Ludwig-Maximilians-Universität München
LFE Medieninformatik
Prof. Dr. Andreas Butz
Henri Palleis

Übungsblatt 8: Shading

Abgabe:

Dieses Übungsblatt ist einzeln zu lösen. Die Lösung ist bis **Donnerstag**, den **19. Juni 2014, 12:00 Uhr s.t.** über UniWorx (https://uniworx.ifi.lmu.de/) abzugeben.

Aufgabe 1: Toon Shading

Beim Toon-Shading handelt es sich um eine einfache Methode, mit der man Objekte nicht-fotorealistisch rendern kann. Dazu wird im einfachsten Fall eine Farbe in wenigen unterschiedlichen Tönen verwendet. Welcher Farbton verwendet wird um ein Pixel zu färben, hängt dabei vom Winkel zwischen der Richtung des Lichts und der Normalen der Oberfläche ab: Je kleiner der Winkel, desto heller der Farbton.

Implementieren Sie entsprechende Vertex- und Fragment-Shader. Gehen Sie wie folgt vor:

Vertex-Shader:

- Berechnen Sie die Position des Knoten in Weltkoordinaten
- Definieren Sie ein virtuelles Punktlicht (d.h. eine Position in Weltkoordinaten)
- Berechnen Sie den Vektor vom Knoten zum virtuellen Licht (Lichtrichtung)
- Normalisieren sie den Normalvektor und den Vektor für die Lichtrichtung (normalize()-Funktion von GLSL)
- Berechnen Sie als Wert für die Intensität den Winkel zwischen Normale und Lichtrichtung
- (sie können die GLSL-Funktion dot(), verwenden, der zwei normalisierte Vektoren als
- Parameter übergeben werden)
- Sorgen Sie dafür, dass die Variable für die Intensität im Fragment-Shader verfügbar ist

Fragment-Shader

- Mit welcher Farbe der Pixel eingefärbt wird, soll von der Intensität abhängen
- Sie könnten z.B. folgende vier Farbtöne verwenden: rgba(1.0,0.5,0.5,1.0), rgba(0.6,0.3,0.3,1.0), rgba(0.4,0.2,0.2,1.0), rgba(0.2,0.1,0.1,1.0)

Abbildung 1Toon-Shading

Aufgabe 2: Phong Shading

Ziel dieser Aufgabe ist es, ein einfaches Phong-Shading zu implementieren. Gehen Sie von einem virtuellen Punktlicht bei Position (50.0, 100.0, -100.0, 1.0) aus. Licht und Material sind dabei folgendermaßen definiert:

Licht: Ambient = (0.2,0.2,0.2,1.0), diffus = (1.0,1.0,1.0,1.0) , spekular = (1.0,1.0,1.0,1.0)

Material: Ambient = (0.0,1.0,1.0,1.0), diffus = (0.8,1.0,0.3,1.0) , spekular = (1.0,1.0,1.0,1.0), shininess = 80.0

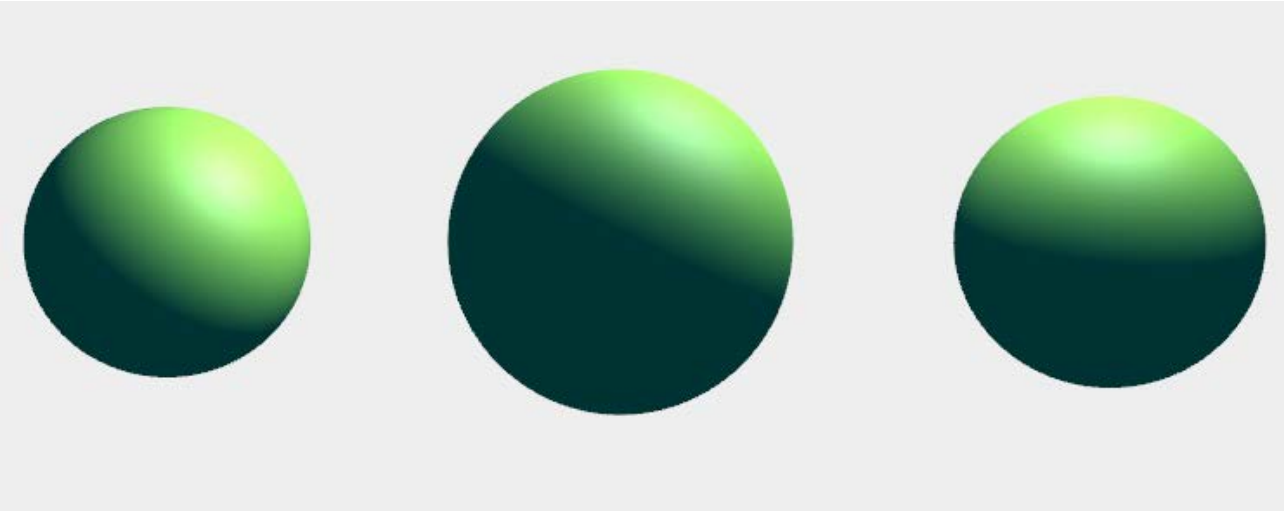

Abbildung 2 **Phong Shading mit GLSL**

Aufgabe 3: Animation mit dem Vertex-Shader

Auf der Vorlesungswebsite finden Sie die Datei plane.html. Ziel dieser Aufgabe ist es, eine Wassersimulation mit Hilfe einer Noise-Funktion (vgl. Übungblatt 4) zu erzeugen. Hierzu sollen sie einen Vertex-Shader programmieren, der die Höhe der einzelnen Knoten fortlaufend neu berechnet. Unter https://github.com/ashima/webgl-noise/blob/master/src/noise3D.glsl finden sie eine GLSL-Implementierung des Noise-Algorithmus von Perlin.

Sie finden in plane.html bereits eine Funktion mit dem Namen *animate()*, die dafür sorgt, dass die Szene mit bis zu 60 fps animiert wird. Dort können sie zusätzlich Werte über die Zeit verändern, die sie als *uniforms* oder *attributes* auch an die Shader übergeben können.

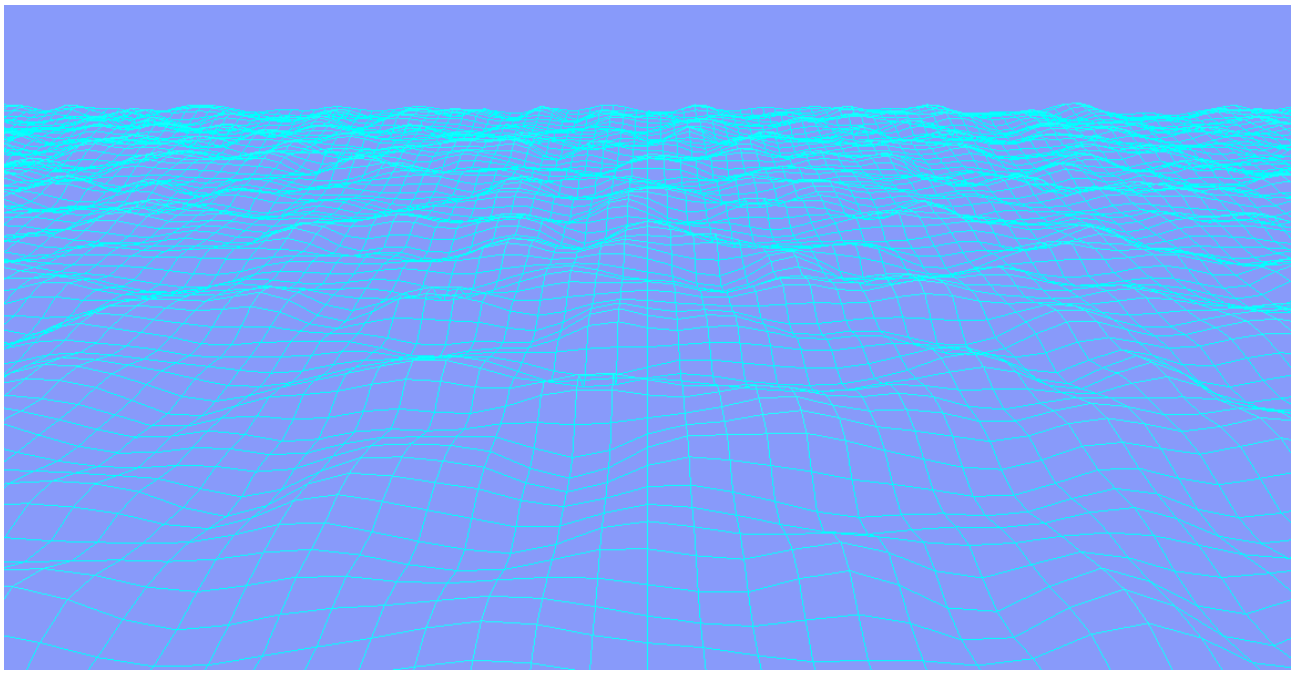

Abbildung 3 Screenshot der Wasser-Simulation

Viel Erfolg.