ILLUSTRATING MULTIVARIABLE CALCULUS AND LINEAR ALGEBRA WITH THE DISK YOU WISH CAME WITH THE BOOK

Paul Bouthellier
Department of Mathematics and Computer Science
University of Pittsburgh-Titusville
Titusville, PA 16354
pbouthe@pitt.edu

In this talk a sequence of 3D animations are used to illustrate concepts of mathematics from mainly the area of calculus III and linear algebra. Other areas such as differential equations and numerical analysis are used when the problems could not be studied without these additional tools. These animations are on a DVD that took over a year to create. In showing this disk to a colleague teaching calc III they stated "I wish that disk came with the book." Hence the title of this talk.

The main idea was to create modeling problems that involve concepts from many areas of mathematics. Too often, I have heard people in industry and computer science say that new graduates know the basic concepts but not how to put everything together. I use the animations in my classes to show students that they need to understand how to put concepts from diverse fields together in order to solve practical problems. The second idea is to take a look at concepts such as Gimbal lock and quaternions which are critical in many areas of science and engineering but rarely get covered in Calc III and Linear Algebra classes.

The packages used in this project were: Studio 3D Max, Maya, Cinema 4D, Poser, Swift3D, Carrara, Paint Shop Pro, Photoshop, and Flash. Programming was done in Python, MaxScript, and ActionScript. To address one question I am frequently asked-yes, it does take a lot of time to learn all these packages, languages, and to get them to interact with each other. However, it only takes a few days (or even hours) to learn enough basics to illustrate the mathematical concepts we may need in class. As always-the objective is to illustrate the mathematical concepts, not to create flashy graphics just for show.

Computer graphics and programming can be used to illustrate mathematical concepts from junior high school through graduate school. Some of the mathematical concepts required in our results are: translations (2D and 3D), rotations in 3D via rotation matrices and quaternions, Boolean adds and subtractions, rotation of cameras (in space and about their axes), Bezier curves and surfaces, scaling, Euler angles, Gimbal lock, extrusions into 3D, interpolation, coordinate systems (local, global, camera, clipping, object-and mapping from one to another), spherical and cylindrical coordinate systems, normal and dot-products, and projecting images onto surfaces.

Modeling Project I: Creating a Book

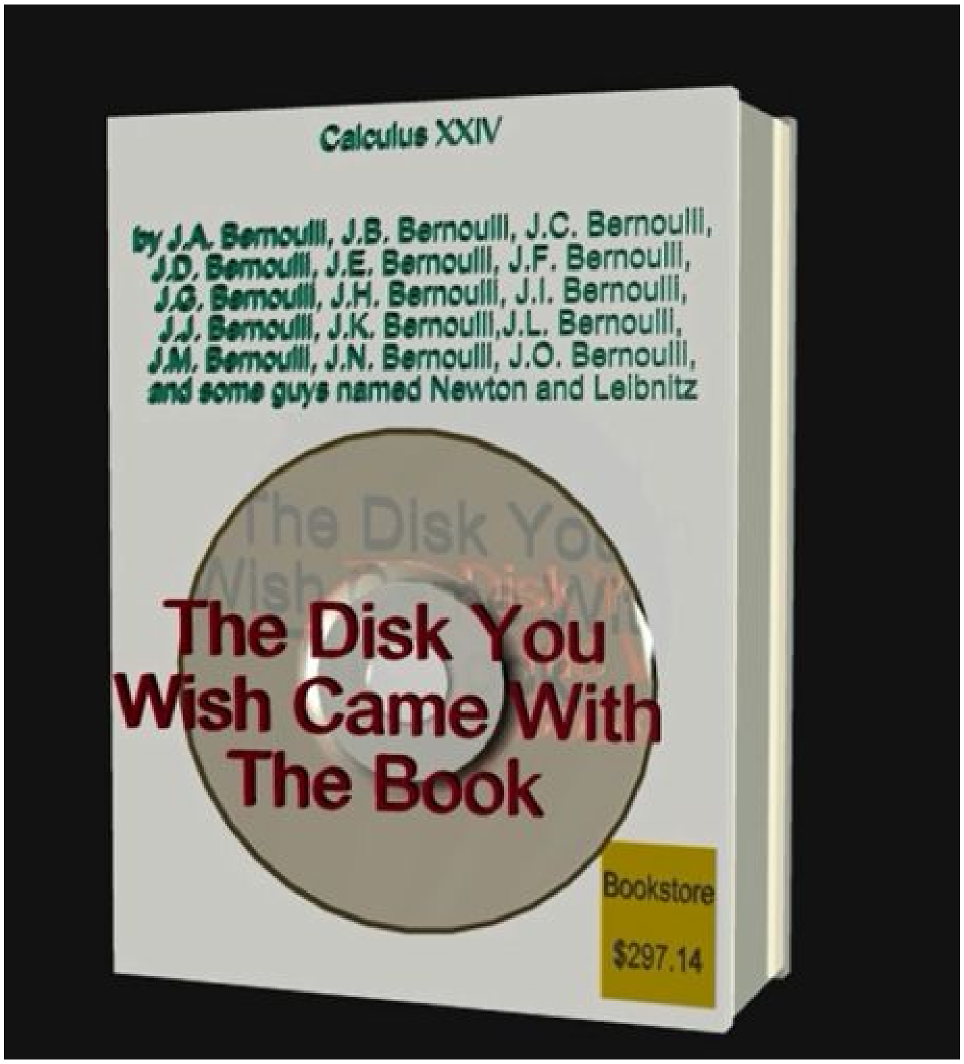

Calculus XXIV
by J.A. Bernoulli, J.B. Bernoulli, J.C. Bernoulli, J.D. Bernoulli, J.E. Bernoulli, J.F. Bernoulli, J.G. Bernoulli, J.H. Bernoulli, J.I. Bernoulli, J.J. Bernoulli, J.K. Bernoulli,J.L. Bernoulli, J.M. Bernoulli, J.N. Bernoulli, J.O. Bernoulli, and some guys named Newton and Leibnitz
The Disk You Wish Came With The Book
Bookstore
$297.14

Figure 1 (Poser, Swift 3D)

In our mathematics and statistics classes we encourage our students to break hard problems into simpler pieces. Almost any manufactured object consists of many small pieces which are then put together by some process.

Basic Steps:

- Create a box in an art package
- Scale the box to create the front cover-This illustrates scaling matrices

- Create a copy of this cover and translate it to create the back cover-this illustrates translations via matrices
- Another scaled and translated box forms the pages
- Two scaled tori are used to create the disk. It may not show up well enough on the image in the final paper-but the edges of the tori are created by polygons which can be used to estimate the circumference and surface area of the disk.
- The 3D text was created in the package Swift3D-text is an excellent example to illustrate the use of Bezier curves for modeling objects.
- The price label was created in Swift3D and projected onto the surface of the book using the package Poser-illustrating another application for projection theorems that we teach in our Calc III and Linear Algebra classes.
- Projections are then used to create the shadows and reflections on the disk.

A way to extend this project: One can put text and images on the front cover, the back cover, the spline, and the inside jacket of the book. This can be done as follows:

- Unfold the geometry of the book
- Project the required elements onto each of the local coordinate systems
- Stitch the local geometries back together into a new 3D object.

This illustrates the concept that to create even a fairly simple object many concepts of mathematics are required.

A simpler example of this that is easy to illustrate in class is that of a dress shirt with a basic pattern. Each part: cuffs, collars, front, back,.. is created as a separate piece, a localized coordinate system. It is then stitched together to form a global coordinate system (the shirt).

Modeling Project II: Modeling Objects by Bezier Curves

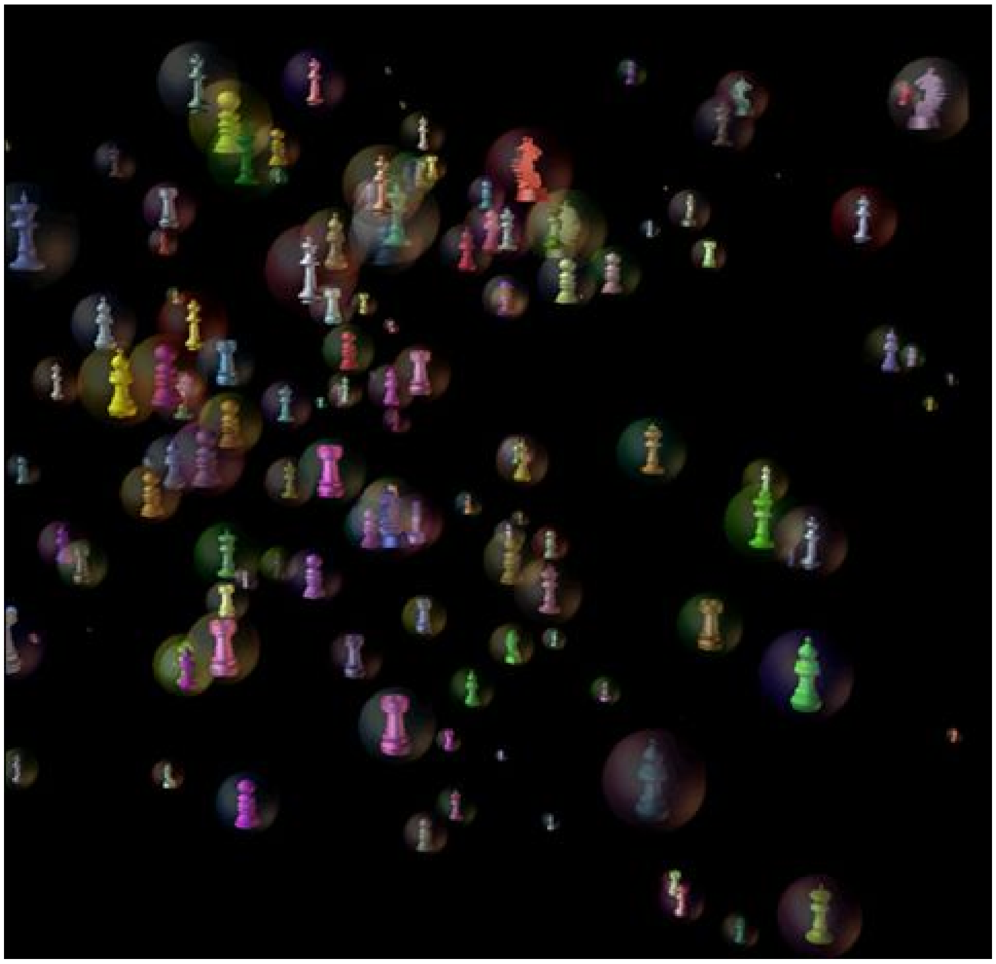
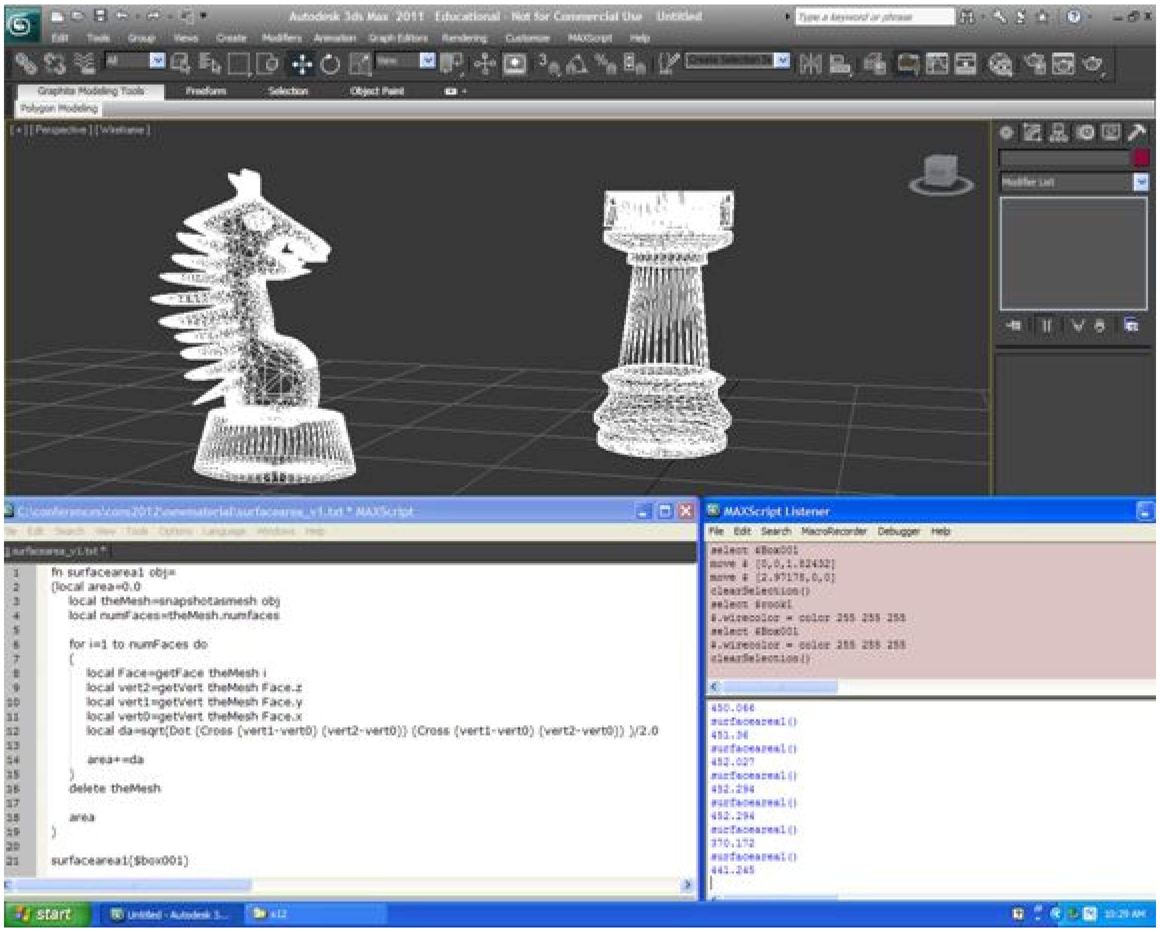


Figure 2 (Swift 3D, Poser)

In order to illustrate mathematical modeling, a 3D chess screensaver was created (a still is shown in Figure 2)

- A complete 6-piece chess set is created via:
  - Bezier curves to create the outlines of each of the pieces which are then rotated about the z-axis to create the basic shapes
  - Boolean adds and subtracts are then used to add in the King's crown, the knight's eyes and ears, the rook's tourettes, etc…
- The pieces and then rotated by using rotation matrices changing the radii and phase angles of each of the pieces.

A related project that works well in calculus 3 and linear algebra is to use the above models to estimate the surface areas and volumes of each of the chess pieces. This is illustrated below. Using the programming language of Studio 3D Max, MaxScript, we can take the vertices from the models and using cross and dot products (as illustrated in the program in Figure 3), approximate the surface areas and volumes of each of the chess pieces.



```
fn surfacearea1 obj=
(local area=0.0
   local theMesh=snapshotasmesh obj
   local numFaces=theMesh.numfaces

   for i=1 to numFaces do
   (
      local Face=getFace theMesh i
      local vert2=getVert theMesh Face.z
      local vert1=getVert theMesh Face.y
      local vert0=getVert theMesh Face.x
      local da=sqrt(Dot (Cross (vert1-vert0) (vert2-vert0)) (Cross (vert1-vert0) (vert2-vert0)) )/2.0

      area+=da
   )
   delete theMesh

   area
)

surfacearea1($box001)
```

```
select $Box001
move $ [0,0,1.82432]
move $ [2.97175,0,0]
clearSelection()
select $rook1
$.wirecolor = color 255 255 255
select $Box001
$.wirecolor = color 255 255 255
clearSelection()
```

```
450.066
surfacearea1()
451.36
surfacearea1()
452.027
surfacearea1()
452.294
surfacearea1()
452.294
surfacearea1()
370.172
surfacearea1()
441.245
```

Figure 3 (Studio 3D Max, Swift 3D, Carrara)

Note: Computer packages have upper bounds on the number of vertices they can use for modeling. Hence, there is a limit to how accurate we can estimate surface areas and volumes using the vertices from computer models.

Modeling Project III: Modeling the Flight of an Object: The Effects of Winds and the Optimal Angle of Launch

The following is one of my favorite examples.

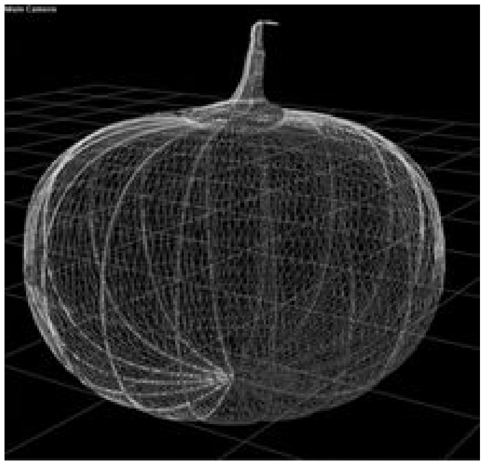


Figure 4 (Carrara/3D Max)

Pumpkin chunkin is a contest about throwing pumpkins via mechanical devices (such as air guns) with the goal of maximizing distance. To model such a contest:

- First a pumpkin is created via a 3d art package (Figure 4 above)
- To get the mass and cross-sectional area of the pumpkin-needed for the flight equations-the vertices of the model can be used to estimate the needed values.
- A system of differential equations with time-varying coefficients are derived to take into account:
    - The speed, angle, and vector of the initial launch
    - The mass of the pumpkin
    - The cross-sectional area and the effects of air-resistance
    - The effects of the wind on the flight
    - Approximating the solutions of the equations by either Euler's method or by a Runge-Kutta method. (Figure 5 below shows three stills from such an animated flight.)

This problem involves mathematical modeling, estimating volumes and surface areas, and creating the path trajectory by vector differential equations.

This problem can also be used to study the flight of objects such as baseballs and golf balls.

Two great aspects of this problem are as follows:

1. One can study the effects of wind on an object-which is much greater than most people think.
2. The optimal launch angle to achieve maximum distance is not 45 degrees but rather about 33 degrees. This is due to the effect of drag caused by the density of the air. This can be illustrated by changing the launch angle in the program and running the simulation again.

Note: A number of the machines used in pumpkin chunkin are now computer controlled. The operator enters the mass and cross sectional area of the pumpkin and the prevailing winds into the computer which guides the machine to get maximum distance on the pumpkin.

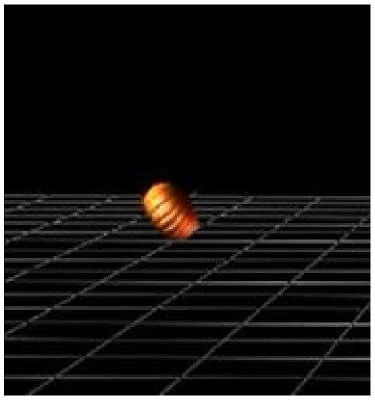


Figure 5a (Poser)

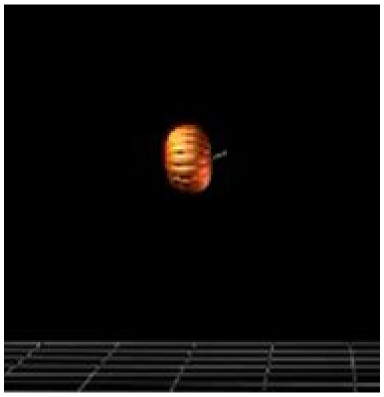


Figure 5b (Poser)

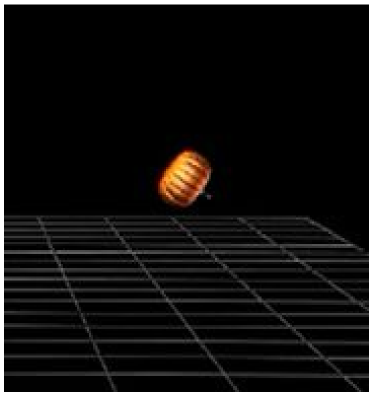


Figure 5c (Poser)

Another nice feature of modeling this problem in a 3D art package is that, once rendered, the result can be viewed from any angle.

Gimbal Lock and Quaternions

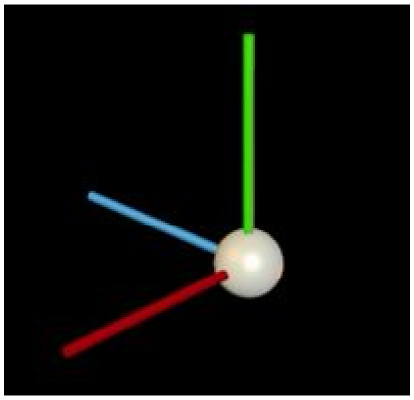


Figure 6a (Swift3D)

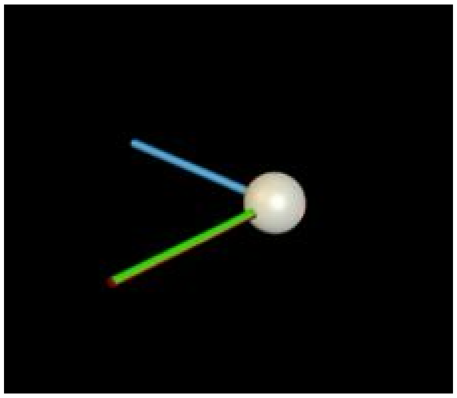


Figure 6b (Swift 3D)

The concepts of gimbal lock and quaternions are something that I feel should be discussed more often in our mathematics classes as they occur so often in the worlds of computer science, physics, and engineering.

What is Gimbal Lock?

Basically gimbal lock is a loss of one degree of freedom when doing rotations. (Look at any pictures of the command modules from the Apollo program-they had an indicator warning that the ship was about to enter Gimbal lock-meaning they could only rotate the ship about 2-axes-beyond major trouble.)

In many applications involving rotations in three-space, rotations are done as follows:

- An order of rotation is specified: for example XYZ. This means that the rotation about the X-axis is done first, then rotation about the Y-axis is done second, and finally the rotation about the Z-axis is done. We shall consider an XYZ rotation sequence order in the following.
- When the rotation about the X-axis is done, the Y and Z-axes are rotated as well.
- When the second axis is rotated, Y in this case, only the third axis Z is rotated with it. If this second axis is rotated either 90 or -90 degrees the Z-axis is rotated onto the X-axis, causing a loss of a degree of freedom in terms of rotations. This is Gimbal lock.

In Figure 6a:

- x-axis points towards the screen
- y-axis: points left
- z-axis: points up

In Figure 6b:

Rotating the y-axis -90 degrees rotates the z-axis onto the x-axis causing a loss of a degree of freedom.

Even if the second rotation is near -90 or 90 degrees the rotations can still become unstable near those angles. The standard way to minimize the problems of Gimbal lock is as follows:

- Decide how much a given object needs to rotate about the various axes.
- Choose a rotation order which assigns the second rotation axis to the angle which requires the minimum amount of rotation.
- Hope this solves any problems.
- Use quaternions.

One final note on the problem on Gimbal lock: When create an animation of a rotating object one can, via interpolation, hit a time (or times) in which the object is either at

Gimbal lock, or is close to Gimbal lock. In these cases the rotational behavior of the object can become quite unstable. Several examples of this are provide on the disk.

Quaternions

Quaternions basically define a rotation, which is specified in a term called q_0 about a vector $q=<q_1, q_2, q_{3>}$. (Four terms-hence quaternion.) Instead of being a sequence of rotations, which is subject to Gimbal lock, quaternions are a single rotation about a vector which does not suffer from Gimbal lock.

The down side to quaternions is that most people do not find them very intuitive. However, given they are used extensively in areas such as computer graphics and modeling, they should, in this authors opinion, be included in courses such as linear algebra.

On a side note: if you teach a course on the History of Mathematics, a wonderful topic is that of the "Quaternion Wars" of the 1890's which was an debate between some of the greatest mathematicians of the era on whether to use quaternions or matrix algebra in scientific applications. The current computer graphics version of this debate is playing out on forums on the Internet and in computer conferences right now. To see some of this ongoing debate, see any game design forum and search for quaternions. Have fun.

A brief descriptive of quaternions is as follows. For far more detail see [1], [2]:

Define $q=q_0+iq_1+jq_2+kq_3=q_{0+}\mathbf{q}$ where

$$i^2=j^2=k^2=ijk=-1$$

$$ij=k=-ji$$

$$jk=i=-kj$$

$$ki=j=-ik$$

Historical note: The above definitions of i, j, k by Hamilton in 1843 was the first use of non-commutative operators in mathematics.

Quaternion multiplication:

where

$$q=q_0+iq_1+jq_2+kq_3=q_{0+}\mathbf{q}$$

$$p=p_0+ip_1+jp_2+kp_3=p_{0+}\mathbf{p}$$

is defined as follows:

$$pq = p_0q_0 - \mathbf{q} \bullet \mathbf{p} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}$$

Given pure quaternions:

$$q = iq_1 + jq_2 + kq_3 = \mathbf{q}$$

$$p = ip_1 + jp_2 + kp_3 = \mathbf{p}$$

$$pq = -\mathbf{p} \bullet \mathbf{q} + \mathbf{p} \times \mathbf{q}$$

Rotations:

Given a unit quaternion

$$\cos(\frac{\theta}{2}) + u\sin(\frac{\theta}{2})$$

Where u in a unit vector in R^3

Defining

$$q^{-1} = q_0 - iq_1 - jq_2 - kq_3 = q_0 - \mathbf{q}$$

and where $v \in R^3$ is represented as a pure quaternion $(0+v)$

$$\mathbf{q}v\mathbf{q}^{-1}$$

is a right-handed rotation of v by an angle of θ about the vector **q.**

References:

[1] Andrew Hanson, "*Visualizing Quaternions*" San Francisco, CA: Morgan Kaufmann, 2006.

[2] Jack B. Kuipers, "*Quaternions and Rotation Sequences*" Princeton, NJ: Princeton University Press, 2002.