UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Computer Graphics

VISUAL COMPUTING GROUP

Prof. Dr. F. Sadlo, D. Aumiller *Heidelberg, November 21, 2016*

Exercise Sheet 5

Assignment 5.1 Euler Angles and even more Transformations [6.5 Points]

Extract the angles ϕ, ψ, and θ for the given matrices. Care for the different sequences in which the rotations are applied:

$$R_{ZYZ} = \begin{pmatrix} 0.7071 & 0 & -0.7071 \\ 0 & -1 & -0 \\ -0.7071 & 0 & -0.7071 \end{pmatrix} \qquad R_{ZYX} = \begin{pmatrix} 0.7500 & -0.6495 & -0.1250 \\ 0.4330 & 0.6250 & -0.6495 \\ 0.5000 & 0.4330 & 0.7500 \end{pmatrix}$$

[*3 Points*]

Computing the inverse of a transformation matrix is time-consuming and may lead to numerical instabilities. To avoid recomputation of inverse matrices, OpenGL makes use of a very simple trick: By pushing the current transformation matrix onto a stack before adding another transformation, it can easily be restored afterwards by simply popping the matrix stack once again. For further reading, you may look at the documentation of `glPushMatrix()` and `glPopMatrix()`. Give pseudocode (e.g., `pushMatrix(), drawTriangle(), popMatrix()`, ...) for the following scenarios:

1. [2D space] The scenario from the last exercise sheet: Given a quadrangle with vertices $V1, ..., V4$, perform a 45° counter-clockwise rotation around $V1$. [*1 Point*]
2. [3D space] Imagine a simple model of the solar system, consisting of the Sun, Earth, and Moon. While the rotation of the Earth around the Sun is planar in the xz-plane, the Earth's axis has an inclination of 23.5°, whilst the Moon is assumed to orbit once again on the xz-plane. Do not only describe how to generate the system, but also perform a rotation of each planet about its axis by an arbitrary nonzero angle. [*2.5 Points*]

Assignment 5.2 Ray Tracing II [4.5 Points]

On the last exercise sheet, you already implemented the intersection calculation for the raytracer, which only respected ambient lighting so far. In this exercise, the Phong lighting model should be completed and extended by the diffuse and specular components. Implement the method `PointLight::ComputeDirectContribution()` in the file `src/Raytracer/Scenes/PointLight.cpp` to calculate the diffuse and specular contribution. You can use your own solution from the last exercise sheet to complete this task. If you were not able to complete this task, you may download the newly uploaded skeleton from the elearning platform.
Hint: If you use your own solution, make sure that the normals in `Sphere::GetIntersection` are calculated correctly.

Submission: November 28, 2016, 6:00 pm via Moodle