Quaternion Sandbox LA Project

1st Shumskyi Dmytro
Lviv, Ukraine
shumskyi.pn@ucu.edu.ua

2nd Stehnii Sviatoslav
Lviv, Ukraine
stehnii.pn@ucu.edu.ua

3rd Yarish Oles
Lviv, Ukraine
yarish.pn@ucu.edu.ua

4th Hryniv Rostyslav
Mentor
Lviv, Ukraine

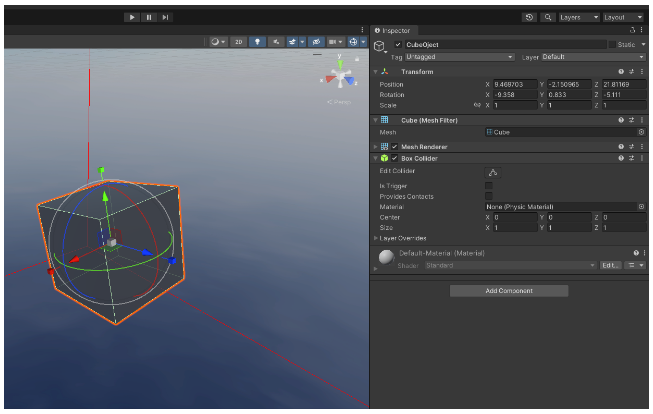
I. Introduction

In this project, our primary focus is to explore and compare different approaches of how rotations and orientations in 3D space can be represented mathematically. In particular, we would like to delve into the concept of quaternions/Euler angles and how 3D transformations are implemented under the hood. Also, for convenient visualization we are going to implement a 3D "sandbox" environment that would allow users to rotate objects and switch between different, so to speak, mathematical back-ends while doing so. The users should also be able to inspect the orientation of the object.

The significance of the quaternions that this project focuses on is that they are widely used in mechanics and form the backbone of many 3D editors.

II. Work review

Concerning the implementation tools, we have made a decision to implement the quaternion/Euler algebra as a library in pure JS and bring things to life with three.js, a JavaScript library known for its capabilities in rendering 3D graphics within web browsers. The above decisions are motivated by the fact that other considered tools and frameworks (for example, implementing the environment with C++ or OpenGL) would require us to focus more on code rather than logic. The results of our work can be examined on this repository.



CubeOject
Static
Tag Untagged   Layer Default

Transform
Position X 9.469703 Y -2.150965 Z 21.81169
Rotation X -9.358 Y 0.833 Z -5.111
Scale X 1 Y 1 Z 1

Cube (Mesh Filter)
Mesh Cube

Mesh Renderer
Box Collider
Edit Collider
Is Trigger
Provides Contacts
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1
Layer Overrides

Default-Material (Material)
Shader Standard

Fig. 1. An example of Unity engine that served as an inspiration for our sandbox and internally heavily relies on both quaternions and Euler angles.

III. Representation approaches outline

A. Axis-angle

This representation consists of 2 parts: the rotation axis vector (unit vector) and the angle of rotation. These are usually combined into a single vector. The Axis-angle Representation has a nice physical interpretation and is quite intuitive, however the calculations involving it may not be the most convenient. On the other hand, the quaternions discussed below represent the similar idea but offer better calculational properties.

B. Rotation matrix

Just like how 2D rotations can be represented as 2x2 matrices, 3D rotations can also be represented by 3x3 matrices. In their simplest form, rotation matrices denote a rotation about a single axis. However, a product of multiple matrices may encode a complete rotation in 3D.

C. Euler angles

The Euler angles representation, first introduced by Leonhard Euler [4], describes rotations in 3D space using three separate angles, typically denoted as α, β, and γ. These angles represent rotations around the axes of a fixed coordinate systems (either global or local to an object) in a specific order. Common sequences of Euler angles include X-Y-Z, Z-Y-X, and many others, depending on the area of application and established conventions, what may also complicate the calculations.

Main idea of this representation is that every 3D rotation can be achieved with the consecutive applications of 3 different rotations. 12 sequences can be used to obtain any rotation in the 3 dimensions. There are 2 groups of these sequences [4]:

Proper Euler angles: (z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y)

Tait–Bryan angles (x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z)

Moreover, rotations can be either intrinsic (the object rotates with respect to its local axes) or extrinsic (object rotates with respect to the global fixed axes), however, there is a rule of thumb: any intrinsic rotation is equal to the extrinsic one applied in reverse axes order but by the same order of angles.

It is important to note that Euler representation suffers from a significant problem called gimbal lock. Gimbal lock occurs when 2 axis get aligned in parallel so the system loses a degree of freedom, meaning the systems goes from 3 dimensions to 2 dimensions. There are only three ways to avoid this problem: changing the rotation sequence, adding an extra gimbal or resorting to quaternions.

D. *Quaternions*

Quaternions were first introduced by the Irish mathematician William Rowan Hamilton in 1843 (3) and later applied to mechanics in three-dimensional space. In their essence, quaternions serve as an extension to complex numbers and initially helped to define the quotient of two vectors in 3D space.

The standard form of a quaternion is defined as follows:

$$q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

where a, b, c, d are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ represent **basis vectors**, or **basis elements**. In some implementations the relative order of the vector (”imaginary”) part of a quaternion $b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ and its scalar part a may be reversed.

Also, the norm of a quaternion is given by:

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2} \tag{1}$$

while its inverse is denoted as follows:

$$q^{-1} = (a; -b; -c; -d) \tag{2}$$

As mentioned before, in practical context quaternions of **unit** length are used to represent rotations. Therefore, their components take on a bit different meaning as demonstrated below [2]:

$$q_\alpha = (cos(\frac{\alpha}{2}); \hat{x} * sin(\frac{\alpha}{2}); \hat{y} * sin(\frac{\alpha}{2}); \hat{z} * sin(\frac{\alpha}{2})) \tag{3}$$

where $\vec{v} = (\hat{x}; \hat{y}; \hat{z})$ represents the axis of rotation and α represents the angle of rotation, in a way similar to Axis-angle approach.

The most handy feature of quaternions is that the product of two **unit** quaternions gives the combination of two rotations represented by them and is usually more easy to calculate as compared to other approaches discussed above. Moreover, quaternions require less conventions to be established and are more convenient when it comes to transforming rotations between global and local spaces, which may be deeply nested in object hierarchy.

IV. Mathematical foundations

In this section we would like to provide the algebra governing the quaternions and briefly discuss the algebra behind other approaches.

A. *Rotation matrices*

The rotations about individual axes are given by the following matrices [1]:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \tag{4}$$

$$R_y(\alpha) = \begin{bmatrix} cos(\alpha) & 0 & sin(\alpha) \\ 0 & 1 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) \end{bmatrix} \tag{5}$$

$$R_z(\alpha) = \begin{bmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

As previously said, simple matrix multiplication $A(\alpha) * B = R$ applied to a column vector like $R * \vec{v}$ yields the vector $\vec{v}$ rotated by the combination of $A(\alpha)$ and $B(\beta)$, where both matrices are assumed to represent rotations about different axes. Nonetheless, as noted before, intrinsic rotation $R = A(\alpha) * B(\beta)$ is equivalent to extrinsic rotation $R = A(\beta) * B(\alpha)$. And the preserved order of matrices is not a mistake: for intrinsic rotations it is assumed that first A rotation is applied and then B, while the order is reversed for extrinsic ones.

B. *Euler angles*

The transition between rotation matrices and Euler angles is quite straightforward. In some sense, rotation matrices are simply a more convenient encoding of Euler angles. For example, the rotation about x-axis of vector $\vec{v} = (x; y; z)^T$ by α is given by:

$$\begin{aligned} x' &= x \\ y' &= cos(\alpha) * y - sin(\alpha) * z \\ z' &= sin(\alpha) * y + cos(\alpha) * z \end{aligned} \tag{7}$$

what, as can be observed, is effectively a decomposition of the corresponding rotation matrix (4).

C. *Quaternions*

The formula of quaternion multiplication stems from the fundamental formulae of **basis elements** multiplication:

$$i^2 = j^2 = k^2 = -1 \quad ijk = -1 \tag{8}$$

$$ij = k = -ji \quad jk = i = -kj \quad ki = j = -ik \tag{9}$$

Eventually, the product of two quaternions, named Hamilton product, is defined as follows [3]:

$$\begin{aligned} q_1 * q_2 = &(a_1 * a_2 - b_1 * b_2 - c_1 * c_2 - d_1 * d_2) \\ &+ \mathbf{i} * (a_1 * b_2 + b_1 * a_2 + c_1 * d_2 - d_1 * c_2) \\ &+ \mathbf{j} * (a_1 * c_2 - b_1 * d_2 + c_1 * a_2 + d_1 * b_2) \\ &+ \mathbf{k} * (a_1 * d_2 + b_1 * c_2 - c_1 * b_2 + d_1 * a_2) \end{aligned} \tag{10}$$

Quaternion multiplication is associative, but not commutative [2].

1) Vector rotation with quaternion: To rotate a vector by the given quaternion, first we need to represent this vector in a form of a quaternion. If $\vec{v} = (x, y, z)$ is the point to be rotated, then it is converted to a quaternion as follows:

$$p = (p_1, p_1, p_2, p_3) = (0, x, y, z) \tag{11}$$

Then we can actually perform a rotation, there are 2 options: passive and active rotation.

Active Rotation: In an active rotation, the object itself rotates in space relative to its own coordinate system. The object’s local vertices are physically rotated.

Passive Rotation: In a passive rotation, the coordinate system in which the object is described or observed is rotated. The object remains fixed in space, but its appearance changes as the viewpoint or reference frame changes.

To rotate a vector using quaternion q:
Active Rotation:

$$p_{new} = q^{-1} * p * q \tag{12}$$

Passive Rotation:

$$p_{new} = q * p * q^{-1} \tag{13}$$

The 3D coordinates of the rotated vector $\vec{v} = (x, y, z)$ is therefore just the vector component of p_{new}.

V. Comparison of point rotation using quaternion and matrix approaches

A. Comparison

Despite the significant difference in execution time, both approaches obviously produce the same final orientations for the objects being rotated. Quaternions provide a more compact representation of rotations compared to matrices. A quaternion only requires four scalar values (three for the rotation axis and one for the rotation angle) compared to the nine values needed to represent a 3x3 rotation matrix. One may think that matrices may be simply stored in Euler form to overcame the described issue, but such solution may later incur extra conversion cost should we ever need to combine two Euler angle rotations. Consequently, the compactness of quaternions reduces memory overhead and simplifies storage and transmission of rotation data, making quaternions more efficient in memory-constrained environments or when working with large datasets.

Quaternion operations, including multiplication and interpolation, can be more computationally efficient compared to matrix operations, especially for chaining multiple rotations or interpolating between orientations. Quaternion multiplication involves fewer operations than matrix multiplication, leading to faster computations, particularly in real-time applications where performance is critical.

These advantages make quaternions particularly well-suited for applications where precise and smooth rotations are essential, such as computer graphics, simulation, animation, robotics, and virtual reality.

VI. Perfromance test

In this segment, we have conducted a performance test to obtain the time required for executing a substantial number of rotations using both matrix and quaternion approaches.

In this comparison, the quaternion approach demonstrated a significantly faster execution time compared to the matrix approach. While the matrix approach took approximately 10.042 seconds, the quaternion approach completed the same task in just 5.539 seconds. This suggests that quaternions offer a more efficient solution for performing rotations in this scenario.

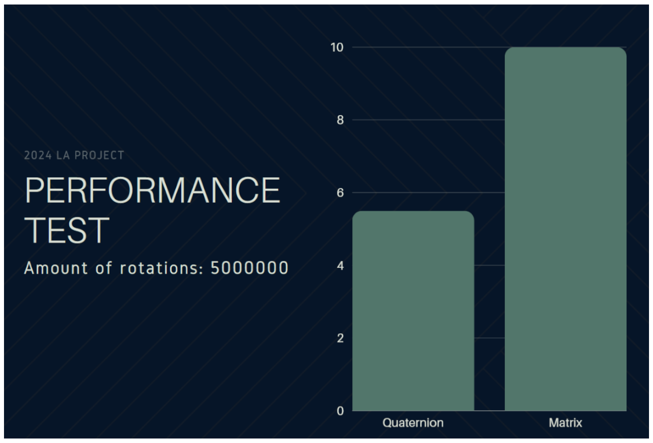
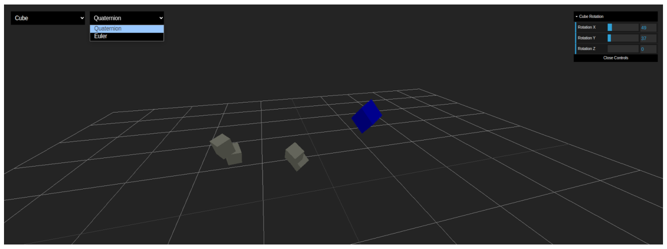


Fig. 2. Perfomance results

VII. Quaternion Sandbox

To illustrate the intricate mathematics behind 3D model rotation, we have developed a React website, leveraging the robust capabilities of the Three.js library. Our primary focus lies in demonstrating the rotation using both Euler angles and Quaternions, two fundamental mathematical representations in 3D graphics.



Rotation X 49
Rotation Y 37
Rotation Z 0

Fig. 3. Blue parent cube with its children. When a rotation is applied to the parent cube, all cubes in the scene rotate as a single solid object, preserving local positions and orientations

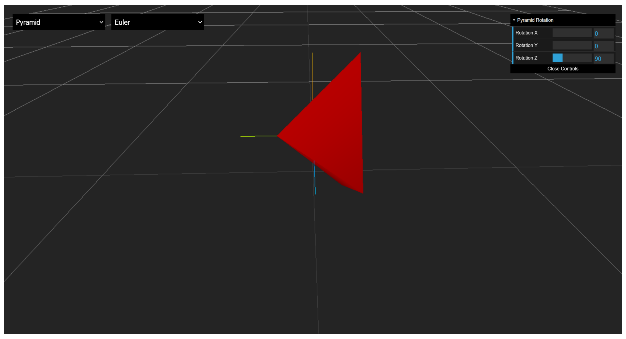
This application empowers users to perform 3D rotations on diverse objects, as exemplified by a cube and its associated child objects in the showcased example. Users are presented with a choice between two rotation methodologies: Euler angles and Quaternions.

The user is also able to rotate objects by swiping across the screen. The back-end of such functionality is based on quaternions. This is to show once again how effective quaternions are: with their help the implementation of this feature was of no difficulty.

The sandbox we constructed also nicely visualizes the phenomenon of gimbal lock. Gimbal lock manifests particularly when rotating around the x and y axes while the z-axis rotation is fixed at 90 degrees.

In such configurations, the axes of rotation become aligned, resulting in the loss of one degree of freedom. Consequently, rotations can exhibit unpredictable behavior, blurring the distinction between rotations along certain axes.

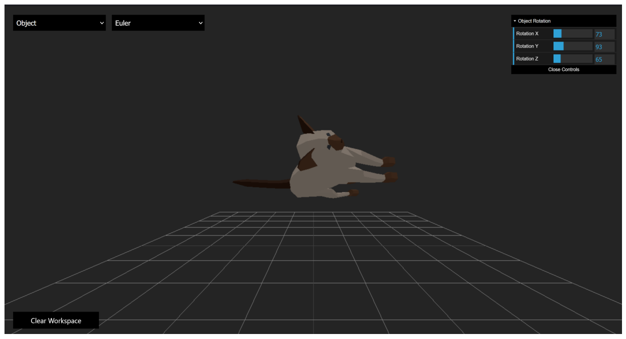
To further illustrate this, consider the accompanying figure depicting a red pyramid rendered within a 3D environment, complemented by axes helpers. This visualization aids in grasping the concept of gimbal lock. As the object rotates, the alignment of rotation axes becomes apparent, underscoring the phenomenon's impact on rotational freedom.



Rotation X 0
Rotation Y 0
Rotation Z 90

Fig. 4. Pyramid gimbal lock

Moreover, user can upload his own model, e.g. cute kitty. All the functionality can also be applied to the uploaded model.



Rotation X 73
Rotation Y 93
Rotation Z 65

Fig. 5. Cat model

References

[1] D. Rose. February, 2015. *"Rotations in Three-Dimensions: Euler Angles and Rotation Matrices. Part 1 - Main Paper"*
[2] D. Rose. May, 2015. *"Rotation Quaternions, and How to Use Them"*
[3] Wikipedia. Quaternion
[4] Wikipedia Euler angles
[5] 3Blue1Brown Home page