Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Steffen Hinderink

UNIVERSITÄT OSNABRÜCK

Computergrafik (SS 2020)

Blatt 7

fällig Sonntag **14. Juni**, 24:00

Verspätet eingereichte Abgaben können nicht berücksichtigt 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 (schwarz) und 255 (weiß). Die Werte der vier Pixel sind: $T(0,0) = 230$, $T(1,0) = 70$, $T(0,1) = 30$, $T(1,1) = 170$. Berechnen Sie, welcher Wert sich an der Stelle $(0.7, 0.1)$ im Interpolationsmodus `NEAREST` (ohne Mipmapping) ergibt.

(b) (2P) Berechnen Sie für die gleiche Textur, welcher Wert sich an der Stelle $(0.7, 0.1)$ im Interpolationsmodus `LINEAR`, d.h. mittels bilinearer Interpolation, (ohne Mipmapping) ergibt. Geben Sie Ihren Rechenweg an.

(c) (2P) Gegeben ist folgende 4×4 Textur (mit Grauwerten). Berechnen Sie alle weiteren Mipmap-Level.

70	210	20	0
0	100	80	200
20	110	140	100
10	180	240	40

(d) (2P) Das Dreieck mit den Eckpunkten $(3,2,1)^\mathsf{T}$, $(7,2,1)^\mathsf{T}$ und $(4,4,2)^\mathsf{T}$ hat die Texturkoordinaten $(0,0)$, $(1,0.5)$ und $(0,1)$. An welcher Stelle (d.h. an welchen Texturkoordinaten) muss die Textur ausgewertet werden, um die Farbe für den Punkt $(6,2,1)^\mathsf{T}$ im Dreieck herauszufinden?

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

(f) (2P) Für das Rendern einer Umgebung wird Spherical Environment Mapping eingesetzt. Berechnen Sie die Texturkoordinaten, an denen die Farbe der Umgebung in Richtung $(1,0,0)^\mathsf{T}$ gespeichert ist. Die Richtung ist dabei bereits in Environment Koordinaten gegeben.

Aufgabe 2 (Praxis: Texturen)

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

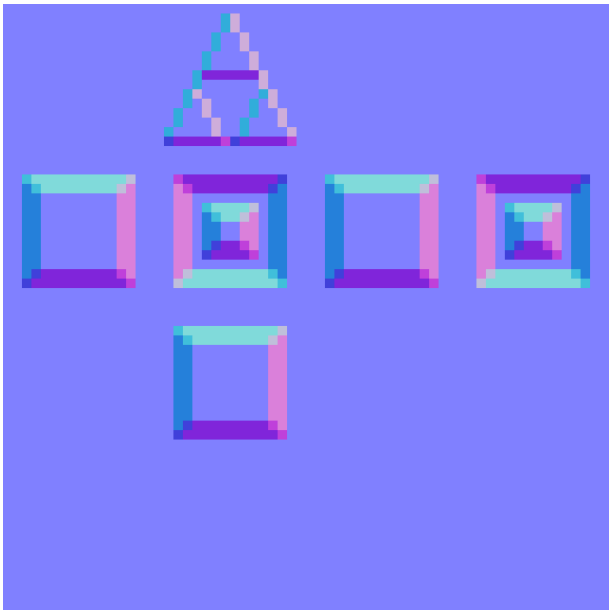
(a) (3P) Dem Vertex Shader werden die Texturkoordinaten vom Hauptprogramm als ein 2D-Vektor-Attribut mit dem Namen `aTextureCoordinate` übergeben. Nehmen Sie diese im Vertex Shader entgegen und reichen Sie diese in geeigneter Weise an den Fragment Shader weiter.

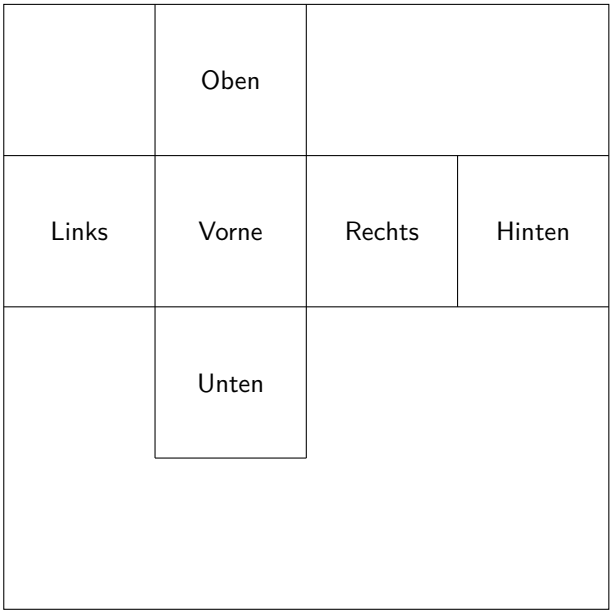
(b) (4P) Im Fragment Shader soll die entsprechende Texturfarbe abgerufen werden. Dazu können Sie die Funktion `texture2D` verwenden. Schauen Sie im übrigen Code nach, um herauszufinden unter welchem Namen die Textur, die vom Hauptprogramm bereits geladen und an WebGL übergeben wird, im Shader verfügbar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

(c) (3P) Lassen Sie die Texturfarbe auf die gleiche Art und Weise in das Phong Beleuchtungsmodell einfließen, wie dies in den vorherigen Übungsblättern die Dreiecksfarbe tat. Beachten Sie dabei, dass `texture2D` RGBA-Werte liefert, das Phong-Modell jedoch nur den RGB-Teil nutzt.

Aufgabe 3 (Bonus: Normal Mapping)

Fügen Sie zusätzlich zur Farbtextur eine weitere Textur mit Normaleninformationen hinzu. Sie können z.B. die links dargestellte nutzen, die zur Farbtextur passt. Beachten Sie, dass die Textur als das rechts dargestellte Würfelnetz gegeben ist.





Nutzen Sie diese Informationen bei der Beleuchtungsberechnung. Beachten Sie, dass Sie verschieden ausgerichtete Normalen für die sechs Seiten des Würfels benötigen. Geben Sie deshalb jeder Würfelseite (durch ihre Vertices) einen Tangentenvektor mit. Die Tangenten werden wie die Normalen des Würfels mit der Normal-Matrix transformiert. Aus einer Tangenten und einer Normalen lässt sich die Bitangente berechnen, die senkrecht zu diesen steht. Tangente, Bitangente und Normale bilden zusammen die TBN-Matrix, mit der die Normalen aus der NormalMap in die Weltkoordinaten transformiert werden können. Beachten Sie weiterhin, dass der positive Wertebereich der Vektoren aus der NormalMap auf das Intervall $[-1, 1]$ abgebildet werden muss. So meint zum Beispiel die Farbe $(0.5, 0.5, 1)$ (dominante Farbe in der NormalMap) die Normale $(0, 0, 1)$.