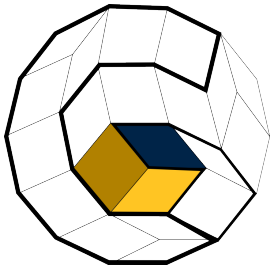
TU BRAUNSCHWEIG
Dr. Georgia Albuquerque
Institut für Computergraphik
Matthias Überheide

April 6, 2018



Real-time Computer Graphics, Summer 2018
Assignment 1

Present your solution to this exercise on Thursday, April 19, 2018.

The exercises are going to take place in the CIP pool, room G40 in Mühlenpfordstraße 23. Please make sure your solutions compile and run on the CIP pool computers. Note that you need a y-number, which can be obtained at the Gauß-IT-Zentrum, to use the computers. If for some reason you are not able to attend the exercise, you may send your solution to ecg@cg.cs.tu-bs.de instead.

This exercise and the corresponding version of the framework can be found on the lectures website (`http://www.cg.cs.tu-bs.de/teaching/lectures/ss18/ecg/`).

Theoretical Tasks

1.1 Dot product (5 Points)

Compute the angle α (in degrees) between the vectors $v_0 = \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix}$ and $v_1 = \begin{pmatrix} 2 \\ 4 \\ 1/2 \end{pmatrix}$.

1.2 Cross product (5 Points)

Given a vector $v_2 = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$, compute two vectors v_3 and v_4 that are perpendicular to each other *and* to the vector v_2. Write down all possible solutions for the vectors v_3 and v_4, given the additional constraint, that the vector v_3 lies in the xy-Plane, i.e. the z-component of this vector is 0.

1.3 Matrix multiplication (5 Points)

Solve the following equations and try to interpret the operation described by the matrix:

- $\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$
- $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$
- $\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

- $\begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \end{pmatrix}$

Practical Tasks

1.4 Setup the Project (5 Points)

In this task you setup the initial version of our framework and ensure that it works on your system. Go to the lecture's website and download the framework for this exercise. As you can see, there are already a lot of files and folder set up. We make use of CMake as a build system, which provides a quite comfortable interface to extend and adapt your project and generate a proper makefile for it. Once you have extracted the files, create a `build` folder and execute the command "`cmake ..`", which will tell the CMake build system to generate a makefile according to your project's configuration defined by the `CMakeLists.txt` in the root folder of the project. This file defines the type, name and structure of your project as well as all basic dependencies and library paths.
After this operation completed successfully, you can make your project (`make`) and generate the executable `ecg_ex01_a`. If everything went without errors you should now be able to start `./ecg_ex01_a` the application and see a colored triangle.

1.5 glm Math (5 Points)

Modern OpenGL does no longer provide methods for linear algebra. glm is a library which aims to provide linear algebra types and functions that are compatible with OpenGL. Open and implement the missing functions in `math3d/Math3d.h` to allow all math related tests in `ecg_ex01_b.cpp` to pass. See `http://glm.g-truc.net/` for more information about *glm*.
Extend `ecg_ex01_b.cpp` to solve the following problem:
Given three points, A, B, and C, forming a triangle (counter clockwise), compute the normal vector defined by that triangle and the angle between the triangle's normal and a vector V.

$$A = \begin{pmatrix} 4 \\ 1 \\ 0 \end{pmatrix}, B = \begin{pmatrix} -1 \\ 3 \\ 2 \end{pmatrix}, C = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}, \text{ and } V = \begin{pmatrix} 1 \\ 1 \\ -3 \end{pmatrix}$$

Output your solution in the console as text.

1.6 Shader Loading (5 Points)

Complete `gl/oglshader/SimpleFileShader.cpp` to load a vertex and fragment shader from given filenames. Test your implementation with `ecg_ex01_c`. After everything is implemented you should see the same result as in `ecg_ex01_a`.