ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics lab

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

Zu lösen bis: 2. Mai 2006

Visual Computing
Transformationen mittels Matrizen und Quaternionen

Ziele

Theorie: Vertiefung der Vorlesung im Bereich Transformationen und Projektionen.
Praxis: Implementierung eines virtuellen Trackballs mittels Quaternionen.

Allgemeine Hinweise

Die theoretische Aufgabe sollte einzeln gelöst werden. Im Unterschied zur letzten Teilaufgabe der praktischen Aufgabe sollte sie wie in einer Prüfung ohne Hilfsmittel gelöst werden können. Sie muss nicht abgegeben werden. Zu dieser Aufgabe wird auch eine Musterlösung ausgehändigt.

Die praktische Aufgabe kann in Zweieteams gelöst und abgegeben werden. Das Codegerüst kann wie üblich direkt von der Webseite geladen, in einem persönlichen Verzeichnis gespeichert und mit der Visual Studio .NET Programmierumgebung bearbeitet werden.

Ressourcen

Webseite der Vorlesung.

1) Theoretische Aufgabe (aus GDV-Prüfung 2005)

a) Kurzaufgaben (Repetition aus der Vorlesung)

I. Zwischen welchen drei Koordinatensystemen unterscheidet man bei der Szenenbeschreibung?

II. In welche zwei Klassen können planare geometrische Projektionen eingeteilt werden?

III. Worin unterscheiden sich oblique Projektionen von orthographischen Projektionen?

IV. Nennen Sie einen Vorteil, der sich durch den Einsatz homogener Koordinaten ergibt.

V. Gegeben seien zwei homogene Punkte $\mathrm{P}_1 = [4\ 3\ 2\ 1]^{\mathrm{T}}$ und $\mathrm{P}_2 = [1\ 2\ 3\ 4]^{\mathrm{T}}$.

Geben Sie den euklidschen Verbindungsvektor $\mathbf{d} \in \mathfrak{R}^3$ an, der von P_1 nach P_2 führt.

b) Homogene Transformationen

I. Zerlegen Sie die Abbildungsmatrix

$$A = \begin{bmatrix} 0 & -1 & 0 & 47 \\ 1 & 0 & 0 & 11 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

in eine lineare Abbildung L und eine Translation T, so dass $A = LT$. (Beachte: nicht $A = TL$!)

II. Geben Sie jeweils eine anschauliche Interpretation für die Abbildungen L und T an. Welche Art lineare Abbildung beschreibt L ?

III. In welcher Reihenfolge werden L und T ausgeführt, wenn A auf einen Punkt angewandt wird?

IV. Es sei $n = [a\ b\ c\ d]^T$ die Normale einer Ebene H in Hesse-Normalform. Für alle Punkte $\mathbf{p} \in H$ gilt also $\mathbf{n} \bullet \mathbf{p} = 0$. Die obige Abbildung A bilde nun alle $\mathbf{p} \in H$ auf Punkte $\mathbf{p}'$ einer zweiten Ebene H' ab ($\mathbf{p}' = A\mathbf{p}$).
Geben Sie eine Matrix $\tilde{A}$ an, die $\mathbf{n}$ auf die Normale $\mathbf{n}'$ der Ebene H' in Hesse-Normalform abbildet ($\mathbf{n}' = \tilde{A}\mathbf{n}$). Berechnen Sie $\tilde{A}$ mit Hilfe der Zerlegung $A = LT$!

c) Quaternionen

I. Stellen Sie das Einheitsquaternion $q = c + xi + yj + zk$ auf, das eine Rotation von 60° um die Achse $[3\ 0\ 4]^T$ beschreibt. Vereinfachen Sie das Quaternion so weit wie möglich. Sie können folgende Wertetabelle benutzen:

φ	$\cos\varphi$	$\sin\varphi$
30°	$\sqrt{3}/2$	1/2
60°	1/2	$\sqrt{3}/2$
120°	-1/2	$\sqrt{3}/2$

II. Berechnen Sie direkt aus q das zugehörige Quaternion der inversen Rotation.

III. Welche elementare Quaternionenoperation haben Sie in II. benutzt?

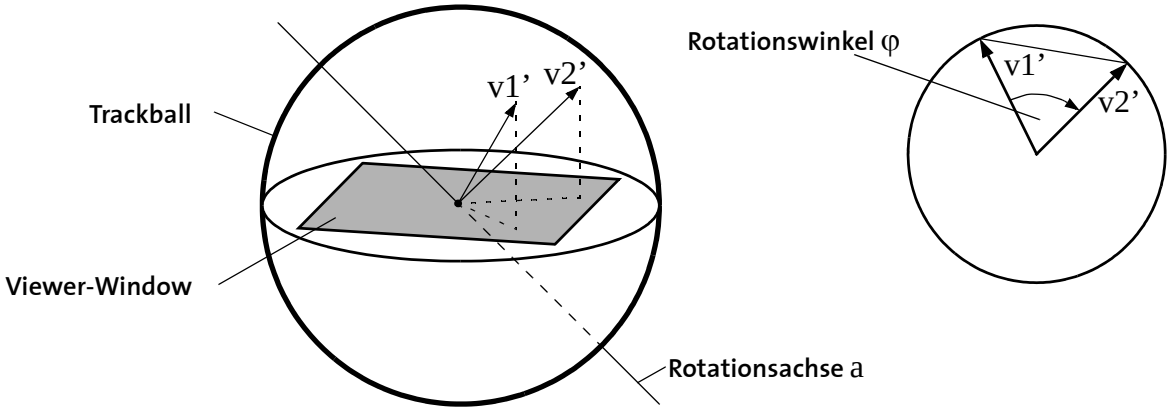
IV. Gegeben sei das Quaternion $r = [c\ x\ y\ z] = [0\ 1\ 0\ 0]$. Mit welcher Rotation korrespondiert r ?

V. Geben Sie die mit r korrespondierende Rotationsmatrix an.

2) Praktische Aufgabe: Trackball

Grundlagen

Um ein an einem Bildschirm zu betrachtendes Objekt interaktiv zu rotieren, eignet sich das Modell eines Trackballs sehr gut. Daher soll in dieser Aufgabe ein sogenannt *virtueller Trackball* realisiert werden, indem die Bewegungen einer Maus entsprechend in Rotationen des Objekts umgewandelt werden. Wie in der Folge noch erklärt, müssen dazu die zweidimensionalen Mauskoordinaten in geeigneter Form auf einen virtuellen Trackball projiziert werden.
Das allgemeine Vorgehen für die Realisierung des Trackballs kann wie folgt beschrieben werden: Aus einer Mausbewegung im Viewer-Window – respektive aus einem linearen Teilstück einer kontinuierlichen Bewegung – muss die Rotationsachse $\mathbf{a}$ sowie der Rotationswinkel φ einer virtuellen Kugel ermittelt werden. Die Rotationsachse ergibt sich aus dem Kreuzprodukt der beiden Vektoren $\mathbf{v1'}$ und $\mathbf{v2'}$, die sich wiederum aus der Projektion der Koordinaten v1 und v2 im Viewer-Window auf den virtuellen Trackball ergeben. Der Rotationswinkel φ kann mittels einfacher trigonometrischer Zusammenhänge berechnet werden.



Aus der Rotationsachse sowie dem Rotationswinkel kann ein Quaternion gebildet werden, welches die Rotation vollständig beschreibt. Für jedes Teilstück einer Bewegung kann ein neues Quaternion berechnet werden, um die aktuelle Rotation zu codieren. Die einzelnen Quaternionen müssen aufmultipliziert werden, damit eine längere Bewegung simuliert werden kann, ohne dass sich numerische Fehler akkumulieren.
Um das rotierte Objekt im Viewer anzuzeigen, muss an OpenGL eine Rotationsmatrix übergeben werden. Dies bedeutet, dass das berechnete Quaternion in eine Rotationsmatrix umgewandelt werden muss.
Ein Quaternion wird in dieser Aufgabe als Vektor `quat[4]` vom Typ `float` behandelt. Dabei haben die einzelnen Koeffizienten die folgende Bedeutung: `q[0]=x; q[1]=y; q[2]=z` und `q[3]=c`. Im Gegensatz zu den Vorlesungsunterlagen und allen mathematischen Abhandlungen **liegt der Realteil eines Quaternions also an der letzten Stelle**. Dies hat den Vorteil, dass alle Operationen auf dem Imaginärteil mittels Vektoroperationen direkt auf `q` durchgeführt werden können, ohne dass Addressarithmetik notwendig wird.

Anweisungen

Die Aufgabe ist in die 3 Teile *Verständnis*, *Implementierung* und *Herleitung* gegliedert.

a) Verständnis des vorgegebenen Codes

Der vorgegebene Code besteht aus den folgenden 3 Modulen:

- GLM: Wavefront .obj file format manipulator
- TB: Trackball module
- Viewer: Display und event handling functions

Erklären Sie in wenigen Worten, wie und warum die Animation der Rotation (kontinuierliches Drehen des Objektes), wie sie in der Vorbesprechung vorgeführt wurde, funktioniert. Erwähnen Sie insbesondere die Mechanismen zum Starten und Stoppen der Animation.

b) Implementierung

Implementieren oder vervollständigen Sie die folgenden Funktionen und Callbacks:

- Funktion `void callback_display(void)`
- Funktion `void _TB_VEC_project(..)`
- Funktion `void _TB_QUAT_norm(..)`
- Funktion `void _TB_QUAT_mult(..)`
- Funktion `void _TB_QUAT_buildQuat(..)`

Die erste dieser Funktionen finden Sie in der Datei `viewer.c`, die restlichen in `trackball.c`. Was in der jeweiligen Funktion zu tun ist, können Sie direkt den Kommentaren im Code entnehmen. Die für die Implementierung benötigten mathematischen Formeln sind nachfolgend gegeben.

- Multiplikation zweier Quaternionen: $qq' = (cc' - u \bullet u') + (u \times u' + \langle cu' \rangle + \langle c'u \rangle)$
  (vgl. Gleichung 3.30 und Regeln 3.31 im Skript)

- Berechnung des Quaternions q aus der Rotationsachse a mit $|a| = 1$ und dem

  Rotationswinkel φ: $q = \left[a_1 \cdot \sin\left(\frac{\varphi}{2}\right), a_2 \cdot \sin\left(\frac{\varphi}{2}\right), a_3 \cdot \sin\left(\frac{\varphi}{2}\right), \cos\left(\frac{\varphi}{2}\right) \right]$

  (vgl. Gleichung 3.46 im Skript sowie die allgemeinen Hinweise auf dem Aufgabenblatt)

c) Herleitung der Rotationsmatrix

Die Funktion `void _TB_QUAT_buildMatrix(GLfloat quat[4])` in der Datei `trackball.c` generiert aus einem Quaternion die entsprechende Rotationsmatrix, die zur Übergabe an die OpenGL benötigt wird – sie ist bereits vollständig vorgegeben. Das Ziel dieser Aufgabe ist die Herleitung der Einträge dieser Matrix.
Gegeben sei also ein Quaternion q. Berechnen Sie $p' = qpq$ für einen allgemeinen, zu rotierenden Punkt p (vgl. Gleichung 3.46 im Skript). Beachten Sie, dass die Rotationsmatrix in homogenen Koordinaten angegeben werden muss.
Für die symbolischen Berechnungen kann zum Beispiel Maple verwendet werden. Nachfolgend finden Sie ein Gerüst sowie eine Liste der weiteren, benötigten Funktionen.

Die folgenden, in `Maple` eingebauten Funktionen werden benötigt:

• Kreuzprodukt	`crossprod(..);`
• Multiplikation mit einem Skalar	`scalarmul(..);`
• Skalarprodukt	`dotprod(..);`
• Auswerten eines Matrix-Ausdruckes	`evalm(..);`
• Extrahieren von Koeffizienten aus einem Ausdruck	`collect(..);`

Codegerüst:

```
> restart;
  with(linalg):
> qpq := proc(q,p)
    local c,u,v,tmp,res;
    res := vector(4,0);
    c: = ...
    u := vector(3, ...);
    v := vector(3, ...);
    ... #calculation according to formula
        #3.44 in lecture notes
    RETURN(res);
  end:
> #vector to be transformed:
  p := vector(4, [...]);
> #quaternion:
  q := ...;
> #do calculation:
  temp := evalm(qpq(q,p));
> #get the entries using 'collect(temp[..], ..);
  ...
```