Computergrafik

Die Mathematik hinter OpenGL, DirectX, etc.

Vom Model-Space zum View-Space

UHLMAN

Inhalt

1 Einleitung

Um eine 3D-Szene fotorealistisch auf einen 2D-Bildschirm abzubilden, wird die Oberfläche der einzelnen Objekte in kleine Untereinheiten (sogenannte Primitive) zerlegt.

2

Dabei wird ein Netz an ***Oberflächenpunkten*** (***Vertex*** - lateinisch: Wirbel, Scheitel. Plural hier: Vertices) koordinatenmäßig erfasst und so die Modelloberfläche durch einfache geometrische Elemente angenähert, meist Vierecke oder Dreiecke.

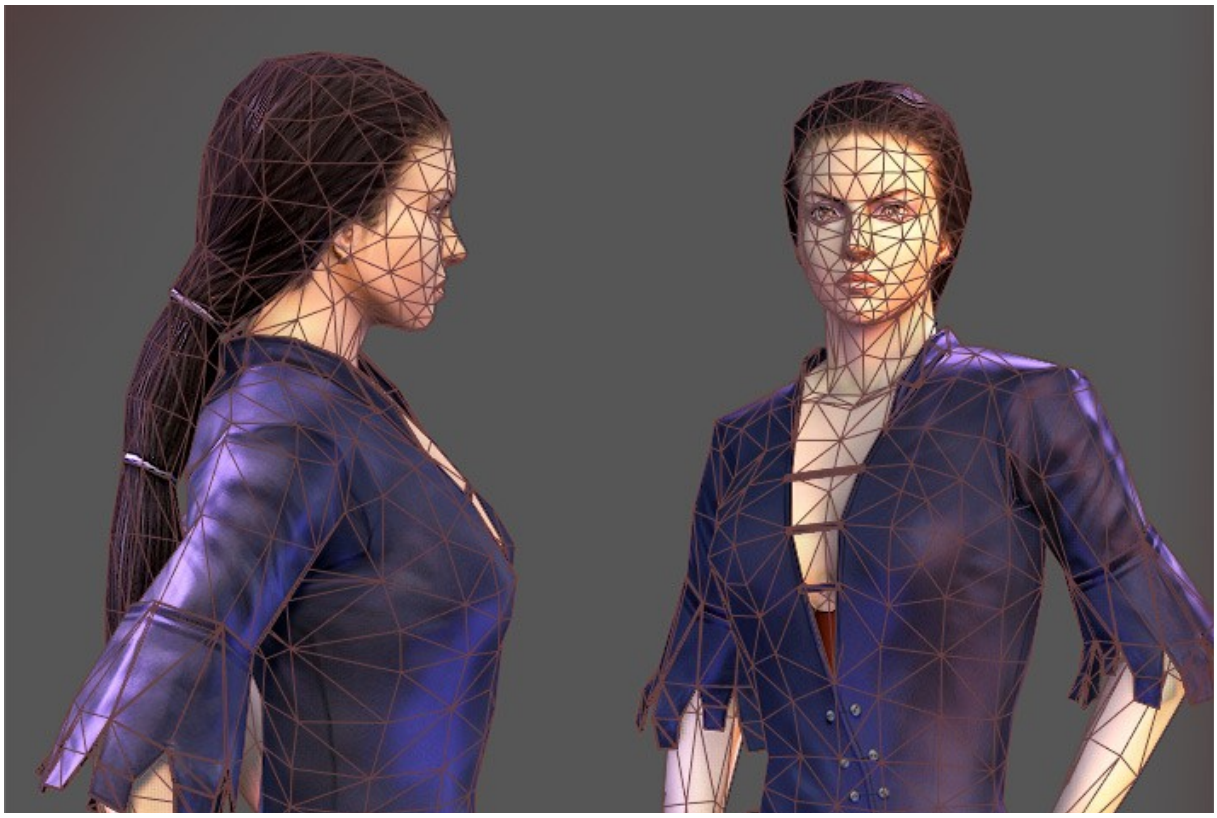


Abb. 1: (https://gdbooks.gitbooks.io/legacyopengl/content/Chapter3/Triangles.html)

Die Rendering-Engine oder Game-Engine wandelt diese bei der Weiterverarbeitung letztlich wieder in Dreiecke um.

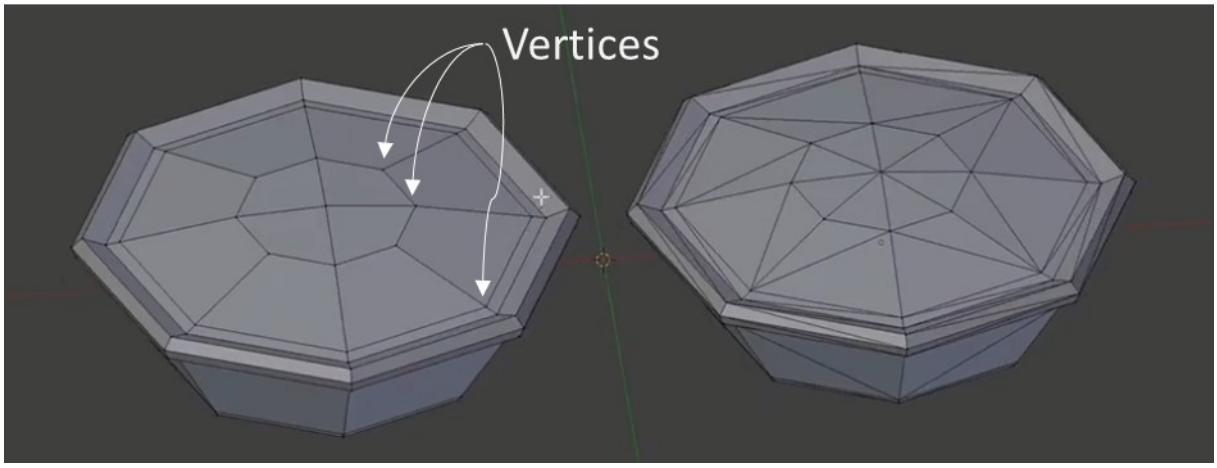
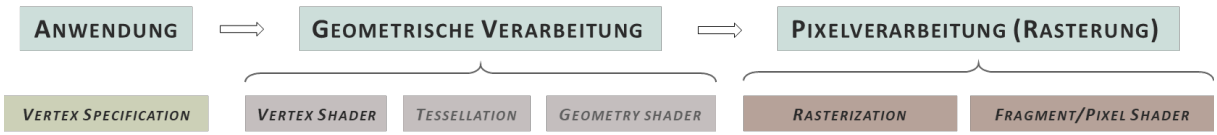


Abb. 2

2 Die Grafik-Pipeline

Die sogenannte **Grafik-Pipeline**, eine Abfolge von Algorithmen zur Darstellung einer 3D-Szene auf einen 2D-Bildschirm, lässt sich grob in drei Abschnitte gliedern. In den einzelnen Abschnitten führen bestimmte Hard- oder Software-Module, sogenannte „ **SHADER** “, die Bearbeitungsschritte des Renderings durch.



Die APPLIKATION (z.B. die Gamesoftware) sendet die Vertices und damit die Grundformen und Netze der Modelle zur Verarbeitung in die Pipeline.

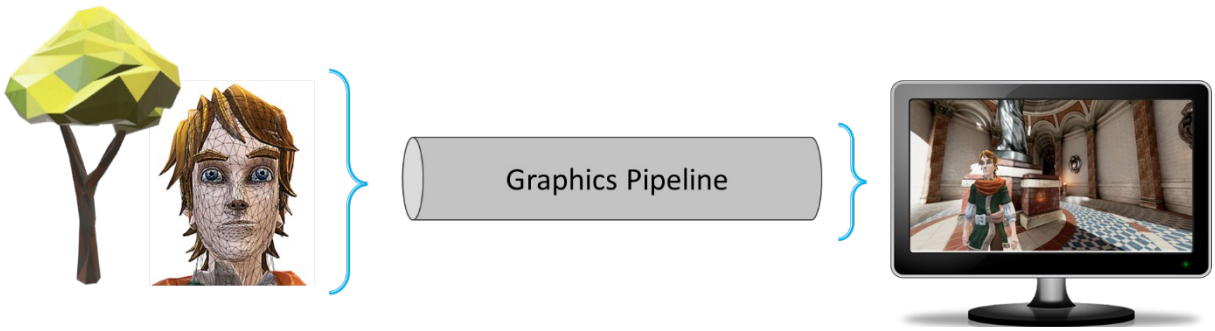


Abb. 3

Zentrales Thema in diesem Teil ist die GEOMETRISCHEN VERARBEITUNG. Sie wird im Wesentlichen vom **VERTEX-SHADER** im Grafikprozessor (GPU) durchgeführt. Im **VERTEX-SHADER** werden die Modelle mit ihrer Netzstruktur durch Skalierung, Drehung und Verschiebung in mehreren Schritten an die richtige Stelle in der Szene platziert.

DIE SCHRITTE IM VERTEX SHADER FÜHREN IN VERSCHIEDENE GEOMETRISCHE RÄUME:

MODEL-SPACE
Der Raum, in dem die einzelnen Modelle, Figuren etc. kreiert werden.

WORLD-SPACE
Der Raum in dem die gesamte Szene dargestellt wird.

VIEW-SPACE
Der Teil der Szene, der für den Betrachter dargestellt werden soll.

(PROJECTION-SPACE (NDC-SPACE)
Mit räumlicher Perspektive versehener VIEW-SPACE, komprimiert dargestellt in einem Standardwürfel. Dieser Schritt wird erst im nächsten Teil abgehandelt.)

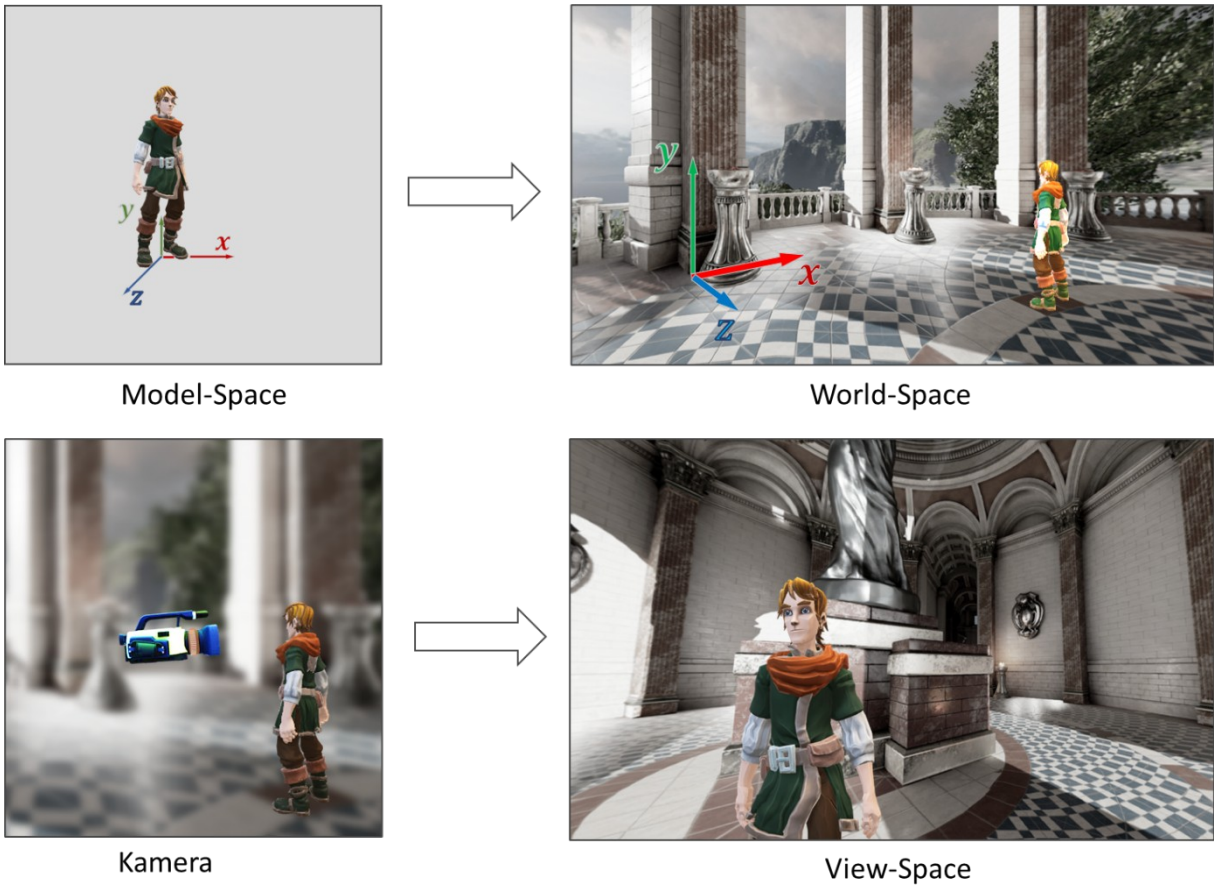


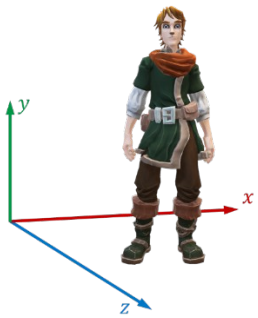
*Abb. 4: Vom Model-Space zum Bildschirm. Der Weg eines virtuellen Modells in seine virtuelle Umgebung und die Darstellung der Szene am Bildschirm. (**Bemerkung**: Die bildlich dargestellten Räume sind in Wirklichkeit nur Zahlenarrays, das Bild entsteht erst am Monitor. Vor allem beinhalten World- und View-Space entgegen der obigen Darstellung noch keine Perspektive.)*

Bei jedem Schritt aus einem Raum in den nächsten werden die Koordinaten aller relevanten Vertices neu berechnet.

KOORDINATENSYSTEME:

Gegenwärtig ist der Standard in der Industrie das rechtshändige XYZ-Koordinatensystem, bei dem x nach rechts zeigt, y nach oben und z nach außen zeigt, also aus dem Bildschirm herauskommt. Grafiksoftware wie Maya und OpenGL verwenden ein rechtshändiges Koordinatensystem, während DirectX, pbrt und PRMan ein linkshändiges Koordinatensystem verwenden. (Wikipedia 2022)

In diesem Skriptum verwenden wir ***rechtshändige Koordinatensysteme***.



3 Die Transformationen im Vertex-Shader

Ein Überblick zur Einordnung: (Vergleiche auch Abb. 4 aus Kapitel 2, Die)

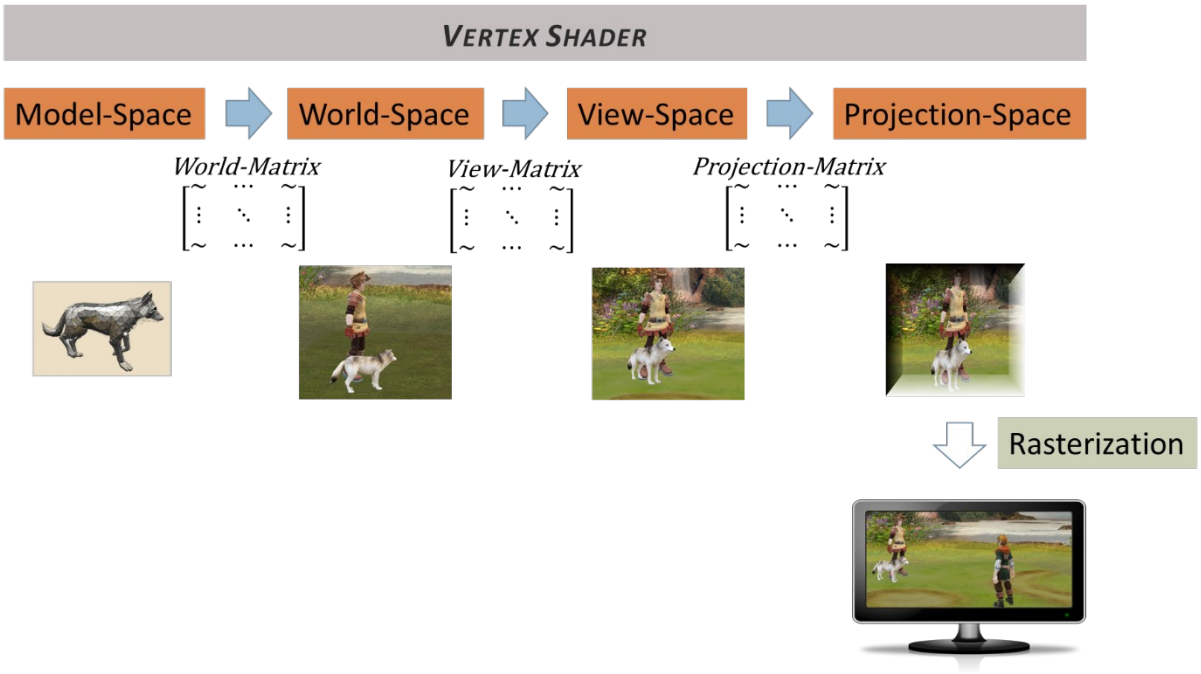


Abb. 5: Vom einzelnen Modell zur Ansicht am Monitor.

Im **VERTEX SHADER** erfolgen die Multiplikationen der Vertices mit geeigneten Transformationsmatrizen und erzeugen so die verschiedenen Ansichten der 3D-Szene.

Dieser Abschnitt soll an Hand eines Beispiels näher erläutert werden.
Beispiel: „Die Begegnung“

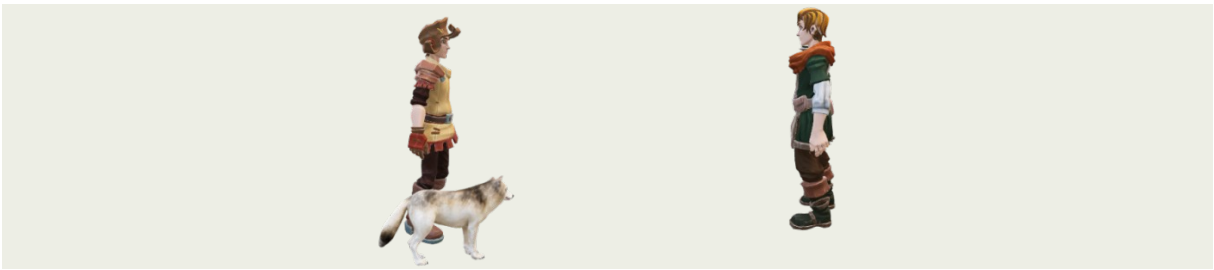


Abb. 6

1.1 Der Model-Space

IM ERSTEN SCHRITT WERDEN DIE GRUNDELEMENTE ENTWORFEN.

Die Festlegung der Vertices erfolgt noch im vorherigen Abschnitt der Pipeline. Der „Raum“, in dem die Objekte konstruiert werden, wird oft MODEL-SPACE genannt. Der Koordinatenursprung liegt dabei in dem für das Modell immanenten Rotationsmittelpunkt.

BEISPIEL (FORTSETZUNG)

ANGABE: Wolf, linke Ohrspitze $Q_0=(5\,2-1.5)$; Maße in cm

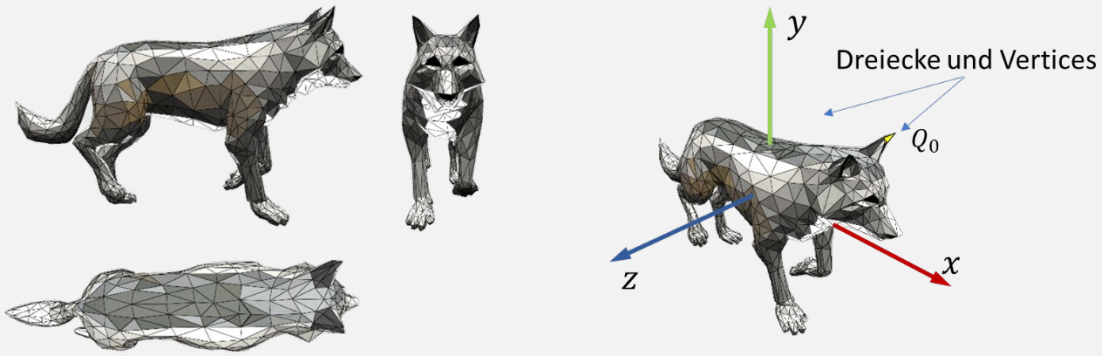


Abb. 7: Linke Ohrspitze $Q_0=(5\,3-1.5)$ und Gittermodell des Wolfes im Model-Space.

1.2 Der World-Space

IM ZWEITEN SCHRITT WERDEN DIE ELEMENTE IN DIE SZENE TRANSFERIERT.

Abb. 8: Der WORLD-SPACE

Die *Model-World-Matrix* M (kurz: *World-Matrix*) transferiert alle Vertices der im MODEL-SPACE entworfenen Modelle in die Szenerie, den WORLD-SPACE.

$$\{Obj\}=M\cdot\{Model\}$$

Das Modell wird dazu auf die korrekte Größe skaliert, in die richtige Position gedreht und an seinen Ort in der Szene transferiert.

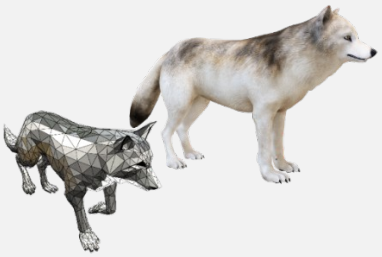
$$\{Obj\} = \underbrace{T \cdot R \cdot S}_{M} \cdot \{Model\}$$

2

BEISPIEL (FORTSETZUNG)

AUFGABENSTELLUNG: Gesucht ist die *Model-World-Matrix* M für das Objekt Wolf.

ANGABEN:



Maßstab $Modell : Objekt = 1:10$

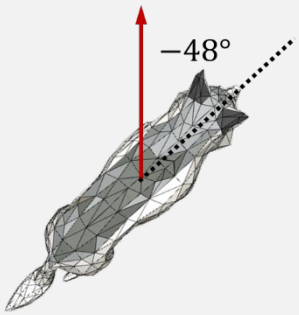
S: **Skalierung** $S = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

R: **Rotation** mit $+48°$ um die y-Achse:

$R = R_y = ¿$

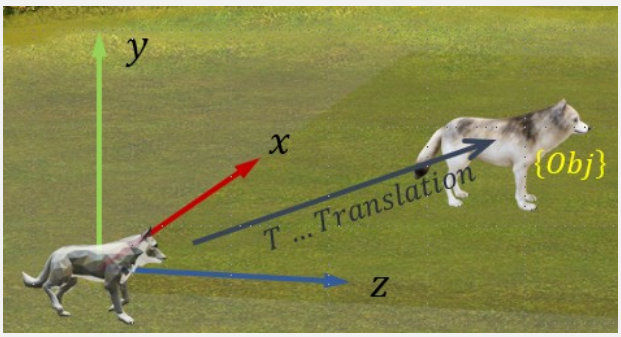
$$¿ \begin{bmatrix} \cos(-48°) & 0 & \sin(-48°) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-48°) & 0 & \cos(-48°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = ¿$$

$$¿ \begin{bmatrix} 0{,}669 & 0 & -0{,}743 & 0 \\ 0 & 1 & 0 & 0 \\ 0{,}743 & 0 & 0{,}669 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



T: **Translation** in den WORLD-SPACE

$$T = \begin{bmatrix} 1 & 0 & 0 & 300 \\ 0 & 1 & 0 & 56 \\ 0 & 0 & 1 & 260 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Nach Skalierung, Drehung und Translation liegt Q_0 Q_W im WORLD-SPACE:

Q_0 Q_W durch $Q_W = T \cdot \left(R_y \cdot \left(S \cdot Q_0\right)\right) = \overbrace{T \cdot R_y \cdot S}^{\text{geometrisch}} \cdot Q_0 = M \cdot Q_0$

Berechnung

Damit berechnet sich die *Model-World-Matrix* zu

$$M = T \cdot R \cdot S = \begin{bmatrix} 1 & 0 & 0 & 300 \\ 0 & 1 & 0 & 56 \\ 0 & 0 & 1 & 260 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0{,}669 & 0 & -0{,}743 & 0 \\ 0 & 1 & 0 & 0 \\ 0{,}743 & 0 & 0{,}669 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \dots$$

… mit dem Ergebnis

$$\text{Ergebnis} \rightarrow \quad M = \begin{bmatrix} 6{,}691 & 0 & -7{,}431 & 300 \\ 0 & 10 & 0 & 56 \\ -7{,}431 & 0 & 6{,}691 & 260 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Beispiel: Der Vertex der Ohrspitze hat im WORLD-SPACE die Koordinaten

$$Q_W = M \cdot Q_0 = \begin{bmatrix} 6{,}691 & 0 & -7{,}431 & 300 \\ 0 & 10 & 0 & 56 \\ -7{,}431 & 0 & 6{,}691 & 260 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 2 \\ -1{,}5 \\ 1 \end{bmatrix} \text{¿}^{*}\text{?} \begin{bmatrix} 344{,}6037 \\ 76 \\ 287{,}1203 \\ 1 \end{bmatrix}$$

* *Ergebnis bei Berechnung mit 10-Nachkommastellen-Genauigkeit*

1.3 Der View-Space

IM DRITTEN SCHRITT WIRD DIE SICHT AUF DIE SZENE DEFINIERT.

Abb. 9: Der VIEW-SPACE

Der VIEW-SPACE ist der WORLD-SPACE aus Sicht der Kamera. Dabei ist die Kamera ebenfalls ein Objekt im WORLD-SPACE.

Die Kamera ist in ihrer Grundform (im MODEL-SPACE) *standardmäßig in Richtung* $(-z)$*-Achse* orientiert. (Abb. 10)

Es gilt die *World-Matrix* der Kamera zu berechnen. Dies ist die Matrix, welche die Kamera im WORLD-SPACE platziert. Sie wird auch als „*Kamera-Matrix*“ C bezeichnet. Der „Up-Vektor“ zeigt in Richtung y-Achse. Von dieser Grundposition aus wird die Kamera über Rotation und Translation in den WORLD-SPACE transferiert.

$$\{K_{World}\} = C \cdot \{K\}$$

2

Die **inverse Matrix** C^{-1} transferiert die Kamera wieder zurück in den URSPRUNG. (Abb. 10)

WORLD-SPACE VIEW-SPACE

Abb. 10

Wird die Transformationsmatrix C^{-1} nicht nur auf die Kamera, sondern auch auf alle anderen Objekte des WORLD-SPACE angewandt, so werden diese mittransferiert. Die Matrix C^{-1} transferiert also den gesamten WORLD-SPACE mit allen in ihm enthaltenen Objekten in den sogenannten VIEW-SPACE, *in dem sich die Kamera im Koordinatenursprung befindet*.

Der VIEW-SPACE stellt den WORLD-SPACE dar, wie wir ihn durch die virtuelle Kamera sehen. C^{-1} wird deshalb auch als *View-Matrix* V bezeichnet.

Als *Model-View-Matrix* $[VM]$ bezeichnet man die Matrix, die die Vertices der Objekte direkt aus dem MODEL-SPACE in den VIEW-SPACE transferiert.

$$C^{-1} \cdot M \cdot \{Model\} = \underbrace{V \cdot M}_{Model-View-¿M-V¿} \cdot \{Model\}$$

Die Matrix M repräsentiert dabei alle Transformationen, die die Vertices des Modells und damit das gesamte Objekt in den WORLD-SPACE gebracht haben.

Beispiel (Fortsetzung)

Die Kamera

a) Kamera-Position $K=(-50\,400\,800)$

b) Kameraneigung: $+20°$ x-Achse ($R_{x:+20°}$) … (Neigen nach unten ↶=↻ math. pos.)

c) Kameradrehung (Schwenk): $-30°$ y-Achse ($R_{y:-30°}$) … (Schwenken nach rechts ↻, math. neg.)

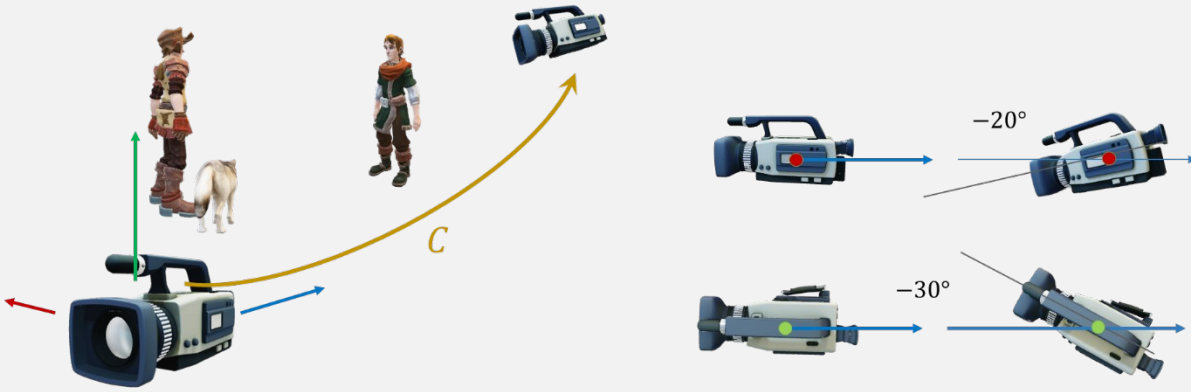


Abb. 11

Die *Model-World-Matrix* C **der Kamera** K: Die Reihenfolge der Transformationen (siehe Angabe) ist wichtig! Zuerst neigen, dann schwenken (dadurch ist die jeweilige Drehachse immer eine Koordinatenachse) und zuletzt verschieben.

(*Matrizenberechnungen mit Online-Rechner https://matrixcalc.org*)

$$R_{x:+20°} \quad R_{y:-30°} \quad T_{\begin{pmatrix}-50\\400\\800\end{pmatrix}} \quad C=T_{\begin{pmatrix}-50\\400\\800\end{pmatrix}} \cdot R_{y:-30°} \cdot R_{x:+20°}$$

Die **Skalierung** der Kamera entfällt bzw. ist $S_C=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$.

Betrachten wir zuerst die **Rotation**en!

$$R_C=\overbrace{R_{y:-30°}\cdot R_{x:+20°}}^{\text{geometrisch}}$$

Berechnung

$$R_C=\begin{bmatrix}0{,}866&0&-0{,}5&0\\0&1&0&0\\0{,}5&0{,}5&0{,}866&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}1&0&0&0\\0&0{,}940&0{,}342&0\\0&-0{,}342&0{,}940&0\\0&0&0&1\end{bmatrix}=\begin{bmatrix}0{,}866&-0{,}171&-0{,}470&0\\0&0{,}940&-0{,}342&0\\0{,}5&0{,}296&0{,}814&0\\0&0&0&1\end{bmatrix}$$

Der **Translation**svektor ist mit $[-50 \quad 400 \quad 800]$ gegeben. Damit ist

$$C = T_C \cdot R_C = \begin{bmatrix} 0{,}866 & -0{,}171 & -0{,}470 & -50 \\ 0 & 0{,}940 & -0{,}342 & 400 \\ 0{,}5 & 0{,}296 & 0{,}814 & 800 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Die *View-Matrix* V (bzw. *Kamera-Matrix*) ist die dazu Inverse Matrix.

Die *View-Matrix* V

$$\text{Ergebnis} \rightarrow \quad V = C^{-1} = \begin{bmatrix} 0{,}866 & 0 & 0{,}5 & -356{,}699 \\ -0{,}171 & 0{,}940 & 0{,}296 & -621{,}386 \\ -0{,}470 & -0{,}342 & 0{,}814 & -537{,}722 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gemeinsam mit der *Model-World-Matrix* M des Wolfes ergibt das die

Model-View-Matrix (MV-Matrix):

$$[MV] = V \cdot M = \begin{bmatrix} 0{,}866 & 0 & 0{,}5 & -356{,}699 \\ -0{,}171 & 0{,}940 & 0{,}296 & -621{,}386 \\ -0{,}470 & -0{,}342 & 0{,}814 & -537{,}722 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6{,}691 & 0 & -7{,}431 & 300 \\ 0 & 10 & 0 & 56 \\ 7{,}431 & 0 & 6{,}691 & 260 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Ergebnis} \rightarrow \quad [MV] = \underbrace{V \cdot M}_{Model¿} View¿ = \begin{bmatrix} 2{,}079 & 0 & -3{,}090 & 33{,}109 \\ -3{,}3455 & 9{,}397 & 3{,}253 & -543{,}058 \\ -9{,}19158 & -3{,}420 & 8{,}937 & -486{,}242 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

NOTATION: $[MV] = [Model¿View] = [M¿V] =: (V \cdot M)$

Z.B.: Aus der Kameraperspektive hat der Vertex P_0 (Ohrspitze) die Koordinaten

$$Q_{View} = V \cdot M \cdot Q_0 = \begin{bmatrix} 2{,}079 & 0 & -3{,}090 & 33{,}109 \\ -3{,}3455 & 9{,}397 & 3{,}253 & -543{,}058 \\ -9{,}19158 & -3{,}420 & 8{,}937 & -486{,}242 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 2 \\ -1{,}5 \\ 1 \end{bmatrix} ¿^{¿} \begin{bmatrix} 48{,}140 \\ -545{,}867 \\ -552{,}446 \\ 1 \end{bmatrix}$$

** Ergebnis bei Berechnung mit 10-Nachkommastellen-Genauigkeit*

$(-z)$ *z-Buffer*

Der $(-z)$-Wert wird im z-Buffer gespeichert!!!

Die negativen Zahlenwerte mögen jetzt überraschen, sind aber aus Sicht der Kamera verständlich. (Vergleiche Abb. 10)

4 Hierarchie

Objekte in einer Computergrafik sind meist insofern komplex, als sie aus etlichen einfacheren Objekten zusammengesetzt sind und diese wiederum aus noch einfacheren, bis dies bei elementaren geometrischen Grundelementen endet. Diese hierarchische Struktur ist der Schlüssel zum Umgang mit Komplexität in der Computergrafik generell und speziell in computergenerierten Animationen, Spielfilmen und Games.

1.1 Erstellen komplexer Objekte

Üblicherweise wird ein Objekt zentral in einem ihm inhärenten Koordinatensystem (MODEL-SPACE) entworfen, wobei meist ein bequemer Bezugspunkt als Ursprung verwendet wird.

Die richtige Platzierung in der Szenerie (WORLD-SPACE) erfolgt anschließend durch eine geeignete Transformationsmatrix, der sog. *World-Matrix*, welche aus einer Verkettung von Skalierungen, Rotationen und Translationen entsteht. Das Objekt wird also auf die richtige Größe gebracht, in die richtige Position gedreht und schließlich an den richtigen Ort im WORLD-SPACE geschoben.

Die Abb. 12 zeigt dies an einem Beispiel einer Gürteltasche. Nehmen wir die Person als Basisobjekt (*Parent-Object*), dann ist der Gürtel ein Unterobjekt (*Child-Object*) und die Tasche ist wiederum ein Unterelement (*Child-Object*) des Gürtels. Objekte auf derselben Stufe der Hierarchie sind sogenannte Geschwister-Objekte (*Sibling-Objects*).

Sobald ein Objekt ein Child-Element eines in der Hierarchie darüber liegenden Parent-Elements ist, betrifft jede Transformation des Parent-Elements auch das Child-Element.

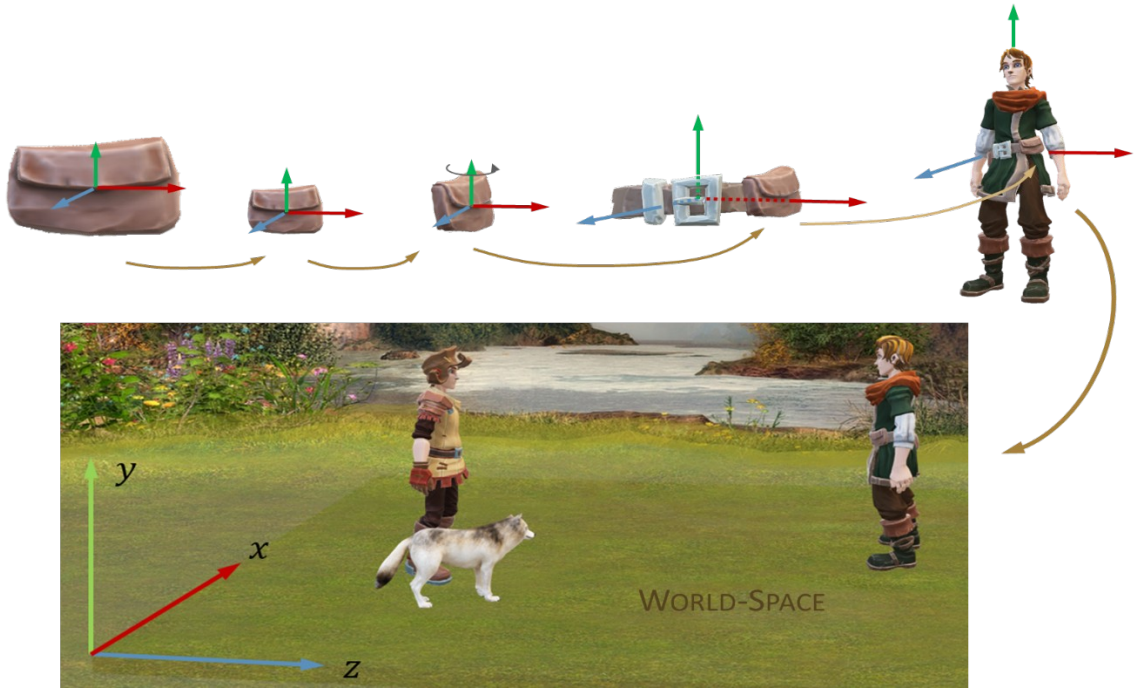


Abb. 12: Vom lokalen Model-Space in den World-Space

SCHRITT 1

Das Grundelement der in Abb. 12 dargestellten Hierarchie ist die Tasche (Futteral F), der durch die Skalierungsmatrix S_F die angepassten Maße für den Gürtel erhält. Der Gürtel (G) ist das Parent-Element des Futterals. Im Koordinatensystem des Gürtels liegt das Futteral nicht im Ursprung. Die Transformationsmatrizen R_F und T_F drehen und schieben es in die richtige Position.

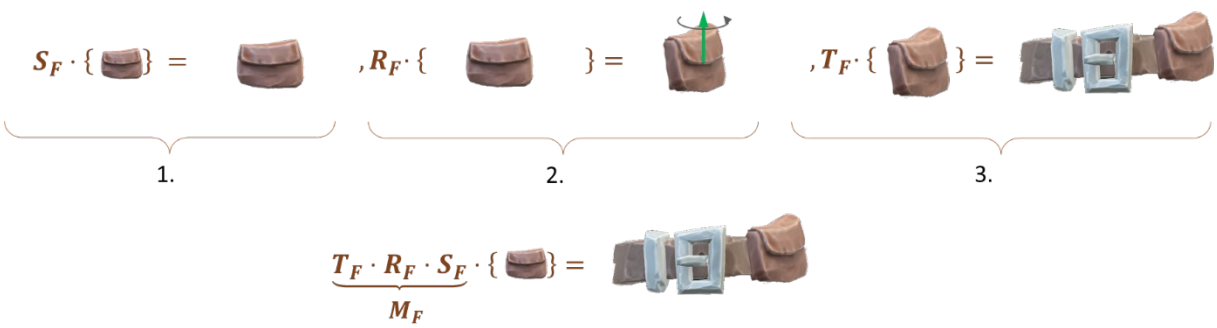
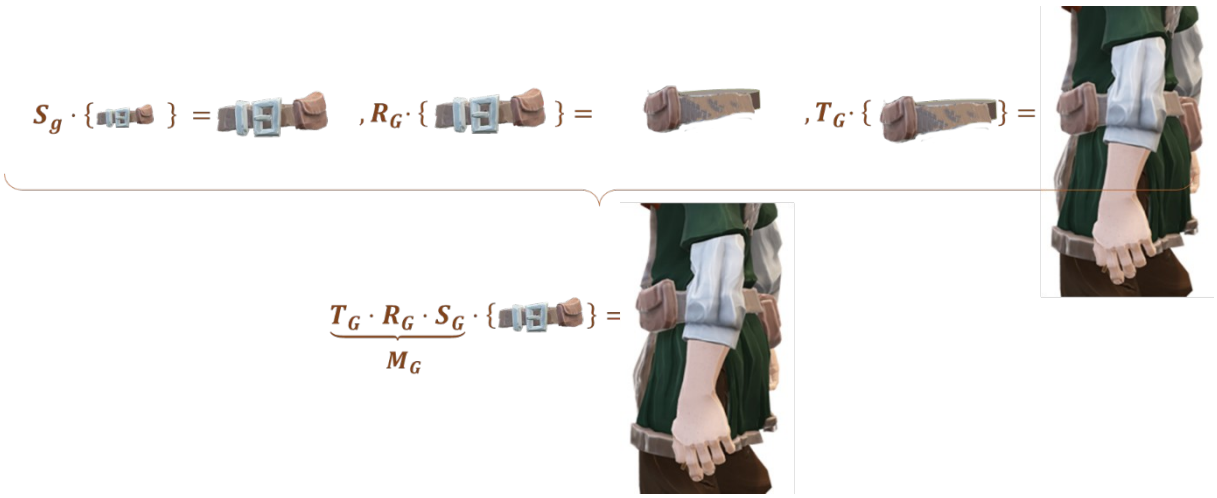


Abb. 13

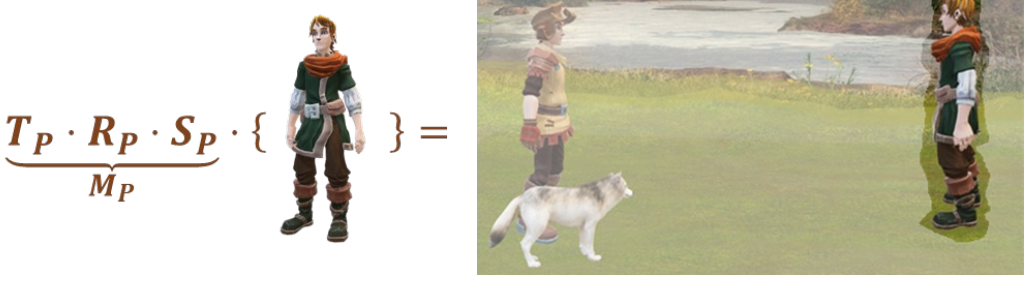
SCHRITT 2

Anschließend betreffen alle Transformationen, die den Gürtel betreffen, auch alle seine Unterelemente.



SCHRITT 3

Zuletzt wird die Person mitsamt ihren Unterelementen in den WORLD-SPACE transferiert.

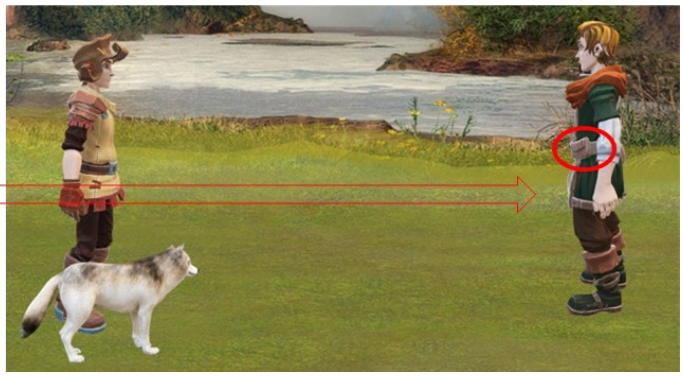


World-Matrix

Die *World-Matrix* bringt ein Objekt *direkt* in den WORLD-SPACE. Am Beispiel des *Futteral* F als Grundelement des Gürtels ist das

$$W_F \;\text{¿}\; M_P \cdot M_G \cdot M_F \;\text{¿}\; \underbrace{T_P \cdot R_P \cdot S_P}_{M_P} \cdot \underbrace{T_G \cdot R_G \cdot S_G}_{M_G} \cdot \underbrace{T_F \cdot R_F \cdot S_F}_{T_F}$$

$W_F \cdot \{$  $\} =$



Bemerkung

Die Rotationsmatrix R hat immer die Form

$$R = \begin{bmatrix} & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

wobei jeweils für eine beliebige aber konkrete Zahl steht.

Die Multiplikation mit der Skalierungsmatrix ergibt

$$R \cdot S = \begin{bmatrix} & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cdot s_x & \cdot s_y & \cdot s_z & 0 \\ \cdot s_x & \cdot s_y & \cdot s_z & 0 \\ \cdot s_x & \cdot s_y & \cdot s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Die Multiplikation mit der Translationsmatrix ergibt dann

$$T \cdot R \cdot S = \begin{bmatrix} \cdot s_x & \cdot s_y & \cdot s_z & t_x \\ \cdot s_x & \cdot s_y & \cdot s_z & t_y \\ \cdot s_x & \cdot s_y & \cdot s_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Die Komponenten der Rotationsmatrix bilden gewissermaßen die Basis, spaltenweise multipliziert mit dem entsprechenden Skalierungsfaktor sowie die letzte Spalte ersetzt durch den Translationsvektor.

5 Die Look-At-Methode

Beispiel (Aufgabe) Kameraposition

2

Es geht hier um die Angabe der Position und die Richtung der Kamera durch Ortsvektor, Up-Vektor und Blickpunkt (Blickrichtung)

Um die Lage einer Kamera festzulegen, braucht man lediglich zwei Punkte. Einen Punkt (K), welcher ihre Position im Raum festlegt, und einen Blickpunkt (B), der definiert, wohin die Kamera blickt. (Look-At-Methode)

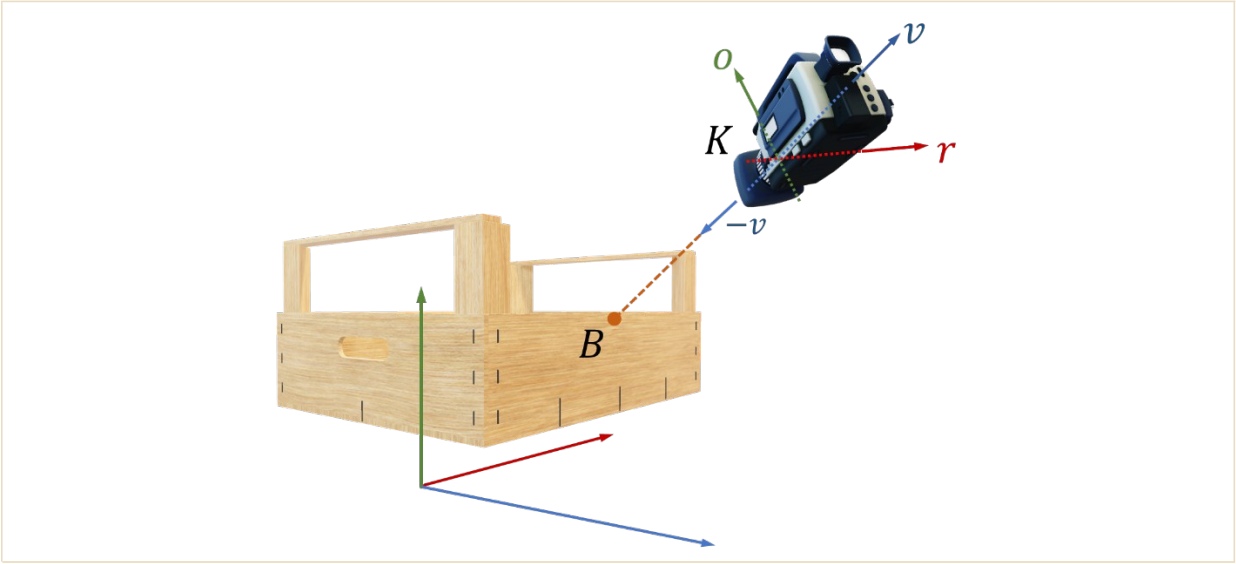


Abb. 14

Angabe:

$K=(-10 \quad 70 \quad 100)$

Blickpunkt $B=(30 \quad 40 \quad 10)$

Aufgabenstellung:

Gesucht ist das kameraeigene Koordinatensystem im World-Space. Um Verwechslungen vorzubeugen werden für die Koordinatenachsen des Kamerasystems statt x, y, z die Bezeichnungen r, o, v für *rechts*, *oben*, *vorwärts* verwendet.

Berechnung:

Leicht einzusehen ist die v-**Achse**. Sie ist der Blickrichtung entgegengesetzt.

$$\acute{v}=\acute{BK}=\begin{bmatrix}-40\\30\\90\end{bmatrix};\hat{v}=\frac{1}{\sqrt{10600}}\begin{bmatrix}-40\\30\\90\end{bmatrix}=\begin{bmatrix}-0{,}389\\0{,}291\\0{,}874\end{bmatrix}$$

Wir gehen davon aus, dass ein aufrechtes Bild durch die virtuelle Kamera erzeugt werden soll. Ein Rollen der Kamera um die v-Achse sei vorerst einmal nicht angenommen. „Aufrecht

ist im WORLD-SPACE gegeben durch die y-Achse und somit durch

$$\acute{y} = \hat{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Für die r-**Achse** ergibt das Kreuzprodukt $\acute{v} \times \acute{y}$ einen Vektor rechtwinklig zu $\acute{v}$ und $\acute{y}$ und somit in r-Richtung.

2

$$\acute{r} = \acute{y} \times \acute{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} -40 \\ 30 \\ 90 \end{bmatrix} = \begin{bmatrix} 90 \\ 0 \\ 40 \end{bmatrix}; \hat{r} = \frac{1}{\sqrt{9700}} \begin{bmatrix} 90 \\ 0 \\ 40 \end{bmatrix} = \begin{bmatrix} 0{,}914 \\ 0 \\ 0{,}406 \end{bmatrix}$$

Die nach oben gerichtete o-**Achse** der Kamera lässt sich erneut mit dem Kreuzprodukt berechnen.

$$\acute{o} = \acute{v} \times \acute{r} = \begin{bmatrix} -40 \\ 30 \\ 90 \end{bmatrix} \times \begin{bmatrix} 90 \\ 0 \\ 40 \end{bmatrix} = \begin{bmatrix} 1200 \\ 9700 \\ -2700 \end{bmatrix}; \hat{o} = \frac{1}{\sqrt{10282}} \begin{bmatrix} 1200 \\ 9700 \\ -2700 \end{bmatrix} = \begin{bmatrix} 0{,}118 \\ 0{,}957 \\ -0{,}266 \end{bmatrix}$$

Ergebnis: ➤

$$\hat{r} = \begin{bmatrix} 0{,}914 \\ 0 \\ 0{,}406 \end{bmatrix}; \hat{o} = \begin{bmatrix} 0{,}118 \\ 0{,}957 \\ -0{,}266 \end{bmatrix}; \hat{v} = \begin{bmatrix} -0{,}389 \\ 0{,}291 \\ 0{,}874 \end{bmatrix}$$

Die Kamera-Matrix C (die *World-Matrix* der Kamera) beinhaltet den Rotationsteil und den Translationsteil. Dabei wird zuerst nur gedreht:

$$C^{\iota} = R_C = \begin{bmatrix} a & d & g & 0 \\ b & e & h & 0 \\ c & f & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Betrachten wir die Achsen-Einheitsvektoren, so ergibt sich folgendes Gleichungssystem:

$$C^{\iota} \cdot \hat{x} = \hat{r}$$
$$C^{\iota} \cdot \hat{y} = \hat{o}$$
$$C^{\iota} \cdot \hat{z} = \hat{v}$$

$$\begin{bmatrix} a & d & g & 0 \\ b & e & h & 0 \\ c & f & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = \hat{r} = \begin{bmatrix} 0{,}914 \\ 0 \\ 0{,}406 \\ 0 \end{bmatrix} \begin{bmatrix} a & d & g & 0 \\ b & e & h & 0 \\ c & f & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} d \\ e \\ f \\ 0 \end{bmatrix} = \hat{o} = \begin{bmatrix} 0{,}118 \\ 0{,}957 \\ -0{,}266 \\ 0 \end{bmatrix}$$

2

$$\begin{bmatrix} a & d & g & 0 \\ b & e & h & 0 \\ c & f & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} g \\ h \\ j \\ 0 \end{bmatrix} = \hat{v} = \begin{bmatrix} -0{,}389 \\ 0{,}291 \\ 0{,}874 \\ 0 \end{bmatrix}$$ Daraus folgt die **Kamera-Matrix**

$$C = \begin{bmatrix} \hat{r}_x & \hat{o}_x & \hat{v}_x & t_x \\ \hat{r}_y & \hat{o}_y & \hat{v}_y & t_y \\ \hat{r}_z & \hat{o}_z & \hat{v}_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}914 & 0{,}118 & -0{,}389 & -10 \\ 0 & 0{,}957 & 0{,}291 & 70 \\ 0{,}406 & -0{,}266 & 0{,}874 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ZUSAMMENFASSUNG:

Look-at-Kameramatrix

$$C = \begin{pmatrix} ¿ & ¿ & ¿ & ¿ \\ \hat{r} & \hat{o} & \hat{v} & K \\ ¿ & ¿ & ¿ & ¿ \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

wobei $\hat{r}, \hat{o}, \hat{v}$ die Einheitsvektoren der x, y, z-Achsen des Kamerasystems darstellen und K der Ort bzw. Ortsvektor der Kamera ist.

6 Beispiele und Aufgaben

1.4 Basisaufgabe

2

Ein Objekt (Referenzpunkt $(0,0,0)$) soll sowohl um die x-Achse als auch um die y-Achse um jeweils 90° gedreht und dann in z-Richtung auf die Hälfte gestaucht und letztlich an den Ort $(8,5,6)$ im WORLD-SPACE verschoben werden. Die zu beachtende Reihenfolge

Skalierung $\rightarrow$ Rotation $\rightarrow$ Translation ergibt eine Multiplikationsreihe, die zu einer einzigen Transformationsmatrix M verkettet werden kann.

$$X' = \left(T \cdot \left(R \cdot \left(S \cdot X\right)\right)\right) = \left(\underbrace{T \cdot R \cdot S}_{M}\right) \cdot X \quad .$$

Genauer: $M = T \cdot R_z \cdot R_y \cdot R_x \cdot S =$ ¿

$$\text{¿} \begin{bmatrix} 1 & 0 & 0 & 8 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & 1 & 0 & 8 \\ 0 & 0 & -0{,}5 & 5 \\ -1 & 0 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Angewandt auf ein Dreieck $\triangle PQR$ mit $P = (2 \mid -4 \mid 1) \; Q = (3 \mid 0 \mid -2) \; R = (1 \mid 2 \mid 5)$ ergibt das

$$\triangle P'Q'R' = M \cdot \left\{ \triangle PQR = \begin{bmatrix} 0 & 1 & 0 & 8 \\ 0 & 0 & -0{,}5 & 5 \\ -1 & 0 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \left\{ \begin{bmatrix} 2 \\ -4 \\ 1 \\ 1 \end{bmatrix} \quad \text{¿} P' = \begin{bmatrix} 4 \\ 4{,}5 \\ 4 \\ 1 \end{bmatrix} \right.$$

$$\left. \begin{bmatrix} 3 \\ 0 \\ -2 \\ 1 \end{bmatrix} \quad \text{¿} Q' = \begin{bmatrix} 8 \\ 6 \\ 3 \\ 1 \end{bmatrix} \right.$$

$$\left. \begin{bmatrix} 1 \\ 2 \\ 5 \\ 1 \end{bmatrix} \quad \text{¿} R' = \begin{bmatrix} 10 \\ 2{,}5 \\ 5 \\ 1 \end{bmatrix} \right.$$

mit $P' = (4 \mid 4{,}5 \mid 4) \; Q' = (8 \mid 6 \mid 3) \; R' = (10 \mid 2{,}5 \mid 5)$

1.5 Look-At-Methode versus Euler-Winkel

Aufgabenstellung: Statt der Look-at-Methode hätte man auch die Kamera im Ursprung des WORLD-SPACE um die einzelnen Koordinatenachsen drehen und anschließend durch Translation in der Szene platzieren können.

2

Berechne die zur vorigen Aufgabe (Beispiel) gehörenden sogenannten „Euler-Winkel“ der Kamera!

a) Drehwinkel α für die Rotation um die x-Achse
b) Drehwinkel β für die Rotation um die y-Achse
c) Drehwinkel γ für die Rotation um die z-Achse

Ergebnis (Lösung): ➤

Betrachten wir die Rotationsmatrix bezüglich der Koordinatenachsen allgemein:

$$C^{¿}=R_C=R_x \cdot R_y \cdot R_z=¿=$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ¿$$

$$¿\begin{bmatrix} \cos(\beta)\cdot\cos(\gamma) & -\cos(\beta)\cdot\sin(\gamma) & \sin(\beta) & 0 \\ \sin(\alpha)\cdot\sin(\beta)\cdot\cos(\gamma)+\cos(\alpha)\cdot\sin(\gamma) & \cos(\alpha)\cdot\cos(\gamma)-\sin(\alpha)\cdot\sin(\beta)\cdot\sin(\gamma) & -\sin(\alpha)\cdot\cos(\beta) & 0 \\ -\cos(\alpha)\cdot\sin(\beta)\cdot\cos(\gamma)+\sin(\alpha)\cdot\sin(\gamma) & \sin(\alpha)\cdot\cos(\gamma)+\cos(\alpha)\cdot\sin(\beta)\cdot\sin(\gamma) & \cos(\alpha)\cdot\cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Das Ergebnis ist die Kamera-Matrix ohne Translation, also die zuvor mit der Look-at-Methode errechnete Matrix $C^{¿}$.

$$C^{¿}=\begin{bmatrix} 0{,}914 & 0{,}118 & -0{,}389 & 0 \\ 0 & 0{,}957 & 0{,}291 & 0 \\ 0{,}406 & -0{,}266 & 0{,}874 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} 0{,}914 & 0{,}118 & -0{,}389 & 0 \\ 0 & 0{,}957 & 0{,}291 & 0 \\ 0{,}406 & -0{,}266 & 0{,}874 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Die markierten Elemente gestatten die Berechnung der „Euler-Winkel“ α, β, γ. Dabei ist zu beachten, nur die Sinuswerte auszuwerten. Die Cosinusfunktion unterscheidet nicht den Drehsinn der Winkel (ob im oder gegen den Uhrzeigersinn), da $\cos(\varphi)=\cos(-\varphi)$, die Sinusfunktion unterscheidet dies schon.

$$¿\begin{bmatrix} \cos(\beta)\cdot\cos(\gamma) & -\cos(\beta)\cdot\sin(\gamma) & \sin(\beta) & 0 \\ \sin(\alpha)\cdot\sin(\beta)\cdot\cos(\gamma)+\cos(\alpha)\cdot\sin(\gamma) & \cos(\alpha)\cdot\cos(\gamma)-\sin(\alpha)\cdot\sin(\beta)\cdot\sin(\gamma) & -\sin(\alpha)\cdot\cos(\beta) & 0 \\ -\cos(\alpha)\cdot\sin(\beta)\cdot\cos(\gamma)+\sin(\alpha)\cdot\sin(\gamma) & \sin(\alpha)\cdot\cos(\gamma)+\cos(\alpha)\cdot\sin(\beta)\cdot\sin(\gamma) & \cos(\alpha)\cdot\cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\sin(\beta) = -0{,}389 \quad \beta = -22{,}892\,°$$
$$-\sin(\alpha)\cdot\cos(\beta) = 0{,}291 \quad \alpha = -18{,}428\,°$$
$$-\cos(\beta)\cdot\sin(\gamma) = 0{,}118 \quad \gamma = -7{,}359\,°$$

1.6 Lego Helikopter

DIE DATEN (MASSE IN MM)

2

- Helikopterkoordinaten in der Szenerie: $H=(500,340,270)$
- Aktuelle Fluggeschwindigkeit: $\acute{v}=(60,0,0)$
- Rotorgeschwindigkeit: 8 Umdrehungen/s
- Abmessungen (Abb. 15)
- Skalierungsfaktor in der Szenerie: $0{,}5$
- Video: 60 fps (frames per second, Bilder pro Sekunde)

AUFGABENSTELLUNG:

a) Berechne die *World-Matrix* des markierten Teils des Rotors!
b) Berechne die *World-Matrix* für das nächste Frame (Bild) im Video!
Beachte die Änderungen bezüglich Ort und Rotorstellung bei gleichbleibender Flugrichtung und unverändertem Rollwinkel)

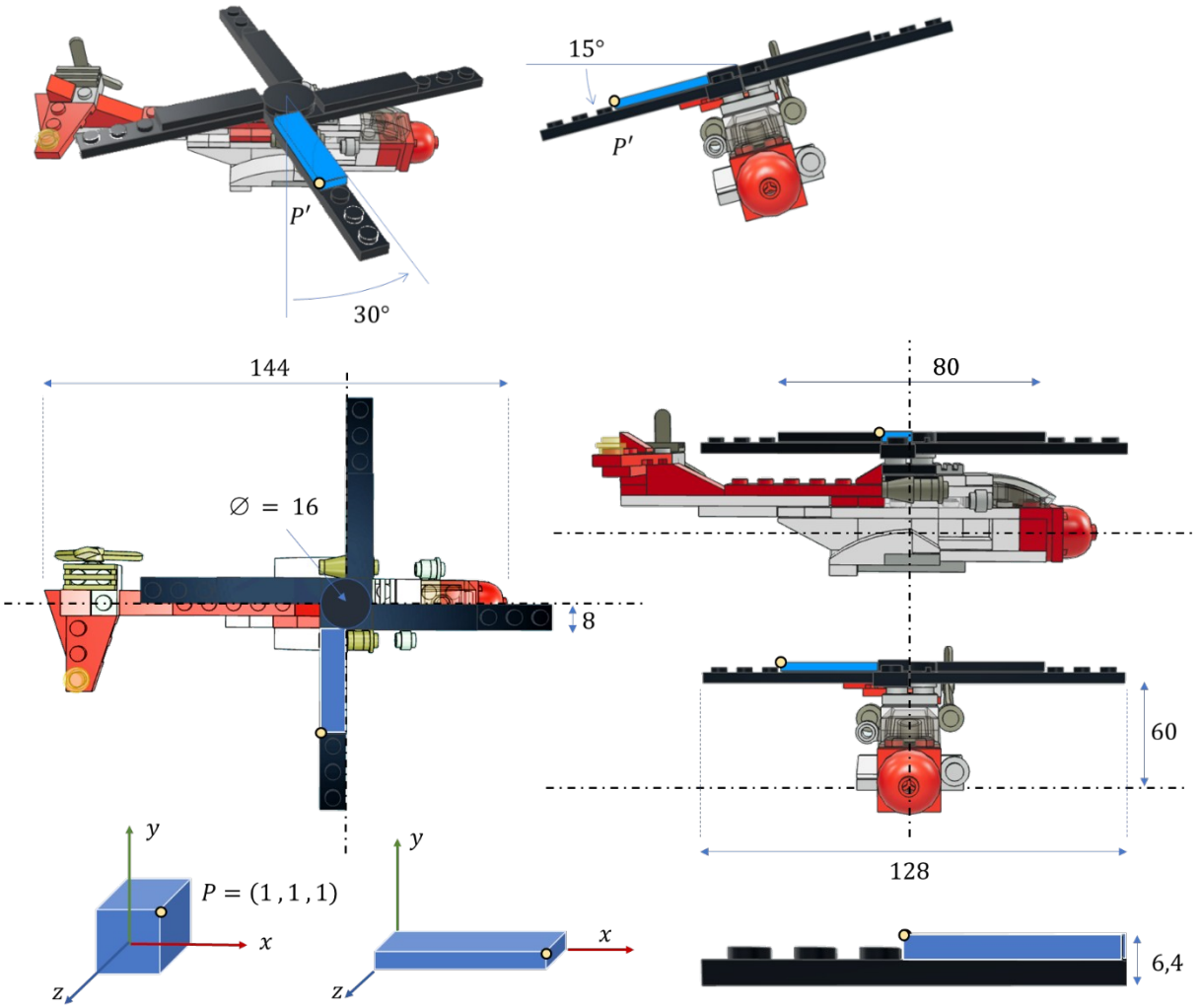


Abb. 15: Die Maße

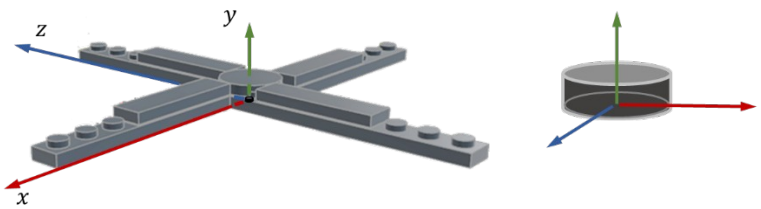


Abb. 16: Der Koordinatenursprung des Rotors liegt im Mittelpunkt Grundkreises des zentralen Zylinders

LÖSUNG Teil a)

DER QUADERBAUSTEIN (Formung und Platzierung auf dem Rotorblatt) a)

Skalierung: Die Grundform aller Quader ist der Einheitswürfel ❒, der über Skalierung die Proportionen des endgültigen Quaders erhalten soll. Die Abmessungen sind $32 \times 3{,}2 \times 8$ mm. Daraus folgt die Skalierungsmatrix

$$S_Q = \begin{bmatrix} 32 & 0 & 0 & 0 \\ 0 & 3{,}2 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation: Drehung des Quaders um $90\,°$ gegen den mathematisch positiven Sinn um die y-Achse ($\theta = -90\,°$).

$$R_Q = \begin{bmatrix} \cos(-90\,°) & 0 & \sin(-90\,°) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-90\,°) & 0 & \cos(-90\,°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

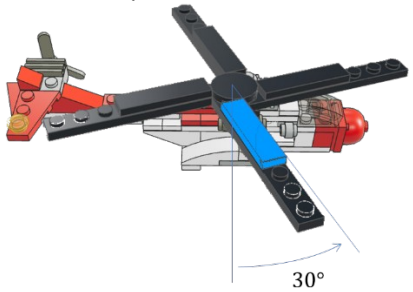
Translation: Verschiebung um $8\,\text{mm}$ in Richtung z-Achse.

$$T_Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Daraus ergibt sich die Transformationsmatrix M_Q zu

$$M_Q \text{¿} T_Q \cdot R_Q \cdot S_Q = \begin{bmatrix} 0 & 0 & -8 & 0 \\ 0 & 3{,}2 & 0 & 0 \\ 32 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

DER ROTOR (Transformation des Rotorblatts)



Skalierung: Eine Skalierung des Rotors ist nicht vorgesehen. Daraus folgt, dass die Skalierungsmatrix mit der Einheitsmatrix identisch ist.

$$S_R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation: Drehung des Rotors um $30\,°$ um die y-Achse ($\rho = 30\,°$).

$$R_R = \begin{bmatrix} \cos(30\,°) & 0 & \sin(30\,°) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(30\,°) & 0 & \cos(30\,°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}866 & 0 & 0{,}5 & 0 \\ 0 & 1 & 0 & 0 \\ -0{,}5 & 0 & 0{,}866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation: Anhebung um $60\,\text{mm}$ nach oben.

$$T_R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 60 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Daraus ergibt sich für den Rotor die Transformationsmatrix M_R zu

$$M_R \text{¿} T_R \cdot R_R \cdot S_R = \begin{bmatrix} 0{,}866 & 0 & 0{,}5 & 0 \\ 0 & 1 & 0 & 60 \\ -0{,}5 & 0 & 0{,}866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2

DER HELIKOPTER (Platzierung des Helikopters im WORLD-SPACE)

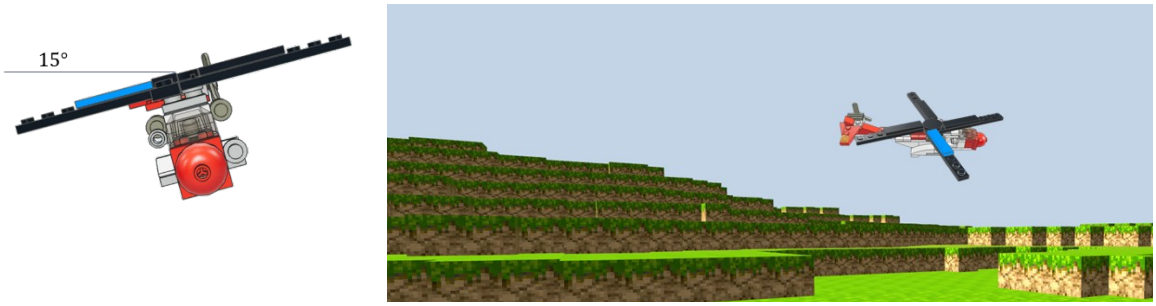


Abb. 17

Skalierung: Implementierung des Helikopters im WORLD-SPACE im Maßstab 1:2. Daraus folgt, die Skalierungsmatrix

$$S_H = \begin{bmatrix} 0{,}5 & 0 & 0 & 0 \\ 0 & 0{,}5 & 0 & 0 \\ 0 & 0 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation: Rollwinkel $-15\,°$ um die x-Achse ($\eta = -15\,°$).

$$R_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-15\,°) & -\sin(-15\,°) & 0 \\ 0 & \sin(-15\,°) & \cos(-15\,°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0{,}966 & 0{,}259 & 0 \\ 0 & -0{,}259 & 0{,}966 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation: Verschiebung in den WORLD-SPACE gemäß der angegebenen Koordinaten.

$$T_H = \begin{bmatrix} 1 & 0 & 0 & 500 \\ 0 & 1 & 0 & 340 \\ 0 & 0 & 1 & 270 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Daraus ergibt sich für den Rotor die Transformationsmatrix M_H zu

$$M_H \text{¿} T_H \cdot R_H \cdot S_H = \begin{bmatrix} 0{,}5 & 0 & 0 & 500 \\ 0 & 0{,}483 & 0{,}130 & 340 \\ 0 & -0{,}130 & 0{,}483 & 270 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Die *World-Matrix* des Quaders ist somit

$$W_Q \text{¿} M_H \cdot M_R \cdot M_Q = \begin{bmatrix} 8 & 0 & -3{,}464 & 502 \\ 3{,}589 & 1{,}546 & 0{,}518 & 369{,}877 \\ 13{,}385 & -0{,}414 & 1{,}932 & 265{,}576 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Jeder Vertex des als Grundelement verwendeten Würfels wird durch W_Q direkt im WORLD-SPACE abgebildet. Zum Beispiel

2

$$V_1 = (1,1,1) \rightarrow W_Q \cdot P = \begin{bmatrix} 8 & 0 & -3{,}464 & 502 \\ 3{,}589 & 1{,}546 & 0{,}518 & 369{,}877 \\ 13{,}385 & -0{,}414 & 1{,}932 & 265{,}576 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 506{,}536 \\ 375{,}529 \\ 280{,}479 \\ (1) \end{bmatrix}$$

$$V_2 = (1,0,1) \rightarrow W_Q \cdot P = \begin{bmatrix} 8 & 0 & -3{,}464 & 502 \\ 3{,}589 & 1{,}546 & 0{,}518 & 369{,}877 \\ 13{,}385 & -0{,}414 & 1{,}932 & 265{,}576 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 506{,}536 \\ 373{,}984 \\ 280{,}893 \\ (1) \end{bmatrix}$$

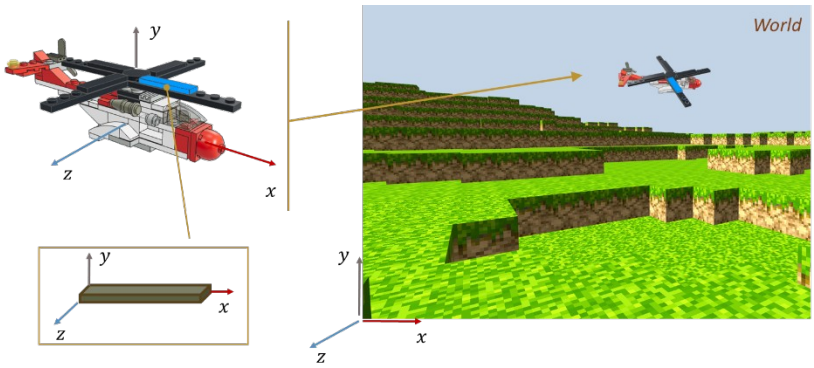
LÖSUNG Teil b)

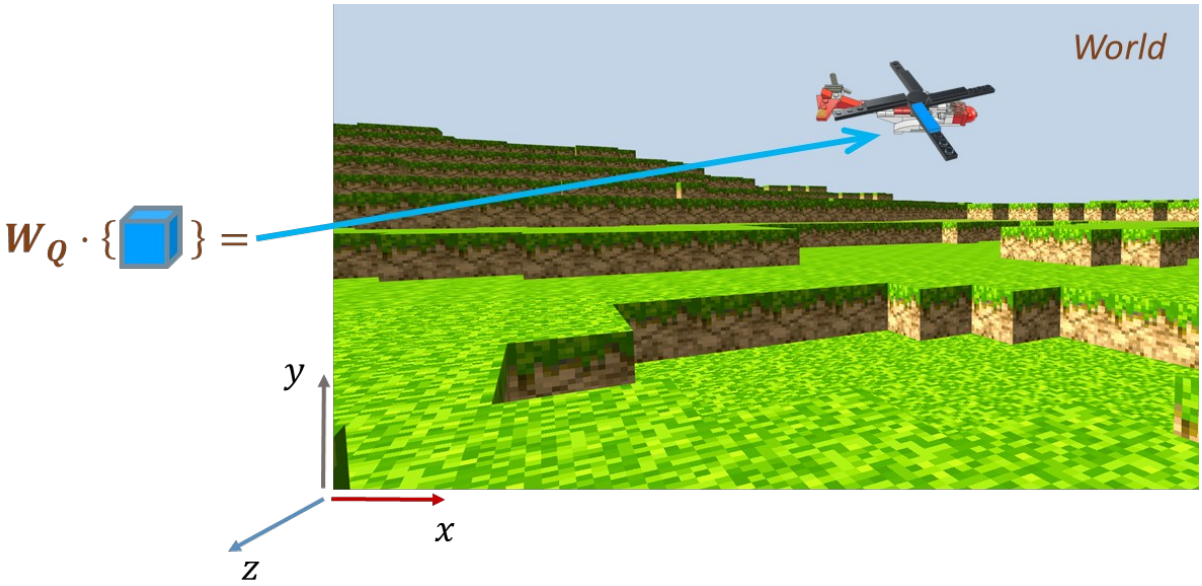
60 fps bedeutet 1/60 s zwischen zwei Frames.

Der Helikopter verändert in dieser Zeit seinen Ort um ***1 mm*** in x-Richtung und der Rotor um einen Winkel von ***48°***.

Das bedeutet in der des betrachteten Bausteins

$$W_Q = M_H \cdot M_R \cdot M_Q \text{¿} T_H \cdot R_H \cdot S_H \cdot T_R \cdot R_R \cdot S_R \cdot T_Q \cdot R_Q \cdot S_Q$$

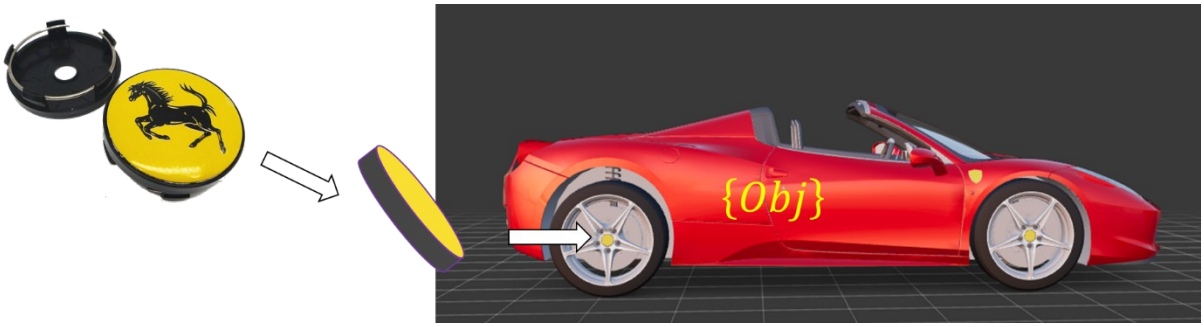




ein Update der Matrizen T_H und R_R!

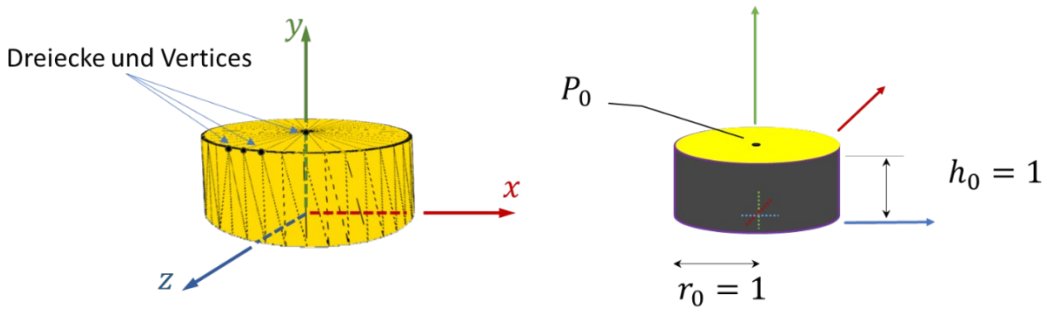
$$T_H = \begin{bmatrix} 1 & 0 & 0 & 500 \\ 0 & 1 & 0 & 340 \\ 0 & 0 & 1 & 270 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow T'_H = \begin{bmatrix} 1 & 0 & 0 & 501 \\ 0 & 1 & 0 & 340 \\ 0 & 0 & 1 & 270 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow M'_H = \begin{bmatrix} 1 & 0 & 0 & 501 \\ 0 & 0{,}966 & 0{,}259 & 340 \\ 0 & -0{,}259 & 0{,}966 & 270 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.7 Ferrari-Radkappe



IM ERSTEN SCHRITT WIRD DAS GRUNDELEMENT ENTWORFEN.

Beispielpunkt $P_0=(0,1,0)$



IM ZWEITEN SCHRITT WIRD DAS GRUNDELEMENT IN DIE SZENE TRANSFERIERT.

$P_0=(0,1,0) \rightarrow P=(2020,404,0) \in \{Obj\}$

BERECHNUNG: Gesucht ist die Model-World-Matrix der Radnabenabdeckung, mit der ihre Koordinaten bezüglich der Szene berechnet werden können.

Model-World-Matrix M ***(Local to World)***

Die tatsächlichen Maße der Radkappe im World-Space:

Abb. 18

BERECHNUNG:

$$P_0 \to P \text{ durch } P = T \cdot \left(R_z \cdot \left(S \cdot P_0\right)\right) = \overbrace{\underbrace{T \cdot R_z \cdot S}_{\text{Berechnung}}}^{\text{geometrisch}} \cdot P_0 = M \cdot P_0$$

2 Betrachten wir zuerst die Skalierung $S = \begin{bmatrix} 33 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 33 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Betrachten wir dann die $+90\,°$-Rotation um die z-Achse:

$$R = R_z = \begin{bmatrix} \cos(90\,°) & \sin(90\,°) & 0 & 0 \\ -\sin(90\,°) & \cos(90\,°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Nach Skalierung und der Drehung liegt $P_0 \to P_1$ auf der x-Achse: $P_1 = (-10, 0, 0)$. Die Translation $P_1 \to P$ in den World-Space ergibt den Translationsvektor $\begin{bmatrix} 2030 & 404 & 0 \end{bmatrix}$

Damit berechnet sich die Model-World-Matrix der Kappe zu

$$M = T \cdot R \cdot S = \begin{bmatrix} 1 & 0 & 0 & 2030 \\ 0 & 1 & 0 & 404 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 33 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 33 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = ¿$$

$$M = \begin{bmatrix} 0 & 10 & 0 & 2030 \\ -33 & 0 & 0 & 404 \\ 0 & 0 & 33 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Im dritten Schritt wird die Sicht auf die Szene definiert.

Der View-Space ist der World-Space aus Sicht der Kamera. Dabei ist die Kamera ebenfalls ein Objekt im World-Space.

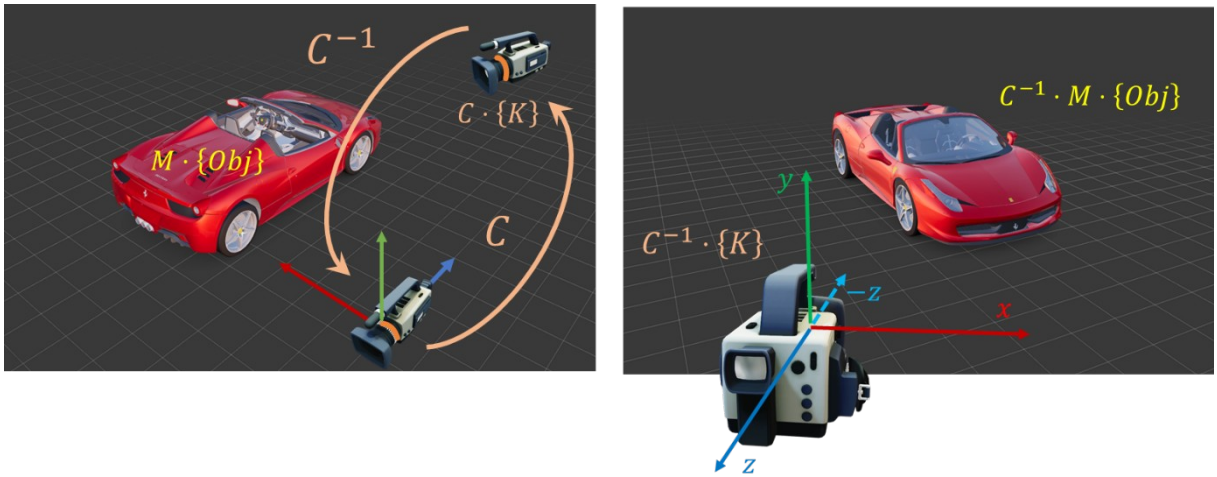


Abb. 19

BERECHNUNG:

(*Matrizenberechnungen mit Online-Rechner https://matrixcalc.org*)

Kamera-Koordinaten

Von der Kamera wird immer angenommen, dass sie im MODEL-SPACE in die $(-z)$-Richtung blickt. Der „Up-Vektor“ zeigt in Richtung y-Achse. Von dieser Grundposition aus wird sie über Rotation und Translation in den WORLD-SPACE transferiert.

Beispiel Radkappe

Rotationen:

a) $-30\,°$ x-Achse ($R_{x:-30\,°}$)

b) $-45\,°$ y-Achse ($R_{y:-45\,°}$)

Translation:

$$\acute{v}=\begin{pmatrix}0\\2000\\7000\end{pmatrix}$$

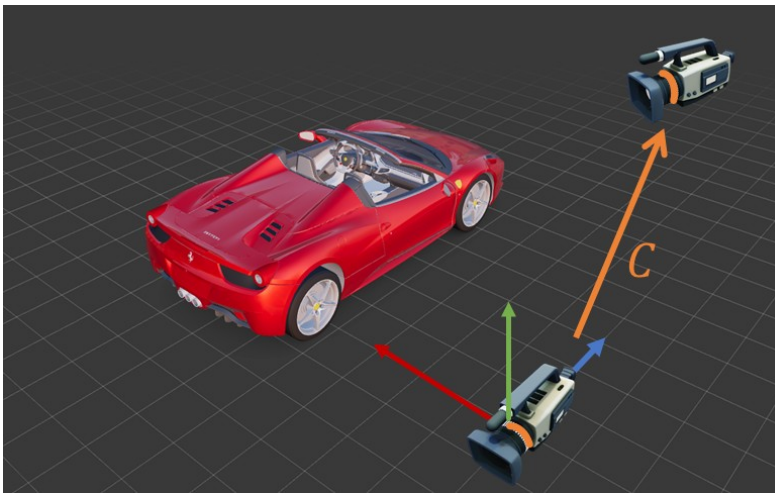


Abb. 20

Kamera: *Model-World-Matrix* C

Die Reihenfolge der Transformationen (siehe Angabe) ist wichtig:

$$R_{x:-30\,°} \rightarrow R_{y:-45\,°} \rightarrow T_{\begin{pmatrix}0\\2000\\7000\end{pmatrix}} \qquad C=T_{\begin{pmatrix}0\\2000\\7000\end{pmatrix}} \cdot R_{y:-45\,°} \cdot R_{x:-30\,°}$$

Die Skalierung entfällt bzw. ist $S_C=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$.

Betrachten wir zuerst die Rotationen!

$$R_C=\overbrace{\underbrace{R_{y:-45\,°} \cdot R_{x:-30\,°}}_{\text{Berechnung}}}^{\text{geometrisch}}$$

$$¿\begin{bmatrix}\cos(-45\,°)&0&\sin(-45\,°)&0\\0&1&0&0\\-\sin(-45\,°)&0&\cos(-45\,°)&0\\0&0&0&1\end{bmatrix} \cdot \begin{bmatrix}1&0&0&0\\0&\cos(-30\,°)&\sin(-30\,°)&0\\0&-\sin(-30\,°)&\cos(-30\,°)&0\\0&0&0&1\end{bmatrix}=¿$$

$$¿\begin{bmatrix} 0{,}707 & 0 & -0{,}707 & 0 \\ 0 & 1 & 0 & 0 \\ 0{,}707 & 0 & 0{,}707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0{,}866 & -0{,}5 & 0 \\ 0 & 0{,}5 & 0{,}866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = ¿$$

$$R_C = \begin{bmatrix} 0{,}707 & -0{,}354 & -0{,}612 & 0 \\ 0 & 0{,}866 & -0{,}5 & 0 \\ 0{,}707 & 0{,}354 & 0{,}612 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2

Der Translationsvektor ist mit $\begin{bmatrix} 0 & 2000 & 7000 \end{bmatrix}$ gegeben. Damit ist

$$C = T_C \cdot R_C = \begin{bmatrix} 0{,}707 & -0{,}354 & -0{,}612 & 0 \\ 0 & 0{,}866 & -0{,}5 & 2000 \\ 0{,}707 & 0{,}354 & 0{,}612 & 7000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Die Inverse $V = C^{-1} = \begin{bmatrix} 0{,}707 & 0 & 0{,}707 & -4949{,}747 \\ -0{,}354 & 0{,}866 & 0{,}354 & -4206{,}925 \\ -0{,}612 & -0{,}5 & 0{,}612 & -3286{,}607 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ergibt gemeinsam mit der

Model-World-Matrix M der Radnabenabdeckung die *Model-View-Matrix*

$$V \cdot M = \begin{bmatrix} 0{,}707 & 0 & 0{,}707 & -4949{,}747 \\ -0{,}354 & 0{,}866 & 0{,}354 & -4206{,}925 \\ -0{,}612 & -0{,}5 & 0{,}612 & -3286{,}607 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 10 & 0 & 2030 \\ -33 & 0 & 0 & 404 \\ 0 & 0 & 33 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underbrace{V \cdot M}_{Local/Model\ ¿} \; View ¿ = \begin{bmatrix} 0 & -7{,}071 & 23{,}335 & -3514{,}321 \\ -28{,}579 & -3{,}536 & 11{,}667 & -4574{,}764 \\ 16{,}5 & -6{,}124 & 20{,}208 & -4731{,}723 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Aus der Kameraperspektive hat der Punkt P_0 die Koordinaten

$$V \cdot M \cdot P_0 = \begin{bmatrix} 0 & -7{,}071 & 23{,}335 & -3514{,}321 \\ -28{,}579 & -3{,}536 & 11{,}667 & -4574{,}764 \\ 16{,}5 & -6{,}124 & 20{,}208 & -4731{,}723 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -3507{,}250 \\ -4578{,}299 \\ -4737{,}847 \\ 1 \end{bmatrix}$$

Die negativen Zahlenwerte mögen jetzt überraschen, aus Sicht der Kamera aber verständlich. ■