Ludwig–Maximilians–Universität München
LFE Medieninformatik
Prof. Dr. Andreas Butz
Alina Hang, Dominikus Baur



# Übungsblatt 5: Shading

**Abgabe:**
Dieses Übungsblatt ist einzeln oder in einer Gruppe zu lösen (wir empfehlen allerdings es allein zu bearbeiten). Die Lösung ist bis **Montag**, den **13. Juni 2011, 12:00 Uhr s.t.** über UniWorx (http://www.pst.ifi.lmu.de/uniworx) abzugeben.
Benennen Sie die Dateien nach dem Schema <Übungsblatt>-<Aufgabe>.<extension>. Packen Sie alle Dateien in eine ZIP-Datei und laden Sie diese bei UniWorx hoch.

**Inhalt:**
Übungsblatt 5 behandelt Shader und die OpenGL ES Shading Language („GLSlang"). Shader sind in WebGL die einzige Möglichkeit mit Farben, Licht und Texturen zu arbeiten. Daher stellt dieses Übungsblatt die Grundlage für kommende Übungsblätter dar.

**Hintergrund:**
In diesem Abschnitt finden Sie in jedem Übungsblatt Hilfsmaterialien, Anleitungen und andere frei verfügbare Informationsquellen die Ihnen bei der Bearbeitung helfen können.
https://www.khronos.org/registry/webgl/specs/1.0/ (Offizielle Spezifikation der Khronos Gruppe)
http://www.khronos.org/registry/gles/specs/2.0/GLSL_ES_Specification_1.0.17.pdf
(Offizielle Spezifikation der OpenGL ES Shading Language – zum Nachschlagen, nicht zum Lesen ;)
http://www.iquilezles.org/apps/shadertoy/ (ShaderToy v0.2 – GLSL Demos)
http://learningwebgl.com/blog/?page_id=1217 (Mutter aller WebGL-Tutorialseiten – arbeiten Sie bitte für dieses Übungsblatt die Tutorials 0 und 1 durch)
http://wiki.delphigl.com/index.php/Tutorial_WebGL (WebGL Tutorial für OpenGL Programmierer)

## Aufgabe 1: Shader und GLSL Grundlagen

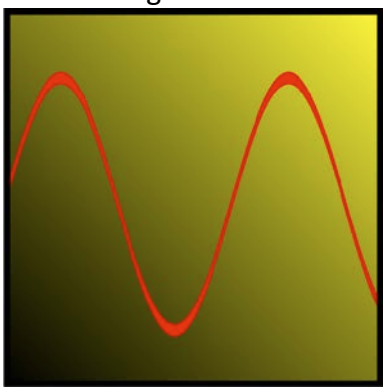
a) Machen Sie sich mit Shading in WebGL/OpenGL ES vertraut. Lesen Sie dazu z.B. die Wikipedia-Seite (http://en.wikipedia.org/wiki/GLSL), LearningWebGL Lesson 2 (http://learningwebgl.com/blog/?p=134) oder die OpenGL ES Shading Language Spezifikation (http://www.khronos.org/registry/gles/specs/2.0/GLSL_ES_Specification_1.0.17.pdf, Seite 7 und Seite 27f.). Was machen Vertex- und Pixelshader? Was ist der Unterschied zwischen attributes, uniform und varying Variablen? Wofür werden sie jeweils eingesetzt? Was wird in den Variablen gl_FragColor und gl_Position gespeichert?
b) Informieren Sie sich jetzt über die verschiedenen Variablentypen in GLSL (matX, vecX, etc) und die verschiedenen vordefinierten Funktionen (z.B. dot, cross). Lesen Sie dazu die Spezifikation ab Seite 17 (Variablen) und Seite 63 (Funktionen) oder dieses Tutorial: http://wiki.delphigl.com/index.php/Tutorial_glsl (Achtung, bezieht sich auf OpenGL SL!)
c) Nehmen Sie jetzt das Codegerüst von der Homepage das einen weißen Würfel anzeigt. Verändern Sie den Fragment Shader so, dass der weiße Würfel in Rot gezeichnet wird.

Verändern Sie den Vertex Shader so, dass alle z-Koordinaten auf 0.5 Einheiten skaliert werden (ein Vertex an der Position (1, 0, -2) soll also z.B. auf der Position (1, 0, -1) landen).

## Aufgabe 2: Fragment Shading

In dieser Aufgabe sollen Sie mit dem Fragment Shader experimentieren. Dieser kann – rein theoretisch – zum Zeichnen beliebiger Formen genutzt werden. Arbeiten Sie mit dem Codegerüst von der Homepage für diese Aufgabe.

a) Initial zeichnet das Codegerüst nur ein weißes Viereck, das beinahe das gesamte Sichtfeld füllt. Passen Sie den Fragment Shader so an, dass das Viereck mit einem gelb-schwarzen Gradienten gefüllt ist, der sich von links unten (schwarz) nach rechts oben (gelb) zieht.



*Tipp*: Die vordefinierte Variable *gl_FragCoord* vom Typ vec2 enthält die Bildschirmposition des Pixels der gerade berechnet wird!

b) GLSL enthält auch grundlegende trigonometrische *Funktionen* (s. Aufgabe 1b)). Lassen Sie den Fragment Shader über dem gelb-schwarzen Gradienten eine Sinuskurve zeichnen. Oben rechts sehen Sie ein Beispiel wie das Endergebnis aussehen könnte.

## Aufgabe 3: Animationen und Shading

Statische Shader sind meist nicht besonders interessant – daher bietet es sich an eine Variable für die Zeit einzuführen und das Bild möglichst oft neu zu zeichnen (diese Schleife heißt im Computergrafikkontext auch *render loop*).

a) Sehen Sie sich einige der Beispiele im Shadertoy (http://www.iquilezles.org/apps/shadertoy/) an. Diese Seite listet auch jeweils direkt den GLSL Code des verwendeten Fragment Shaders auf (und wundern Sie sich auch nicht, falls nicht alle Beispiele im Browser lauffähig sind).

b) Nutzen Sie das Codegerüst für diese Aufgabe. Dieses enthält einen render loop und eine Variable *time*, die jeweils von 0.0 bis 1.0 hoch- und dann wieder runtergezählt wird (falls Sie mehr über die Funktionsweise erfahren möchten, lesen Sie WebGL Lesson 3: http://learningwebgl.com/blog/?p=239). Lassen Sie zuerst den Vertex Shader wieder die z-Koordinate manipulieren: Diesmal soll die z-Achse abhängig von time skaliert werden. Bei time = 0.0 sollen alle z-Koordinaten gleich 0 sein. Bei time = 1.0 sollen alle z-Koordinaten normale Werte erhalten. Interpolieren Sie dazwischen.

c) Lassen Sie den Fragment Shader die Helligkeit des Würfels von schwarz (time = 0) bis weiß (time = 1) setzen.