Студент 1 курса 101 группы ИЦТМС ***Абрамова Н.Д.***
Научный руководитель доцент, кан. филол. наук ***А.Н. Сак***

SCALAR AND VECTOR PRODUCT IN COMPUTER GAMES.

QUATERNIONS.

A method used in modern game engines is called quaternions.

Quaternions can be described with the formula $q = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$, where **i, j** and **k** are unit-vectors, and represents the imaginary part (like in complex numbers) and **w, x, y** and **z** belong to the set of real numbers. [1]

To begin with, multiplication of quaternions looks like $[u, v] = uv + (u, v)$. The rules of multiplication are caused by behavior of quaternions. Its imaginary part act like a vector and real part act like a scalar.

If we wanted to get a cross product, it would be non-commutative, while only the scalar-scalar and scalar-vector products are commutative. Nevertheless, let's get it:

$$(s + \boldsymbol{v})(t + \boldsymbol{w}) = (st - (\boldsymbol{v}, \boldsymbol{w})) + (s\boldsymbol{w} + t\boldsymbol{v} + [\boldsymbol{v}, \boldsymbol{w}])$$

In this formula (s + **v**) and (t + **w**) are vector parts added to scalars.

In a deal with inverse for non-zero quaternion we confront ensuring case: it equals for left and right inverse and is calculated such way:

$$(s + \boldsymbol{v})^{-1} = \frac{s - v}{s^2 + |v|^2}$$

Furthermore, the definition of a vector can't be changed by multiplying by -1. It means, that quaternions multiplied by 1 and -1 determine the same spin.

Additionally, consider multiplication of different quaternions. Such way, a difficult spin can be represented. Resulted composition of rotations depends on order of multiplication of quaternions. All the same, from point of mathematics doesn't matter on **q** or on **p** you multiply at first (q and p are quaternions).

$$pq\boldsymbol{v}(pq)^{-1} = pq\boldsymbol{v}q^{-1}p^{-1}$$

Namely in computer games quaternions are used for rotation of three-dimensional models. Appropriate vectors, angles and formulas are used. It supposed to be much more difficult than usage a dot or vector product, but in a point of fact quaternions are not so hard on acting as in understanding. They accelerate the performance of a game.

Based on the formula, vector **v** is rotated by an angle **θ**, moreover, this angle must be positive. It is possible due to vector **u**, that forms the direction of an axis. Around the axis rotation occurs. You can present a direction of vector **u** as a direction of right screw moving. [2]

$$v' = qvq^{-1} = \left(cos\frac{\theta}{2} + \boldsymbol{u}\,sin\frac{\theta}{2}\right)\boldsymbol{v}\,\left(cos\frac{\theta}{2} - \boldsymbol{u}\,sin\frac{\theta}{2}\right)$$

$$\boldsymbol{v}' = v\,cos^2\frac{\theta}{2} + (uv - vu)sin^2\frac{\theta}{2}cos\frac{\theta}{2} - uvu\,sin^2\frac{\theta}{2} =$$

$$= v\,cos^2\frac{\theta}{2} + 2[u,v]\sin\frac{\theta}{2}\cos\frac{\theta}{2} - \big(\,v(u,v) - 2\,u(u,v)\big)sin^2\frac{\theta}{2} =$$

$$= v\left(cos^2\frac{\theta}{2} - sin^2\frac{\theta}{2}\right) + [u,v]\left(2sin\frac{\theta}{2}cos\frac{\theta}{2}\right) + u(u,v)\left(2sin^2\frac{\theta}{2}\right) =$$

$$= v\cos\theta + [u,v]\sin\theta + u(u,v)(1-\cos\theta) =$$

$$= \big(v - u(u,v)\big)\cos\theta + [u,v]\sin\theta + u(u,v) =$$

It can lead us to the final formula, where vector **v** is presented in form of two components: parallel and perpendicular to the **u** axis.

$$= v_{\perp}\cos\theta + [u, v_{\perp}]\sin\theta + v_{||}$$

As a result, a formula for rotating a vector by an angle θ is obtained. The advantage of using quaternions is the ability to integrate into 3D space. With the help of quaternions, any rotations are described, which is much more difficult to implement using the methods of vector, scalar products, rotation matrices or the Euler method.

Approaching the conclusion, let's consider the action of quaternions using the example of my computer game project called Arkanoid.

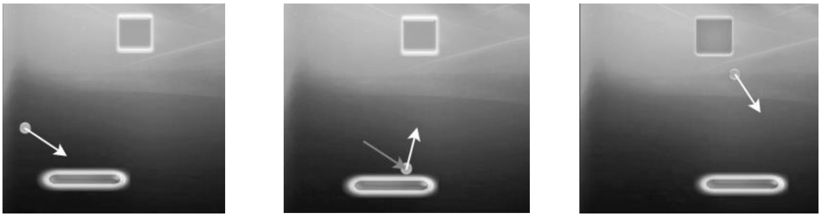
Although the game is implemented within a plane, the quaternion method is convenient because it does not need a voluminous description and does not slow down the gameplay. As you can see, quaternions are much more difficult to understand than they are to put into practice.

$q = w + (x, y, z) = \cos\left(\theta/_2\right) + \boldsymbol{u} \sin\left(\theta/_2\right)$ – a mathematical formula to apply looks much more difficult than integration in programming.

initialBall = Instantiate (ballPrefab, startingPosition, Quaternion.identity); [4]

The method itself is described in the c# libraries. The game is implemented using the Unity game engine.

Example:



a *b* *c*

Pict. 1. Bouncing ball, rotated by quaternion method
a) The ball flies towards the paddle

b) The ball bounces off the paddle

c) The ball bounced off the block

REFERENCES:

1. *Kantor I.L., Solodovnikov A.S.* Hypercomplex Numbers. M.: Nauka, 1973. 144 p.
2. *Arnold V. I.* Geometry of complex numbers, quaternions and spins. M.: MTsNMO, 2002. 40 p.
3. *Goldman R.* Understanding quaternions, Graphical Models. V. 73, Issue 2, 2011, p. 21-49
4. *Richter J.* CLR via C#. Programming with Microsoft .NET Framework version 4.0 in C#. M.: Piter, Sankt-Petersburg, 2013. p. 333-346.