

Real Time Rendering Aufgabe 1: Simple Lighting

Diese Aufgabe dient zur Einarbeitung in die Verwendung der OpenGL Shading Language und von Qt/OpenGL. Sie beschäftigen sich mit der Entwicklung von Shader-Programmen zur Darstellung von verschiedenen Materialeigenschaften.

Abgabe: siehe Moodle!

Demonstrieren Sie in der Übung nach der Abgabe Ihre Lösung. Alle Gruppenmitglieder müssen dazu anwesend sein. Ausschlaggebend ist neben der eigentlichen Lösung vor allem die Qualität Ihrer Erklärungen.

Bewertung: Werden zwei der drei Teilaufgaben gelöst, wird die Aufgabe maximal mit *gut* bewertet. Ein *sehr gut* ist nur bei Lösung aller Teilaufgaben möglich. Verspätete Abgaben werden mit 2/3 Notenabzug pro angefangener Woche Verspätung belegt.

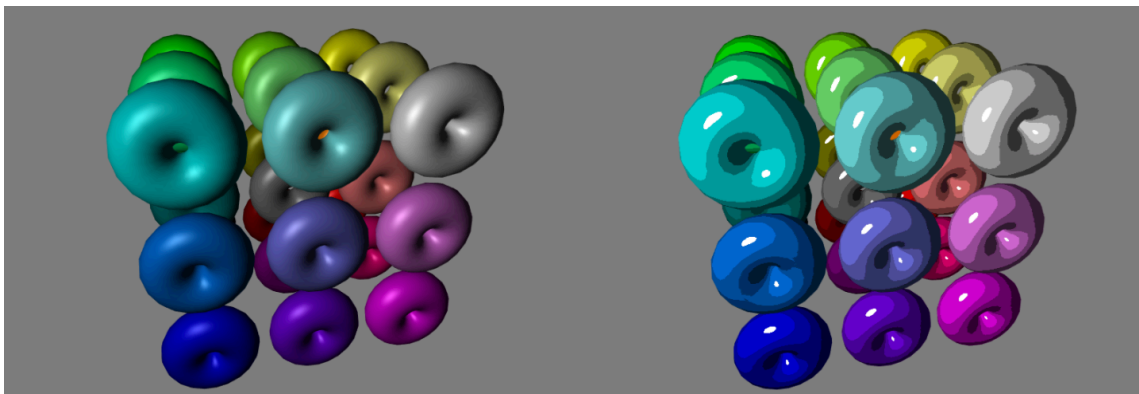
Aufgabe 1.1: Toon Shader

Recherchieren Sie die Technik des *Toon-Shading* (auch bekannt als *Cel-Shading*). Vorsicht: zu diesem Thema gibt es viel „Müll“ im Netz. Dokumentieren Sie im Sourcecode Ihre Quelle(n), und erklären Sie bei der Demo, was für die Qualität Ihrer Quelle spricht (siehe auch Qualitätskriterien weiter unten!)

Kopieren und modifizieren Sie den Phong-Shader so, dass die berechnete Oberflächenintensität, wie im Bild rechts zu sehen, in mehreren Abstufungen diskretisiert wird (Verlauf wird durch grobe Stufen ersetzt). Achten Sie dabei auf folgende

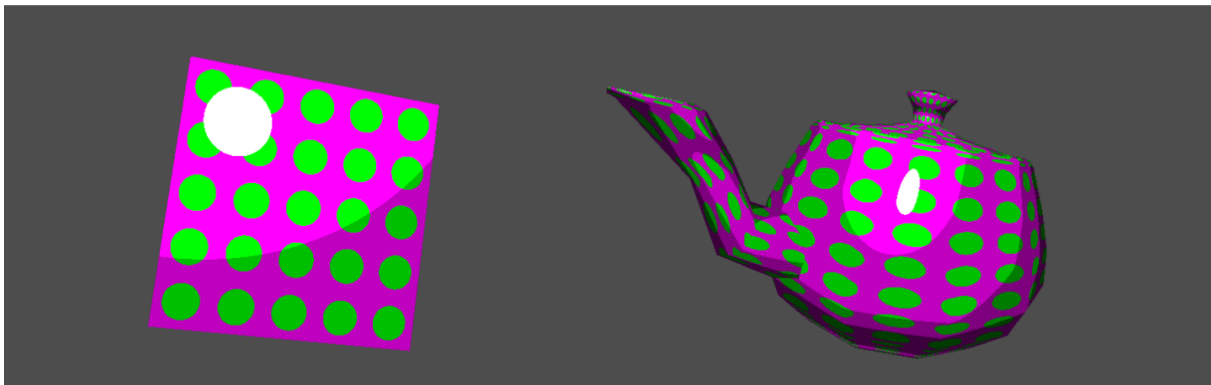
Qualitätskriterien:

- korrekte Wiedergabe des weißen Glanzlichts;
- gleichmäßige Verteilung der Farbstufen über die Einfallswinkel (Achtung genau überlegen was ndotl ist)
- nur intuitive Parameter, keine „magic numbers“ im Code, die man nicht erklären kann.
- die Anzahl der Farbstufen soll interaktiv einstellbar sein
- da Verzweigungen schlecht für die Performanz sind, vermeiden Sie soweit wie möglich if-then-else im Shadercode.



Aufgabe 1.2: Dots

Kopieren und modifizieren Sie Ihren Toon-Shader aus Aufgabe 1.1 so, dass die Oberfläche nicht einfarbig erscheint, sondern mit gleichmäßig verteilten andersfarbigen Punkten übersät ist. Farbe, Dichte und Radius der Punkte sollen über drei Uniform-Variablen von geeignetem Typ aus der Anwendung heraus über die Tastatur steuerbar sein.



Zur Implementierung benötigen Sie Zugriff auf die Texturkoordinaten für jedes Fragment. Dazu muss das entsprechende Attribut `in vec2 texcoord` im Vertex-Programm entgegengenommen werden (siehe auch Beispiele im SU).

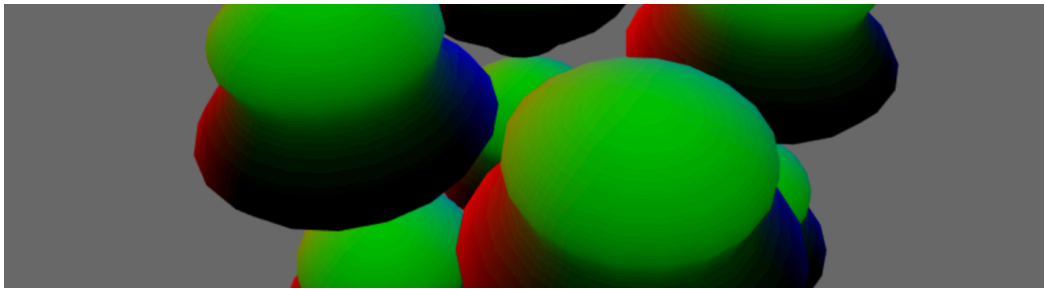
Bitte beachten Sie: es werden lediglich Texturkoordinaten benötigt, nicht jedoch Texturen selbst. Die Verwendung eines Texture-Samplers und der dazugehörigen Look-Up Funktion ist zur Lösung der Aufgabe nicht erwünscht.

Beachten Sie weiterhin, dass einige OBJ-Modelle eine recht unintuitive Verteilung der Texturkoordinaten verwenden.

Aufgabe 1.3: Prozeduraler Veränderung der Oberfläche

Implementieren Sie einen weiteren prozeduralen Oberflächen-Shader ihrer Wahl. Lassen Sie sich dabei z.B. von Beispielen aus dem Web inspirieren. Oder modifizieren Sie die Dots aus Aufgabe 1.2 so, dass das Punktmuster z.B. durch ein Schachbrettmuster ersetzt wird. Beachten Sie, dass Sie die Funktionsweise Ihres Shaders bei der Demo erklären sollen.

Prozedurale Shader sind nicht notwendigerweise statisch, sondern lassen sich durch Übergabe eines Zeitparameters per Uniform-Variable auch animieren. Siehe das „Wobbel“-Beispiel im SU.



Fügen Sie ihrer Szene weitere Modelle hinzu, die dieses Material gut demonstrierbar machen.