Ludwig-Maximilians-Universität München
LFE Medieninformatik
Prof. Dr. Andreas Butz
Dominikus Baur, Sebastian Boring

Sommersemester 2009
Übungsblatt 6
29. Mai 2009

Computergrafik 1

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum Freitag den **5. Juni 2009, 12:00 Uhr s.t.** per Email abzugeben.

Inhalt:
Dieses Blatt dient zum Erlernen von grundlegenden Konzepten der Beleuchtung und Texturierung. Hierzu werden Sie Ihre bereits erstellten Objekte (*Quader*, *Pyramide*, *Kegel* und *Kugel*) um Texturkoordinaten sowie Normalen erweitern, um eine korrekte Darstellung zu ermöglichen. Zusätzlich lernen Sie, Lichtquellen (hier: *Ambientes Licht*, *Punktlicht* und *Spotlicht*) zu platzieren, um eine Beleuchtung der Szene realisieren zu können. Pro Aufgabe können maximal zehn (Aufgaben 20 und 21 maximal 20) Punkte erreicht werden. Zum Bestehen des Übungsblatts müssen in **JEDER** Aufgabe mindestens fünf (Aufgabe 20 und 21 mindestens 10) Punkte erreicht werden.

Geben Sie insgesamt (d.h. ein Projekt für alle Aufgaben zusammen) alle benötigten Header-, Source-, und Projektdateien (von *Qt Creator* erzeugt) mit ab, d.h. `*.h`, `*.cpp` und `*.pro` Dateien. Abgaben die nicht kompilieren werden mit 0 Punkten bewertet. Fassen Sie alle Aufgaben zu einer zip-Datei zusammen und senden Sie diese an: `cg1_ss09@medien.ifi.lmu.de`.

Aufgabe 19 Normalen der Vertices (10 Punkte)

Diese Aufgabe dient zum Erlernen der Erzeugung von Normalen für jeden Vertex eines Objekts. *OpenGL* bietet hierfür die Funktion `glNormal3d` an. Im Folgenden wird beschrieben, wie die Normalen für jedes Objekt zu setzen sind. Erweitern Sie die oben genannten Objekte entsprechend der Vorgabe in dieser Aufgabe.

a) *Quader*: Die Normalen jedes Vertex entsprechen der Flächen-Normalen an der dieser beteiligt ist. Die Normale zeigt hierbei vom Quader weg, d.h. sie zeigt nach aussen. Da die Kanten des Quaders „harte“ Kanten sind (d.h. sie fließen farblich nicht ineinander über), hat jeder Vertex drei Normalen. Genauer gesagt existieren an jedem Eckpunkt des Quaders ohnehin drei Vertices, die dann jeweils zueinander unterschiedliche Normalen haben. Hierzu ein Beispiel:

```
glBegin(GL_QUADS);
  glVertex3d(1.0, -1.0, 1.0);
  glVertex3d(1.0, 1.0, 1.0);
  glVertex3d(-1.0, 1.0, 1.0);
  glVertex3d(-1.0, -1.0, 1.0);
glEnd();
```

Die erzeugten Punkte würden prinzipiell zwei Normalen zulassen, nämlich $\vec{n} = (0,0,1)^T$, sowie $\vec{n} = (0,0,-1)^T$. Wenn man jetzt noch weiß, dass der erzeugte Quader seinen Mittelpunkt im Ursprung hat, ist die korrekte Normale durch $\vec{n} = (0,0,1)^T$ gegeben. Diese Normale wäre für die vier erstellten Vertices gültig. Erzeugen Sie nun die Normalen für alle Vertices des Quaders (**2 Punkte**).

b) *Pyramide*: Die Normalen einer Pyramide sind ähnlich zu erstellen, wie die eines Quaders. Für die Grundfläche ist die Vorgehensweise exakt der bei einer Seitenfläche des Quaders. Für die Mantelflächen (bzw. die Dreiecke) gilt dann, dass die Normale an jedem Vertex gleich der Normale der Fläche ist. Die Flächennormale können Sie sich durch das Vektorprodukt aus zwei Vektoren der Fläche berechnen. **Hinweis:** Achten Sie darauf, dass die Normalen „nach aussen“ zeigen. Erzeugen Sie nun die Normalen für alle Vertices der Pyramide (**2 Punkte**).

c) *Kegel*: Der Kegel ist eine spezielle Form der Pyramide. Auch hier gilt für die Grundfläche wieder die gleiche Normale. Für die Mantelflächen benötigen Sie jedoch zusätzlich die Fläche **vor** und die Fläche **hinter** der aktuellen Fläche. Da die Kanten auf dem Kegelmantel nicht mehr sichtbar sein sollen, müssen Vertices, die in zwei unterschiedlichen Flächen involviert sind, jedoch an der gleichen Position liegen, die gleiche Normale aufweisen. Der einfachste Weg besteht nun darin, den Mittelwert aus der Normalen der aktuellen Fläche und der Normale der vorherigen (bzw. nachfolgenden) Fläche zu berechnen. Diese gemittelte Normale ist dann die korrekte Normale für **alle** Vertices, die an dieser Stelle gezeichnet werden. Erzeugen Sie nun die Normalen für alle Vertices des Kegels (**3 Punkte**).

d) *Kugel*: Bei der Kugel besteht das gleiche Problem wie bei einem Kegel, d.h. alle Flächen sollen später ineinander übergehen. Daher müssen auch hier die Normalen gemittelt werden. Allerdings gibt es auch ein deutlich einfacheres Verfahren: Die Normale zu einem Vertex entspricht dem normierten Vektor vom Ursprung zum jeweiligen Vertex. Erzeugen Sie nun die Normalen für alle Vertices der Kugel (**3 Punkte**).

Aufgabe 20 Beleuchtung (20 Punkte)

Diese Aufgabe dient zum Erlernen der grundlegenden Verfahren eine Szene zu beleuchten. *OpenGL* bietet hierfür zahlreiche Funktionen an, die bestimmte Lichtparameter beeinflussen können. Im Folgenden werden Sie drei Lichttypen erzeugen: *Ambientes Licht*, *Punktlicht* und *Spotlicht*.

a) Erzeugen Sie sich ein weiteres Szene-Objekt namens *Plane*. Dieses Objekt ist eine Ebene, die aus einer gewissen Anzahl von horizontalen (x-Achse) und vertikalen (z-Achse) Segmenten besteht. **Hinweis:** Verwenden Sie eine höhere Anzahl von Segmenten, um später eine reellere Beleuchtung zu erhalten (**2 Punkte**).

b) Erzeugen Sie sich nun eine Klasse *SceneLight*, die folgende Parameter erhält: Eine Position, an die die Lichtquelle gesetzt werden soll, eine Farbe (d.h. ein Objekt vom Typ `QColor`) für den *ambienten* Anteil des Lichts und eine Farbe für den *spiegelnden* Anteil des Lichts. Erstellen Sie zusätzlich eine Methode `setup(int index)`, die nicht direkt von *SceneLight* implementiert wird (**3 Punkte**).

c) Erstellen Sie sich nun ein Objekt namens *PointLight*, das eine Punktlichtquelle repräsentiert. Diese erbt von *SceneLight* und erhält keine zusätzlichen Parameter (**2 Punkte**).

d) Erstellen Sie nun noch ein Objekt namens *SpotLight*, das einen „Strahler“ repräsentiert. Dieser kann entweder direkt von *SceneLight* oder von *PointLight* erben. Zusätzlich erhält diese Lichtquelle folgende Parameter: eine Richtung, in die sie strahlen soll, einen Öffnungswinkelwinkel (**Hinweis:** Dieser Winkel ist der *halbe* Öffnungswinkel und liegt im Bereich von 0° bis 90°!) sowie einen Exponenten (im Bereich von 0 bis 128), der beschreibt, wie „hart“ bzw. „weich“ das Licht des Strahlers ist (**3 Punkte**).

e) Implementieren Sie nun die `setup`-Funktionen dieser Lichtquellen. Verwenden Sie als Lichtparameter für *OpenGL* immer `GL_LIGHT0 + index` und setzen Sie alle Werte, die für die Lichtquelle relevant sind (**2 Punkte**).

f) Fügen Sie nun jeweils mindestens eine Instanz dieser Lichtquellen zur Szene hinzu, und lassen Sie diese bei jedem Rendervorgang durch Aufruf der `setup`-Funktion mit berechnen. **Hinweis:** Vergessen Sie nicht `GL_LIGHTING` zu aktivieren. Deaktivieren Sie das Lighting-Verfahren dann später wieder **bevor** Sie die Achsen eines Objekts zeichnen (**4 Punkte**).

g) Erweitern Sie die Szene um eine ambiente Lichtquelle. Setzen Sie dazu **zu Beginn** jedes Rendervorgangs die Parameter `GL_LIGHT_MODEL_AMBIENT` auf eine von Ihnen definierte Farbe mit Hilfe der Funktion `glLightModelfv`. **Hinweis:** Um eine sinnvolle Ausleuchtung zu erhalten, verwenden Sie am Besten die Farbe *weiß* (**4 Punkte**).

Aufgabe 21 Texturen (20 Punkte)

In dieser Aufgabe erlernen sie die Texturierung der in der Szene befindlichen Objekte. Hierzu müssen den Objekten (bzw. jedem Vertex) noch *Texturkoordinaten* hinzugefügt werden. Für zweidimensionale Texturen bietet *OpenGL* die Funktion `glTexCoord2d` an.

a) Geben Sie jedem Objekt (d.h. jedem Vertex) Texturkoordinaten mit Hilfe der Funktion `glTexCoord2d`. Beachten Sie, dass die Texturkoordinate $(0,0)$ die *linke untere* und die Texturkoordinate $(1,1)$ die *rechte obere* Ecke der Textur beschreibt (**4 Punkte**).

b) Erzeugen Sie sich eine Klasse namens *Texture*, die später für genau eine Textur bereitgestellt wird. Der Konstruktor erhält einen `QString`, der den Pfad zu einem Bild definiert. Geben Sie der *Texture* außerdem eine Variable vom Typ `GLuint`, in der später die ID der Textur gespeichert wird. Achten Sie bei den Bildern darauf, dass deren Abmessungen grundsätzlich im Format 2^n sein müssen. Beachten Sie ferner, dass manche Rechner (so wie in *OpenGL* standardisiert) keine Texturen mit Abmessungen größer als 512 verwenden können. Die auf der Homepage bereitgestellten Texturen sind teilweise jedoch größer, d.h. falls diese auf Ihrem System nicht funktionieren, dürfen Sie diese verkleinern (**2 Punkte**).

c) Laden Sie im Konstruktor einer *Texture* nun das Bild in ein `QImage` Objekt und erzeugen Sie sich die Textur. Gehen Sie dazu folgendermaßen vor:

- Ein `QImage` enthält eine Methode namens `load`, der lediglich ein Pfad übergeben werden muss. Verwenden Sie diese Methode, um das Bild zu laden.
- Konvertieren Sie dieses Bild nun in ein gültiges *OpenGL*-Format. **Hinweis:** Die *statische* Funktion `QImage convertToGLFormat(const QImage& img)` die in der Klasse `QGLWidget` hilft hier ungemein.
- Generieren Sie sich nun die Textur, d.h. verwenden Sie die Funktion `glGenTextures` mit der zuvor angelegten ID der *Texture*. Verwenden Sie nun die Funktionen `glEnable` und `glBindTexture`, um die Textur temporär an *OpenGL* zu binden. Dies ist nötig, um der Textur noch weitere Parameter zu geben.
- Erzeugen Sie nun die Textur mit Hilfe der Funktion `glTexImage2D`. **Hinweis:** Das zuvor erzeugte und konvertierte `QImage` enthält bereits alle nötigen Informationen (z.B. durch die Funktionen `width()`, `height()` und `bits()`). Setzen sie nun noch die Texturparameter für den Verkleinerungs- und Vergrößerungsfilter. Hierfür können Sie die Funktion `glTexParameteri()` verwenden.

(**6 Punkte**)

d) Erweitern Sie die Klasse *SceneObject* um eine *Textur*, d.h. geben Sie dieser Klasse eine Variable für die zu speichernde Textur. Erweitern Sie zudem die Funktion `render` derart, dass die Textur gezeichnet werden kann. Verwenden Sie dazu wiederum die Funktionen `glEnable` und `glBindTexture`. Deaktivieren Sie nach dem Zeichnen `GL_TEXTURE_2D` wieder, damit nicht-texturierte Objekte korrekt dargestellt werden können (**4 Punkte**).

e) Laden Sie sich nun die Datei *textures.zip* herunter, und texturieren Sie die Objekte *Quader* (Box.png), *Pyramide* (Pyramid.png), *Kegel* (Cone.png), *Kugel* (Sphere.png) und *Ebene* (Plane.png). Auf der Homepage ist das Ergebnis der Texturierung dargestellt (**4 Punkte**).