ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

cgl computer graphics lab

Prof. Markus Gross
Remo Ziegler / Christian Vögeli / Daniel Cotting

Zu lösen bis: 9.5.2006

Visual Computing I
Lighting - Shading Models - Multipass Rendering

Ziele

Vertiefung der Vorlesung im Bereich Beleuchtungsmodelle & Projektionen.
Kennen lernen von Multipass-Rendering Verfahren.
Generierung von qualitativ ansprechenden Bildern.

Allgemeine Hinweise

Das Codegerüst kann wie üblich direkt von der Webseite geladen, in einem persönlichen Verzeichnis gespeichert und mit der Visual C++ Programmierumgebung bearbeitet werden.

Ressourcen

Webseite der Vorlesung: `http://graphics.ethz.ch/`

Theoretische Aufgabe

1) Beleuchtungsmodell

Geben Sie zu den unten aufgeführten Bildern an, mit welchem Beleuchtungsmodell sie erstellt wurden. Begründen sie ihre Antwort.

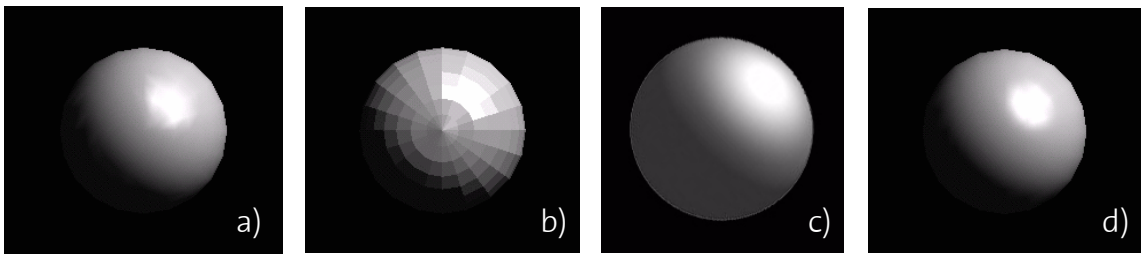
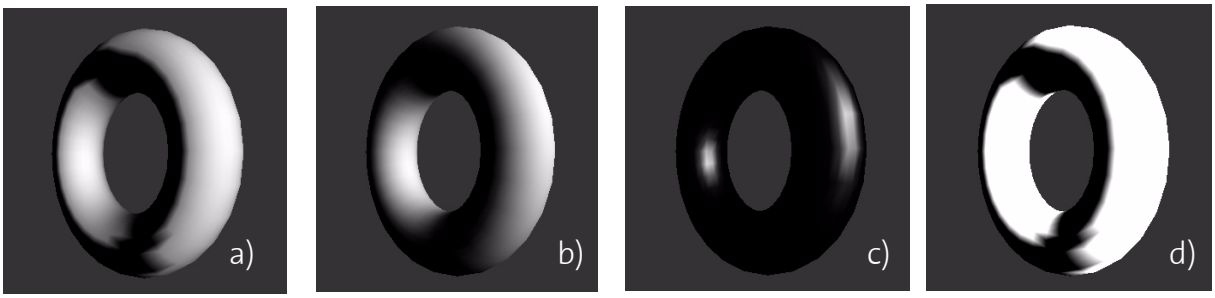


Figure 1: Es kann auch mehrere Lösungen geben. Geben sie in solchen Fällen an, welche Annahmen für das Modell getroffen werden müssen.

2) Beleuchtungseigenschaften

Die unten aufgeführten Bilder wurden alle mittels Gouraud-Shading berechnet. Das verwendete Modell unterstützt ambiente-, diffuse- und spekulare Reflexionseigenschaften. Pro Bild wird jedoch nur eine dieser Komponenten verwendet. Geben Sie jeweils pro Bild an, welche Komponente in der Materialbeschreibung verwendet wurde. Falls zwei Bilder dieselbe Materialeigenschaft haben, erklären Sie warum die Bilder trotzdem unterschiedlich aussehen. Dabei gilt die Annahme, dass die Lich-

tquelle immer am gleichen Ort ist und ambientes, diffuses und spekulares Licht ausstrahlt.



Praktischer Teil

Bei einem Multipass-Rendering Verfahren wird – wie es der Name schon andeutet – zur Generierung eines Bildes dieselbe Szene mehrfach gerendert. Es werden im Falle dieser Übung mehrere Einzelbilder mit verschiedenen Lichtquellen berechnet und in einem Akkumulationsbuffer aufaddiert. Das Ergebnisbild stellt somit eine Mittelung aller berechneten Einzelbilder dar und es ergibt sich ein Bild mit mehreren Lichtquellen. Berechnet man nun für die Lichtquelle in jedem Einzelbild jeweils die Schatten der Objekte, so ist es durch das Multipass-Rendering möglich, ein Bild mit weichen Schatten für eine räumlich ausgedehnte Lichtquelle zu generieren. Da OpenGL keinen Szenengraph kennt, können Schatten nicht direkt von OpenGL erzeugt werden. Die Schatten müssen explizit berechnet werden, indem die Objekte von der Lichtquelle aus auf alle Oberflächenelemente der Szene projiziert werden, welche im Bereich eines Schattens liegen.
Fügt man über die Serie der Einzelbilder ein sogenanntes Jittering der Kameraposition ein, so kann ein Vollbild-Antialiasing erreicht werden. Moderne 3D Beschleuniger für PCs bieten Verfahren für Vollbild-Antialiasing direkt in Hardware an.

3) Setzen von Materialeigenschaften

Implementieren Sie die Funktion

- `void setMaterial(...)`

welche die Materialeigenschaften der darzustellenden Objekte setzt.

Durch die Funktion `setMaterial` sollen die Oberflächeneigenschaften von Objekten definiert werden. Die Funktion wird für jedes zu zeichnende Objekt mit den entsprechenden Parametern aufgerufen werden. Daher sind die Eigenschaften auch anhand der Übergabeparameter zu setzen.
OpenGL unterscheidet zwischen ambienter (`GL_AMBIENT`), diffuser (`GL_DIFFUSE`) und spekularer (`GL_SPECULAR`) Lichtreflexion, wobei die Glanzeigenschaft von glatten Oberflächen durch einen zusätzlichen Parameter (`GL_SHININESS`) beeinflusst werden kann. Zusätzlich gibt es die Möglichkeit, selbstleuchtende Objekte zu beschreiben (`GL_EMISSION`).
All diese Eigenschaften sollen gesetzt werden. Dazu kann die folgende OpenGL-Funktion verwendet werden:

- `glMaterialf`

4) Setzen von Spot-Lichtquellen

Implementieren Sie die Funktion

- `void setLighting(void)`

welche eine Spot-Lichtquelle definiert und die Beleuchtung in OpenGL aktiviert.

In OpenGL kann ein Spotlight durch das Setzen bestimmter Parameter für eine Lichtquelle simuliert werden. Dazu ist die Beleuchtungsrichtung (`GL_SPOT_DIRECTION`), der CutOff-Winkel (`GL_SPOT_CUTOFF`) und der Exponenten n (`GL_SPOT_EXPONENT`) der Lichtstärkeverteilung $\cos^n(\alpha)$ zu definieren. Der CutOff-Winkel gibt den maximalen Winkel α eines Lichtstrahls der Lichtquelle zur Beleuchtungsrichtung an und definiert so einen Lichtkegel. Der Attenuationsfaktor einer Lichtquelle wird in OpenGL wie folgt berechnet

$$\text{attenuation} = \frac{1}{k_c + k_l d + k_q d^2}$$

wobei

- `d` = Distanz zwischen Lichtquelle und betrachtetem Punkt
- k_c = `GL_CONSTANT_ATTENUATION`
- k_l = `GL_LINEAR_ATTENUATION`
- k_q = `GL_QUADRATIC_ATTENUATION`

Setzen Sie nun in der Funktion `setLighting` die vorgegebenen Variabeln auf sinnvolle Werte. Diese Werte werden anschliessend benötigt, um die entsprechenden Zustände in der OpenGL-StateMachine zu definieren. Setzen Sie die Attenuationsparameter so, dass die damit definierte Lichtquelle eine zur Distanz linear abnehmende Beleuchtungsstärke aufweist. Ändern Sie nichts an der Position des Spotlights – diese wird an einer anderen Stelle gesetzt. Lassen Sie das Licht in die Richtung der positiven x-Achsen leuchten.

Welche OpenGL-Funktionen für das Setzen dieser Eigenschaften einer Lichtquelle sowie für die Aktivierung der Beleuchtung verwendet werden müssen, entnehmen Sie direkt dem Quellcode.

5) Schattengenerierung

Implementieren Sie die Funktionen

- `void generateShadowProjections(...)`

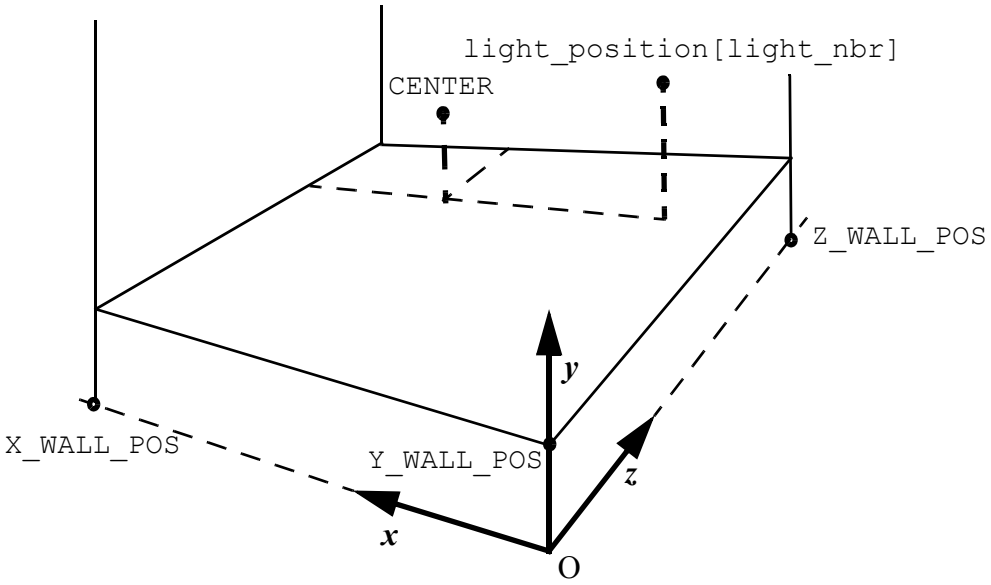
welche die Projektionsmatrix für die Schattenprojektion auf eine Wand oder den Boden nicht aber auf andere Gegenstände generiert, sowie

- `void drawShadow(...)`

welche die Schattenprojektionen ausführt.

Die Schatten aller in der Szene vorkommenden Objekte sollen auf die beiden Wände und auf den Boden der Szene projiziert werden. Für jede dieser drei Projektionen soll in der Funktion `generateShadowProjections` abhängig von der aktuellen Lichtposition eine Projektionsmatrix berechnet werden. Die aktuelle Position der Lichtquelle ist im Array `light_position` zugänglich (siehe auch Aufgabe 6).
Damit sich die Projektionen vereinfachen ist das Koordinatensystem der Szene so gewählt, dass die gesamte Szene in den positiven Achsenrichtungen liegt. Zudem

stehen alle Wände senkrecht zu einer der Koordinatenachsen.



Die Konstanten `X_WALL_POS`, `Y_WALL_POS` und `Z_WALL_POS` geben die Positionen der Wände und des Bodens als Abstand vom Ursprung vor. `CENTER` bezeichnet das Zentrum der dargestellten Objekte – alle Objekte in der Szene werden relativ zu diesem Referenzpunkt gezeichnet.
Wichtig zu beachten ist, dass bei der Projektion auf den Boden – im Gegensatz zu den Projektionen auf die Wände – in die negative Achsenrichtung projiziert wird, weil sich die Ebene, auf welche projiziert wird, auf der negativen y-Achse befindet. Beim Generieren der dafür benötigten Matrix ist sicherzustellen, dass bei einer Multiplikation von Punkten mit dieser Matrix keine negativen homogene Koordinaten entstehen, weil OpenGL keine negativen homogenen Koordinaten kennt.
Als Vorlage für die Berechnung der Matrizen können die Betrachtungen im Kapitel 4.2.1 auf Seite 69 im Skript dienen.
Wenden Sie die generierten Matrizen in der Funktion `drawShadow` an. In derselben Funktion werden die Objekte – nun als Schatten – nochmals gezeichnet. Damit die Berechnung der Matrizen einfacher wird, soll die gesamte Szene vor Anwendung einer der Schattenprojektionsmatrizen so translatiert werden, dass die aktuelle Lichtquelle in den Ursprung zu liegen kommt. Nach der Multiplikation mit der Matrix muss die Szene selbstverständlich zurücktranslatiert werden. Diese Vorgehensweise **muss** bei der Generierung der Matrizen berücksichtig werden.
Hinweise auf allfällige zu verwendenden OpenGL-Funktionen können wiederum direkt den Kommentaren im Quellcode entnommen werden. Zu beachten ist, dass Matrizen in OpenGL als Arrays dargestellt werden. So entspricht beispielsweise der Array m[16] mit 16 Elementen der folgenden Matrix:

$$\begin{bmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{bmatrix}$$

6) Multipass-Rendering mit ausgedehnter Lichtquelle

Implementieren Sie die Funktion

- `void buildLightSource(...)`

welche eine eine räumliche, rechteckförmige Lichtquelle bestehend aus mehreren Spotlichtquellen, definiert und erweitern Sie die Callback-Funktion

- `void callback_display(void)`

für Multipass-Rendering.

Durch die Definition einer, aus mehreren Spotlichtquellen bestehenden, räumlichen Lichtquelle und Akkumulation der für die einzelnen Spotlichtquellen gerenderten Bildern lässt sich ein Bild mit weichen Schatten berechnen.
Definieren Sie eine räumliche Lichtquelle, indem Sie in der oben aufgeführten Funktion `buildLightSource` den globalen Array `light_position` durch weitere Lichtquellen ergänzen. In diesem Array sind Einträge für (`nbr_y` * `nbr_z`) Positionen von Lichtquellen vorgesehen, wobei `nbr_y` die Anzahl der Lichtquellen in y-Richtung und `nbr_z` die Anzahl der Lichtquellen in z-Richtung vorgeben.
Setzen sie in der Funktion `callback_display` den Aufruf `displayOnePass(...)` in eine Schleife, in welcher Sie über die verschiedenen Lichtquellen (`0 <= light_nbr < AL_NBR_OF_LIGHTS`) iterieren. Initialisieren Sie vor Eintritt in die Schleife den Accumulation-Buffer. Bilden Sie das arithmetische Mittel über alle Einzelbilder mit Hilfe des Accumulation-Buffers.
Wenn die Akkumulation korrekt funktioniert, können Sie die Anzahl der Lichtquellen anhand der Konstanten `AL_NBR_Y_LIGHTS` und `AL_NBR_Z_LIGHTS` im Header-File erhöhen, um weichere Schattenverläufe zu bekommen. Mit den Konstanten `AL_DELTA_Y_LIGHTS` und `AL_DELTA_Z_LIGHTS` können Sie auch den Abstand zwischen den Lichtquellen in den beiden Achsenrichtungen verändern.

7) Vollbild-Antialiasing

Probieren Sie das bereits im Code eingebaute Vollbild-Antialiasing aus, indem Sie die Präprozessor-Variable `AL_ANTI_ALIASING` auf der Zeile 15 im c-File setzen (durch Entfernen des Kommentarzeichens). Dadurch wird in der Funktion `displayOnePass` an der entsprechenden Stelle der Code vom Präprozessor nicht mehr ausmaskiert.
Nach einer Neukompilierung wird das Bild mit Vollbild-Antialiasing gerendert.