Übung Nr. 6:

Im Programm `ObjElabGLaaTx.c` soll die Darstellung texturierter Objekte implementiert werden (Menüpunkt 'T'); zur Orientierung sind die Bezeichner `WITH_TEXR` und `MORE_TEXR` vorbereitet worden. Die Stellen, an denen (insg. ca. 20 Zeilen) Code fehlt, sind durch `MORE_TEXR` gekennzeichnet, indizierte Bitmaps zur Umwandlung des Würfelmodells in einen Spielwürfel nach Abb. 1 sind in der Datei `diceTex.h` enthalten; das entsprechende MS-VC-Projekt kann unter http://homepages.thm.de/christ/ heruntergeladen werden.

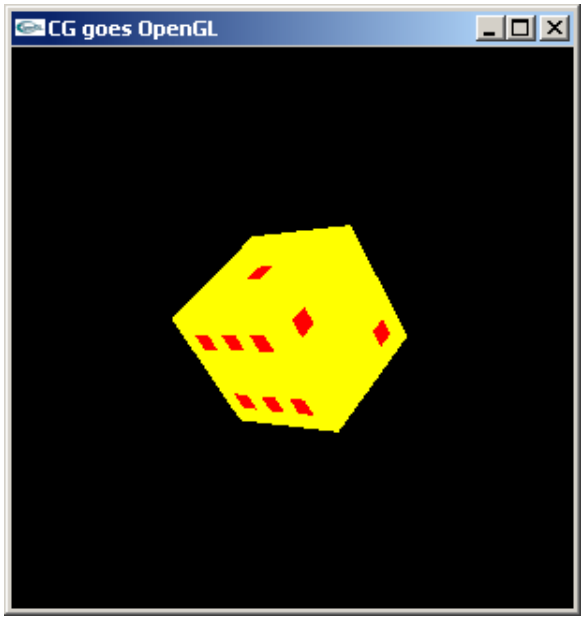

Abb. 1 Texturierter Würfel

Wie in der Vorlesung besprochen, werden zur Einrichtung und Verwaltung von Textur-Objekten mit OpenGL vier Schritte benötigt, nämlich

1. die Vereinbarung programm-interner „Texturnamen“ mit `glGenTextures()`,
2. die Ersteinrichtung und die Nutzung von Textur-Objekten durch `glBindTexture()`,
3. die Besetzung der jeweils aktuellen Textur mit RGB(A)-Daten über `glTexImage*()` und
4. die Zuordnung von Textur- / Objektkoordinaten mit Hilfe von `glTexCoord*()` sowie die Einstellung der Vergrößerungs- und Verkleinerungsfilter mittels `glTexParameter*()`.

In der vorliegenden Implementierung kommt der Funktion `TexInit()` eine besondere Bedeutung zu; darin sind die einmalig benötigten unter den o.a. Schritten integriert: Die Generierung von Textur-Namen (Schritt 1) und die Ersteinrichtung der Textur-Objekte (als erster Teil von Schritt 2) sind schon korrekt eingetragen. Außerdem ist in der Funktion `drawObj()`, im Rahmen der Behandlung der Punkte jeder zu zeichnenden Objektfläche, der erste Teil von Schritt 4 bereits implementiert.

Zur weiteren Behandlung der Aufgabe wird folgendes Vorgehen empfohlen:

Beim Compilieren der Projektes wird darauf hingewiesen, daß die dreifach indizierte Speicherung der Farbwerte für die 6 Texturen (`texels[][][]`) nicht benutzt wird. Dies muß nachgeholt werden, weil sonst die Bitmaps unter `diceTex[][]` (Datei `diceTex.h`) nicht verwendet werden können.

Wie aus der Theorie bekannt und an der Programmstruktur zu erkennen ist, sollen für jede Fläche aus jedem Wert des `char`-Feldes `diceTex[][]` (Datei `diceTex.h`) drei Werte (für Rot, Grün und Blau) der Textur `texels[][][]` entstehen. Wie weiterhin Abb.1 zu entnehmen ist, soll dabei die Farbe der Würfel-Augen rot, jene des Hintergrunds gelb sein. Prinzipiell kann man hier zunächst eine reine Umspeicherung vornehmen, die später angepaßt wird. (OpenGL erwartet eine Textur-Speicherung von, bei der die Texel aller Zeilen von links nach rechts angetroffen werden, beginnend mit der untersten Texturzeile.)

Eine Verbindung zwischen den nun gespeicherten Texturen und den nunmehr gesetzten RGB-Werten muß noch (Schritt 3, s.o.) hergestellt werden; dafür ist ein Platz kurz vor dem Ende der Funktion vorgesehen.

Den Anblick eines texturierten Objekts verschafft erst die Mitteilung an das System, wie die Textur bei Vergrößerungen und Verkleinerungen zu filtern ist (Schritt 4). Hierbei ist zu berücksichtigen, daß im Menü (Funktion `key()`) vorgesehen ist, anhand der Variablen `texFil` die Filterung zwischen `GL_NEAREST` und `GL_LINEAR` wechseln zu lassen (Betätigung der Leertaste). Ist diese Lücke geschlossen, so sind zwei texturierte Versionen zu sehen. Dabei fällt auf, daß alle Objektflächen nach derselben Bitmap texturiert werden (Abb. 2). Ursache hierfür ist wiederum, daß (gemäß Schritt 2) die Festlegung der jeweils aktuellen, individuellen Textur für jede Fläche fehlt, was verständlicherweise beim Zeichnen des grafischen Objektes (in welcher Funktion?) seinen Platz hat.

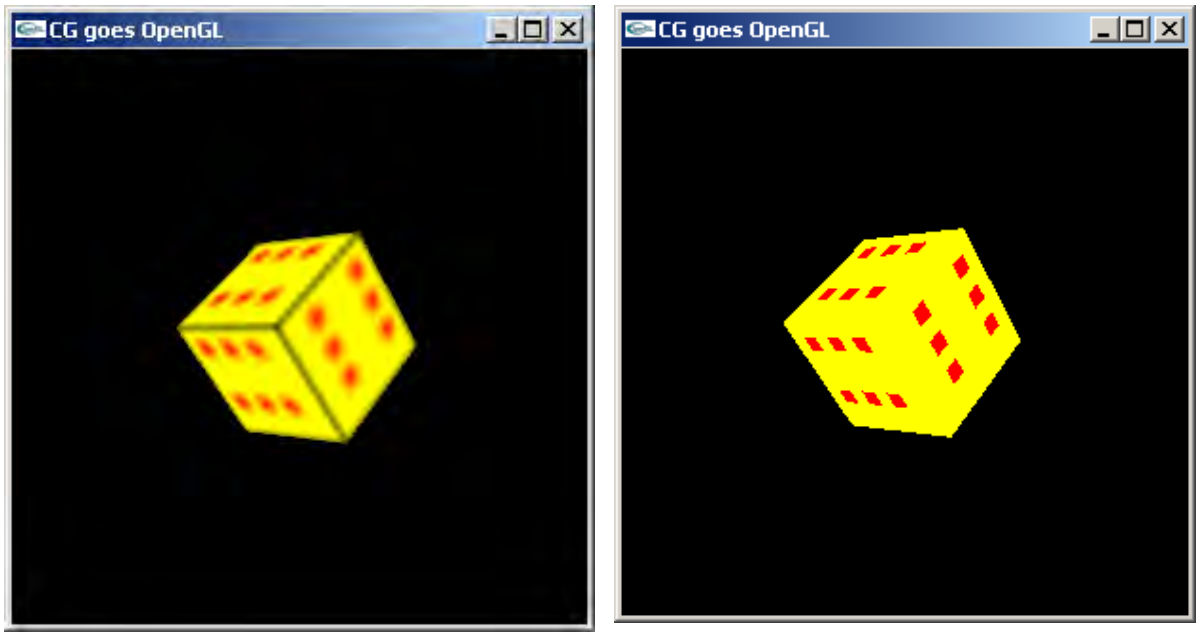

Abb. 2 Würfel bei Textur-Filterung mit `GL_LINEAR` (li.) und `GL_NEAREST` (re.)

Nach Behebung auch dieses Mangels ist eine erste Version des texturierten Modells nach Abb. 3 (li.) zu sehen.

Nun sollte überprüft werden, ob die Texturen korrekt aus den Bitmaps übertragen werden. Dazu kann für das Spielwürfel-Feld mit der Eins eine Bitmap mit der bildlichen Darstellung der Ziffer '1' aktiviert werden. Bei richtiger Übertragung der Werte zwischen den Arrays `diceTex[][]` und `texels[][][]` sollten Einstellungen wie jene in Abb. 3 (re.) entstehen.

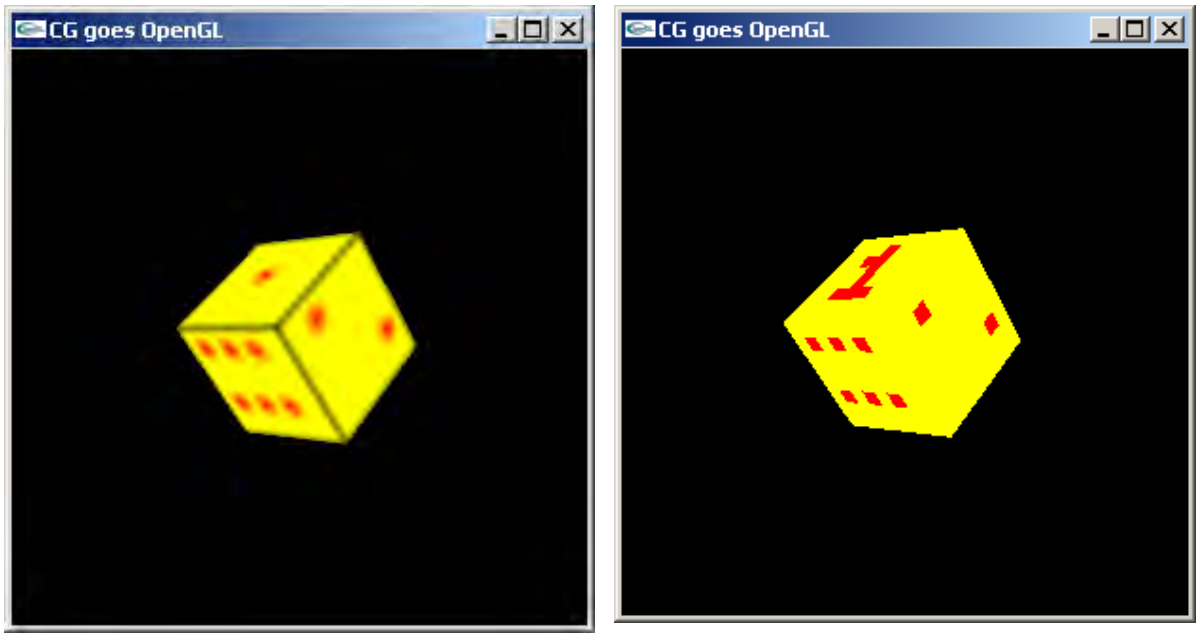

Abb. 3 Würfel mit korrekter Textur-Zuordnung bei Filterung mit `GL_LINEAR` (li.) und mit bildlicher Darstellung bei `GL_NEAREST` (re.)

Eine letzte Schwäche kennzeichnet noch das Ergebnis der Texturierung: Wie z.B. anhand der Darstellung der Zahl 6 sofort zu erkennen ist, beträgt die optimale Kantenlänge für eine Spielwürfel-Textur 7 Texel; dies mußte hier jedoch wegen der von OpenGL geforderten Bindung an Zweier-Potenzen auf 8 Texel erweitert werden und macht sich beim erstmaligen Laden des Modells störend bemerkbar.

Dieser Makel läßt sich schnell durch entsprechende Korrektur der Bezeichner `LFT`, `BTM`, `RIT` und `TOP` beheben (Abb. 4).

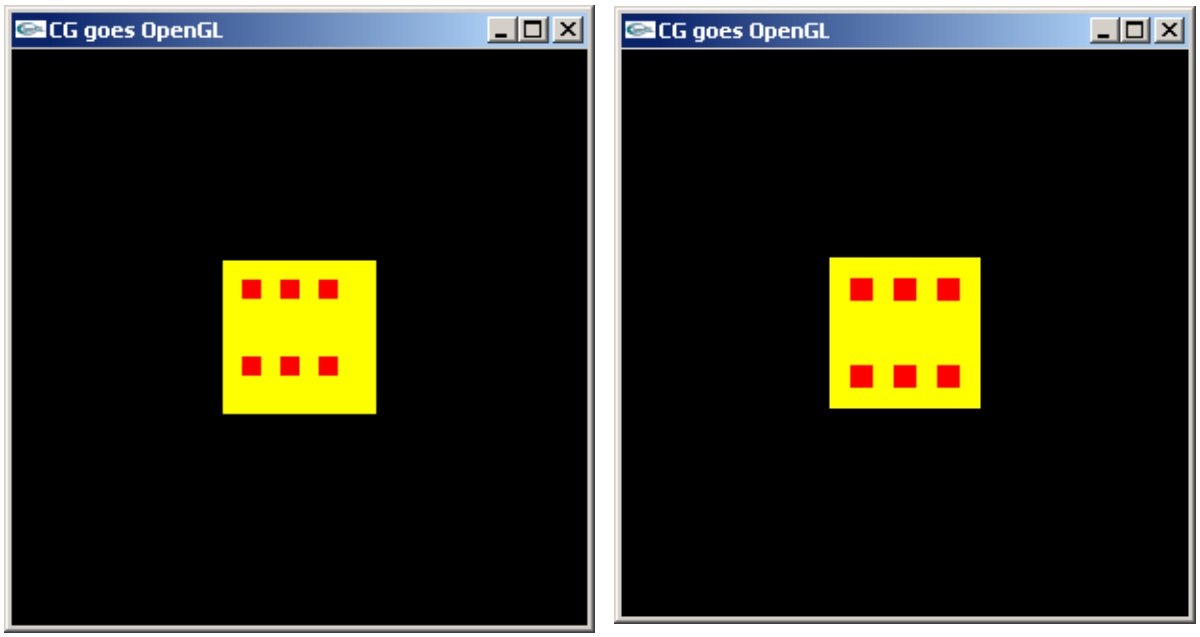

Abb. 4 Würfel vor (li.) und nach (re.) Zentrierung der Textur (bei Filterung mit `GL_NEAREST`)