Übungen zu Computergraphik I

– Übungsblatt 3 –

Lehrstuhl für Computergraphik und Multimediasysteme

Rene Winchenbach, Jan Mußmann

Abgabe:	Bis spätestens 5. November 2019, 10 Uhr
Besprechung:	Dienstag 12. November und Mittwoch 13. November 2019
Gesamtpunktzahl nach Übungsblatt 3:	19 von 65

Hinweis: Die Programmieraufgaben müssen per E-Mail **an Jan Mußmann** eingereicht werden. Geben Sie dabei bitte immer Ihren **Namen**, Ihre **Matrikelnummer**, sowie Ihre **Übungsgruppe (Di. / Mi.)** an. Geben Sie nur die von Ihnen **geänderten Dateien** ab (`GLWidget.cpp`, `vs.glsl` und `fs.glsl`).

Aufgabe 1 Skalarprodukt (1 Punkt)

Gegeben sei das gleichschenklige Dreieck aus Abbildung 1, das durch die beiden Vektoren $\hat{\mathbf{u}}$ und $\hat{\mathbf{v}}$ aufgespannt wird. Die Vektoren $\hat{\mathbf{u}}$ und $\hat{\mathbf{v}}$ haben beide die Länge 1 und schließen den Winkel α ein. Da $\hat{\mathbf{u}}$ die Länge 1 hat, ist die Länge der orthogonalen Projektion von $\hat{\mathbf{v}}$ auf $\hat{\mathbf{u}}$ genau $\cos\alpha$. Die Länge des durch die Projektion aufgespannten Vektors, der in Abbildung 1 gestrichelt dargestellt ist, entspricht genau $\sin\alpha$.

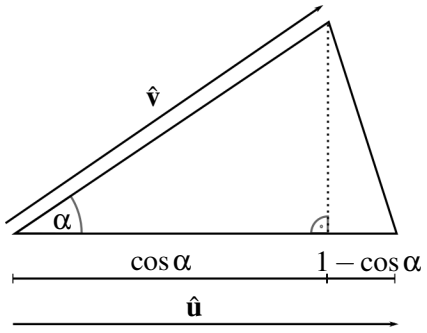

Abbildung 1: *Ein gleichschenkliges Dreieck mit Winkel α.*

Für das linke Teildreieck liefert der Satz von Pythagoras die Gleichung

$$\cos^2\alpha + \sin^2\alpha = \|\hat{\mathbf{v}}\|^2 .$$

1.1 Leiten Sie eine entsprechende Formel für das rechte Teildreieck her.

1.2 Vergewissern Sie sich durch Umformen Ihrer Gleichung, dass für das obige Dreieck

$$\hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = \|\hat{\mathbf{u}}\| \, \|\hat{\mathbf{v}}\| \cos\alpha$$

gilt.

Aufgabe 2 OpenGL Theorie (2 Punkte)

2.1 Erklären Sie was ein **Vertex Array Object** (VAO) ist. Erklären Sie insbesondere den Zusammenhang bzw. Unterschied zwischen VAO und **Buffer** (geneuaer: **Vertex Buffer Object** (VBO)).

2.2 Ein **Shader-Programm** besteht aus einem **Vertex-Shader** und einem **Fragment-Shader**. Erklären Sie in welcher Reihenfolge diese Shader aufgerufen werden und auf welchen Daten (Punkte / Pixel / ...) diese Shader jeweils arbeiten.

Aufgabe 3 OpenGL Praxis (4 Punkte) Nehmen Sie als Ausgangsbasis das auf unserer Website bereitgestellte Programmgerüst `ueb04.zip`. Um das Projekt zu kompilieren, folgen Sie der Anleitung auf der CG1 Übungsseite.

3.1 Die Methode `initializeBuffers()` ist im aktuellen Programmgerüst unvollständig, uzw. wird lediglich das **Vertex Array Object** (VAO) erstellt (durch `glGenVertexArrays()`).
Erweitern Sie die Methode `initializeBuffers()` so dass sie, wie in der Vorlesung (Folie 3.20) besprochen, die Positionsdaten aus `geometry` auf die Grafikkarte überträgt.
Hinweis: Wenn Sie alles korrekt gemacht haben, sollten Sie ein fliegendes Dreieck auf Ihrem Monitor sehen.

3.2 Der Parameter `color` im Vertex-Shader (Datei `vs.glsl`) wird aktuell nicht ausgewertet. Stattdessen wird im Fragment-Shader (Datei `fs.glsl`) die Farbe auf `(0, 0, 0, 1)` gesetzt.
Ändern Sie sowohl den Vertex-Shader als auch den Fragment-Shader so dass die Farbe auf `color` gesetzt wird (d.h. geben Sie den Wert von `color` vom Vertex-Shader weiter an den Fragment-Shader und setzen sie den Wert von `fcolor` entsprechend).

3.3 Definieren Sie das in der Zeichnung dargestellte sternenförmige Objekt mit der Längeneinheit d, die der Funktion `setupStar()` als Parameter übergeben wird (vgl. Abb. 2).

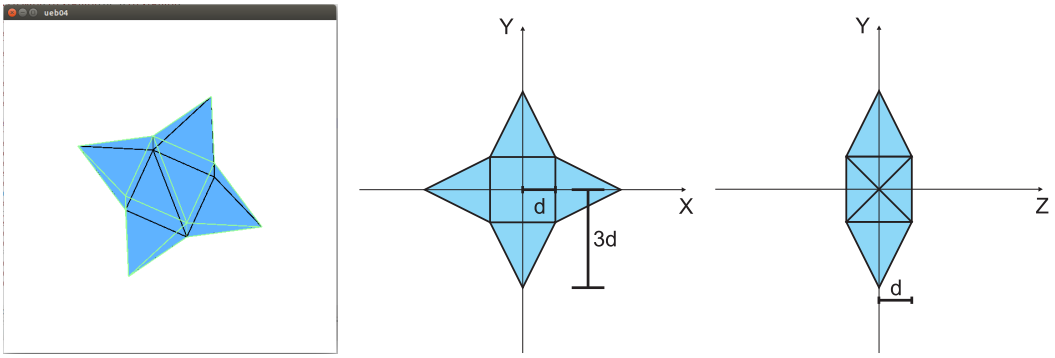

Abbildung 2: *Sternenförmiges Objekt*