Computergrafik I
Kapitel 4: Geometrische Transformationen

Wintersemester 2014/2015

Prof. Dr. Timo Ropinski
Forschungsgruppe Visual Computing

Übersicht

4.1 Mathematische Grundlagen

Vektor und Matrixrechnung

Vektorrepräsentation 1/2

- Vektoren werden zur Darstellung von geometrischen Objekten (Koordinaten) verwendet
- Vektoren haben reelle Zahlen als Komponenten, sind also Elemente aus $\mathbb{R}^2, \mathbb{R}^3$ oder $\mathbb{R}^4$
- (Eck-)Punkte eines Polygons werden als Spaltenvektoren dargestellt

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \qquad \vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \qquad \vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ v_w \end{pmatrix}$$

Vektorrepräsentation 2/2

- Längenberechnung eines Vektors:

$$|\vec{v}| = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2}$$

- Vektoren mit der Länge $|\vec{v}| = 1$ nennt man Einheitsvektoren oder normalisierte Vektoren
- Normalisieren von Vektoren:

$$\vec{n} = \frac{1}{|\vec{v}|}\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

Skalarprodukt

- Skalarprodukt (*dot product*)
  - Gegeben: Vektoren $\vec{u}$ und $\vec{v}$ im $\mathbb{R}^n$
  - Skalarprodukt ist eine reell wertige Funktion:

$$\vec{u} \cdot \vec{v} = u_x \cdot v_x + v_y \cdot v_y + u_z \cdot v_z + \cdots$$

- Geometrische Interpretation
  - Winkel α zwischen zwei Vektoren $\vec{u}$ und $\vec{v}$:

$$\cos\alpha = \frac{\vec{u} \cdot \vec{v}}{\overrightarrow{|u|} \cdot \overrightarrow{|v|}}$$

  - Winkel α zwischen zwei normalisierten Vektoren $\vec{u}$ und $\vec{v}$:

$$\cos\alpha = \vec{u} \cdot \vec{v}$$

Kreuzprodukt

- Kreuzprodukt (*cross product*, *vector product*)
  - Gegeben zwei Vektoren $\vec{u}$ und $\vec{v}$ im $\mathbb{R}^3$
  - Kreuzprodukt $\vec{u} \times \vec{v}$ von $\vec{u}$ und $\vec{v}$ ist ein Vektor:
- Geometrische Interpretation
  - $\vec{u} \times \vec{v}$ steht senkrecht auf der durch $\vec{u}$ und $\vec{v}$ aufgespannten Ebene
  - Die Länge von $u \times v$ ist gleich dem Flächeninhalt des von $\vec{u}$ und $\vec{v}$ aufgespannten Parallelogramms

$$\vec{u} \times \vec{v} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \times \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} u_y \cdot v_z - u_z \cdot v_y \\ u_z \cdot v_x - u_x \cdot v_z \\ u_x \cdot v_y - u_y \cdot v_x \end{pmatrix}$$

Matrizen

- Einsatz: Beschreibung von Transformationen
- Hauptsächlich quadratische Matrizen 2×2, 3×3, 4×4
- Einheitsmatrix I: Alle Aij = 0 (i ≠ j) und alle Aii = 1 (Hauptdiagonale)
- Inverse Matrix M^{-1} für quadratische Matrizen: $M^{-1} \cdot M = M \cdot M^{-1} = I$
- Transponierte Matrix M^T einer (m×n)-Matrix: (n×m)-Matrix gespiegelt an der Hauptdiagonale

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \qquad \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

Matrixrechnung 1/2

- Matrix multipliziert mit einer Zahl

$$\alpha \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} = \begin{pmatrix} \alpha \cdot a & \alpha \cdot b & \alpha \cdot c & \alpha \cdot d \\ \alpha \cdot e & \alpha \cdot f & \alpha \cdot g & \alpha \cdot h \\ \alpha \cdot i & \alpha \cdot j & \alpha \cdot k & \alpha \cdot l \\ \alpha \cdot m & \alpha \cdot n & \alpha \cdot o & \alpha \cdot p \end{pmatrix}$$

- Addition zweier Matrizen

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} + \begin{pmatrix} \alpha & \beta & \gamma & \delta \\ \varepsilon & \zeta & \eta & \theta \\ \iota & \kappa & \lambda & \mu \\ \nu & \xi & o & \pi \end{pmatrix} = \begin{pmatrix} a+\alpha & b+\beta & c+\gamma & d+\delta \\ e+\varepsilon & f+\zeta & g+\eta & h+\theta \\ i+\iota & j+\kappa & k+\lambda & l+\mu \\ m+\nu & n+\xi & o+o & p+\pi \end{pmatrix}$$

Matrixrechnung 2/2

- Matrixmultiplikation
  - Gegeben: (m×n)-Matrix A und (p×m)-Matrix B
  - Ergebnis: (p×m)-Matrix C
  - Matrixmultiplikation ist nicht kommutativ!
  - Wichtiger Spezialfall: Matrix-Vektor-Multiplikation
  - Motivation: Matrix beschreibt Transformation, Vektor einen Eckpunkt

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} a \cdot \alpha + b \cdot \beta + c \cdot \gamma \\ d \cdot \alpha + e \cdot \beta + f \cdot \gamma \\ g \cdot \alpha + h \cdot \beta + i \cdot \gamma \end{pmatrix}$$

$$C_{i,j} = \sum_{k=1}^{m} A_{k,i} \cdot B_{j,k}$$

Matrizen für Lineare Abbildungen

- Lineare Abbildungen können durch Matrizen beschrieben werden:
  - Wir bezeichnen eine solche lineare Abbildung als Transformation.

$$q = T \cdot p \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \\ v_w \end{pmatrix} \; mit \; p, q \in \mathbb{R}^4$$

4.2 Grundtransformationen

Verschiebung, Skalierung und Rotation

Verschiebung 1/2

- Matrixdarstellung einer Verschiebung

$$T(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T(d_x, d_y, d_z) \cdot p = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x + d_x \\ p_y + d_y \\ p_z + d_z \\ 1 \end{pmatrix}$$

Verschiebung 2/2

- Eigenschaften der Verschiebung
  - Bewahren Längen und Winkel
  - Identität: T(0, 0, 0) = E (Einheitsmatrix)
  - Vertauschung von Verschiebungen:

    T(d1x, d1y, d1z) • T(d2x, d2y, d2z)
    = T(d2x, d2y, d2z) • T(d1x, d1y, d1z)
  - Verkettung von Verschiebungen:

    T(d1x, d1y, d1z) • T(d2x, d2y, d2z)
    = T(d1x + d2x, d1y + d2y, d1z + d2z)
  - Inverse einer Verschiebung:

    T –1(dx, dy, dz) = T(–dx, –dy, –dz)

Rotation 1/3

- Matrixdarstellung einer Rotation um die x-, y- bzw. z-Achse:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation 2/3

- Rotationen erfolgen im rechtshändigen Koordinatensystem so, dass – wenn wir entlang einer positiven Achse zum Ursprung schauen - eine 90° Rotation um diese Achse die positiven Achsen wie folgt ineinander überführt:
  - Rotation um
    - x-Achse überführt +y nach +z
    - y-Achse überführt +z nach +x
    - z-Achse überführt +x nach +y

Rotation 3/3

- Eigenschaften von Rotationen:
  - Bewahren Längen und Winkel
  - Identität: Rx(0) = Ry(0) = Rz(0) = E (Einheitsmatrix)
  - Verkettung von achsengleichen Rotationen:
    Ra(θ) • Ra(ϕ) = Ra(θ + ϕ)
  - Vertauschung von achsengleichen Rotationen:
    Ra(θ) • Ra(ϕ) = Ra(ϕ) • Ra(θ)
  - Inverse einer Rotation:
    Ra(θ) –1 = Ra(–θ) = Ra(θ)T
  - Rotationen um verschiedene Achsen sind nicht vertauschbar!

Skalierung 1/2

- Matrixdarstellung einer Skalierung:

$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Anwendung: Größe ändern
- Uniforme Skalierung: sx = sy = sz

$$S(s_x, s_y, s_z) \cdot p = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} s_x \cdot p_x \\ s_y \cdot p_y \\ s_z \cdot p_z \\ 1 \end{pmatrix}$$

Skalierung 2/2

- Eigenschaften von Skalierungen:
  - Bewahren nicht die Länge, bewahren nur bei uniformer Skalierung die Winkel
  - Identität: S(1, 1, 1) = E (Einheitsmatrix)
  - Vertauschung von Skalierungen:
    S(s1x, s1y, s1z) • S(s2x, s2y, s2z)
    = S(s2x, s2y, s2z) • S(s1x, s1y, s1z)
  - Verkettung von Skalierungen:
    S(s1x, s1y, s1z) • S(s2x, s2y, s2z)
    = S(s1x · s2x, s1y · s2y, s1z · s2z)
  - Inverse einer Skalierung:
    S–1 (sx, sy, sz) = S(1/sx, 1/sy, 1/sz)

Spiegelung

- Matrixdarstellung einer Spiegelung an der xy-Ebene:

$$M_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

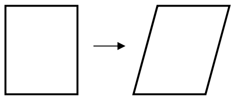
- Anwendung:
  - Beispiel: Konvertierung zwischen linkshändigem und rechtshändigem Koordinatensystem
- Eigenschaften: Bewahrt Längen und Winkel, nicht aber die Polygonorientierung (!)

$$M_z \cdot p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ -p_z \\ 1 \end{pmatrix}$$

Scherung

- Matrixdarstellung einer Scherung:

$$SH(h_{xy}, h_{xz}, h_{yx}, h_{yz}, h_{zx}, h_{zy}) = \begin{pmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & h_{yz} & 0 \\ h_{zx} & h_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Anwendung 

$$SH_{xy}(a) = \begin{pmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad SH_{xy}(a) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + ay \\ y \\ z \\ 1 \end{pmatrix}$$

4.3 Komposition von Transformationen

Zusammenfassung mehrerer Transformationsschritte

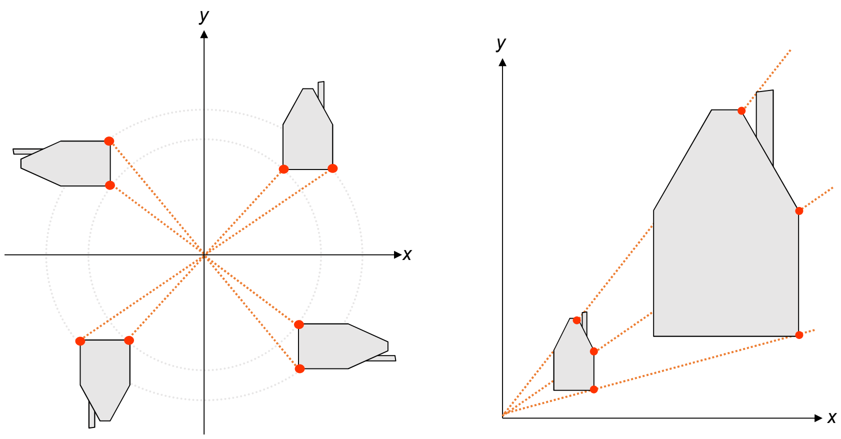
Komposition von Transformationen

- Komposition von Transformationen T_i erfolgt durch Multiplikation der Transformationsmatrizen
  - Interpretation: zunächst wird auf p Transformation T_n angewandt, dann T_{n-1}, … , und am Ende T_1
- Anwendung der Ergebnismatrix $T_{composite}$ statt der Einzelmatrizen auf die Punkte p geometrischer Objekte ist effizienter

$$\left(T_1 \cdot \left(T_2 \cdot \ldots \cdot \left(T_n \cdot p\right)\right)\right) = \left(T_1 \cdot T_2 \cdot \ldots \cdot T_n\right) \cdot p = T_{composite} \cdot p$$

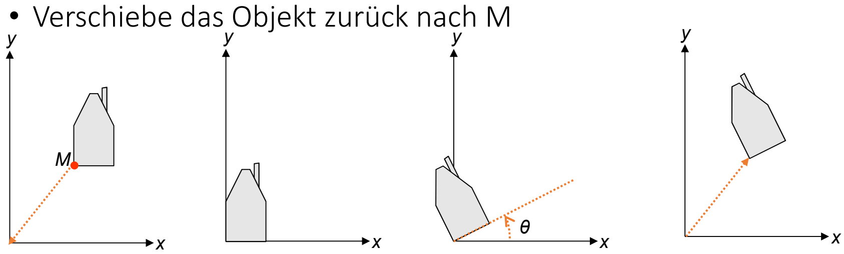
Komposition von Transformationen

- Rotation und Skalierung finden relativ zum Ursprung statt



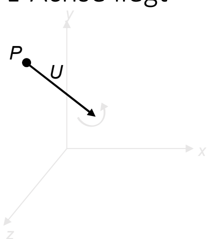
Komposition von Transformationen

- Um Objekte an einem beliebigen Bezugspunkt zu rotieren (bzw. zu skalieren), ist eine Folge von Transformationen notwendig
- Beispiel: Rotation um M = (xM, yM, zM)
  T(xM, yM, zM) • Rz(θ) • T(–xM, –yM, –zM)
  - Verschiebe das Objekt von M in den Ursprung O
  - Rotiere (bzw. skaliere) das Objekt
  - Verschiebe das Objekt zurück nach M



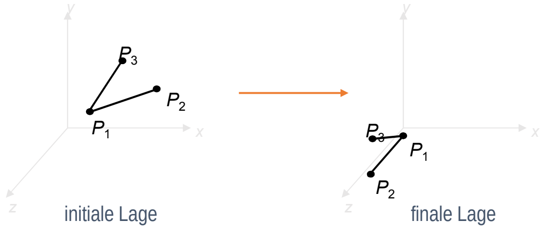
Komposition von Transformationen

- Rotation um eine beliebige Achse
  - Achse gegeben durch Schwerpunkt P und Richtung U = (ux, uy, uz)
  - Rotation um θ Grad
- Konstruktion der Transformation als Transformationssequenz:
  - Schwerpunkt P in den Ursprung verschieben
  - Drehung um die y-Achse, so dass U in der yz-Ebene liegt
  - Drehung um die x-Achse, so dass U auf der z-Achse liegt
  - Drehung um die z-Achse um θ Grad
  - Rückdrehung um die x-Achse
  - Rückdrehung um die y-Achse
  - Rückverschiebung nach P



Komposition von Transformationen

- Transformation gerichteter Liniensegmente (1)
  - Gegeben Liniensegmente P1P2 und P1P3.
    Transformiere sie so in die yz-Ebene, dass P1P2 auf der z-Achse liegt.
  - Die Transformation soll Längen und Winkelverhältnisse nicht ändern
  - Weg 1: Ermittlung der Drehwinkel der einzelnen Rotationen
  - Weg 2: Nutzung der Eigenschaften orthogonaler Matrizen



Komposition von Transformationen

- Transformation gerichteter Liniensegmente (2)
  - Weg 1 - Transformationsfolge:
    - 1. Verschiebe P1 = (x1, y1, z1) in den Ursprung
    - 2. Rotiere so um die y-Achse, dass P1P2 in der yz-Ebene liegt
    - 3. Rotiere so um die x-Achse, dass P1P2 auf der z-Achse liegt
    - 4. Rotiere so um die z-Achse, dass P1P3 in der yz-Ebene liegt
  - Schritt 1:
    - Translation: T(–x1, –y1, –z1)
    - Neue Punktkoordinaten:

$$P_1'(0,0,0)$$
$$P_2'(x_2 - x_1, y_2 - y_1, z_2 - z_1)$$
$$P_3'(x_3 - x_1, y_3 - y_1, z_3 - z_1)$$

Komposition von Transformationen

- Transformation gerichteter Liniensegmente (3)
  - Schritt 2: Rotation von $P_1' P_2'$ um die y-Achse in die yz-Ebene
    - Rotationswinkel: –(90° – θ) = θ – 90°
    - Rotation: Ry(θ – 90°)
    - Neue Punktkoordinaten:

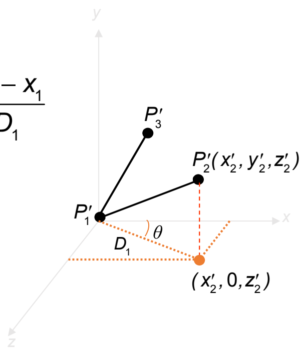
$$\cos(\theta - 90°) = \sin(\theta) = \frac{z_2'}{D_1} = \frac{z_2 - z_1}{D_1}$$

$$\sin(\theta - 90°) = -\cos(\theta) = -\frac{x_2'}{D_1} = -\frac{x_2 - x_1}{D_1}$$

$$D_1 = \sqrt{(x_2')^2 + (z_2')^2}$$

$$P_i'' = R_y(\theta - 90°) \cdot P_i'$$

$$P_2'' = (0, y_2 - y_1, D_1)$$

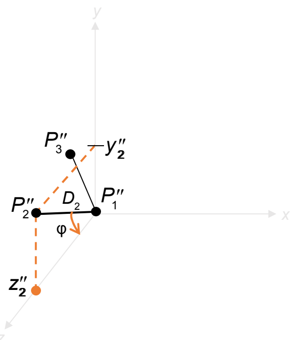


Komposition von Transformationen

- Transformation gerichteter Liniensegmente (4)
  - Schritt 3: Rotation von $P_1'' P_2''$ um die x-Achse auf die z-Achse
    - Rotationswinkel: ϕ
    - Rotation: Rx(ϕ)
    - Neue Punktkoordinaten:

$$\cos(\varphi) = \frac{z_2''}{D_2}, \sin(\varphi) = \frac{y_2''}{D_2}$$

$$D_2 = |P_1'' P_2''| = |P_1 P_2|$$



$$P_i''' = R_x(\varphi) \cdot P_i'' = R_x(\varphi) g R_y(\theta - 90°) \cdot P_i'$$

$$P_2''' = R_x(\varphi) \cdot P_2'' = R_x(\varphi) g R_y(\theta - 90°) \cdot P_2' = (0, 0, |P_1 P_2|)$$

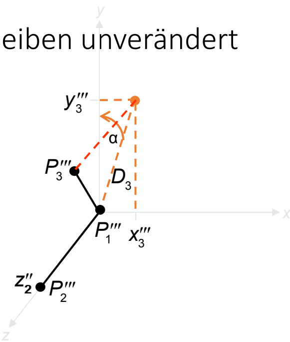
Komposition von Transformationen

- Transformation gerichteter Liniensegmente (5)
  - Schritt 4: Rotation von $P_1'''P_3'''$ in die yz-Ebene
    - Rotationswinkel: α
    - Rotation: Rz(α)
  - Transformationsfolge:
  - Rigid-Body-Transformation:
    keine Verzerrungen, Längen und Winkel bleiben unverändert

$$\cos(\alpha) = \frac{y_3'''}{D_3},\ \sin(\alpha) = \frac{x_3'''}{D_3}$$

$$D_3 = \sqrt{(x_3''')^2 + (y_3''')^2}$$

$$R_z(\alpha) \cdot R_x(\varphi) \cdot R_y(\theta - 90°) \cdot T(-x_1, -y_1, -z_1)$$



Komposition von Transformationen

- Transformation gerichteter Liniensegmente (6)
  - Weg 2 – Nutzung der Eigenschaften orthogonaler Matrizen
    - Jede Zeile und Spalte entspricht einem Vektor der Länge 1
    - Zeilen- und Spaltenvektoren sind zueinander orthogonal (Skalarprodukt = 0).
    - Jeder Zeilenvektor rotiert auf eine Hauptachse (Rx auf x-Achse, Ry auf y-Achse, Rz auf z-Achse), die Koordinatenachsen rotieren auf die Spaltenvektoren (x-Achse auf R1, y-Achse auf R2, z-Achse auf R3).
  - Gesucht: Rotationsmatrix
  - Überlegung 1:
    - Rz ist der Einheitsvektor, der auf die z-Achse rotiert. Deshalb:

$$R = \begin{pmatrix} r_{1x} & r_{2x} & r_{3x} \\ r_{1y} & r_{2y} & r_{3y} \\ r_{1z} & r_{2z} & r_{3z} \end{pmatrix} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} R_1 & R_2 & R_3 \end{pmatrix}$$

$$R_z = (r_{1z}, r_{2z}, r_{3z})^T = \frac{P_1P_2}{|P_1P_2|}$$

Komposition von Transformationen

- Transformation gerichteter Liniensegmente (7)
  - Überlegung 2:
    - Rx ist der Einheitsvektor, der senkrecht zur Ebene, die durch P1, P2 und P3 definiert ist, steht:
  - Überlegung 3:
    - Ry steht senkrecht auf Rx und Rz:
  - Transformationsfolge:

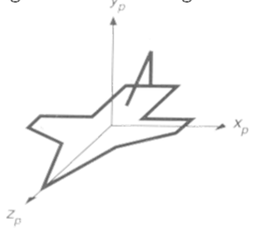
$$R_x = (r_{1x}, r_{2x}, r_{3x})^T = \frac{P_1P_3 \times P_1P_2}{|P_1P_3 \times P_1P_2|}$$

$$R_y = (r_{1y}, r_{2y}, r_{3y})^T = R_z \times R_x$$

$$\begin{pmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} gT(-x_1, -y_1, -z_1)$$

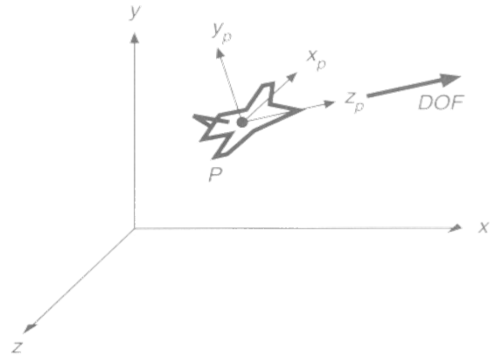
Komposition von Transformationen

- Richtungstransformation (direction of flight, DOF)
  - Fragestellung:
    - Gegeben:
      - Objekt im xp, yp, zp-Koordinatensystem, zentriert im Ursprung
      - Punkt P, Richtungsvektor DOF
    - Gesucht:
      - Transformation, nach deren Ausführung das Objekt im Punkt P zentriert ist, die zp-Achse in die ausgezeichnete Richtung zeigt, wobei xpzp-Ebene nicht durch die Richtungstransformation gedreht werden soll.



Komposition von Transformationen

- Richtungstransformation (1)
  - Transformationsfolge:
    - 1. Rotiere Objekt, so dass es in Richtung DOF zeigt
    - 2. Verschiebe das Objekt vom Ursprung nach P



Komposition von Transformationen

- Richtungstransformation (2)
  - Konstruktion der Rotationsmatrix
    - Die zp-Achse muss auf DOF abgebildet werden:
    - Die xp-Achse muss auf einen horizontalen (d.h. zur y-Achse senkrechten) Vektor, der senkrecht zu DOF ist, abgebildet werden:
    - Die yp-Achse muss auf R3 × R1 abgebildet werden:

$$R = \begin{pmatrix} r_{1x} & r_{2x} & r_{3x} \\ r_{1y} & r_{2y} & r_{3y} \\ r_{1z} & r_{2z} & r_{3z} \end{pmatrix}$$

$$R_3 = (r_{3x}, r_{3y}, r_{3z})^T = \frac{DOF}{|DOF|}$$

$$R_1 = (r_{1x}, r_{1y}, r_{1z})^T = \frac{y \times DOF}{|y \times DOF|}$$

$$R_2 = (r_{2x}, r_{2y}, r_{2z})^T = \frac{DOF \times (y \times DOF)}{|DOF \times (y \times DOF)|}$$

Komposition von Transformationen

- Richtungstransformation (3)
  - Transformationsfolge:
    - Zuerst rotiere das Objekt in die DOF-Richtung.
    - Dann schiebe es an die Stelle P.

$$T(x_p, y_p, z_p) \cdot \begin{pmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^T$$

  - Degenerierter Fall:
    - Falls DOF = y, dann gibt es unendlich viele horizontale Vektoren. Die Kreuzprodukte y × DOF und DOF × (y × DOF) sind in diesem Fall 0.

4.4 Koordinatensystemwechsel

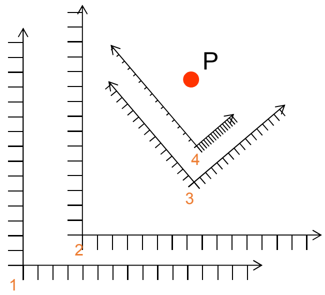
Transformationen zur Änderung des Koordinatensystems

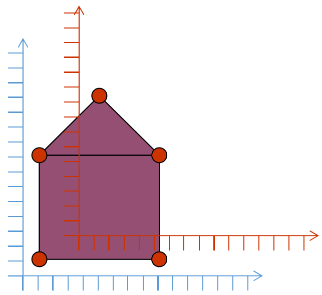
Koordinatensystemwechsel

Alternative Sichtweise auf Transformationen:

Bislang: Jede Transformation überführt eine Menge von Punkten in eine Menge transformierter Punkte ⇒ Koordinatensystem bleibt unverändert, die Lage der Punkte im Koordinatensystem wird verändert.

Alternativ: Jede Transformation ändert das Koordinatensystem, in dem eine Menge von Punkten beschrieben ist ⇒ Das Objekt wird in unterschiedlichen Koordinatensystemen dargestellt, die Koordinatensysteme werden gewechselt.





Koordinatensystemwechsel

- Anwendung:
  - Einzelne Objekte einer Szene besitzen ihr eigenes, lokales Koordinatensystem.
  - Die lokalen Koordinatensysteme der Objekte müssen in ein gemeinsames Koordinatensystem, das Weltkoordinatensystem, überführt werden.
- Transformation zwischen Koordinatensystemen (KS):
  - Mi←j bezeichne den Wechsel vom KSj in das KSi
  - P(i) bezeichne die Darstellung des Punktes P im KSi
  - Invertierung: Mi←j = (Mj←i) –1
  - Transition: Mi←j • Mj←k = Mi←k

Koordinatensystemwechsel

- Beispiel 1:
  - Transformationsfolge:
    - Rotation und Skalierung eines Objektes um seinen Mittelpunkt M und anschließende Verschiebung in den Punkt Q
    - Transformationen zwischen den Koordinatensystemen:
      - Angewendet auf Punkte:

$$T_{composite} = T(x_Q, y_Q, z_Q) \cdot R_z(\theta) \cdot S(s_x, s_y, s_z) \cdot T(-x_M, -y_M, -z_M)$$

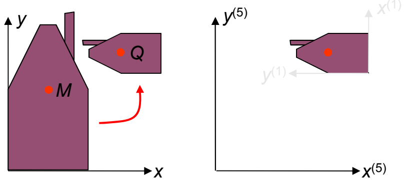
$$M_{5\leftarrow 1} = M_{5\leftarrow 4} \cdot M_{4\leftarrow 3} \cdot M_{3\leftarrow 2} \cdot M_{2\leftarrow 1} = T_{composite}$$

$$M_{1\leftarrow 5} = M_{1\leftarrow 2} \cdot M_{2\leftarrow 3} \cdot M_{3\leftarrow 4} \cdot M_{4\leftarrow 5} = T^{-1}_{composite}$$

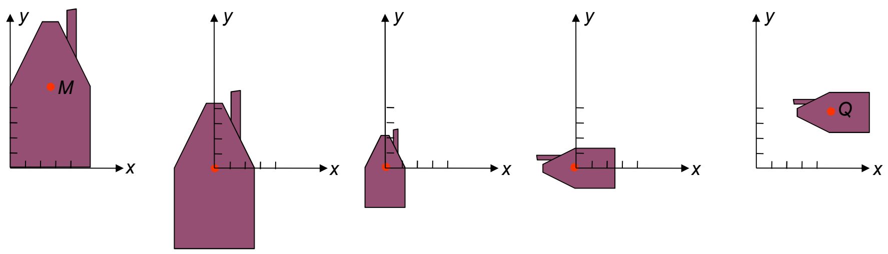
$$= T(x_M, y_M, z_M) \cdot S(1/s_x, 1/s_y, 1/s_z) \cdot R_z(-\theta) \cdot T(-x_Q, -y_Q, -z_Q)$$

$$P^{(5)} = M_{5\leftarrow 1} \cdot P^{(1)}$$

$$P^{(1)} = M_{1\leftarrow 5} \cdot P^{(5)}$$



Koordinatensystemwechsel

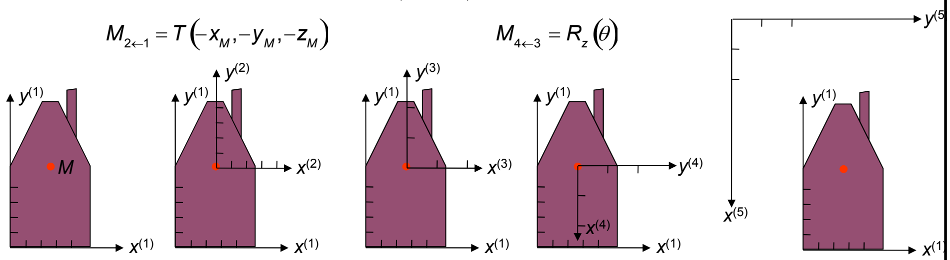


$$M_{3\leftarrow 2} = S(s_x, s_y, s_z)$$

$$M_{5\leftarrow 4} = T(x_Q, y_Q, z_Q)$$

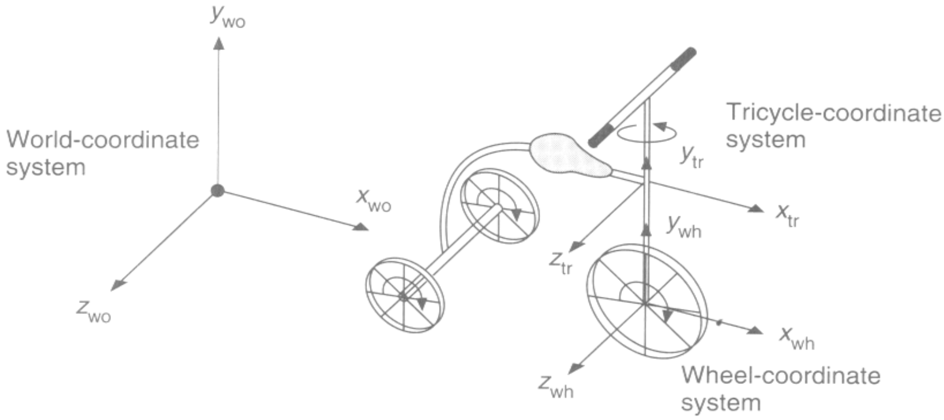
$$M_{2\leftarrow 1} = T(-x_M, -y_M, -z_M)$$

$$M_{4\leftarrow 3} = R_z(\theta)$$



Koordinatensystemwechsel

- Beispiel 2 (Dreirad):
  - Koordinatensysteme:
    - Welt-KS (world) - Dreirad-KS (tricycle) - Rad-KS (wheel)



Koordinatensystemwechsel

- Beispiel 2 (Dreirad):
  - Fragestellung:
    - Hierarchische Darstellung der geometrischen Transformationen?
      - ⇒ Szenengraph als Werkzeug zur hierarchischen Modellierung
    - Abstimmung der Rotationen in unterschiedlichen Koordinatensystemen?

```
Dreirad ----------------------------------- Translation T_wo
 ├─ Lenkung ------------------------------- Rotation R_y-tr(α)
 │   ├─ Lenkstange
 │   └─ Vorderrad ------------------------- Rotation R_z-wh(β')
 └─ Rahmen
     ├─ Rahmenstangen
     ├─ Sitz
     ├─ Hinterachse
     └─ Hinterräder ----------------------- Rotation R_z-wh(β'')
         ├─ Hinterrad1
         └─ Hinterrad2
```

Dreirad: Translation T_{wo}; Lenkung: Rotation $R_{y\text{-}tr}(\alpha)$; Vorderrad: Rotation $R_{z\text{-}wh}(\beta')$; Hinterräder: Rotation $R_{z\text{-}wh}(\beta'')$

4.5 Transformationen in OpenGL

Umsetzung mit GLM und GLSL

Geometrische Transformationen in OpenGL

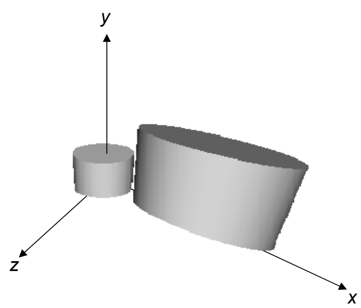
- Unterstützte Koordinatensysteme
  - Gleitkomma-Koordinatensystem für 2D und 3D
    - Ganzzahliges Koordinatensystem (2D) als Spezialfall
  - Repräsentation von Vektoren:
    - 2D-Vektor: Array mit 2 Elementen
    - 3D-Vektor: Array mit 3 Elementen
  - Repräsentation von Matrizen:
    - 3×3-Matrizen durch zweidimensionales Array mit $3\cdot3$ Elementen
    - 4×4-Matrizen durch zweidimensionales Array mit $4\cdot4$ Elementen
- Funktionen für geometrische Transformationen
  - OpenGL spezifiziert keine Klassen für Vektoren und Matrizen (... Hardware-nahe Schnittstelle!)
  - GLM kann für die Verwaltung von Vektoren und Matrizen benutzt werden

OpenGL Transformationspipeline

- Model-View-Matrix
  - Transformationsmatrix für modellraumbezogene Transformationen
  - Enthält zusätzlich die Transformation der Szene in das Kamerakoordinatensystem (View-Transformation, nicht Projektion!)
- Transformationspipeline
  - Jeder Eckpunkt wird von OpenGL zuerst mit der aktuellen Model-View-Transformation, dann mit der aktuellen Projektionstransformation multipliziert
  - In Standard-Szenendarstellungen gibt es eine einzige Projektionstransformation (NP = 1), eine einzige View-Transformation
    (NV = 1) und je nach Hierarchietiefe beliebig viele Modelltrans-formationen

Geometrische Transformationen in OpenGL

- Beispiel 1: Translation und Skalierung
  - Zeichnen von Zylinderobjekten
    - drawCylinder zeichnet einen im Ursprung zentrierten, bzgl. der y-Achse stehenden Zylinder mit vorgebenem Radius und Höhe.
    - Durch Translation wird dieser "Einheitszylinder" ggf. positioniert und
    - durch Skalierung ggf. in seinen Proportionen verändert.



Geometrische Transformationen in OpenGL

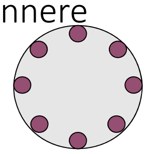
- Beispiel 2: Spezifikation von Matrizen
  - Spezifikation expliziter Matrixkoeffizienten

```
void draw(...) {
  ...
  GLfloat M[4][4];
  M[0][0] = ....;
  ...
  M[3][3] = ....;

}
```


Beispiel Säulenhalle 1/5

- Säulenhalle besteht aus Bodenfläche, Deckelfläche und Säulen
- Säulen sind entlang des Randes der Bodenfläche gleichmäßig verteilt
- Gesamthöhe einer Säule beträgt H, wobei insgesamt 20% auf die Sockel entfallen
- Jede Säule hat 2 Sockel unten und 2 Sockel oben
- Radien: Säulenmittelstück 70%, innere Sockel 90%, äußere Sockel R

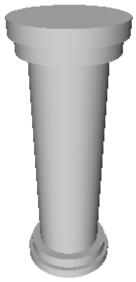


Beispiel Säulenhalle 2/5

- Zeichnen eines oben und unten verschlossenen Zylinders: Jede Säule soll in der y-Achse "stehen" sowie oben und unten eine Deckelfläche besitzen.
- Primitiv Disk: in die xy-Ebene eingebettet
- Primitiv Cylinder: besteht nur aus Mantelfläche, Höhenachse = z-Achse
  - Tessellation: Anzahl der Polygonunterteilungen.

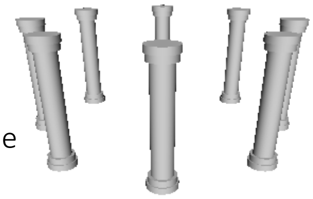
Beispiel Säulenhalle 3/5

- Zeichnen einer Säule:
  - Komposition aus 5 Zylinderobjekten
  - Positionierung der Teile durch Translation entlang der y-Achse
  - Inkrementelle Translation jeweils um die Teilsäulenhöhe



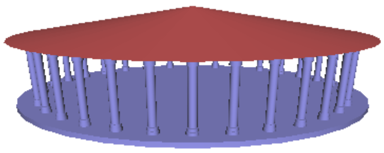
Beispiel Säulenhalle 4/5

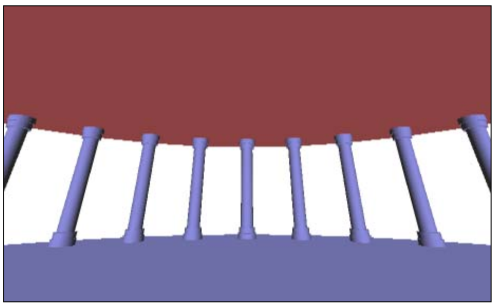
- Zeichnen der Hallensäulen
  - Säulen sind gleichmäßig auf einem Kreis mit Radius *hallr* verteilt
  - Anzahl *n* und Radius *colr* der Säulen sowie Bodenradius *hallr* wählbar



Beispiel Säulenhalle 5/5

- Zeichnen der Halle
  - Zylinder als Bodenfläche
  - Kegel als Dach





4.6 Weiterführende Literatur

Zugrundeliegende und ergänzende Quellen

Literatur

- J. Foley, A. van Dam, S. Feiner: Computer Graphics: Principles and Practice (3. Auflage), Addison-Wesley 2013.
  - Kapitel 10: Transformations in Two Dimensions
  - Kapitel 11: Transformations in Three Dimensions
- P. Shirley, M. Ashikhmin, S. Marschner: Fundamentals of Computer Graphics (3. Auflage), AK Peters 2009.
  - Kapitel 6: Transformation Matrices