Virtual Reality
Winter Term 2018/2019

Assignment Sheet 1

Throughout the lab classes, we will use *Python* to write Virtual Reality applications in our framework *avango-guacamole*. Python is a dynamically-typed (and thus interpreted), object-oriented, high-level programming language designed to be simple and effective for task automatization. If you need help with learning Python, good tutorials are available on the official Python homepage (http://docs.python.org/3/tutorial/) and on *Tutorialspoint* (http://www.tutorialspoint.com/python3/). In this assignment sheet, you will be asked to solve some basic tasks with the Python standard library (i.e. external packages should be avoided) in order to be prepared for the upcoming lab classes.

This assignment sheet should make you familiar with the programming language Python. **General knowledge of programming is a prerequisite for this course**. Group work in pairs of two is permitted. You are required to submit this assignment by **28 October 2018, 11:55 pm**. Please submit your solutions on Moodle.

This assignment sheet contains tasks worth **22 points** and will be weighted by **10%** for your total lab class grade.

Python and Matrix Basics

Only the `math` package provided by Python is permitted for the following tasks. Please write and submit your code in a **single *.py-file** with a suitable main function. We use **column-major** representations of transformation matrices.

Exercise 1.1 (6 points)

Write a Python class `TMatrix`, which is able to store a 4x4 transformation matrix. Write a constructor creating the identity matrix and a constructor in which all 16 values can be set individually. Furthermore, implement a member function `mult(self, other_matrix)`, which returns the product of the stored transformation matrix with `other_matrix`. Illustrate that your code is working by multiplying $A \cdot B$ and printing the result.

$$\mathrm{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad \mathrm{B} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 2 & 4 & 6 & 8 \\ 10 & 12 & 14 & 16 \end{bmatrix}$$

Exercise 1.2 (5 points)

Implement the **free** functions `make_trans_mat(x, y, z)`, `make_rot_mat(degree, axis)` and `make_scale_mat(sx, sy, sz)`, which create instances of `TMatrix` expressing translation, rotation and scale matrices, respectively. For the scope of this exercise, it is acceptable if `axis` is either `'x'`, `'y'` or `'z'`, but keep in mind that is not always the case in more complex settings. Illustrate that your code is working by printing `make_trans_mat(1,2,3)`, `make_rot_mat(45, 'x')`, `make_rot_mat(90, 'y')`, `make_rot_mat(120, 'z')` and `make_scale_mat(1,2,3)`. Have a look at the lecture *Scenegraph Basics* or at https://www.tutorialspoint.com/computer_graphics/3d_transformation.htm to see how to construct the matrices.

Exercise 1.3 (3 points)

Write a Python class `Vector4`, which is able to store a 3D point/vector in homogeneous coordinates. Implement a **free** function `euclidean_distance(point, point)`, which computes the euclidean distance between two 3D points. Illustrate that your code is working by printing `euclidean_distance(Vector4(2, 4, 6, 2), Vector4(0, 0, 0, 1))`.

Exercise 1.4 (2 points)

Extend the class `TMatrix` by a **member** function `mult_vec(self, vector)`, which returns a new vector representing the given vector transformed by the matrix. Illustrate that your code is working by printing $A \cdot v$, where $v = [1, 2, 3, 1]^T$.

Exercise 1.5 (2 points)

For the following tasks, we use $trans(x, y, z)$, $rot(degree, axis)$ and $scale(x, y, z)$ to denote translation, rotation and scale matrices, respectively.

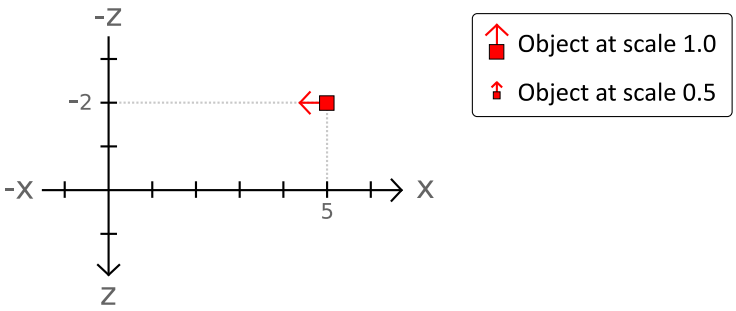
Representing rotations using angles around the cardinal axes (Euler angles) can suffer from disambiguities, which means that different combinations of Euler

angles can lead to the same visual rotation. Find a pair of angles α and β such that

$$rot(90, x) \cdot rot(\alpha, z) = rot(\beta, y) \cdot rot(90, x)$$

holds. Illustrate the correctness of your solution using your previously written code from Exercise 1.2.

Exercise 1.6 (4 points)



The above coordinate system shows a top shot of the scene onto the xz-plane of a scene. The transformation matrix $trans(5, 0, -2) \cdot rot(90, y)$ is visualized by a red square (position) and a red arrow (orientation). Scaling of the object is shown by the size of the square. Visualize the following matrices accordingly. Pay careful attention to the matrix orders, which play an important role for the resulting transformations. Rotations are performed mathematically positive (counter-clockwise).

- $rot(90, y) \cdot trans(5.0, 0.0, -2.0)$
- $trans(-4.0, 0.0, 2.0) \cdot scale(0.5)$
- $scale(0.5) \cdot trans(-4.0, 0.0, 2.0)$
- $trans(-2.0, 0.0, 4.0) \cdot scale(0.5) \cdot rot(180, y)$