ETH zürich

computer graphics laboratory

Prof. Markus Gross

Visual Computing
Exercise 8: Matrices and Quaternions

Hand-out: 22.11.2022

Goals

- Learning transformations used typically in a 3D graphics application
- Getting used to different representations of transformations
- Implementing an animated scene

Exercise 1) Theory

a) **Short Questions**

I. What are the three coordinate systems used to describe the scene?

II. State one advantage of using homogeneous coordinates.

III. Given two homogeneous points $\mathbf{p_1} = [4\ 3\ 2\ 1]^T$ and $\mathbf{p_2} = [1\ 2\ 3\ 4]^T$, compute the Euclidian displacement vector $\mathbf{d} \in \mathrm{R}^3$ from $\mathbf{p_1}$ to $\mathbf{p_2}$.

b) **Homogenous Transformations**

I. Decompose the matrix

$$\mathbf{A} = \begin{bmatrix} 0 & -1 & 0 & 47 \\ 1 & 0 & 0 & 11 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

into a linear transformation **L** and a translation **T**, so that **A** = **LT**. (Note: not **TL**!)

II. Please describe in words what the transformations **L** and **T** represent.

III. In which order are **L** and **T** processed, if applied to a point in 3D?

IV. Let **n** be the normal of a plane H in "Hessian normal form". Thus, for all points $\mathbf{p} \in \mathrm{H}$, we have $\mathbf{p}^T\mathbf{n} = 0$. The linear transformation A given above maps all $\mathbf{p} \in \mathrm{H}$ to points $\mathbf{p}'$ in a second plane H' ($\mathbf{p}' = \mathbf{Ap} \in \mathrm{H}'$). Compute a matrix **B** that maps **n** to the normal **n'** of the plane H' such that $\mathbf{n}' = \mathbf{Bn}$. (Hint: Compute **B** using the decomposition $\mathbf{A} = \mathbf{LT}$.)

c) **Quaternions**

I. Assemble the unit quaternion **q** = c + xi + yj + zk which describes a rotation of 60^o around the rotation axis $[3\ 0\ 4]^T$. Simplify the quaternion as much as possible.

II. Directly compute the quaternion of the inverse rotation from $\mathbf{q}$.

III. Which elementary quaternion operation have you used in II?

IV. Given the quaternion $\mathbf{r} = [c\ x\ y\ z]^{\mathrm{T}} = [0\ 1\ 0\ 0]^{\mathrm{T}}$. What rotation does $\mathbf{r}$ correspond to?

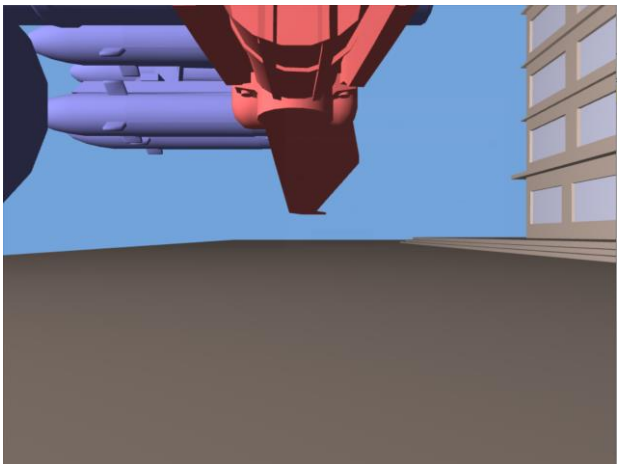
V. Please specify the rotation matrix that corresponds to $\mathbf{r}$.

Exercise 2) Practice

In this exercise you will implement a full animated 3D scene. The scene will consist of a helicopter with rotating rotors and a tower on which the helicopter is about to land.

a) **Getting used to the framework**

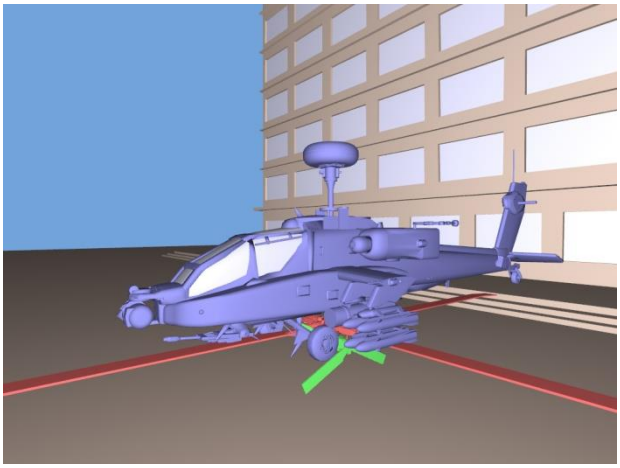
I. Make sure the framework compiles without errors.
II. Get used to the framework by browsing the classes and running the code. The first picture you will see will not be that interesting:



b) **Implementing the classes**

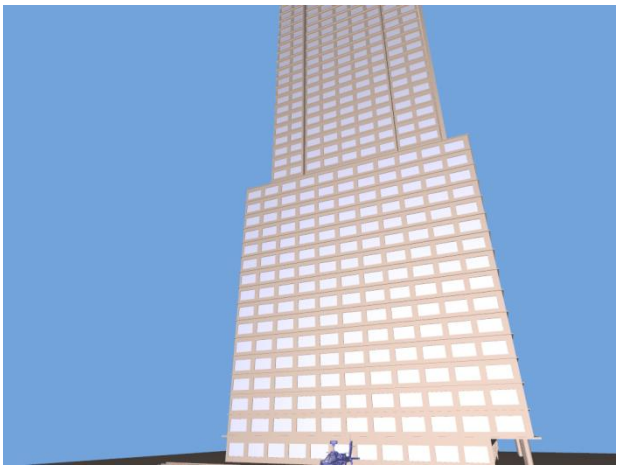
I. Step 1: In this step you will implement the missing functions of the Matrix class (Helpers/Matrix.cpp). The parts of the code that should be filled in are marked with the line //Step 1. These functions are used for various matrix computations for transformations in the scene.

After filling in the functions, go to exercises.h and comment in #define STEP 1 and comment out the other defines for STEP. Now you should be able to see the following rendering:



II. Step 2: You will now use the functions of the Matrix class you implemented to get the camera moving and rotating. The parts of the code that should be filled are marked with the line //Step 2.

After filling in the functions, go to exercises.h and comment in #define STEP 2 and comment out the other defines for STEP. Now you should be able to move and rotate the camera and see the scene from different views:



III. Step 3: Now it is time to place everything in the right place and animate the scene. Use the functions of the Matrix class again to place the helicopter on the top of the building (there is a helicopter landing field on the top) and place the rotors so that they are in the correct parts of the helicopter. Then, make the rotors rotate! The parts of the code that should be filled are marked with the line //Step 3.

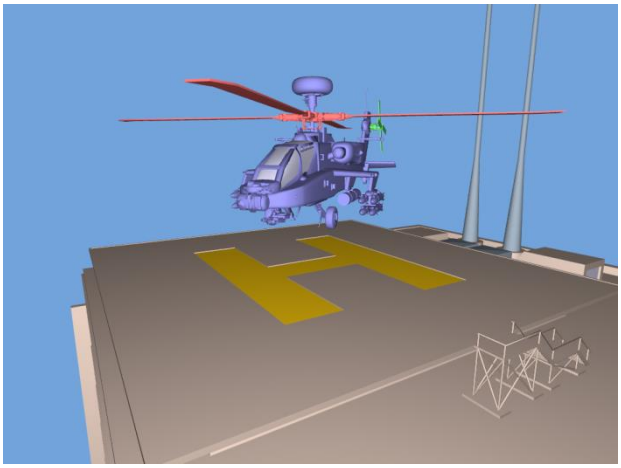
You may use the following parameters for this so that everything is placed and moving correctly:

Scale the helicopter so that it is 3 times bigger.
Rotate the helicopter by 320 degrees around the y axis.
Translate the helicopter by the vector $[40\ \ 158\ \ -5]$.

The rotors should also be translated with respect to the helicopter by the following: for rotor1: $[0.0\ \ 2.8\ \ 0]$, for rotor2: $[8.09\ \ 3.27\ \ 0.24]$.

Rotor1 should turn around the y axis and rotor2 around the z axis.

After filling in the functions, go to exercises.h and comment in #define STEP 3 and comment out the other defines for STEP. Now you should see the following animated scene with rotors rotating (you should climb up to see it!):



IV. Step 4: In this step you will implement missing functions of the Quaternion class (Helpers/Quaternion.h). The parts of the code that should be filled in are marked with the line //Step 4.

V. In this last step, you will implement a nicely moving camera view on a given path. The parts of the code that should be filled are marked with the line //Step 5.

After filling in the functions, go to exercises.h and comment in #define STEP 5 and comment out the other defines for STEP. Now you should be able to see the camera moving around the building.