Ludwig-Maximilians-Universität München
LFE Medieninformatik
Prof. Dr. Andreas Butz
Dipl. Inf. Otmar Hilliges

Sommersemester 2008
Übungsblatt 4
15. Mai 2008

3D Programmierpraktikum

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum 29. Mai 2008 abzugeben (14 Tage Bearbeitungszeit).

Inhalt:
Dieses Übungsblatt gibt eine Einführung in die wesentlichen Eigenschaften von Rotationsmatrizen zur Beschreibung dreidimensionaler Orientierung. Außerdem werden Transformationsmatrizen sowie homogene Koordinaten eingeführt.

Aufgabe 9 (H) Rotationsmatrizen

Bewegungen von Objekten im mehrdimensionalen Raum können auf verschiedene Weisen dargestellt werden. Ein Beispiel ist die Darstellung durch Translationen und Rotationen. Rotationen können durch Matrizen beschrieben werden. Ein Punkt $\vec{x}$ im zweidimensionalen Raum kann mit Hilfe einer Matrix um den Winkel α (in mathematisch positiver Drehrichtung) rotiert werden:

$$R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

a) Berechnen Sie den Punkt der durch die Rotation des Punktes $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ um 90° entsteht. Zeigen Sie, dass $R_\alpha \cdot R_\beta \cdot \vec{x} = R_\beta \cdot R_\alpha \cdot \vec{x}$ für beliebige α und β gilt. Warum ist diese Aussage immer gültig?

Jede Rotation im Dreidimensionalen kann intuitiv dargestellt werden indem die Basisvektoren des $\mathbf{R}^3$ auf drei neue Basisvektoren abgebildet werden. In diesem neuen Koordinatensystem wird dann das Objekt dargestellt. Diese lineare Abbildung entspricht genau einer 3×3 Matrix deren Spalten die neuen Basisvektoren sind.

b) Geben Sie die Rotationsmatrix an, die den Punkt $x = (4,0,0)^T$ auf den Punkt $(0,4,0)^T$ abbildet.

c) Kann es Transformationen geben bei denen nur eine Koordinatenachse verändert wird? Falls ja, welche Art von Transformationen sind dies?

Eine Rotation im 3D-Raum um eine beliebige Achse x und einen beliebigen Winkel f, kann nach Euler immer in eine Serie von Rotationen um drei linear unabhängige Achsen zerlegt werden. Hierbei ist ω die Rotation um die x-Achse,

$$R(\omega,0,0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{pmatrix}$$

φ die Rotation um die y-Achse,

$$R(0,\varphi,0) = \begin{pmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{pmatrix}$$

und κ die Rotation um die z-Achse.

$$R(0,0,\kappa) = \begin{pmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

d) Zeigen Sie, dass gilt: $R(\omega,0,0)\cdot R(0,\varphi,0)\cdot R(0,0,\kappa)\cdot\vec{x} \neq R(0,0,\kappa)\cdot R(0,\varphi,0)\cdot R(\omega,0,0)\cdot\vec{x}$. Begründen Sie diese Eigenschaft.

e) Wie lautet die Matrix, die eine solche Rotation rückgängig macht?

Aufgabe 10 (H) Homogene Transformationsmatrizen

Die *Translationsmatrix* $T = (t_x, t_y, t_z^T)$ stellt eine Verschiebung vom Ursprung des Koordinatensystems dar. Die allgemeine Transformationsgleichung

$$p' = R\cdot p + t$$

ist nicht linear und kann daher nicht invertiert werden. Für Szenegraphen ist die Invertierung jedoch zwingend notwendig. Durch Hinzufügen einer weiteren Dimension, d.h.

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \mapsto \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda\cdot p_x \\ \lambda\cdot p_y \\ \lambda\cdot p_z \\ \lambda \end{pmatrix}$$

kann diese Gleichung wieder in eine lineare Abbildung umgewandelt werden:

$$p^* = \begin{pmatrix} p_{x'} \\ p_{y'} \\ p_{z'} \\ 1 \end{pmatrix} = \begin{pmatrix} & & & t_x \\ & R(\omega,\varphi,\kappa) & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$

Diese 4×4 Matrixdarstellung rigider Transformationen im 3D-Raum ist in der Computergraphik sehr gebräuchlich und wird *Homogene Transformationsmatrix* genannt.

Zeigen Sie, dass die o.g. lineare Gleichung genau der Berechnung $p' = R\cdot p + t$ entspricht.

Aufgabe 11 (P) Ein interaktiver 3D-Viewer

In dieser Aufgabe erstellen Sie ein Tool zur Betrachtung dreidimensionaler Szenen. Dazu muss zunächst eine solche Szene erstellt werden. Verwenden Sie hierzu wieder den Code aus Aufgabe 7 für die grafische Ausgabe. Die Szene soll einen Tisch und vier Stühle enthalten, die rund um den Tisch angeordnet sind. Auf die Orientierung der Stühle ist selbstverständlich zu achten.

a) Erstellen Sie eine Klasse *SceneObject*, die die abstrakte Methode `void render()` besitzt. Implementieren Sie zwei weitere Klassen *Table* und *Chair*, die von *SceneObject* erben und die abstrakte Methode überschreiben.

b) Implementieren Sie nun die Zeichnungsfunktionen der Objekte. Ein *Tisch* besteht mindestens aus einer Tischplatte und vier Tischbeinen, weitere Features dürfen selbstverständlich eingebaut werden. Jeder *Stuhl* besteht mindestens aus einer Sitzfläche, vier Stuhlbeinen sowie einer Rückenlehne. Auch hier sollen Ihrer Fantasie keine Grenzen gesetzt werden.

c) Platzieren Sie die Stühle rund um den Tisch. Achten Sie darauf, dass sie richtig orientiert sind, d.h. dass die Rückenlehne an der richtigen Stelle angezeigt wird. Verwenden Sie insbesondere die Funktionen `glPushMatrix()` und `glPopMatrix()`, um das Erstellen übersichtlich und einfach zu halten.

d) Erstellen Sie eine Grundfläche, die nicht aus einem einzigen Viereck besteht, d.h. es sollen viele Vierecke in einer Ebene angeordnet werden. Dies führt zu schöneren Lichteffekten.

e) Die Interaktion in dem 3D-Viewer soll folgende Kriterien erfüllen:

- **Bewegen der Szene:** Durch die Bewegung der Maus bei gedrückter *linker* Maustaste soll die Szene nach links- bzw. rechts (abhängig vom x-Wert) und nach oben bzw. unten (abhängig vom y-Wert) bewegt werden. Bei der Bewegung mit gedrückter *rechter* Maustaste soll die Szene nach vorne bzw. hinten (abhängig vom x-Wert) bewegt werden. Achten Sie darauf, dass es keine Sprünge der Bewegung zu Beginn der Transformation gibt, d.h. speichern Sie die Mausposition an der das erste Mal ein Drücken der Taste erkannt wurde.
- **Drehen der Szene:** Durch die Bewegung der Maus bei gedrückter *linker* Maustaste soll die Szene um die y-Achse (abhängig vom x-Wert) und um die x-Achse (abhängig vom y-Wert) gedreht werden. Bei der Bewegung mit gedrückter *rechter* Maustaste soll die Szene um die z-Achse (abhängig vom x-Wert) gedreht werden. Achten Sie auch hier auf unerwünschte Sprünge. Weiterhin soll ein Bezugspunkt (fest gegenüber der Kameraposition) bei der Rotation vorgegeben werden (z.B. der Punkt (0, 0, -10)).
- **Wechseln des Interaktionsmodus:** Durch Drücken der Taste *T* soll der Interaktionsmodus zwischen *Bewegen* und *Drehen* gewechselt werden.
- **Rücksetzen der Szene:** Durch Drücken der Taste *Enter* soll die Szene in ihren ursprünglichen Zustand zurücktransformiert werden.

f) Geben Sie den Stühlen, dem Tisch und der Grundfläche unterschiedliche Materialien. Dies beinhaltet vor allem die Werte für die ambiente und diffuse Farbe, den Glanz und die Glanzlichtspiegelung.

g) Erstellen Sie ein Licht, das Sie irgendwo in der Szene platzieren. Es ist ein omnidirektionales Licht. Erweitern Sie es um Parameter, wie z.B. *ambiente* und *diffuse Farbe*, *Glanzfarbe*, *Position* und *Abschwächung*. Bei den angegebenen Interaktionen soll dieses Licht *mit* der Szene bewegt werden, d.h. es hat gegenüber allen Szeneobjekten eine feste Position.

h) Erstellen Sie eine weitere Lichtquelle. Geben Sie dieser andere Parameter (v.a. im Bezug auf *Farben* und *Position*). Das Licht ist ein *Spotlight* soll nicht senkrecht über dem Tisch positioniert sein. Lassen Sie es beständig um den Tisch kreisen und erlauben Sie dem Benutzer die Geschwindigkeit zu erhöhen (Pfeiltaste rechts) bzw. zu erniedrigen (Pfeiltaste links). Deuten Sie das Licht mit einer kleinen Drahtgitterkugel (`glutWireSphere(...)`) an, damit der Benutzer die Position des virtuellen Lichts nachvollziehen kann.

i) Richten Sie das *Spotlight* immer genau auf den Tisch aus, so dass der Tisch (auch bei Bewegung des Lichts) eine optimale Ausleuchtung von dieser Lichtquelle erhält. Schalten Sie die Lichtreflexion auf dem Boden für dieses Licht aus, da dies häufig zu unschönen Effekten führt (das globale Licht soll den Boden jedoch ausleuchten).

j) Erzeugen Sie zwei kleine Quadrate, die dem Benutzer in seiner Ansicht oben und horizontal zentriert erscheinen. Eines davon ist *rot* und eines *grün*. Die Farben sind abhängig vom gewählten Interaktionsmodus. Diese Icons bewegen sich bei jeder Transformation durch den Benutzer mit dessen Ansicht mit. Achten Sie darauf, dass sie immer sichtbar sind, unabhängig davon, wie die Szene gerade dargestellt wird.

   Beispiel: Das linke Quadrat erscheint *rot* und das rechte *grün*, wenn der Drehmodus aktiv ist. Ist der Bewegungsmodus aktiv, so ist das linke Quadrat *grün* und das rechte *rot*.

k) **Optional:** Stellen Sie diese Quadrate mit *Alpha-Blending* dar, damit der Benutzer auch hinter diesen die Szene noch erkennen kann.