Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Axel Schaffland, M.Sc.

UNIVERSITÄT OSNABRÜCK

Computergrafik (SS 2018)

Übung 7

fällig Montag **4. Juni**, 14:20

- Geben Sie bei jeder Abgabe alle Ihre Matrikelnummern auf jedem Blatt an.
- Verspätet eingereichte Abgaben können nicht gewertet werden.

Aufgabe 1 (Theorie: Texturen)

(a) (1P) Eine Textur T mit 2×2 Pixeln ist gegeben. Die Textur enthält keine RGB-Farben, sondern nur Grauwerte, d.h. jedes Texel hat einen Wert zwischen 0.0 (schwarz) und 255 (weiß). Die Werte der vier Pixel sind: T(0;0) = 15, T(1;0) = 140, T(0;1) = 0, T(1;1) = 60. Berechnen Sie, welcher Wert sich an der Stelle $(0.2; 0.6)$ im Interpolationsmodus `NEAREST` (ohne Mipmapping) ergibt.

(b) (3P) Berechnen Sie für die gleiche Textur, welcher Wert sich an der Stelle $(0.2; 0.6)$ im Interpolationsmodus `LINEAR`, d.h. mittels bilinearer Interpolation, (ohne Mipmapping) ergibt.

(c) (3P) Gegeben ist folgende 4×4 Textur (mit Grauwerten). Berechnen Sie alle weiteren Mipmap-Level.

10	20	0	100
110	180	220	60
80	200	240	40
0	20	140	100

(d) (2P) Das Dreieck mit den Eckpunkten $(2, 2, 4)$, $(6, 2, 4)$, $(4, 3, 5)$ hat die Texturkoordinaten $(0, 0)$, $(1, 0)$, $(0.5, 1.0)$. An welcher Stelle (d.h. an welchen Texturkoordinaten) muss die Textur ausgewertet werden, um die Farbe für den Punkt $(3, 2, 4)$ im Dreieck herauszufinden?

(e) (1P) Sei W eine quadratische Textur, die (unkomprimiert) n Bytes Speicher belegt. Wieviel zusätzlicher Speicher ist ungefähr nötig, um alle Mipmap-Level von W vorberechnet bereitzuhalten?

Aufgabe 2 (Praxis: Texturen)

Ergänzen Sie Vertex Shader und Fragment Shader derart, dass eine Textur auf dem Würfel angezeigt wird.

(3P) Dem Vertex Shader werden die Texturkoordinaten vom Hauptprogramm als ein 2D-Vektor-Attribut mit dem Namen `aTextureCord` übergeben. Nehmen Sie diese im Vertex Shader entgegen und reichen Sie diese in geeigneter Weise an den Fragment Shader weiter.

(4P) Im Fragment Shader soll die entsprechende Texturfarbe abgerufen werden. Dazu können Sie die Funktion `texture2D` verwenden. Schauen Sie im übrigen Code nach, um herauszufinden unter welchem Namen die Textur, die vom Hauptprogramm bereits geladen und an WebGL übergeben wird, im Shader verfügbar ist.

(3P) Setzen Sie dann die (bereits vorberechneten) Terme des Phong Beleuchtungsmodells zusammen und lassen Sie dabei die abgerufene Texturfarbe so einfließen, wie dies bei der vorigen Übung die Dreiecksfarbe tat. Beachten Sie dabei, dass `texture2D` RGBA-Werte liefert, das Phong-Modell jedoch nur den RGB-Teil nutzt. Setzen Sie dann die Fragmentfarbe entsprechend (wobei wieder RGBA-Werte benötigt werden).

Aufgabe 3 (Bonus: Normal Mapping)

Fügen Sie zusätzlich zur Farbtextur eine weitere Textur mit Normaleninformationen hinzu. Sie können z.B. die unten dargestellten nutzen. Nutzen Sie diese Informationen bei der Berechnung der Beleuchtung im Phong Shading. Beachten Sie, dass Sie verschieden ausgerichtete Normalen für die sechs Seiten des Würfels benötigen. Daher müssen entweder sechs verschiedene Texturen bzw. Texturkoordinatenbereiche verwendet werden, oder Sie geben für jede Würfelseite eine Transformationsmatrix als Attribut mit, um so die Normaleninformation aus einer einheitlichen Normal Map entsprechend transformieren zu können. Beachten Sie dann noch, dass (genau wie bei den Vertex- oder Face-Normalen) die NormalMatrix angewendet werden muss.