Prof. G. Zachmann
P. Dittmann
H. Meißenhelter

Universität Bremen
Fachbereich 3
Arbeitsgruppe CGVR
30. November 2020

Wintersemester 2020/21

Übungen zu Computergraphik - Blatt 4

Abgabe am 07.12.2020, 23:59 Uhr

Für diese Übungen verwenden Sie das Framework „ShaderIntro“. In dem Projekt gibt es für jede Aufgabe ein eigenes cmake-Target bzw. Executable: Tetrahedron, Wave, Model.

Aufgabe 1 (Vertex Shader - Tetrahedron, *3 Punkte*)

Ihnen liegt eine binäre Datei Tetrahedron vor. Ein Kollege hat die Vertices und Color für ein Tetraeder in ein Array hartkodiert. Die Farben hat er im HSV-Format festgelegt, da OpenGL nur RGB unterstüzt, werden die Werte falsch interpretiert. Der Kollege ist nun im Urlaub und Ihnen fällt ein, dass die Shader zur Laufzeit von der GPU kompiliert werden. In einem Buch finden Sie einen Algorithmus, um vom HSV-Farbraum in den RGB-Farbraum zu konvertieren. Implementieren Sie diesen Algorithmus als eine Funktion in dem Vertex Shader „tetrahedron_vertex.glsl“:

When $0 \le H < 360$, $0 \le S \le 1$ and $0 \le V \le 1$:

$$C = V \times S$$

$$X = C \times (1 - |(H / 60°) \bmod 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \le H < 60° \\ (X, C, 0) & , 60° \le H < 120° \\ (0, C, X) & , 120° \le H < 180° \\ (0, X, C) & , 180° \le H < 240° \\ (X, 0, C) & , 240° \le H < 300° \\ (C, 0, X) & , 300° \le H < 360° \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

Aufgabe 2 (Fragment Shader - Wave,Water,Ocean, *2 Punkte*)

In dieser Aufgabe wird ein Wasser/Wellen-Bild geladen und auf einer horizontalen Ebene gesetzt. Ihre Aufgabe ist es nun diese Bild zu animieren. Dazu müssen Sie nur den Fragment Shader „wave_fragment.glsl“ bearbeiten. Dort finden zwei Effekt-Funktionen, wo Sie einen Effekt implementieren sollen. Der erste Effekt soll das Bild zu Ihnen hin bewegen. Beim zweiten Effekt sollen Sie die sin- und cos-Funktion verwenden, um die Animation etwas zufälliger wirken zu lassen (Wasserrichtung hierbei egal). Im Code sind die Attribute kommentiert. Verwenden Sie die Uniformvariable *time* zum Animieren. Achten Sie darauf, dass ihre Animation nicht zu schnell ist bzw. realistisch aussieht.

Aufgabe 3 (C++ und OpenGL - Model, *5 Punkte*)

In dieser Übung laden Sie ein 3D-Modell mit einem Bild und rendern es. Der Code, um das Bild zu laden, ist bereits vorhanden. Auch ist eine Funktion *loadOBJ* vorhanden (siehe Headerfile: objloader.hpp), um das Mesh zu laden. Sie editieren hierbei nur die „ModelWindow.[h|cpp]“ Dateien.

Legen sie ein Struct VertexData an, dass die Vertices,Normals und UVs(Bild/Textur-Koordinaten) speichert. Schauen Sie welche Attribute in den entsprechenden Shadern sind und bereiten Sie OpenGL auf das Mesh (VertexData) vor mit Hilfe der in der Vorlesung vorgestellten Funktionen vor: glGenBuffers, glGenVertexArrays, ...

Tipp:
Schauen Sie in WaveWindow.cpp