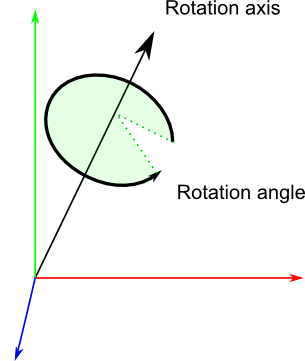
**Quaternion Visual Representation OpenGL**

Proposal:

I will visually be showing quaternion rotation with a cube as it would in a game. This is to test out my Quaternion Class I created a while back and ass onto it to test if it is game engine ready. Visually, it will rotate the cube much like it should in a game engine.

I will take the transformation matrix and show it rotating from point to point visually using OpenGL in C++.



*// RotationAngle is in radians*

x **=** RotationAxis.x **\*** sin(RotationAngle **/** 2)

y **=** RotationAxis.y **\*** sin(RotationAngle **/** 2)

z **=** RotationAxis.z **\*** sin(RotationAngle **/** 2)

w **=** cos(RotationAngle **/** 2)

Having already a Quaternion Class: <https://github.com/LShelton/Overloaded-Quaternion-Class> I will be adding more onto this program, which then allows my program to be fully tested outside of math to a graphical representation, again to test the game engine aspects of rotation

*Example:*

**Reading quaternions**

This format is definitely less intuitive than Euler angles, but it’s still readable: the xyz components match roughly the rotation axis, and w is the acos of the rotation angle (divided by 2). For instance, imagine that you see the following values in the debugger: [ 0.7 0 0 0.7 ]. x=0.7, it’s bigger than y and z, so you know it’s mostly a rotation around the X axis; and 2\*acos(0.7) = 1.59 radians, so it’s a rotation of 90°.

Similarly, [0 0 0 1] (w=1) means that angle = 2\*acos(1) = 0, so this is a unit quaternion, which makes no rotation at all.