Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Axel Schaffland, M.Sc.

UNIVERSITÄT OSNABRÜCK

Computergrafik (SS 2018)

Übung 2

fällig Montag **23. April**, 14:20

- Geben Sie bei jeder Abgabe alle Ihre Matrikelnummern auf jedem Blatt an.
- Verspätet eingereichte Abgaben können nicht gewertet werden.

Aufgabe 1 (Theorie: Transformation und Projektion von 3D Punkten)

(a) (4P) Projezieren und dehomogenisieren Sie die acht Eckpunkte des Würfels, welcher die Kantenlänge 2 besitzt und dessen Zentrum bei $(0, -1, -3, 1)$ liegt, mittels der perspektivischen Standardprojektion P_{std}. Zeichnen Sie die Ergebnisse (genauer: die x- und y-Koordinaten der Ergebnisse) in ein 2D-Koordinatensystem und verbinden Sie die Punkte (den Würfelkanten entsprechend), so dass sich ein perspektivisches Abbild des Würfels ergibt.

(b) (2P) Bestimmen Sie die LookAt-Matrix für eine Kamera an Position $(1, 2, 3)$, welche in Richtung $(\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}})$ blickt und den up-Vektor $(0, 1, 0)$ besitzt.

(c) (2P) Berechnen Sie **mittels Skalarprodukt und/oder Kreuzprodukt**:

- die Länge des Vektors $(4, 0, 3)$;
- den Winkel zwischen den Vektoren $(2, 0, 0)$ und $(3, 3, 0)$;
- einen Vektor senkrecht sowohl zu $(2, 3, 1)$ als auch zu $(0, 4, 2)$;
- einen Vektor $\vec{v}$, so dass $((3, 1, 0) - \vec{v})$ senkrecht zu $\vec{v}$ ist.

Aufgabe 2 (Praxis: Transformation und Projektion von 3D Punkten)

(a) (2P) Ergänzen Sie den Code für Übungsblatt 2 um eine Funktion, die ein Array von 3D Punkten erzeugt.

Die Funktion soll mindestens acht Punkte zurückgeben. Diese können eine beliebige nicht-ebene Figur, beispielsweise die Ecken eines Würfels, bilden, und sollen in allen Dimensionen auf den Wertebereich von -1 bis $+1$ beschränkt sein. Schreiben Sie die Punkte in die Variable `pointarray`.

Zur Verwaltung der Punkte wird Ihnen die `Point3D`-Klasse zu Verfügung gestellt. Zur Dehomogenisierung eines Punktes besitzt jene die Funktion `dehomogen()`.

(b) (5P) Ergänzen Sie den Code für Übungsblatt 2 um eine Funktion, die ein Array von 3D Punkten rotiert.

Zur Verwaltung der Matrizen wird Ihnen die Klasse `Matrix4D` zur Verfügung gestellt. Diese stellt auch Funktionen zur Vektormultiplikation `mulVec` und Matrixmultiplikation `mulMat` bereit.

Die über die Slider eingestellten Rotationswinkel stehen Ihnen **in Grad** in den Variablen `rotX`, `rotY` und `rotZ` zur Verfügung. Berechnen Sie daraus eine Transformationsmatrix, die alle drei Rotationen kombiniert.

Berechnen Sie zudem eine LookAt-Matrix: Die Kamera-Position soll am Punkt $(0, 0, \texttt{camZ}, 1)$ liegen – `camZ` lässt sich über einen Slider variieren. right-Vektor (1,0,0,0), up-Vektor (0,1,0,0), und direction-Vektor (0,0,-1,0). Das heißt die Kamera liegt in der z-Achse und schaut in Richtung $-z$.

Sorgen Sie dafür, dass die kombinierten Rotationen und die LookAt-Matrix auf alle Punkte angewendet werden.

Hinweis: Sie können die Größe des Canvas mit der Konstanten `size` anpassen, um ggf. das Rendering zu beschleunigen oder eine größere Anzeige zu erreichen.

(c) (5P) Ergänzen Sie den Code von Übungsblatt 2 um eine Funktion, die die transformierten 3D Punkte in eine 2D Bildebene projiziert.

Die Matrix `projMat` ist vorinitialisiert mit einer Parallelprojektion. Dadurch fehlt perspektivische Verkürzung. Zudem ist die Projektion nur ca. 2×2 groß (da die Punkte auf den Wertebereich von -1 bis $+1$ beschränkt waren), wohingegen das Canvas deutlich größer ist (`size`$\times$`size`).

Ersetzen sie `projMat` daher durch eine perspektivische Projektion, mit Projektionszentrum im Ursprung und Bildebene $z = -1$, gefolgt von einer Skalierung der x- und y-Koordinaten mit dem Faktor `size`$/3$ (die z-Koordinate wird im Folgenden nicht mehr benötigt und kann daher unverändert gelassen werden).

Wenden Sie dann `projMat` auf alle Punkte des `pointarray` an, und schreiben Sie die **dehomogenisierten** Ergebnisse in das Array `pointsProj`.

Aufgabe 3 (Bonus: Virtual Trackball Metapher)

Ergänzen Sie den Code um eine Funktion, die die Rotation der Punktmenge mittels Maus ermöglicht. Dazu soll die Virtual Trackball Metapher verwendet werden. Sie können sich dabei an der Beschreibung unter `http://web.cse.ohio-state.edu/~shen.94/781/Site/Slides_files/trackball.pdf` orientieren.

Die vorherigen Mauskoordinaten werden Ihnen in den Variablen `x1` und `y1`, die aktuellen Mauskoordinaten in den Variablen `x2` und `y2` zu Verfügung gestellt. Daraus können Sie die Rotationsachse und den Rotationswinkel berechnen. Ein geeigneter Radius für den Virtual Trackball ist etwa `size`$/4$, doch Sie können mit verschiedenen Radien experimentieren.

Modifizieren Sie Ihre Lösung von Aufgabe 2b derart, dass die in dieser Bonusaufgabe berechnete Mausrotationsmatrix *zusätzlich* für die Transformation der Punkte verwendet wird.