Universität Leipzig
Institut für Informatik
Abteilung Bild- und Signalverarbeitung

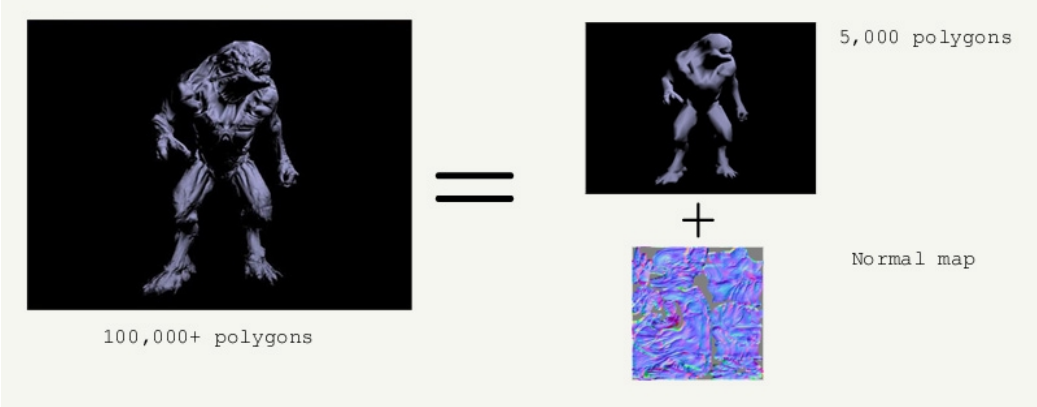
Computergrafik-Praktikum
Sommersemester 2013
Betreuer: Steven Schlegel (schlegel@informatik.uni-leipzig.de)

Aufgabenblatt 5 – Abgabe in der Woche vom 8. Juli

# Normal Mapping und Toon Shader

## Aufgabe 1: Normal Mapping

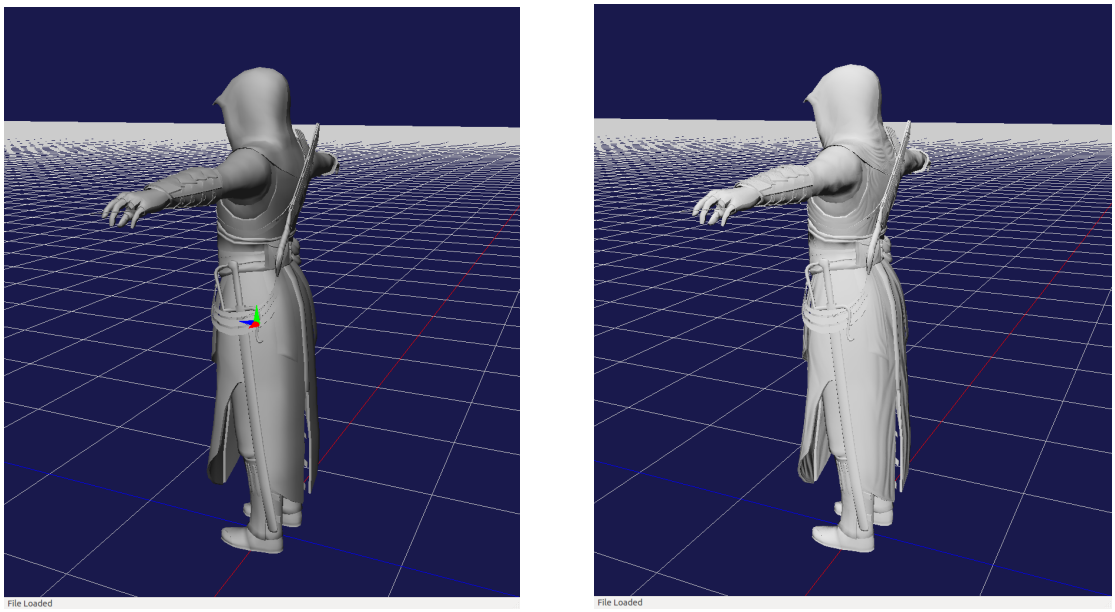
Ein beliebtes Mittel um den Detailgrad von 3D-Objekten zu erhöhen ist das Normal-Mapping. Im Normalfall wird das 3D-Objekt mit sehr vielen Polygonen (und dadurch vielen Details) erstellt. Die Normalen dieses High-Poly-Models werden in einer Textur – der Normalmap - gespeichert. Dann wird (durch ein spezielles Verfahren) die Anzahl der Polygone des Modells erheblich reduziert. Wenn dieses und andere detaillierte 3D-Modelle ansonsten z.B. in ein Spiel einfließen sollen, würde es zu einem starken Einbruch der Framerate des Spiels kommen. Deswegen greift man eben auf das Low-Poly-Modell zurück, um das Spiel z.B. auch auf schwächeren Rechnern lauffähig zu machen
Wenn man aber jetzt bei der Auswertung des Beleuchtungsmodells eben nicht die Eckpunktnormalen dieses Low-Poly-Modells berücksichtigt, sondern die (interpolierten) Normalen aus der vorher erstellten Normalmap, kann man mit sehr geringen Kosten das Modell detailreicher erscheinen lassen, als es eigentlich ist.




Die Aufgabe besteht darin, den Phong Shader aus der letzten Aufgabe um die Funktionalität des Normalmappings zu erweitern. Das Model des Altair besitzt eine Normalmap. Sie können sich auch nach weiteren Modellen mit Normalmaps umsehen. Belesen Sie sich selbständig, wie Normalmapping mittels GLSL im Detail funktioniert.

**Hinweise:**
1. Die Normalmap wird dem Shader auf Textur-Unit 1 übergeben
2. Die Farbtextur kann mit „*uniform sampler2D basemap;*“ im Shader ausgelesen werden, die Normalmap mittels „*uniform sampler2D normalmap;*“.
3. Beim Normalmapping sind die Koordinaten der Normale in der Textur nicht in Weltkoordinaten gegeben, sondern im sogenannten Tangent-Space. Zum Umrechnen von Weltkoordinaten in Tangent-Space Koordinaten benötigen Sie den Tangential- und den Binormalvektor. Diese werden normalerweise im Host-Programm (nicht im Shader) berechnet. Es existiert aber eine Heuristik, um diese Größen im Shader auszurechnen: (t-tangential, b-binormal)

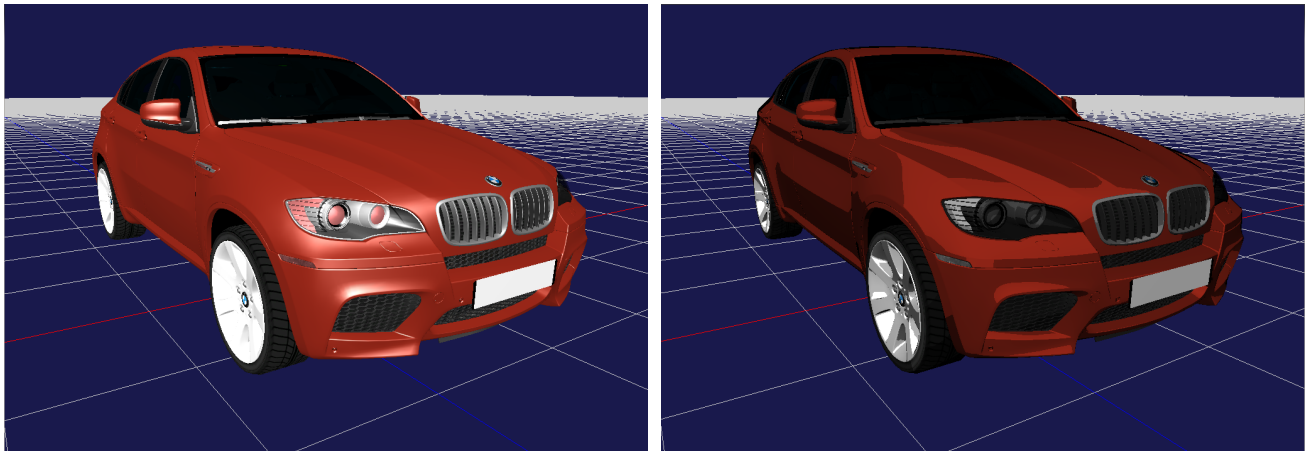
```
vec3 Q1 = dFdx(vertex);
vec3 Q2 = dFdy(vertex);
vec2 st1 = dFdx(gl_TexCoord[0]);
vec2 st2 = dFdy(gl_TexCoord[0]);
vec3 t = normalize(Q1*st2.t - Q2*st1.t);
vec3 b = normalize(-Q1*st2.s + Q2*st1.s);
```



Altair (Phong Shading, ohne Diffustextur): links ohne Normalmap, rechts mit Normalmap

**Aufgabe 2: Toon Shader**

Die letzte Aufgabe besteht darin einen Toon-Shader (auch Cel-Shader genannt) zu schreiben, der den Objekten der 3D-Szene einen Cartoon – Look verpasst. Dieser Effekt wirkt natürlich am Besten bei Objekten, die von Haus aus nicht versuchen realistisch auszusehen (also z.B. nicht mit einer Textur überzogen sind, die dem Objekt eigentlich mehr Realismus verleihen soll – wie z.B. bei Altair).
Hauptmerkmal des Toon Shaders sind die starken Farbabstufungen, die Sie implementieren sollen. Darüber hinaus wird der Cartoon Eindruck verstärkt, wenn Sie eine schwarze Silhouette um das Objekt zeichnen (das können Sie optional implementieren).
Auch zu diesem Thema gibt es sehr viele Beispiele im Netz.



BMW: links Phong Beleuchtung, rechts Phong Beleuchtung + Toon Shader