Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Steffen Hinderink

UNIVERSITÄT OSNABRÜCK

Computergrafik (SoSe 2023)

Blatt 2

fällig Sonntag **7. Mai**, 24:00

Verspätet eingereichte Abgaben können nicht berücksichtigt werden.

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

(a) (5P) Projizieren und dehomogenisieren Sie die acht Eckpunkte des Würfels, welcher die Kantenlänge 8 besitzt und dessen Zentrum bei $(4, 2, -6, 1)^{\mathsf{T}}$ 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) (3P) Bestimmen Sie die LookAt-Matrix für eine Kamera an Position $(3, 0, 2)^{\mathsf{T}}$, welche in Richtung $(-\frac{4}{5}, 0, \frac{3}{5})^{\mathsf{T}}$ blickt und den *up*-Vektor $(0, 1, 0)^{\mathsf{T}}$ besitzt.

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

- die Länge des Vektors $\vec{v} = (-5, 4, -20)^{\mathsf{T}}$.
- den Winkel zwischen den Vektoren $\vec{v} = (0, 8, 8)^{\mathsf{T}}$ und $\vec{w} = (0, 0, -2)^{\mathsf{T}}$.
- einen Vektor senkrecht sowohl zu $\vec{v} = (6, 36, -4)^{\mathsf{T}}$ als auch zu $\vec{w} = (1, 1, 1)^{\mathsf{T}}$
- einen Vektor $\vec{v}$, so dass $\vec{w} = ((-1, 8, 7) - \vec{v})^{\mathsf{T}}$ senkrecht zu $\vec{v}$ ist.

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

(a) (1P) Ergänzen Sie die Funktion `createPoints`, 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.

Zur Verwaltung der Punkte steht Ihnen die Klasse `Point` zur Verfügung. Zur Dehomogenisierung eines Punktes besitzt jene die Funktion `dehomogen`.

(b) (4P) Ergänzen Sie den ersten Teil der Funktion `transform`. In diesem Teil soll die ModelView-Matrix aus Rotationsmatrizen und einer LookAt-Matrix berechnet werden.

Für die Rotationsmatrizen stehen Ihnen die über die Slider eingestellten Rotationswinkel **in Grad** in den Variablen `rotX`, `rotY` und `rotZ` zur Verfügung.

Die LookAt-Matrix soll bewirken, dass die Kamera-Position am Punkt $(0, 0, \texttt{camZ})$ liegt. `camZ` lässt sich über einen Slider variieren. Der *right*-Vektor soll $(1, 0, 0)$, der *up*-Vektor $(0, 1, 0)$ und der *direction*-Vektor $(0, 0, -1)$ sein. Das heißt die Kamera liegt in der z-Achse und schaut in Richtung $-z$.

Zur Verwaltung der Matrizen steht die Klasse `Matrix` zur Verfügung.

(c) (5P) Ergänzen Sie den zweiten Teil der Funktion `transform`. In diesem Teil soll eine Projektionsmatrix berechnet werden und diese mit den Matrizen aus Aufgabe b auf alle Punkte angewendet werden, die im Parameter `points` übergeben werden.

Die Matrix `projectionMat` ist vorinitialisiert mit einer Parallelprojektion. Dadurch fehlt die perspektivische Verkürzung. Zudem ist die Projektion nur 2×2 groß, da die Punkte auf den Wertebereich von -1 bis $+1$ beschränkt waren, wohingegen das Canvas deutlich größer ist (500×500).

Ersetzen sie `projectionMat` 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 250. Die z-Koordinate wird im Folgenden nicht mehr benötigt und kann daher unverändert gelassen werden.

Wenden Sie dann alle Transformationen (Rotationen, LookAt und Projektion) vereint auf alle Punkte an und geben Sie die **dehomogenisierten** Ergebnisse zurück.

Aufgabe 3 (Bonus: Virtueller Trackball)

Ergänzen Sie den Code, so dass eine Rotation der Punktmenge mit der Maus möglich ist. Dazu soll ein virtueller Trackball verwendet werden. In der Übung 2 und den dazugehörigen Folien wird dieser erklärt.

Die vorherigen und aktuellen Mauskoordinaten stehen Ihnen in den Variablen `x0` und `y0`, sowie `x1` und `y1` zur Verfügung. Daraus können Sie die Rotationsachse n und den Rotationswinkel α berechnen. Dafür kann die Klasse `Vector` hilfreich sein. Ein geeigneter Radius für den virtuellen Trackball ist $500/\texttt{camZ}$, dadurch können die Punkte unabhängig von ihrer Entfernung zur Kamera rotiert werden. Modifizieren Sie Ihre Lösung von Aufgabe 2b derart, dass die in dieser Bonusaufgabe berechnete Mausrotationsmatrix *zusätzlich* für die Transformation der Punkte verwendet wird.