Computergrafik

Die Mathematik hinter OpenGL, DirectX, etc.

3D-Transformationen & Homogene Koordinaten

UHLMANN

Inhalt

1 Einleitung

Um die virtuellen Objekte an ihre vorgesehene Position in der 3D-Szene zu platzieren, müssen sie zuvor richtig skaliert, gedreht und verschoben werden.

2

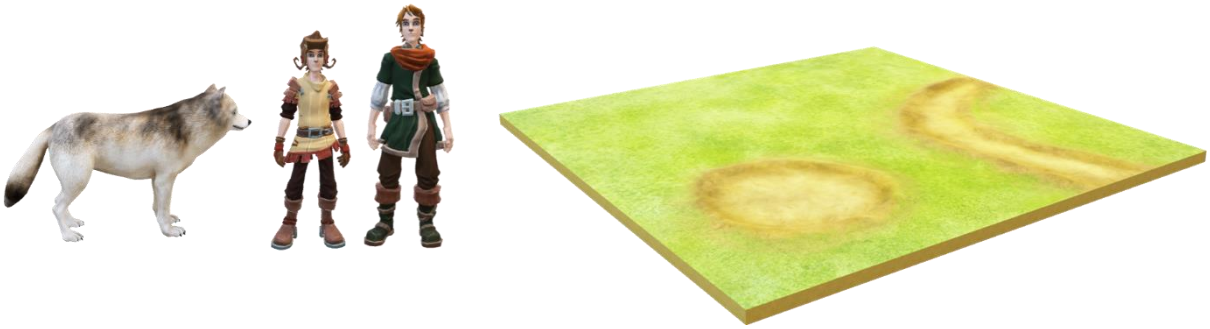


Abb. 1

Die dafür benötigten mathematischen Verfahren beruhen auf der Multiplikation der Vertex-Vektoren $\boldsymbol{X}$ mit quadratischen (3×3)- bzw. (4×4)-Transformationsmatrizen.

Die Matrix-Multiplikation ist assoziativ, aber im Allgemeinen nicht kommutativ!!!

$$\left.\begin{aligned} A \cdot B &\neq B \cdot A \\ A \cdot \boldsymbol{X} &\neq \boldsymbol{X} \cdot A \end{aligned}\right\}$$

$$\left.\begin{aligned} (A \cdot B) \cdot C &= A \cdot (B \cdot C) \\ A \cdot (B \cdot \boldsymbol{X}) &= (A \cdot B) \cdot \boldsymbol{X} \end{aligned}\right\}$$

Im Besonderen kann eine Folge von Transformationen eines Vertex $\boldsymbol{X}$ zu einer einzigen Transformationsmatrix zusammengefasst werden.

$$M_3 \cdot (M_2 \cdot (M_1 \cdot \boldsymbol{X}) = \underbrace{(M_3 \cdot M_2 \cdot M_1)}_{M} \cdot \boldsymbol{X} = M \cdot \boldsymbol{X}$$

2 3D-Transformationen

Die MATHEMATISCHEN GRUNDLAGEN dabei sind die Koordinatentransformationen bei SKALIERUNG, ROTATION UND VERSCHIEBUNG. Naheliegenderweise erzielt man

- Skalierung durch Multiplikation mit einem Vergrößerungs- oder Verkleinerungsfaktor,
- Verschiebung durch Addition eines Verschiebungsvektors und
- Drehung durch Multiplikation mit Drehmatrizen.

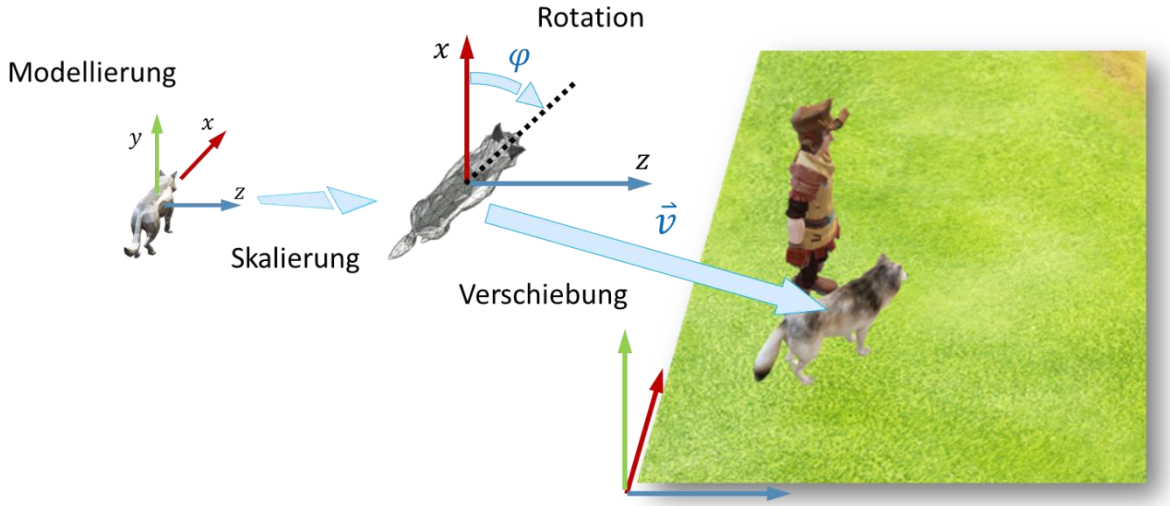


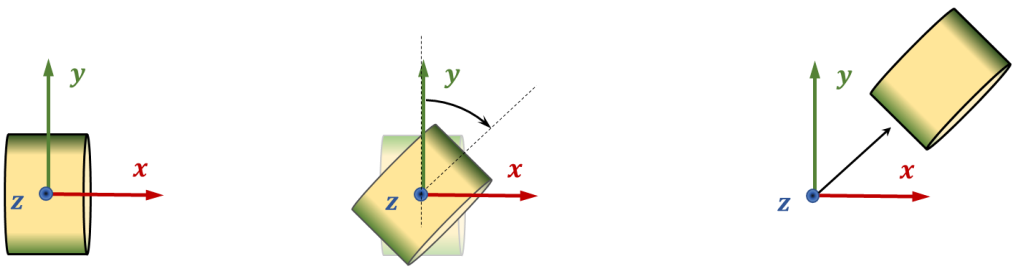
Abb. 2

Die Reihenfolge ist wichtig!

Die Reihenfolge der Transformationen ist entscheidend! Üblicherweise wird ein Modell zuerst im MODEL-SPACE skaliert, dann in die passende Stellung gedreht, bevor es schlussendlich an den richtigen Ort geschoben wird.

Abb. 4 zeigt den Einfluss der Reihenfolge bei Rotation und Translation.

Zuerst *a) drehen, dann verschieben*



1. $R_{(-45°)}$ 2. $T_{(\nearrow)}$

Abb. 3

versus *b) zuerst verschieben, dann drehen*

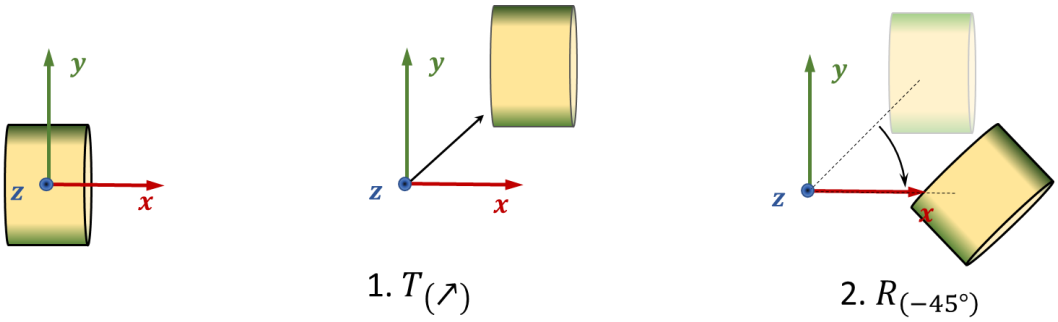


Abb. 4

Da Drehmatrizen um den Koordinatenursprung drehen, ist das Ergebnis bei umgekehrter Abfolge ein anderes. Die vorgeschriebene Reihenfolge ist

1. Skalierung → **2. Rotation** → **3. Translation**

Angewandt auf einen Vertex $\boldsymbol{X}$ ergibt das also zuerst $S(X)$, dann $R\big(S(X)\big)$ und zuletzt $T\left(R\big(S(X)\big)\right)$.

$$\boldsymbol{X}' = \left(T\left(R(S(\boldsymbol{X}))\right)\right),$$

2.1 Skalierung

Skalierung mit Sperrung der Seitenverhältnisse

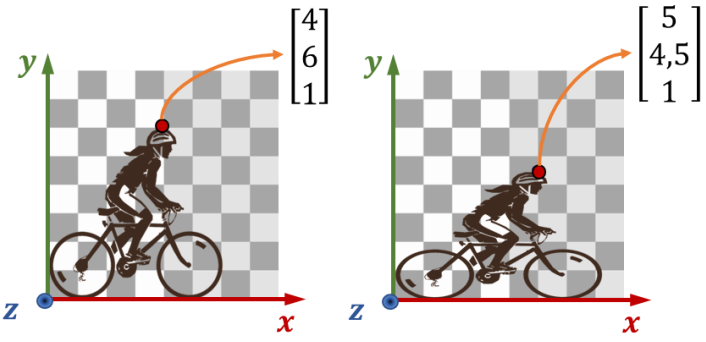
Werden die Vertices einer Figur mit einem generellen Skalierungsfaktor c multipliziert, dann entspricht das einer Skalierung mit Sperrung der Seitenverhältnisse.

$$2{,}0 \cdot \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \cdot 2{,}0 \\ 2 \cdot 2{,}0 \\ 1 \cdot 2{,}0 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 2 \end{bmatrix}$$

bzw. in Form einer Matrix: $\begin{bmatrix} 2{,}0 & 0 & 0 \\ 0 & 2{,}0 & 0 \\ 0 & 0 & 2{,}0 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 2 \end{bmatrix}$

Skalierung ohne Sperrung der Seitenverhältnisse

Dabei kann jede Koordinate unterschiedlich skaliert werden. (Abb. 5)



$$\begin{bmatrix}4\\6\\1\end{bmatrix} \quad \begin{bmatrix}5\\4{,}5\\1\end{bmatrix}$$

$$\begin{bmatrix}4{,}0 \cdot 1{,}20\\6{,}0 \cdot 0{,}75\\1{,}0 \cdot 1{,}00\end{bmatrix} = \begin{bmatrix}5{,}0\\4{,}5\\1{,}0\end{bmatrix}$$

bzw. $$\begin{bmatrix}1{,}2 & 0 & 0\\0 & 0{,}75 & 0\\0 & 0 & 1{,}0\end{bmatrix} \cdot \begin{bmatrix}4\\6\\1\end{bmatrix} = \begin{bmatrix}5{,}0\\4{,}5\\1{,}0\end{bmatrix}$$

Abb. 5

Skalierungsmatrix $S = \begin{bmatrix}c_x & 0 & 0\\0 & c_y & 0\\0 & 0 & c_z\end{bmatrix}$

2.2 Rotation

2.2.1 2D-Rotation

Rotation in der $[xy]$-Koordinatenebene **um den Koordinatenursprung**:

Abbildungsmatrix: $D = \begin{bmatrix}\cos(\varphi) & -\sin(\varphi)\\\sin(\varphi) & \cos(\varphi)\end{bmatrix}$

(Begründung siehe Anhang 3.1 unten)

BEISPIEL

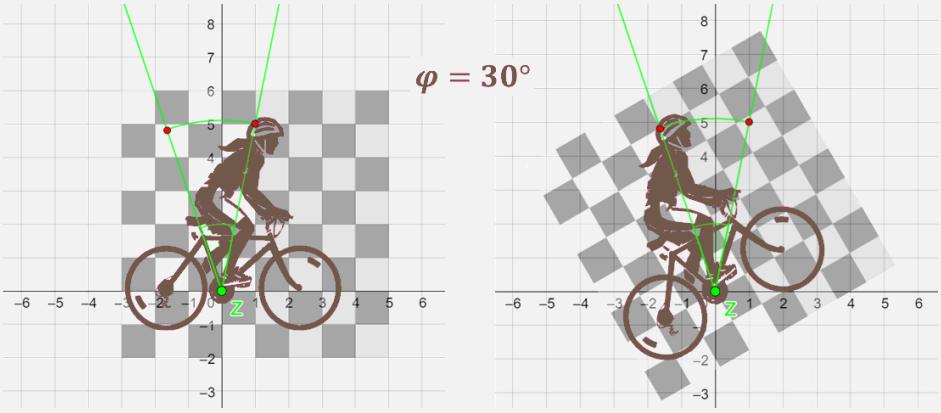


Abb. 6

Drehung um Zentrum $Z = (0{,}0)$ und Drehwinkel $\varphi = 30°$.

$$R = \begin{bmatrix}\cos(\varphi) & -\sin(\varphi)\\\sin(\varphi) & \cos(\varphi)\end{bmatrix} = \begin{bmatrix}0{,}866 & -0{,}500\\0{,}500 & 0{,}866\end{bmatrix}$$

$$X = \begin{bmatrix}1\\5\end{bmatrix} \rightarrow X' = R \cdot X = \begin{bmatrix}0{,}866 & -0{,}500\\0{,}500 & 0{,}866\end{bmatrix} \cdot \begin{bmatrix}1\\5\end{bmatrix} = \begin{bmatrix}-1{,}634\\4{,}830\end{bmatrix}$$

Allgemein:

$$X' = R \cdot X = \begin{bmatrix}\cos(\varphi) & -\sin(\varphi)\\\sin(\varphi) & \cos(\varphi)\end{bmatrix} \cdot \begin{bmatrix}x\\y\end{bmatrix}$$

2.2.2 3D-Rotation

Rotation um eine Koordinatenachse im Koordinatenursprung

Es geht um die gleiche Rotation wie eben in 2.2.1, allerdings in 3D formuliert. Dreidimensional gesehen ist obige Rotation eine Rotation um die z-Achse.

Wir verwenden wie üblich ein rechtshändiges Koordinatensystem. Die z-Achse zeigt dabei aus der Bildebene heraus, weshalb wir immer in Richtung $(-z)$-Achse blicken.

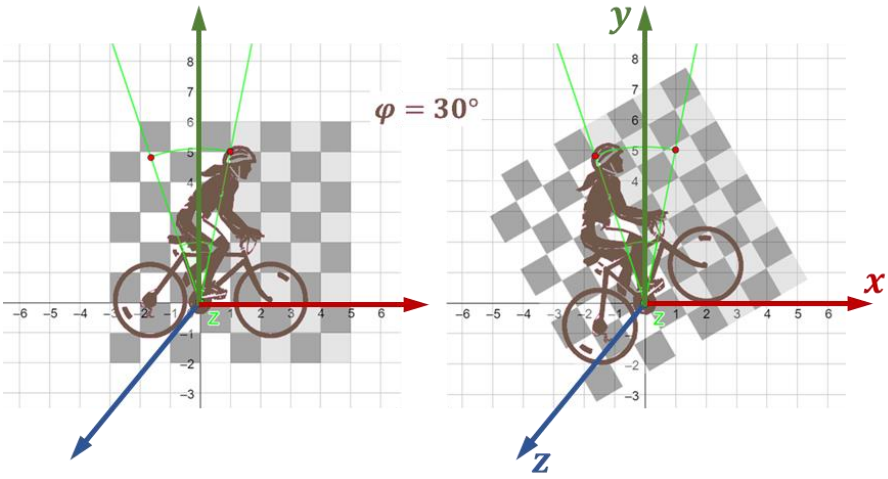


Abb. 7: Rotation um die z-Achse

Die Rotationsmatrix muss jetzt 3dimensional geschrieben werden. Bei einer Drehung um die z-Achse ändern sich nur die x- und y-Koordinaten. Die z-Koordinaten aller Punkte eines Objekts ändern sich dabei nicht.

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

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

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

Bei einer Drehung um die y-Achse ändert sich die y-Koordinate nicht, der Rest entspricht einer 2dim-Drehung in der $[xz]$-Koordinatenebene.

$$X' = R_y \cdot X = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\beta) \cdot x + \sin(\beta) \cdot z \\ y \\ \cos(\beta) \cdot z - \sin(\beta) \cdot x \end{bmatrix}$$

7

Die **Reihenfolge der Drehungen** ist dabei wesentlich! Die Matrix-Multiplikation ist **assoziativ**, aber **nicht kommutativ**! (Abb. 8 und Abb. 9)

$$R_z \cdot R_y \cdot R_x \neq R_x \cdot R_z \cdot R_y$$

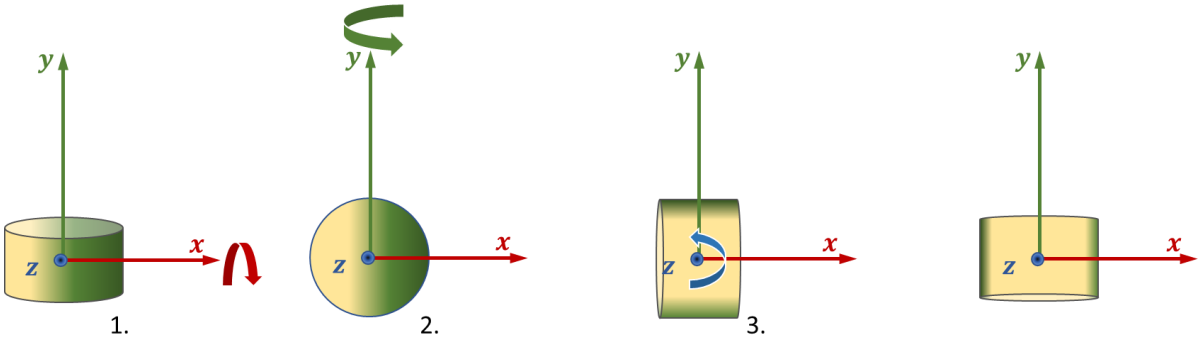


Abb. 8: zuerst x, dann y, dann z $\rightarrow \left(R_z \cdot \left(R_y \cdot (R_x \cdot X)\right)\right) = (R_z \cdot R_y \cdot R_x) \cdot X$ *, jeweils um 90° gegen den Uhrzeigersinn*

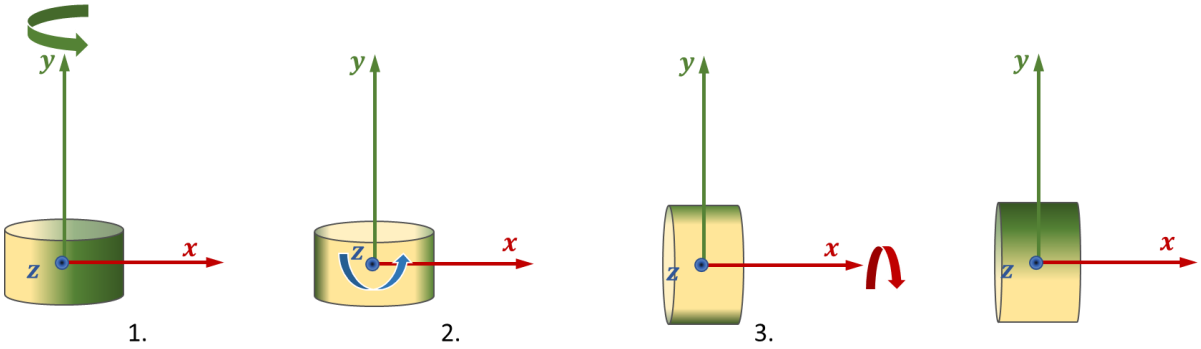


Abb. 9: zuerst y, dann z, dann x $\rightarrow \left(R_x \cdot \left(R_z \cdot \left(R_y \cdot X\right)\right)\right) = (R_x \cdot R_z \cdot R_y) \cdot X$ *, jeweils um 90° gegen den Uhrzeigersinn*

Beispiel

Drehung von $X = (1, 5, -2)$ um die x-Achse: $\alpha = 20°$
y-Achse: $\beta = 60°$
z-Achse: $\gamma = 90°$

$$X' = \left(R_z \cdot \left(R_y \cdot (R_x \cdot X)\right)\right) = \left(\underbrace{R_z \cdot R_y \cdot R_x}_{\hat{R}}\right) \cdot X = R \cdot X =$$

$$= \begin{bmatrix} 0 & -0{,}5 & 0{,}866 \\ 0{,}94 & -0{,}296 & -0{,}171 \\ 0{,}342 & 0{,}814 & 0{,}470 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 5 \\ -2 \end{bmatrix} = \begin{bmatrix} -4{,}232 \\ -0{,}198 \\ 3{,}472 \end{bmatrix}$$

2.3 Translation

Die Translation kann durch die Addition der Vertices mit dem Verschiebungsvektor dargestellt werden.

Verschiebungsvektor: $X' = X + \vec{v}$

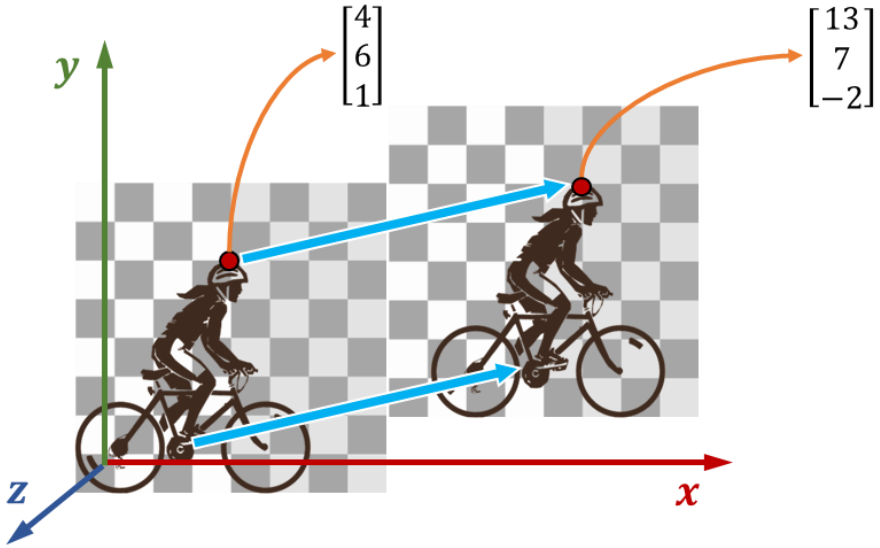


Abb. 10: Translation

$$X' = X + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad \Longrightarrow \quad X' = X + \vec{v} = \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix} + \begin{bmatrix} 9 \\ 1 \\ -3 \end{bmatrix} = \begin{bmatrix} 13 \\ 7 \\ -2 \end{bmatrix}$$

Es geht aber bedeutend eleganter, und zwar ebenfalls mit Matrizen durch Verwendung sogenannter „Homogener Koordinaten“. Siehe Kapitel 2.4 „Transformationen in homogenen Koordinaten“

2.4 Transformationen in homogenen Koordinaten (3D → 4D)

Während die Rotation und die Skalierung mit Multiplikation geeigneter Matrizen berechnet werden kann, gelingt dies bei der Translation so nicht.

KUNSTGRIFF

Die Verwendung homogener Koordinaten bedeuten (vorübergehend) die Erweiterung der Dreidimensionalität durch eine vierte Koordinate:

$$(x, y, z) \rightarrow (x, y, z, w)$$

Punkte $w = 1$ $(x, y, z) \to (x, y, z, 1)$

Vektoren $w = 0$ $(x, y, z) \to (x, y, z, 0)$

Skalierungsmatrix $S = \begin{bmatrix} c_x & 0 & 0 & 0 \\ 0 & c_y & 0 & 0 \\ 0 & 0 & c_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\det(S) = c_x \cdot c_y \cdot c_z \neq 0$ falls $c_x, c_y, c_z \neq 0$

Rotationsmatrizen $R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\det(R_x) = 1$

$R_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\det(R_y) = 1$

$R_z = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\det(R_z) = 1$

Translationsmatrix $T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\det(T) = 1$

Durch diese Erweiterung lassen sich alle geometrischen Transformationen als

Multiplikation mit Matrizen

bewerkstelligen.

BEISPIEL $X = (1, 5, -2)$

1. Skalierung: $S = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0{,}5 \end{bmatrix} \longrightarrow \begin{bmatrix} 7 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

2. Rotation $\varphi = +60°$ um die y-Achse: $R_y = \begin{bmatrix} \cos(60°) & 0 & \sin(60°) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(60°) & 0 & \cos(60°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

3. Translation $\vec{v} = \begin{bmatrix} 5 \\ -3 \\ 11 \end{bmatrix} \longrightarrow T = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$X' = (\mathrm{T} \cdot R_y \cdot \mathrm{S}) \cdot X =$$

$$= \left(\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0{,}5 & 0 & 0{,}866 & 0 \\ 0 & 1 & 0 & 0 \\ -0{,}866 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 7 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \cdot X =$$

$$= \begin{bmatrix} 7 \cdot 0{,}5 & 2 \cdot 0 & 0{,}5 \cdot 0{,}866 & 5 \\ 7 \cdot 0 & 2 \cdot 1 & 0{,}5 \cdot 0 & -3 \\ 7 \cdot -0{,}866 & 2 \cdot 0 & 0{,}5 \cdot 0{,}5 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 5 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 7{,}634 \\ 7 \\ 4{,}438 \\ 1 \end{bmatrix} \widehat{=} \begin{bmatrix} 7{,}634 \\ 7 \\ 4{,}438 \end{bmatrix}$$

Skalierung→Rotation→Translation folgt dabei immer folgendem Schema:

$$\begin{bmatrix} 1 & 0 & 0 & \boldsymbol{t_x} \\ 0 & 1 & 0 & \boldsymbol{t_y} \\ 0 & 0 & 1 & \boldsymbol{t_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha & \beta & \gamma & 0 \\ \delta & \varepsilon & \eta & 0 \\ \vartheta & \mu & \lambda & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{c_x} & 0 & 0 & 0 \\ 0 & \boldsymbol{c_y} & 0 & 0 \\ 0 & 0 & \boldsymbol{c_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha \cdot \boldsymbol{c_x} & \beta \cdot \boldsymbol{c_y} & \gamma \cdot \boldsymbol{c_z} & \boldsymbol{t_x} \\ \delta \cdot \boldsymbol{c_x} & \varepsilon \cdot \boldsymbol{c_y} & \eta \cdot \boldsymbol{c_z} & \boldsymbol{t_y} \\ \vartheta \cdot \boldsymbol{c_x} & \mu \cdot \boldsymbol{c_y} & \lambda \cdot \boldsymbol{c_z} & \boldsymbol{t_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

BEMERKUNG: Da die Transformationsmatrizen quadratisch und ihre Determinanten ungleich Null sind, sind sie invertierbar. Jede Transformation kann also wieder rückgängig gemacht werden.

$$R^{-1}_{x,y,z,\varphi} = R_{x,y,z,-\varphi} \quad \text{und} \quad T^{-1}_{t_x,t_y,t_z} = T_{-t_x,-t_y,-t_z}$$

2.5 Weitere Algorithmen der Vektorrechnung

DAS KREUZPRODUKT - ZUR BERECHNUNG VON FLÄCHEN, ROTATIONSWINKELN, ACHSENVEKTOREN ETC.

Dreiecksfläche $A_\triangle = \frac{1}{2} \cdot |\vec{a} \times \vec{b}|$

Rotationswinkel: Ein Objekt soll von einer Anfangsposition in eine gewisse Endposition gedreht werden. Das Kreuzprodukt dient dabei dazu, die Rotationsachse zu finden (Abb. 11)

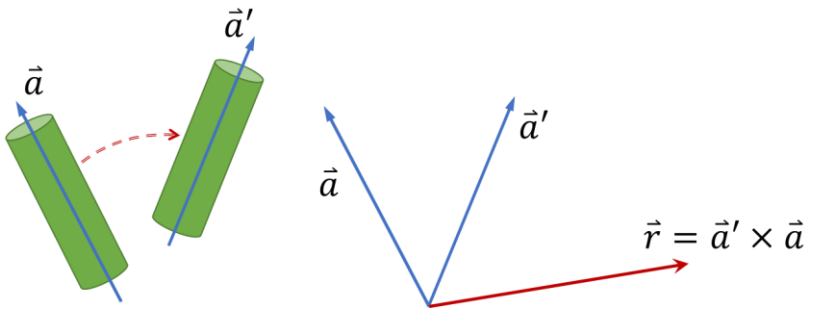


Abb. 11: Bestimmung der Rotationsachse mit Hilfe des Vektorproduktes

Achsenvektoren Gesucht ist das vervollständigte Koordinatensystem der Kamera. Gegeben sind bereits die Einheitsvektoren $\hat{y}'$ und $\hat{z}'$:

$$\hat{x}' = \hat{y}' \times \hat{z}'$$

Das Skalarprodukt - Zur Berechnung von Winkeln

Beispiel: Das Skalarprodukt liefert obigen Rotationswinkel.

$$\varphi = \measuredangle(\vec{a}, \vec{a}') = \cos^{-1}\left(\frac{\vec{a} \cdot \vec{a}'}{a \cdot a'}\right)$$

$$(a = |\vec{a}|\,;\; a' = |\vec{a}'|)$$

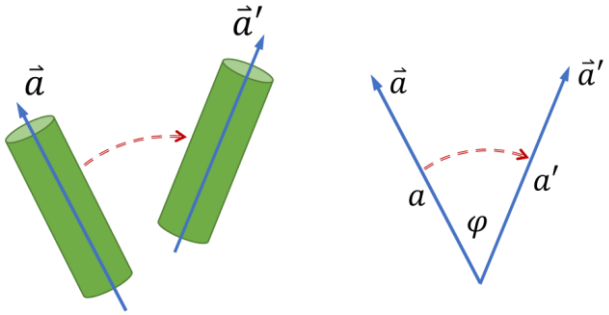


Abb. 12

Sind die Vektoren normiert, also im Fall von $a = a' = 1$, dann gilt $\varphi = \cos^{-1}(\vec{a} \cdot \vec{a}')$

3 Anhang

3.1 Begründung der Drehmatrix-Formeln

12

(ANHANG ZUR 2D-ROTATION)

Betrachten wir zuerst nur die Drehung der **Basisvektoren** $\hat{x} = \begin{bmatrix}1\\0\end{bmatrix}$ und $\hat{y} = \begin{bmatrix}0\\1\end{bmatrix}$.

$$\hat{x} = \begin{bmatrix}1\\0\end{bmatrix} \rightarrow \hat{x}' = \begin{bmatrix}\cos(\varphi)\\\sin(\varphi)\end{bmatrix} \text{ und } \hat{y} = \begin{bmatrix}0\\1\end{bmatrix} \rightarrow \hat{y}' = \begin{bmatrix}-\sin(\varphi)\\\cos(\varphi)\end{bmatrix}$$ (Siehe Abb. 13)

Eine allfällige Drehmatrix muss also folgende Bedingung erfüllen:

$$D \cdot \begin{bmatrix}\mathbf{1}\\\mathbf{0}\end{bmatrix} = \begin{bmatrix}d_{11} & d_{12}\\d_{21} & d_{22}\end{bmatrix} \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}d_{11}\\d_{21}\end{bmatrix} = \begin{bmatrix}cos(\varphi)\\sin(\varphi)\end{bmatrix}$$

$$d_{11} = \cos(\varphi)\,; d_{21} = \sin(\varphi)$$

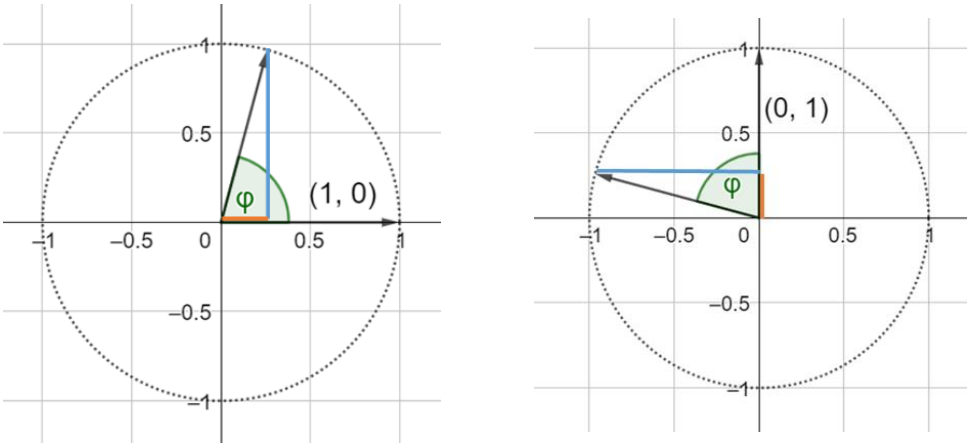


Abb. 13

Ähnliches gilt für den zweiten Basisvektor:

$$D \cdot \begin{bmatrix}\mathbf{0}\\\mathbf{1}\end{bmatrix} = \begin{bmatrix}d_{11} & d_{12}\\d_{21} & d_{22}\end{bmatrix} \cdot \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}d_{12}\\d_{22}\end{bmatrix} = \begin{bmatrix}-\sin(\varphi)\\\cos(\varphi)\end{bmatrix}$$

$$d_{12} = -\sin(\varphi)\,; d_{22} = \cos(\varphi)$$

$$D = \begin{bmatrix}\cos(\varphi) & -\sin(\varphi)\\\sin(\varphi) & \cos(\varphi)\end{bmatrix}$$

Dass diese Drehmatrix für jeden Punkt $X = \begin{bmatrix}x\\y\end{bmatrix}$ gültig ist, zeigt folgende Überlegung.

$$D \cdot X = D \cdot \begin{bmatrix} x \\ y \end{bmatrix} = D \cdot \left(\begin{bmatrix} x \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ y \end{bmatrix} \right) = D \cdot \begin{bmatrix} x \\ 0 \end{bmatrix} + D \cdot \begin{bmatrix} 0 \\ y \end{bmatrix} = \cdots$$

$$\ldots = D \cdot x \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + D \cdot y \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = x \cdot \left(D \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) + y \cdot \left(D \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) =$$

$$= x \cdot \begin{bmatrix} cos(\varphi) \\ sin(\varphi) \end{bmatrix} + y \cdot \begin{bmatrix} -sin(\varphi) \\ cos(\varphi) \end{bmatrix} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} =$$

$$= \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{bmatrix} \cdot X$$

Die Drehmatrix, die die Basisvektoren dreht, dreht auch jeden beliebigen Punkt.

3.2 3D-Rotation um eine beliebige Achse

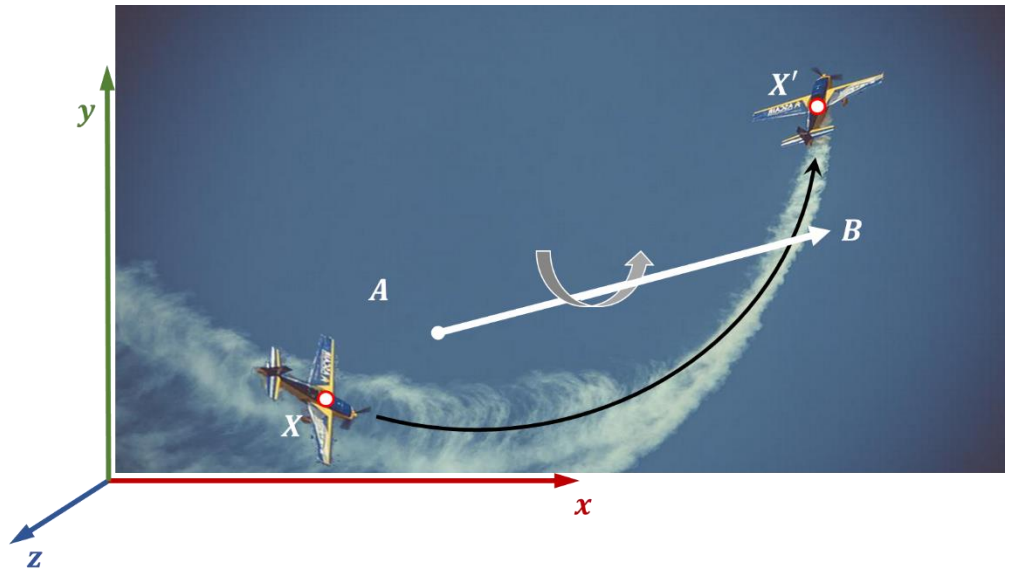


Abb. 14

Die Rotationsmatrix $\boldsymbol{R}_{\vec{r},\alpha}$ für eine Drehung einer Figur um den Winkel α um eine beliebige Achse $\vec{r} = \overrightarrow{AB}$ erfordert mehrere Schritte. Dabei wird zuerst die Rotationsachse $\vec{r}$ in die z-Achse des Koordinatenursprungs transferiert („*World to Local*“), dort die gewünschte Drehmatrix $\boldsymbol{R}_{z,\alpha}$ angehängt und danach wieder zurück transferiert („*Local to World*“).

1. **TRANSFORMATION** „*World to Local*“

a. Translation der Drehachse

$$\vec{r} = \overrightarrow{AB}$$

in den Ursprung des Koordinatensystems:

$$T_{\vec{r}\to\mathcal{O}} \mathrel{\widehat{=}} T_{-A} = \begin{bmatrix} 1 & 0 & 0 & -a_x \\ 0 & 1 & 0 & -a_y \\ 0 & 0 & 1 & -a_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

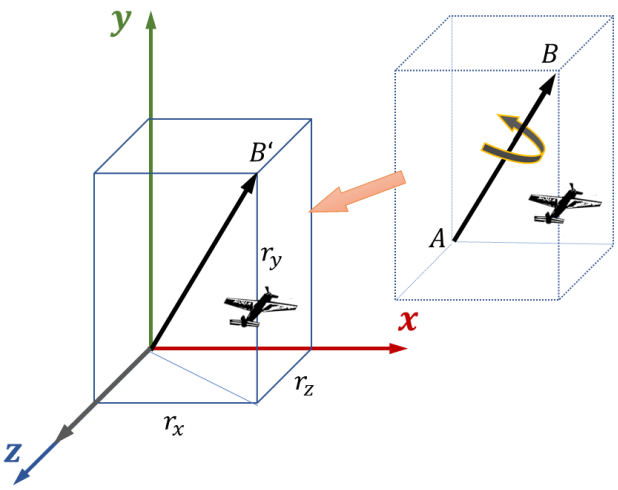


Abb. 15: Translation in den Ursprung

b. Drehung in die $[yz]$-Ebene:

$$\Theta = \tan^{-1}\left(\frac{r_x}{r_z}\right)$$

$$R_{[\to yz]} = \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

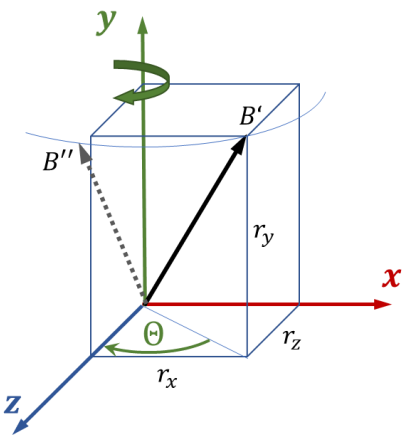


Abb. 16: Drehung in die $[yz]$-Ebene

c. Drehung um die x-Achse in die z-Achse:

$$\phi = \tan^{-1}\left(\frac{r_y}{\sqrt{r_x^2 + r_z^2}}\right)$$

$$R_{[z]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

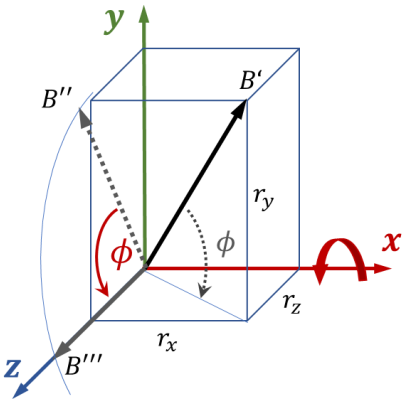
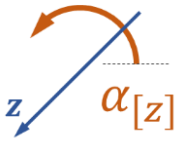


Abb. 17: Drehung um die x-Achse

2. **Rotationsmatrix für den gewünschten Drehwinkel** α um die z-Achse:

$$\boldsymbol{R}_{\boldsymbol{z},\boldsymbol{\alpha}} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3. **Rücktransformation** „*Local to World*“:

   Die Rücktransformation nach der Drehung erfolgt in umgekehrter Reihenfolge:

$$\boldsymbol{R}_{[z]}^{-1} \rightarrow \boldsymbol{R}_{[\rightarrow yz]}^{-1} \rightarrow \boldsymbol{T}_{\vec{r}\rightarrow\mathcal{O}}^{-1}$$

   Dies ergibt für jeden Vertex Y im lokalen Modell die Multiplikationskette

$$\left(T_{\vec{r}\rightarrow\mathcal{O}}^{-1} \cdot \left(R_{[\rightarrow yz]}^{-1} \cdot \left(R_{[z]}^{-1} \cdot Y\right)\right)\right)$$

   (Alle Translations- und Rotationsmatrizen sind invertierbar!)

Auf Grund der Assoziativität der Matrixmultiplikation lässt sich die gesamte Transformationskette für die Rotation um die Achse $\vec{r} = \overrightarrow{AB}$ in einer einzigen Transformationsmatrix $\boldsymbol{R}_{\vec{r},\boldsymbol{\alpha}}$ zusammenfassen.

$$\boldsymbol{R}_{\vec{r},\boldsymbol{\alpha}} = \underbrace{T_{\vec{r}\rightarrow\mathcal{O}}^{-1} \cdot R_{[\rightarrow yz]}^{-1} \cdot R_{[z]}^{-1}}_{Local\ to\ World} \cdot \boldsymbol{R}_{\boldsymbol{z},\boldsymbol{\alpha}} \cdot \underbrace{R_{[z]} \cdot R_{[\rightarrow yz]} \cdot T_{\vec{r}\rightarrow\mathcal{O}}}_{World\ to\ Local}$$

Die 3D-Rotation um eine beliebige Rotationsachse lässt sich somit auch durch eine einzige Matrixmultiplikation mit allen Vertices X des Modells berechnen.

$$X' = \boldsymbol{R}_{\vec{r},\boldsymbol{\alpha}} \cdot X$$

Bildliche Zusammenfassung:

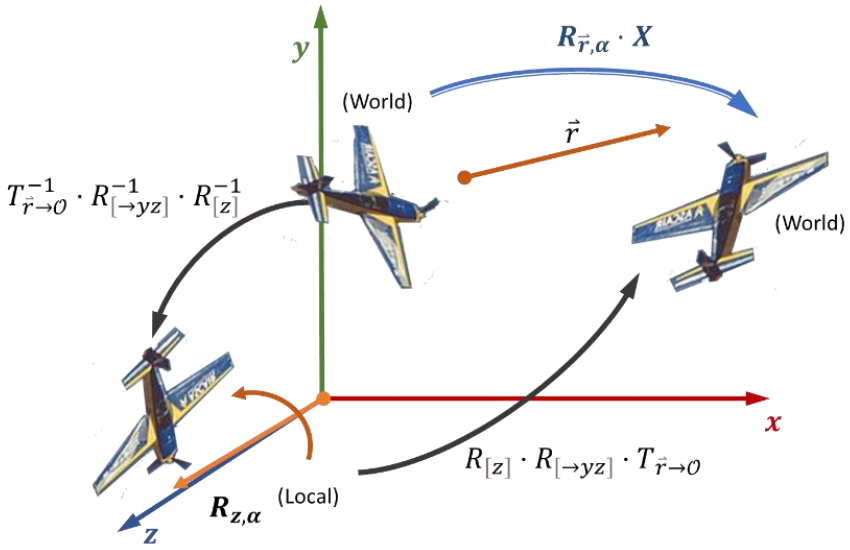


Abb. 18: Transformation $X' = \boldsymbol{R}_{\vec{r},\alpha} \cdot \boldsymbol{X}$