### Lab10 – Quaternion 3D Rotation

# Content

[Lab10 – Quaternion 3D Rotation 1](#_Toc103079178)

[1. Content 1](#_Toc103079179)

[2. Learning objectives 1](#_Toc103079180)

[2.1. Exam objectives 1](#_Toc103079181)

[2.2. Supportive objectives 1](#_Toc103079182)

[2.2.1. Self-support by two softwares: 1](#_Toc103079183)

[3. Exercises 2](#_Toc103079184)

[3.1. Basic exercises 2](#_Toc103079185)

[3.1.1. Pen and paper practice is key 3](#_Toc103079186)

[3.2. Bridging exercises 4](#_Toc103079187)

[3.3. Contextual practice 6](#_Toc103079188)

[4. Referecences 7](#_Toc103079189)

[4.1. Basics 7](#_Toc103079190)

[4.1.1. VecMathLexer 7](#_Toc103079191)

[4.2. Demos in programming 7](#_Toc103079192)

[4.2.1. Math engine 7](#_Toc103079193)

[4.2.2. Linear interpolation between orientation vectors 7](#_Toc103079194)

[4.2.3. Gimbal lock explained 7](#_Toc103079195)

# Learning objectives

## Exam objectives

By the end of this lab you should be able to (pen and paper):

* Perform 3D rotations conventionally through multiplication of matrices
* Perform 3D rotations through multiplication of quaternions
* Practice the previous mixed in the trigonometric and the vectorial shape of the quaternions

We advise you to **make your own summary of topics** which are new to you.

## Supportive objectives

### Self-support by two softwares:

More specifically related to the above you should in **GeoGebra Online** <https://geogebra.org/classic>

* Perform 3D rotations conventionally through multiplication of matrices
* Visualize the original vertex and its rotation image in the 3D space

More specifically related to the above you should via **VecMathLexer** (see LEHO)

* Compute 3D rotations through multiplication of quaternions
* Practice the previous in the trigonometric and the vectorial shape of the quaternions

# Exercises

Dependent of the lab session you may work individually or teamed (organized by the lab attendant). In either case make sure that throughout the course of this lab, you backup sufficiently your solution file on your local machine as

**1DAExx-0y-name1**(+name2+name3).GGB given **xx**=groupcode, **0y**=labindex

## Basic exercises

As a developer, awareness and mastering of the quaternion rotation is multiple advantageous:

1. However underneath manual exercising seems to show more calculation load on the quaternion side, the opposite is true when performed by machines. (Hint: imagine solving exercise 3 by matrices only)
2. Rotation by quaternions allows for interpolation (means: fast-forwarding directions between two given vector orientations by linearly combining both) which is technically impossible to run by conventional matrices.
3. Conventional matrix rotation suffers from what is known as ‘gimbal lock’ (means: loosing one degree of freedom in a 3D spatial motion, when a rotated standard axis gets parallel to another original standard axis).
4. Quaternion rotation around a given axis is unambiguous compared to the various Rx, Ry, Rz permutations which achieve the matrix equivalent for it.
5. And in general, having an alternative (quaternion instead of conventional) approach in store pays off, sooner than later.

### Pen and paper practice is key

For most of the underneath exercises do at least one pen and paper attempt in your own workbook, before checking your results machine wise.

**Exercise 1**

1. Write down the standard matrix for z-axis rotation over , and apply this 3D-rotation to the point P(1, 0, 0) yielding its rotation image point P’
2. Redo this rotation by quaternions:
3. Construct the required unit rotation quaternion *r*
4. Write down the pure quaternion *p* to carry the given point P
5. Apply the 3D-rotation by the so-called ‘sandwich’ multiplication

**Halfway check:** unit rotation quaternion *r* = [ 0.707 , (0, 0, 0.707)]

**Exercise 2**

1) Write down the standard matrix for a rotation around the x-axis over , apply this 3D-rotation to the point Q(2, 3, 1) yielding its rotation image Q’

1. Redo this rotation by quaternions:
2. Construct the required unit rotation quaternion *r*
3. Write down the pure quaternion *q* to carry the given point Q
4. Apply the 3D-rotation by the so-called ‘sandwich’ multiplication

**Halfway check:** unit rotation quaternion *r* = [ -0.5 , (0.866, 0, 0)]

**Exercise 3**

Calculate the image coordinates of the point H(1,1,0) upon rotation over a straight angle around the direction vector by applying quaternion arithmetic.

1. Construct the required unit rotation quaternion *r*
2. Write down the pure quaternion *h* to carry the given point H
3. Apply the 3D-rotation by the so-called ‘sandwich’ multiplication

**Halfway check:** unit rotation quaternion *r* = [ 0 , (0.408, 0.816, 0.408)]

## Bridging exercises

**Exercise 4**

To save time, use the **VecMathLexer** (see LEHO) tool to do the calculations. Focus on the conceptual understanding of the exercises.We hereby orient a camera with quaternion rotations.

**Given**:

1. A camera at location C(0.5, 1, 1) looking at a target placed in T(1, 4, 1.5)
2. The camera orientation vectors are defined by its
   1. Forward axis
   2. Upward axis
   3. Right hand side axis

**Question:**

Rotate the camera **only** over the **up axis** to make it face the target. (It is important to understand that we need two rotations over two different axises to be able to “look” at the target directly. However in this exercise we are only interested in the rotation around the up axis, as we will cover the second rotation later on).

Write down the angle of rotation, the quaternion and the resulting rotated forward vector and rotated right vector.

**Hint:** Sketch the situation in a right handed 3D frame!

**Halfway check:** unit rotation quaternion *r* = [ 0.763 , (0, 0, 0.646)]

**Exercise 5**

To save time, use the **VecMathLexer** (from LEHO) to do the calculations. Focus on the conceptual understanding of the exercises.We outline overhere the so-called `Look-at transformation’ by use of quaternions.

**Given**:

1. A spaceship that is modeled as following:
   * Forward axis
   * Upward axis:
2. A docking bay at the following location vector:

**Question:**

Calculate a quaternion rotation that will point the forward axis of this spaceship towards the docking bay. (As a check to see if the quaternion is correct, apply the quaternion with the sandwich method to the forward axis, the angle between the rotated forward axis and the docking bay should be zero. (Parallelism to be confirmed either by the cross or the dot product)).

Write down the axis of rotation, the angle of rotation, the quaternion and finally the rotated forward vector.

**Hint**: Sketch the situation in a right handed 3D frame!

Main difficulty is finding a rotation axis, for which you need to consider the cross product of two directions in this setup.

**Halfway check:** unit rotation quaternion *r* = [0.724 , (0, -0.676, 0.135)]

## Contextual practice

**Exercise 6**

To save time, use the **VecMathLexer** (from LEHO) to do the calculations. Focus on the conceptual understanding of the exercises.

Exercise 4 rotated the camera towards the target, but we are not yet looking directly at the target T. Determine now a *second rotation* around the rotated right axis of the camera which will point the camera perfectly at the target T.

**Questions**:

* Write down the second angle of rotation, the axis of rotation you use and the quaternion for this rotation.
* Finally write down the final quaternion that can rotate the camera in *one step.*
* Write down the final forward vector and check if this vector is parallel with the vector from the camera position C and the target T. (Parallelism to be confirmed either by the cross or the dot product).

**Exercise 7**

In exercise 5 we assumed that the forward vector will rotate in the correct direction towards the docking station. Is the assumption correct? Explain in your own words using the definitions of cross product and dot product why or why not.

**Yes**

Could we make the same assumption in exercise 4? Explain.

**No**

# Referecences

## Basics

### VecMathLexer

<https://leho-howest.instructure.com/courses/11650>

## Demos in programming

### Math engine

<https://www.boost.org/doc/libs/1_66_0/libs/math/doc/html/quaternions.html>

### Linear interpolation between orientation vectors

<http://sjbrown.co.uk/2002/05/01/representing-rotations-in-quaternion-arithmetic/>

### Gimbal lock explained

<https://www.youtube.com/watch?v=zc8b2Jo7mno>