CS-461 COMPUTER GRAPHICS

ASSIGNMENT - 3

-MAYANK WADHWANI
(170101038)

PROBLEM STATEMENT

Brief discussion on Quaternions.

SOLUTION

BRIEF INTRODUCTION

Quaternions in simple words is a kind of a number system that came at a time when complex numbers were well understood by the scholars. The scholars visualized complex numbers as imaginary 2-D in nature. Hamilton (the mathematician behind the quaternions) wanted to represent a similar analogy for higher dimensions. The main problem that arised was:

When we multiply 2 complex numbers say a+ib and c+id, we get ac-bd + i(ad+bc), this is because we have a rule i*i = -1,
Now if we assume, complex numbers with only 2 imaginary parts (which was what Hamilton in the early days thought of), then we would get an extra ij and a ji term when we multiply. Hamilton found it very difficult to find proper expressions for the same. We also had to ensure that the division algebra should hold, if we multiple 2 non zero vectors, then the product should be non zero.
So in short we cannot work with only 2 imaginary parts, that is when it struck Hamilton, we realized that this problem can be solved easily if we use 3 imaginary parts instead of just 2.

So in this number system, we represent a number as a + bi + cj + dk where a,b,c,d are scalers as usual and i,j,k are special symbols representing the imaginary parts respectively.

APPLICATIONS

As Computer Graphics enthusiasts, the main area that quaternions find the most applications is **rotation** of objects as we shall see below how it is implemented with respect to any axis in 3-D. Moreover, they also find their applications in areas of computer vision, navigation, molecular dynamics, flight dynamics.

PROBLEMS WITH OLD METHODS

The primitive methods that were used before quaternions involved matrix multiplications. Matrix multiplications uses three different axes and rotation is performed along each axes with respect to angles called as Euler angles produces the output but this method suffers from a problem of ‘Gimbal Lock’, in simple words, we cannot use matrix multiplication if the axes get aligned on top of each other. Plus matrix multiplication requires a lot of memory usage also which for higher matrices slows computation.

And so, quaternions were found very helpful here. Rotating a point with respect to any axis was now very easy thanks to this number system.

HOW TO ROTATE (THE METHOD)
Let us have an initial point in the 3-D space as. xi + yj + zk, we have to rotate this point.
Let us first represent the axis of rotation by ai + bj + ck.
We first convert this vector and represent as a quaternion. Let angle of rotation be theta. We do this by adding a scaler quantity of cos(theta/2) and multiplying sin(theta/2) to ai+bj+ck.

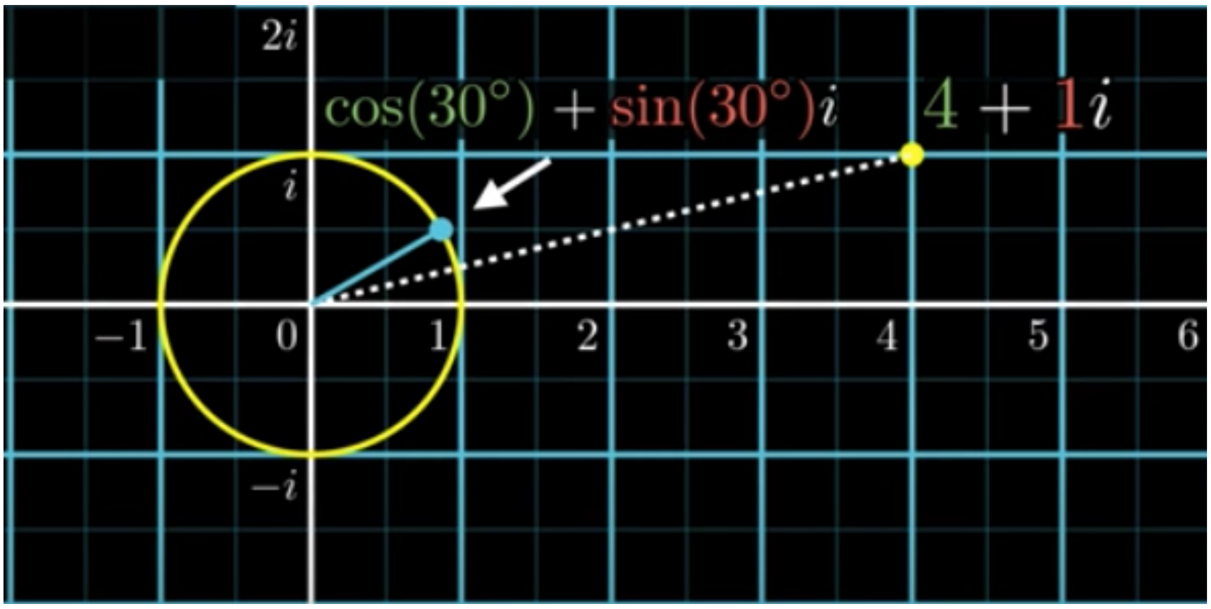
So we get the following quaternion -> p = cos(theta/2) + sin(theta/2) (ai + bj + ck)

Now we calculate the inverse of this quaternion p'
We had the initial quaternion as q=xi + yj + zk

Then, the final rotated point will be q_rotated = p * q * p'

This is the beauty of quaternion. Calculating this product in a way rotates the point. Many programmers take this a black box to rotate the objects. An apple developer said in an interview that he didn't understand anything about quaternions but had been using it as a black box to ship dozens of products involving some graphics.

A SMALL ANALOGY
We can make a small analogy with the complex numbers here. Just like in 2 dimensions, to rotate an initial point a+bi, we simply used to multiply cos(theta) + i sin(theta) to get the updated point, ie. we used this as a black box, multiplied by the cos(theta) + i sin(theta) and got the rotated point.
We can think of the quaternions in a similar way, take them as a black box, multiply p*q*p' and see the magic!



So these were quaternions, which were called as redundant and evil at the time of invention and later saw so many applications in various scientific and computer science domains