HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
Fakultät für Informatik
Labor für Computergrafik
Prof. Dr. G. Umlauf

Konstanz, 08.03.2016

Aufgabe 1 & 2 & 3

„Geometrisches Modellieren“

Besprechung und Abgabe spätestens am 20.04.2016, F031/F033.

Vorbemerkung:
Benutzen Sie für diese Übung **keine** OpenGL, GLUT oder GLAUX Funktionen, die Projektionen und Rotationen berechnen! Benutzen Sie die zur Verfügung gestellten Vektor und Matrix Klassen.

Programmgerüst für die Aufgabe:
Laden Sie sich die zip-Datei zur Übung von der web-Seite der Vorlesung:

- Darin enthalten ist ein VC-Projekt Version 2010.
- Darin enthalten sind weiter:
  - `main.cpp:`
    Programmgerüst, das die Benutzung einer Display-Funktion (zum Refresh des Doublebuffer) und eine Keyboard-Funktion zur Steuerung enthält.
  - `color.h, vec.h, mat.h:`
    Implementierungen einer Farb-, Vektor- und Matrixklasse. Einige Funktionen sind bereits exemplarisch implementiert. Implementieren Sie bei Bedarf die fehlenden Funktionen. Orientieren Sie sich dabei an dem existierenden Code.
  - `viewSystem.h, viewSystem.cpp:`
    Implementierungen des Sichtsystems, bestehend aus Augpunkt (`EyePoint`), Blickrichtung (`ViewDir`) und Bildebene (`ViewUp, ViewHor`) in homogenen Koordinaten. Das Sichtsystem übernimmt die 3d-2d-Projektione, globale Koordinatensystem-Transformationen sowie affine Abbildungen. Sie verfügt über drei Modi zur Auswahl, welche Implementierung der affinen Abbildungen verwendet werden soll:
    - `VIEW_MATRIX_MODE, VIEW_FORMULA_MODE:` Implementierung mit 4x4 Matrizen (funktionsfähig).
    - `VIEW_QUATERNION_MODE:` Implementierung mit Quaternionen (siehe Aufgabe 02).
  - `quader.h, quader.cpp:`
    Implementierungen einer Klasse zur Repräsentation von Quader-Objekten.
  - `quaternion.h, quaternion.cpp:`
    Implementierungen einer Klasse zur Repräsentation und Verwendung von Quaternionen.

Die Funktionalität aus beiden Aufgaben wird anhand des Source-Codes überprüft!

HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
Fakultät für Informatik
Labor für Computergrafik
Prof. Dr. G. Umlauf

Aufgabe 01 (Allgemeine Orientierung berechnen)

In dem Programmgerüst kann die Szene nur aus der Ansicht des Sichtsystem (`ViewSystem`) betrachtet werden, das beliebig im Raum positioniert sein kann. Gegeben sei also das Sicht-Koordinatensystem:

- Eine allgemeine Position des Augpunktes (in homogenen Koordinaten) `EyePoint`.
- Eine allgemeine Sichtrichtung `ViewDir`.
- Ein allgemeiner View-Up-Vektor `ViewUp`.
- Die zusätzliche (implizierte) Richtung `ViewHor = ViewDir x ViewUp`.

Implementieren im Sie in `viewSystem.cpp` die Methoden

```
CMat4f getTransform?(),
```

die die Lage des Sicht-Koordinatensystem als 4×4-Matrix zur Transformation in das Welt-Koordinatensystem berechnet (siehe Abbildung 1). Diese führt auch auf eine Matrix, die Punkte aus dem Welt-Koordinatensystem in das Sicht-Koordinatensystem überführt.

Wenn Sie diese Methoden implementiert haben, sollte die Szene mit den Tastenkürzeln aus Aufgabe02 im `VIEW_FORMULA_MODE` (in `main.cpp` setzen) manipulierbar sein.

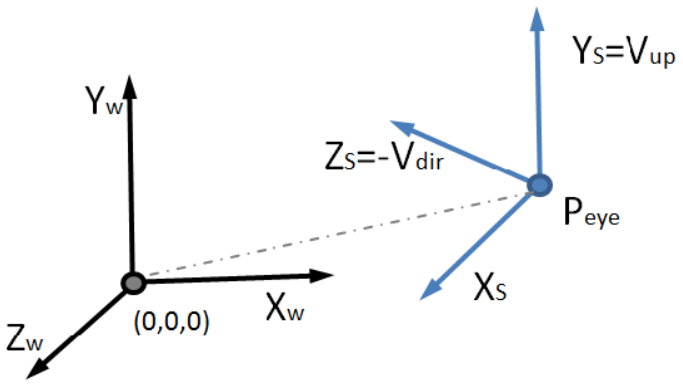

Abbildung 1 Allgemeine Sichttransformation.

HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
Fakultät für Informatik
Labor für Computergrafik
Prof. Dr. G. Umlauf

Aufgabe 02 (Quaternionen)

Schreiben Sie nun ein Programm so um, dass es im `VIEW_QUATERNION_MODE` verwendet werden kann. Erstellen Sie dazu für die Keyboard-Interaktion die fehlenden Methoden und Operationen im `viewSystem.cpp` und `quaternion.cpp`:

a. X, Y und Z rotieren das Sicht-Koordinatensystem in positiver Richtung um die x-, y- und z-Achse des Welt-Koordinatensystems und x, y und z drehen in negativer Richtung um die jeweilige Achse.
b. A, B und C bzw. a, b und c rotieren das Sicht-Koordinatensystem in entsprechender Richtung um die jeweilige Achse des Sicht-Koordinatensystems: A, a: ViewDir-Vektor, B, b: ViewUp-Vektor, B, b: ViewHor-Vektor.
c. U, V, W, u, v und w verschieben das Sicht-Koordinatensystem entlang der Achsen des Welt-Koordinatensystems in die entsprechenden Richtungen: U, u: x-Achse, V, v: y-Achse, W, w: z-Achse (nicht Teil der Aufgabe).
d. R setzt das Sicht-Koordinatensystem in den Anfangszustand (deckungsgleich mit dem Welt-Koordinatensystem) zurück (nicht Teil der Aufgabe).
e. f und F ändert den Fokalabstand des Sichtsystems (nicht Teil der Aufgabe).

Zum Zeitpunkt der Initialisierung der Applikation sollen Welt-Koordinatensystem und Sicht-Koordinatensystem deckungsgleich sein.

Aufgabe 03 (Rotationsinterpolation)

Implementieren Sie in Ihrem Programm drei Methoden zur Interpolation von Rotationen. Wählen Sie dazu eine Start- und eine Endposition Ihrer Szene und interpolieren Sie zwischen diesen beiden Positionen entsprechende Zwischenpositionen mit den folgenden drei Methoden, siehe [Shoemaker 1985] und [Kremer 2008].

a. Lineare Interpolation (LERP) zwischen den beiden Positionen.
b. Spherical linear interpolation (SLERP) zwischen den beiden Positionen.
c. Normalized SLERPs (NLERP) zwischen beiden Positionen.

Besprechung und Abgabe spätestens am 20.04.2016, F031/F033.