Technische Universität München
Fakultät für Informatik
Prof. Gudrun Klinker, Ph.D.

Sommersemester 2004
Übungsblatt 2
4. Mai 2004

Praktikum Augmented Reality

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum 11.05.2004 abzugeben.

Aufgabe 5 (P) ARToolkit

Schreiben Sie ein einfaches Programm welches das ARToolkit verwendet; bauen Sie dazu aber nicht auf einem der mitgelieferten Beispielprogramme auf. Verwenden Sie nur Variablen die Sie tatsächlich benötigen, schreiben Sie nur Funktionen die Sie tatsächlich benutzen.

Beginnen Sie dazu mit einer leeren Datei in die Sie immer mehr Code einfügen.

Aufgabe 6 (P) Mehrere Marker mit ARToolkit

In dieser Aufgabe lernen Sie, mehrere unterschiedliche Marker mit dem ARToolkit zu verwenden. Beispiele für verschiedene Marker finden Sie im ARToolkit Verzeichnis unter Patterns, Sie sollen aber Ihre eigenen Marker entwickeln.

a) Erstellen Sie auf Basis der mitgelieferten Marker zwei neue, eigene Marker. Trainieren Sie die neuen Marker mit Hilfe des Programms mk_patt. Eine Anleitung dazu finden Sie im ARToolkit Manual auf der ARToolkit Homepage. Die Muster in den Markern dürfen auch Farbe enthalten. Mehr Information dazu ist in dem Beispielprogramm modeTest enthalten.

b) Ändern Sie ihren Programmtext so ab daß auf die beiden neuen Marker erkannt werden. Ändern Sie gegebenenfalls das Aussehen Ihrer Marker, wenn sie schlecht erkannt oder schlecht unterschieden werden.

c) Machen Sie sich mit der `glut`-Funktion `glutKeyboardFunc` vertraut, die vom ARToolkit in der Datei `lib/SRC/Gl/gsub.c` verwendet wird. Sorgen Sie dafür, dass auf den in der vorherigen Teilaufgabe erkannten Markern zwei verschiedene Objekte angezeigt werden, die sich auf Tastendruck (z.B. der Leertaste) individuell verändern lassen.

Aufgabe 7 (P) Translationen von Objekten

Modifizieren Sie das soeben geschriebene Programm derart, dass Sie die auf den Markern gerenderten Objekte mit den Cursor-Tasten relativ zum Marker verschieben können. Hierbei soll es möglich sein, durch Drücken der Taste „`R`“ die Translation zurückzusetzen.

Machen Sie sich zum Lösen dieser Aufgabe zuerst mit den im ARToolkit verwendeten Koordinatensystemen vertraut, eine gute Erklärung finden Sie in der auf der Praktikumshomepage hinterlegten Präsentation von Hirokazu Kato. Die eigentliche Translation von Objekten geschieht durch Veränderung der OpenGL-Projektionsmatrix mittels `glTranslate3f` an geeigneter Stelle.

Aufgabe 8 (H) Quaternionen

Ähnlich wie komplexe Zahlen zur Darstellung von Rotationen im zweidimensionalen benutzt werden, können die von *William Rowan Hamilton* 1843 entdeckten *Quaternionen* zur Darstellung von Rotationen im dreidimensionalen Raum verwendet werden. In dieser Aufgabe sollen Sie ein paar der wesentlichen Eigenschaften von Quaternionen zeigen.

Quaternionen sind hyperkomplexe Zahl vom Rang 4 und lassen sich wie folgt darstellen:

$$q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3 = q_0 + \mathbf{q}$$

wobei für $\mathbf{i}, \mathbf{j}, \mathbf{k}$ gilt:

$$\begin{aligned} \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} &= -1 \\ \mathbf{ij} = \mathbf{k} &= -\mathbf{ji} \\ \mathbf{jk} = \mathbf{i} &= -\mathbf{kj} \\ \mathbf{ki} = \mathbf{j} &= -\mathbf{ik} \end{aligned}$$

Die *Addition* zweier Quaternionen ist als Addition der entsprechenden vier Komponenten definiert; zwei Quaternionen sind genau dann gleich, wenn alle vier Komponenten gleich sind.

Die Darstellung $q = q_0 + \mathbf{q}$ eines Quaternions kann auch als Kombination eines Skalars mit einem dreidimensionalen Vektor interpretiert werden. Das Quaternion $q^* = q_0 - \mathbf{q}$ eines Quaternions $q = q_0 + \mathbf{q}$ wird *komplex konjugierte von q* genannt.

Die Norm $N(q)$ eines Quaternions ist wie folgt definiert:

$$\begin{aligned} N^2(q) &= q^* q \\ &= q_0^2 + q_1^2 + q_2^2 + q_3^2 = |q|^2 \end{aligned}$$

Ein *Einheitsquaternion* ist ein Quaternion mit der Norm $N(q) = 1$.

a) Zeigen Sie: Die Multiplikation bei Quaternionen ist nicht kommutativ. Berechnen Sie hierzu auf der Basis der obigen Gleichungen das Produkt $r = pq$ zweier Quaternionen $q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$ und $p = p_0 + \mathbf{i}p_1 + \mathbf{j}p_2 + \mathbf{k}p_3$.

b) Zeigen Sie: Das Einselement der Quaternionenmultiplikation ist $q = 1 + \mathbf{0}$.

c) Zeigen Sie: Das inverse Element der Quaternionenmultiplikation ist $q^{-1} = q^*/N^2(q)$.

d) Zeigen Sie: $(pq)^* = q^* p^*$

Aufgabe 9 (P,H) Rotationen mit Quaternionen

Man kann zeigen, dass für jedes Einheitsquaternion $q = q_0 + \mathbf{q}$ die Operation $\mathbf{v}' = q\mathbf{v}q^*$ einen in homogenen Koordinaten dargestellten Punkt $\mathbf{v}$ im dreidimensionalen Raum auf einen Punkt $\mathbf{v}'$ derart abbildet, dass dies einer Drehung im Raum um die Achse $\mathbf{q}$ um den Winkel $2\text{acos}(q_0)$ entspricht.

a) Berechnen Sie den Rotationsoperator $L_q(\mathbf{v}) = q\mathbf{v}q^*$ in Matrixdarstellung (d.h. als 4×4 Matrix R_q mit $\mathbf{v}' = R_q\mathbf{v}$).

b) Zeigen Sie: Das Produkt pq der Quaternionen p und q definiert einen Rotationsoperator L_{pq} der einer Sequenz von Rotationsoperatoren, L_p gefolgt von L_q entspricht.

c) Implementieren Sie in C eine Funktion `quaternionToMatrix`, die ein Einheitsquaternion in eine homogene Rotationsmatrix wandelt. Testen Sie Ihre Funktion mitttels des auf der Praktikumshomepage zur Verfügung gestellten Programms.

Aufgabe 10 (H) Verwendung von CVS

CVS ist das meistverwendete System zur Versionsverwaltung bei Gruppenprojekten. Im Abschlussprojekt, das Sie in wenigen Wochen machen werden, muss CVS verwendet werden. Um mit diesem System vertraut zu werden, bitten wir Sie, sich schon jetzt in CVS einzuarbeiten.

a) Benutzen Sie, wie auf der Praktikumshomepage erklärt, ein privates CVS-Repository zur Verwaltung Ihrer Lösungen.

b) Checken Sie jede von der Praktikumsleitung akzeptierte Lösung in das Repository mit dem CVSROOT `:ext:<login>@cvsbruegge.in.tum.de:/cvs/arprakt` ein. Benutzen Sie das Modul `loesungen` und legen Sie sich darin ein Verzeichnis mit ihrem Login als Namen an.