_0, and a vector part, $\vec{q}$. Let $\vec{v} = \vec{a} + \vec{n}$, where $\vec{a}$ is the component of $\vec{v}$ along the vector part of the quaternion q, and $\vec{n}$ is the component of $\vec{v}$ normal to the vector part of q. So we can rewrite $L_q(\vec{v})$ as

$$\begin{aligned} L_q(\vec{v}) &= L_q(\vec{a} + \vec{n}) \\ &= L_q(\vec{a}) + L_q(\vec{n}). \quad \text{(by Proposition 3)} \end{aligned}$$

First we need to show that $L_q(\vec{a}) = \vec{a}$ since $\vec{a}$ lies along the vector $\vec{q}$ and will be preserved under a rotation. Because $\vec{a}$ lies along the vector $\vec{q}$, it can be written as $\vec{a} = k\vec{q}$ for some $k \in \mathbb{R}$.

$$\begin{aligned} L_q(\vec{a}) &= L_q(k\vec{q}) \\ &= kL_q(\vec{q}) \\ &= kq(\vec{q})q^* \\ &= k[(q_0^2 - |\vec{q}|^2)\vec{q} + 2(\vec{q}\cdot\vec{q})\vec{q} + 2q_0(\vec{q}\times\vec{q})] \quad \text{(by Proposition 2)} \\ &= k[(q_0^2 - |\vec{q}|^2)\vec{q} + 2|\vec{q}|^2\vec{q}] \\ &= k[q_0^2 - |\vec{q}|^2 + 2|\vec{q}|^2]\vec{q} \\ &= k[q_0^2 + |\vec{q}|^2]\vec{q} \\ &= k|q|^2\vec{q} \\ &= k\vec{q} \qquad \text{(since } \vec{q} \text{ is a unit quaternion)} \\ &= \vec{a}. \end{aligned}$$

So $\vec{a}$ is preserved under our operator, as required. Next we must show that $L_q(\vec{n})$ rotates the vector $\vec{n}$ throuh an angle 2θ about $\vec{q}$. Recall that, as demonstrated in Eq. (26),

$$q = q_0 + \vec{q} = \cos(\theta) + \vec{u}\sin(\theta) \quad (\text{where } \vec{u} = \frac{\vec{q}}{|\vec{q}|}).$$

Using Eq. (28), we may write

$$\begin{aligned} L_q(\vec{n}) &= (q_0^2 - |\vec{q}|^2)\vec{n} + 2(\vec{q}\cdot\vec{n})\vec{q} + 2q_0(\vec{q}\times\vec{n}) \\ &= (q_0^2 - |\vec{q}|^2)\vec{n} + 2q_0(\vec{q}\times\vec{n}). \qquad (\because \vec{q}\perp\vec{n}) \\ &= (q_0^2 - |\vec{q}|^2)\vec{n} + 2q_0|\vec{q}|(\vec{u}\times\vec{n}) \qquad (\because \vec{u} = \frac{\vec{q}}{|\vec{q}|}). \end{aligned}$$

We know that $\vec{u}\perp\vec{n}$, so let $\vec{n}_\perp = \vec{u}\times\vec{n}$. Then

$$L_q(\vec{n}) = (q_0^2 - |\vec{q}|^2)\vec{n} + 2q_0|\vec{q}|\vec{n}_\perp.$$

Making use of the trigonometric representation of q in Eq. (26), recall that $q_0 = \cos\theta$ and $|\vec{q}| = \sin\theta$. Then

$$\begin{aligned} L_q(\vec{n}) &= (\cos^2\theta - \sin^2\theta)\vec{n} + 2\cos\theta\sin\theta\vec{n}_\perp \\ &= \cos(2\theta)\vec{n} + \sin(2\theta)\vec{n}_\perp. \end{aligned}$$

From this we see that our operator L_q does, in fact, rotate the vector $\vec{v}$ through an angle 2θ about $\vec{q}$ as the axis of rotation. But we still do not know if the rotation operator preserves the vector's length, which is what must be shown next. Let $\vec{m} = L_q(\vec{n})$. We are to show that $|\vec{m}| = |\vec{n}| = |\vec{n_\perp}|$. We know $n \perp \vec{n}$, so the angle between them is $\frac{\pi}{2}$.

$$\begin{aligned} |\vec{n}_\perp| &= |\vec{n} \times \vec{u}| \\ &= |\vec{n}||\vec{u}|\sin(\frac{\pi}{2}) \\ &= |\vec{n}| \qquad \text{(since } \vec{u} \text{ is by definition a unit vector).} \end{aligned}$$

Therefore $|\vec{n}_\perp| = |\vec{n}|$. By Proposition 4, we know that $|\vec{m}| = |\vec{n}| = |\vec{n_\perp}|$. Now we can conclude that the operator L_q rotates a vector through an angle 2θ about $\vec{q}$ while preserving the length. Fitting this all together, we have shown that

$$\begin{aligned} \vec{w} = q\vec{v}q^* = L_q(\vec{v}) &= L_q(\vec{a} + \vec{n}) \\ &= L_q(\vec{a}) + L_q(\vec{n}) \\ &= \vec{a} + \vec{m}. \end{aligned}$$

where $\vec{m} = \cos(2\theta)\vec{n} + \sin(2\theta)\vec{n}_\perp$ and $|\vec{m}| = |\vec{n}| = |\vec{n_\perp}|$. It is clear then that $\vec{m}$ is a rotation of $\vec{n}$ through 2θ. Since $\vec{w} = \vec{a} + \vec{m}$, it is then also clear that $\vec{w} = q\vec{v}q^*$ is a vector $\vec{v}$ rotated through 2θ about $\vec{q}$ as an axis. ■

6.1 Applying Theorem 1

Theorem 1 shows that to rotate a point about an arbitrary axis using quaternions is fairly straightforward. Let us walk through a brief example to familiarize ourselves with applying Theorem 1. Let $a \in \mathbb{R}^3$ such that $a = 3\mathbf{i} + 2\mathbf{j} - 5\mathbf{k}$. We want to rotate point a about the vector $b = -4\mathbf{i} + 2\mathbf{j}$ at an angle of $\theta = \frac{\pi}{5}$. Using Theorem 1, it is not difficult to find a unit quaternion q that will accomplish such a rotation.

$$\begin{aligned} q &= \cos\frac{\theta}{2} + \frac{\vec{b}}{|\vec{b}|}\sin\frac{\theta}{2} \\ &= \cos\frac{\pi/5}{2} + \frac{-4\mathbf{i} + 2\mathbf{j}}{(-4)^2 + 2^2}\sin\frac{\pi/5}{2} \\ &= \cos\frac{2\pi}{5} + \mathbf{i}\frac{-5}{20}\sin\frac{2\pi}{5} + \mathbf{j}\frac{2}{20}\sin\frac{2\pi}{5} \\ &= \cos\frac{2\pi}{5} - \mathbf{i}\frac{1}{4}\sin\frac{2\pi}{5} + \mathbf{j}\frac{1}{10}\sin\frac{2\pi}{5}, \end{aligned}$$

so

$$\begin{aligned} q_0 &= \cos\frac{2\pi}{5} = 0.951 \\ q_1 &= -\mathbf{i}\frac{1}{4}\sin\frac{2\pi}{5} = 0.077 \\ q_2 &= +\mathbf{j}\frac{1}{10}\sin\frac{2\pi}{5} = .031 \\ q_3 &= 0. \end{aligned}$$

Now we have defined a quaternion q such that $q\vec{a}q^* = \vec{a}\,'$ where $\vec{a}\,'$ is the point $\vec{a}$ rotated about vector $\vec{b}$ at an angle of $\frac{\pi}{5}$. Now that q is known, the operator L_q is applied.

$$\begin{aligned} L_q(\vec{v}) &= qvq^* \\ &= (0.951 - \mathbf{i}0.077 + \mathbf{j}0.031 + \mathbf{k}0) \\ &\quad (2\mathbf{i} + 3\mathbf{j} - 1\mathbf{k}) \\ &\quad (0.951 + \mathbf{i}0.077 - \mathbf{j}0.031 - \mathbf{k}0) \\ &= 1.80\mathbf{i} + 2.90\mathbf{j} - 0.588\mathbf{k} \end{aligned}$$

The process is very straighforward, and most of the steps shown are only simplification. There are methods to use 3×3 matrices for rotations about arbitrary vectors, but quaternions are simple to use and computationally efficient.

6.2 Avoiding Gimbal Lock

In order to avoid gimbal lock in the 3D virtual world, quaternions must perform rotations about an arbitrary axis defined in the world. Quaternions are very efficient at calculating creating these types of rotations, as seen previously. With the method of rotating about an arbitrary axis, there is only one axis of rotation and degrees of freedom cannot be lost, therefore gimbal lock cannot occur. Quaternion rotations do not require a set of pre-defined rotation axes as Euler angle sequences do, but can change its single axis continuously. If the application requires Euler angles, then you can still use them with quaternions (as shown in Appendix B). Since the use of quaternions can easily take advantage of both Euler angle sequences and rotation about an arbitrary vector they are a very powerful tool.

The visualization program titled "Quaternions" is an example of quaternions being applied to rotations. Notice that with this application, the axis of rotation do not change depending upon previous rotations. The quaternions in this particular example are the coordinate axis and they remain fixed.

6.3 Other Reasons to Use Quaternions

Beyond the scope of this paper, but worth mentioning are a few other reasons the use of quaternions are often desired over standard rotation matrices for 3D graphics. Quaternions are computationally more efficient in many cases, and require fewer mathematical operations than multiplying rotation matrices [6]. In the case of data transfer, quaternions are more efficient with 4 elements to describe rotation compared to the 9 elements of a rotation matrix. Because of floating point cutoffs, rotations will inevitably distort. With quaternions, distortion can be handled by re-normalizing the quaternions – a much more efficient and straightforward process than with rotation matrices, which requires calculating the the determinant of matrices.

7 Conclusion and Further Studies

Euler angle sequences and quaternions are both common methods to use for rotations in $\mathbb{R}^3$. Euler angle sequences, however, encounter the obstacle of gimbal lock. Quaternion rotation does not encounter gimbal lock, and in that respect are advantageous over Euler angle sequences. They rotate objects about an arbitrary vector unlike Euler angle sequences which have a set of three pre-defined rotation vectors. Despite some of the complexities of quaternion algebra, achieving rotations is a straightforward and efficient process.

We have explored and concluded that quaternions are beneficial over Euler angle sequences, but there are still many unanswered questions as to the usefulness of quaternions compared to other methods of rotation using rotation matrices. Computationally, quaternions are said to be much more efficient in many respects compared to any rotation matrix system. An in-depth study would be beneficial to find if this is, in fact, the case. Another benefit of quaternions that is beyond the reach of this paper is that they are much more efficient at creating smooth interpolated motion [6].

Appendix A - Visualization Program Descriptions

Euler Angles

"Euler Angles" gives a visual representation of the xyz (yaw-pitch-roll) Euler angle rotation sequence. Each colored ring, or "gimbal," contains a rotation axis. This setup is modeled to resemble a 3 dimensioned gyroscope, where the term gimbal originates from. The red gimbal contains the z-axis, the green gimbal contains the y-axis, and the blue gimbal contains the $z-$axis. The three buttons on the bottom labeled x, y, and z are used to rotate the airplane. To rotate the airplane, position the mouse over the desired button, then click and drag the mouse horizontally. When rotating about the y-axis, you might notice that the displayed angle does not smoothly between 0 and 360 degrees. This is caused by the fact that the game engine used to develop the software handles rotations internally with quaternions, and converting to Euler angle values presents a few problems. Technically, the program still employees quaternions for rotations, but it is being forced into behaving like an Euler angle system.

Quaternions

"Quaternions" gives a visual representation of quaternions being used for rotation. In this particular example, the three rotations are the static coordinate axis. Similar to "Euler Angles," rotations can be accomplished by utilizing the buttons at the bottom of the window. This program, however, has an additional feature that Euler Angles did not contain. You can move the object directly by positioning the mouse over the airplane, and dragging the curser. The vector and amount of rotation is defined by the movement of the mouse in the vertical and horizontal directions.

These programs have not been thoroughly tested under the Microsoft Windows operating system, so please be aware that the programs may not be entirely stable. If you encounter any problems, bugs, or have any questions you can email me at grav0145@morris.umn.edu and I will try to resolve any issues you may have with the program.

Appendix B - Converting Euler Angles to Quaternions

The familiar yaw-pitch-roll convention (or any of the 12 Euler angle conventions) can be achieved with quaternions. Let ψ describe the yaw angle, θ describe the pitch angle, and ϕ describe the roll angle. The convention we are using correlates the yaw, pitch and roll to rotations around the z-axis, y-axis, and x-axis respectively. Let $L_{q_z}(\vec{v})$ be the rotation about the z-axis (yaw), then by Theorem 1,

$$q_z = \cos\frac{\psi}{2} + \mathbf{k}\sin\frac{\psi}{2}. \tag{30}$$

Note that we are defining our q_z with $\frac{\psi}{2}$ inside the sine and cosine functions so that L_{q_z} rotates $\vec{v}$ about the z-axis at an angle of ψ rather than 2ψ. Similarily, let us define Lq_y and Lq_x such that

$$\begin{aligned} q_y &= \cos\frac{\theta}{2} + \mathbf{j}\sin\frac{\theta}{2} \\ q_x &= \cos\frac{\phi}{2} + \mathbf{i}\sin\frac{\phi}{2}. \end{aligned}$$

Now let us define L_q as the successive rotation about the x-axis, y-axis, and then the z-axis. So

$$\begin{aligned} L_q(\vec{v}) &= Lq_x(Lq_y(Lq_z(\vec{v})) \\ &= q_x q_y q_z \vec{v} q_z^* q_y^* q_x^*. \end{aligned}$$

Therefore, $q = q_x q_y q_z$. Using Eq. (15), multiply $q_x q_y q_z$ to find our four components of q. After the tedious calculations, we find that

$$\begin{aligned} q_0 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\ q_1 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\ q_2 &= \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\ q_3 &= \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2}. \end{aligned} \tag{31}$$

Now we have a rotation operator, L_q, that allows us to work in the familiar convenion of yaw-pitch-roll. The large difference, however, is that now we do not encouter the stumbling block of gimbal lock and can obtain smooth, continuous rotations without the risk of losing a degree of freedom. Let it be known that this is not restricted to zyx-convention. All other conventions, such as zxz (spherical), can be used but different values for the components of q will result.

Gimbal Lock with Quaternions

Utilizing the four components in Eq. (31), it can be shown that gimbal lock can occur with quaternions. As with the rotation matrices, let $\theta = 0$. The resulting components of the quaternion, q, for this sequence is

$$\begin{aligned} q_0 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\psi}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\psi}{2}\right] \\ q_1 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\psi}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\cos\frac{\phi}{2}\right] \\ q_2 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\psi}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\phi}{2}\right] \\ q_3 &= \frac{\sqrt{2}}{2}\left[\sin\frac{\psi}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\phi}{2}\right] \end{aligned}$$

Now consider the case where $\psi = 0$ and ϕ be an arbitrary angle. Then

$$\begin{aligned}
q_0 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\phi}{2}\right] \\
q_1 &= \frac{\sqrt{2}}{2}\left[\sin\frac{\phi}{2}\right] \\
q_2 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\phi}{2}\right] \\
q_3 &= \frac{\sqrt{2}}{2}\left[-\sin\frac{\phi}{2}\right]
\end{aligned}$$

Next examine the case where $\phi = 0$ and $-\psi$ be an arbitrary angle. To differentiate between the first and second case, notate the following quaternion as p. Then

$$\begin{aligned}
p_0 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\psi}{2}\right] \\
p_1 &= \frac{\sqrt{2}}{2}\left[\sin\frac{\psi}{2}\right] \\
p_2 &= \frac{\sqrt{2}}{2}\left[\cos\frac{\psi}{2}\right] \\
p_3 &= \frac{\sqrt{2}}{2}\left[-\sin\frac{\psi}{2}\right]
\end{aligned}$$

So we have shown that $p = q$. Once again we lose a degree of freedom and experience gimbal lock! This shows that quaternions, if not used properly, can still encounter gimbal lock. We are still experiencing gimbal lock because we are still using a sequence of three rotations. It is a common misconception that quaternions "magically" get rid of the problem of gimbal lock. It should be understood that gimbal lock still can occur with quaternions if not used correctly.

References

[1] Kuipers, Jack B. Quaternions and Rotation Sequences. Princeton: Princeton University Press, 1999.

[2] Goldstein, Herbert. Classical Mechanics. Second Edition. Reading: Addison-Wesley Publishing Company, Inc., 1981.

[3] Altmann, Simon L. Rotations, Quaternions, and Double Groups. New York: Oxford University Press, 1986.

[4] Thornton, Stephen T.,Jerry B. Marion. Classical Dynamics of Particles and Systems. Fifth Edition. Belmont: Thomson Learning, Inc., 2004.

[5] Eric W. Weisstein. "Quaternion." From *MathWorld*–A Wolfram Web Resource. <http://mathworld.wolfram.com/Quaternion.html>

[6] Mukundand, R. *Quaternions: From Classical Mechanics to Computer Graphics, and Beyond.* Proc. 7th ATCM Conf. (2002), pp. 97-106.

Acknowledgements: Many thanks to Dr. Byungik Kahng for much of his time for discussion and help as my advisor for this paper, Dr. Barry McQuarrie for providing valuable input as a second reader, and Christopher Orth who provided TeX help and revisions.