HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

3D Transformationen

Folien zur Vorlesung am 14. + 21.06.2024

Mathematische Grundlagen von Computergrafik und Gestaltung (MKB2) + STEM1 (GMB1)

Hinweis: ergänzend zum Foliensatz findet man Videos und interaktive Übungen unter:

3Blue1Brown: Linear Algebra: 3D transformations

Immersive Math Kapitel 6: Matrices

3D Transformationsmatrizen

Matrix * Vektor = Ein Vektor aus den Skalarprodukten der Zeilenvektoren mit dem Vektor

$$M\vec{x} = \vec{b}$$

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} m_{11}x_1 + m_{12}x_2 + m_{13}x_3 \\ m_{21}x_1 + m_{22}x_2 + m_{23}x_3 \\ m_{31}x_1 + m_{32}x_2 + m_{33}x_3 \end{pmatrix}$$

Transformationsmatrizen

Erinnerung: Die Koordinaten eines Vektors sind als die Faktoren der Basisvektoren zu verstehen.

$$\vec{u} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = 1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Multipliziert man die Matrix mit den Basisvektoren ergeben sich die Spalten der Matrix.

Das gilt auch in 3D.

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} m_{11} \\ m_{21} \\ m_{31} \end{pmatrix}$$

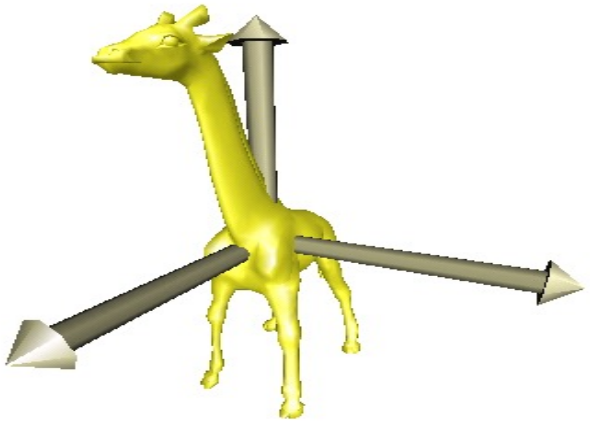
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} m_{12} \\ m_{22} \\ m_{32} \end{pmatrix}$$

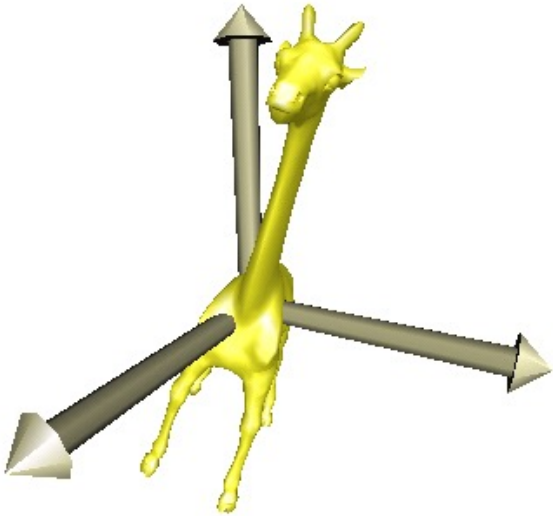
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} m_{13} \\ m_{23} \\ m_{33} \end{pmatrix}$$

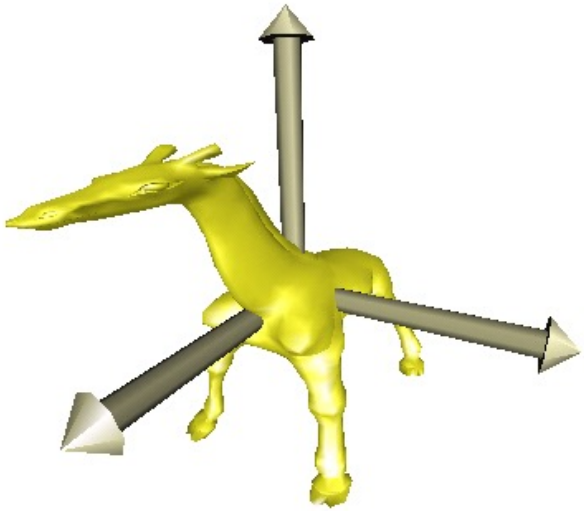
Transformationen

Transformation + relative Koordinaten

- Rotation
- Skalierung
- Scherung (ohne Abbildung)



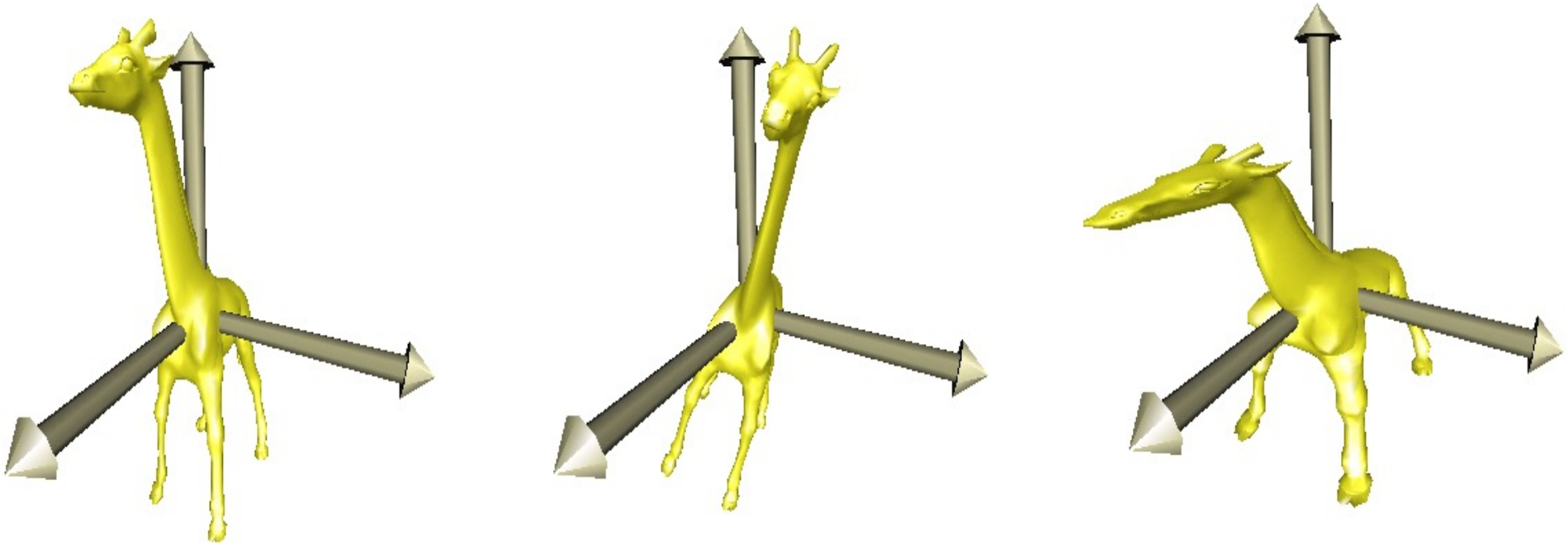




Skalierung, Scherung, Rotation

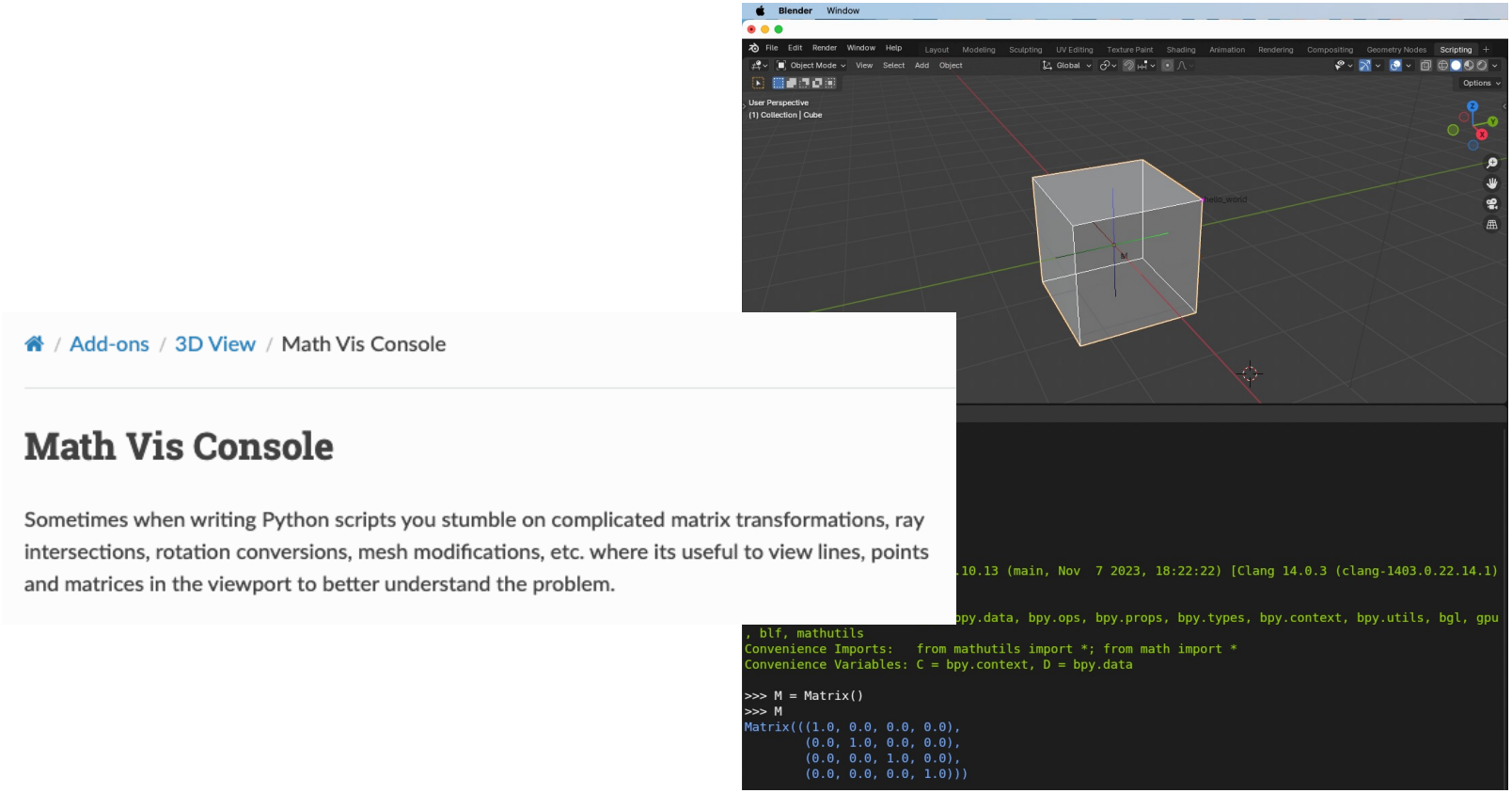
Die Transformationen Skalierung, Scherung, Rotation lassen den Ursprung unverändert.

- Es sind lineare Transformationen.
- 3x3-Matrizen sind ausreichend.



Homogene Koordinaten

In der Computergrafik arbeitet man mit 4x4 Matrizen und 4-dimensionalen Vektoren.



Translationen

Wie beschreiben wir die Verschiebung (Translation) eines Punkts im Raum?

2D Translation als Matrix-Vektor-Multiplikation?

$$
\begin{aligned}
x' &= x + t_x \\
y' &= y + t_y
\end{aligned}
$$

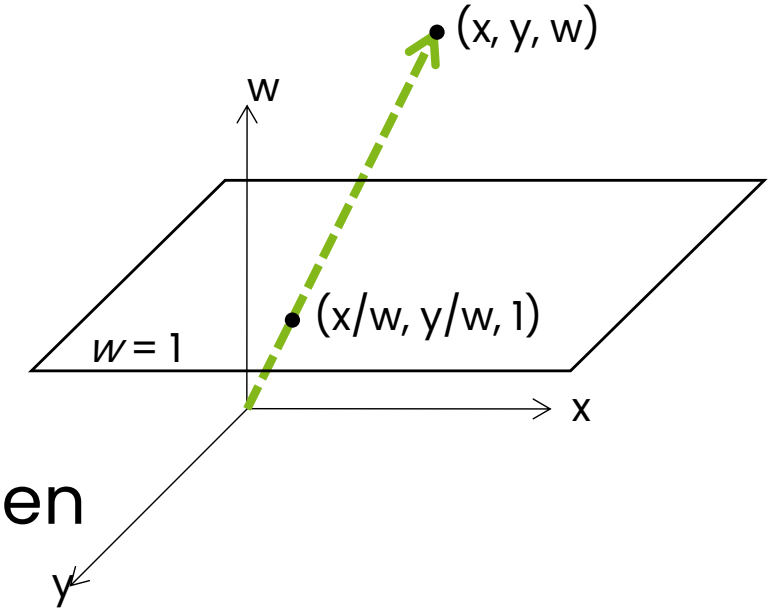
Nein!

Die Translation ist keine lineare Operation auf 2D Koordinaten.

Homogene Koordinaten

Trick: Wir fügen eine weitere Koordinate ein…

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Homogene Bildkoordinaten

Rückkonvertierung *aus* homogenen Koordinaten

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Translation

Problem: Die Translation ist keine lineare Transformation im 2D.
Lösung: homogene Koordinaten verwenden

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Translation

Problem: Die Translation ist auch keine lineare Transformation im **3D**.

Lösung: homogene Koordinaten verwenden

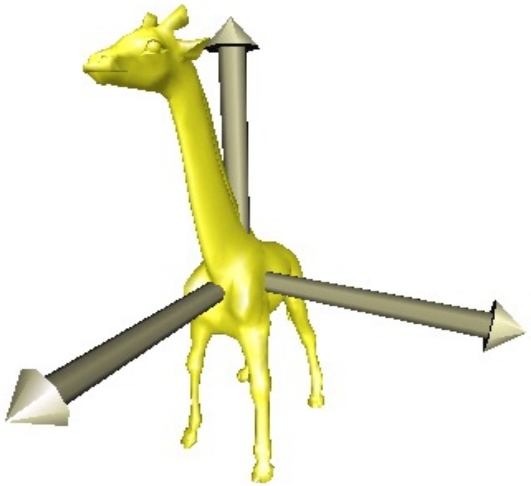
$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix}$$

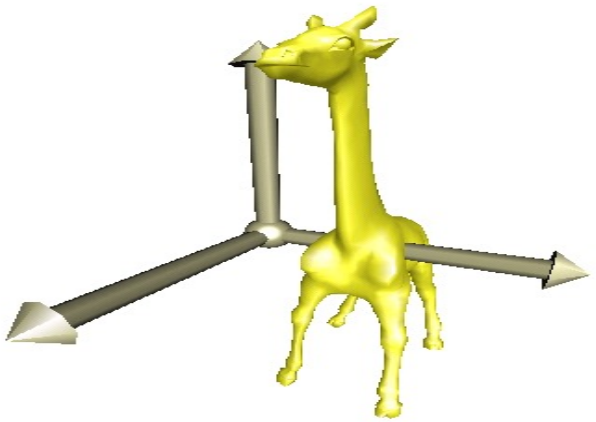
Translation

Translationen sind affine Transformationen

- Der lineare Teil einer Translation T ist die Identität
- Man verwendet eine 4x4-Matrix zur Beschreibung einer Translation um den Vektor $\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$



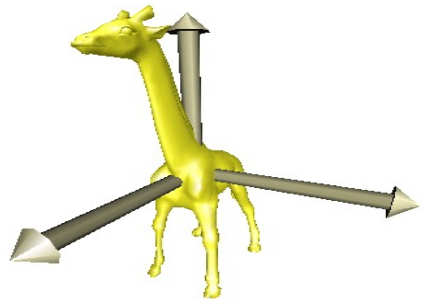
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

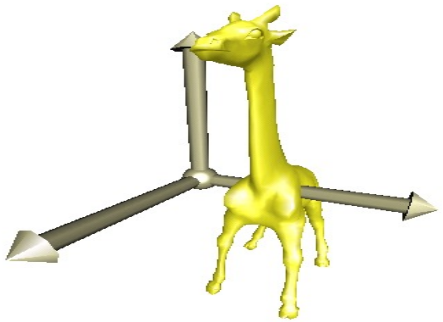


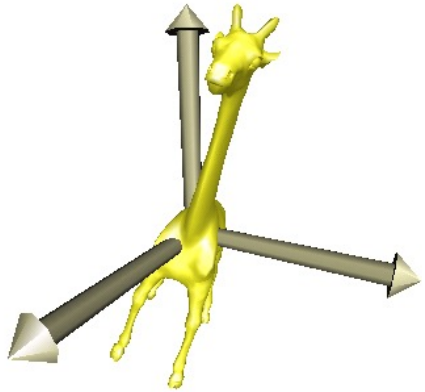
Transformationen

Transformation + relative Koordinaten

- Translation
- Rotation
- Skalierung
- Scherung (ohne Abbildung)







affine Transformationen

Affine Transformation als 4x4-Matrix

Koordinatensystem ist problemabhängig

Affine Räume haben

- keine festen Ursprung
- keine festen Achsen

Lineare Transformation + additiver Term = affine Transformation

$$F(x,y,z) = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Schreibweise mittels **4x4-Matrix**:

$$F(x,y,z,w) = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotation, Skalierung, Scherung, Spiegelung

Translation

Affine Transformation als 4x4-Matrix

Schreibweise mittels **4x4-Matrix**:

$$F(x,y,z,w) = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Affine Abbildungen werden durch homogene 4x4 Matrizen beschrieben

- einheitliche Darstellung und einfache Implementierung
- Hintereinanderausführung verschiedener Transformationen $\boldsymbol{F_i}$
- nur Multiplikation der Matrizen

$$\begin{aligned} \overrightarrow{x_1} &= F_1 \cdot \vec{x} \\ \overrightarrow{x_2} &= F_2 \cdot \overrightarrow{x_1} \\ &\dots \\ \overrightarrow{x_n} &= F_n \cdot \overrightarrow{x_{n-1}} \end{aligned}$$

$$\overrightarrow{x_n} = F_n \cdot \ \dots \cdot F_2 \cdot F_1 \cdot \vec{x}$$

Homogene Koordinaten zur Unterscheidung von Punkt und Vektor in der Computergrafik

Koordinaten beschreiben Punkte.

Vektoren beschreiben eine Richtung und einen Betrag.

Punkt ≠ Vektor

Punkt:

$$p = \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$

Vektor:

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix}$$

Rechnen mit Matrizen funktioniert wie gewünscht.

Homogene Koordinaten zur Unterscheidung von Punkt und Vektor in der Computergrafik

HOCHSCHULE FURTWANGEN UNIVERSITY HFU

Punkt - Vektor Rechenregeln

Punkt + Punkt =

undefiniert

Vektor + Vektor =

Vektor

Punkt + Vektor =

Punkt

Punkt – Punkt =

Vektor

Skalar * Vektor =

Vektor

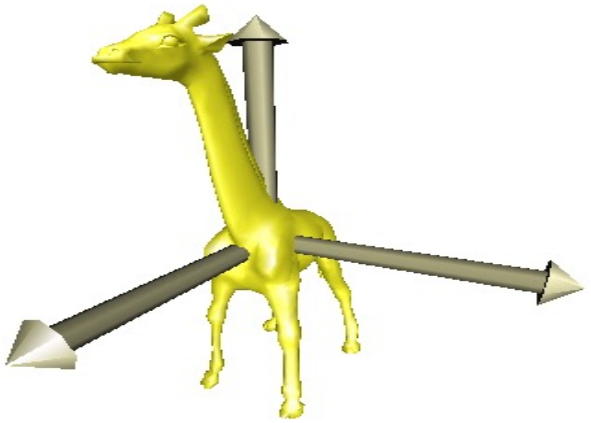
Skalar * Punkt =

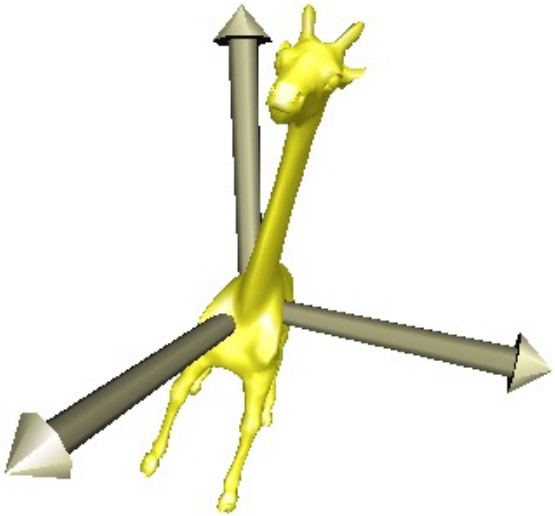
Punkt (wenn Skalar = 1), Vektor (wenn Skalar = 0), undefiniert (sonst)

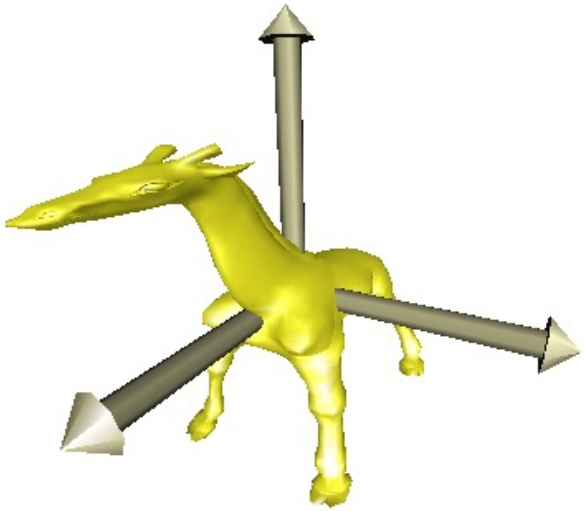
Zurück zu den linearen Transformationen

Lineare Transformationen

- Rotation
- Skalierung
- Scherung (ohne Abbildung)







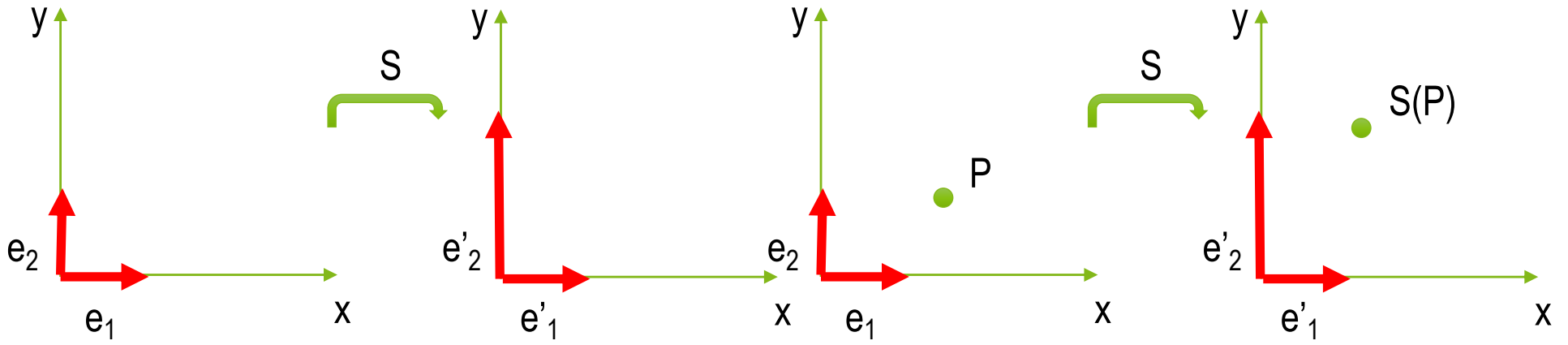
Skalierung

Eine Skalierung S ergibt für die Basisvektoren:

- $S\left(\begin{pmatrix}1\\0\\0\end{pmatrix}\right) = \begin{pmatrix}s_1\\0\\0\end{pmatrix}$
- $S\left(\begin{pmatrix}0\\1\\0\end{pmatrix}\right) = \begin{pmatrix}0\\s_2\\0\end{pmatrix}$
- $S\left(\begin{pmatrix}0\\0\\1\end{pmatrix}\right) = \begin{pmatrix}0\\0\\s_3\end{pmatrix}$

Skalierung eines Punkts ≠ Skalarmultiplikation

Visualisierung für 2D



Skalierung

Die zugehörige 3 x 3 Matrix ergibt sich daher als

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix}$$

Isotrope Skalierung vs. anisotrope Skalierung

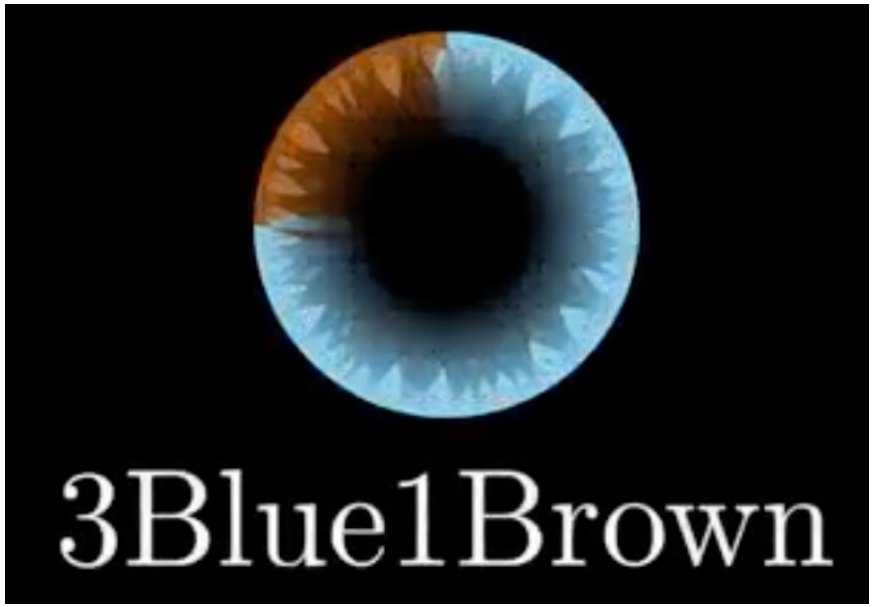
Wenn $s_1 = s_2 = s_3 = s$ gilt, wird die gleiche Skalierung für alle Koordinatenachsen angewendet. Das nennt man isotrope Skalierung.

In homogenen Koordinaten sieht die 4x4 Matrix so aus:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Einstieg am 21.6.

- Video über 3D Transformation als Wrap-Up.

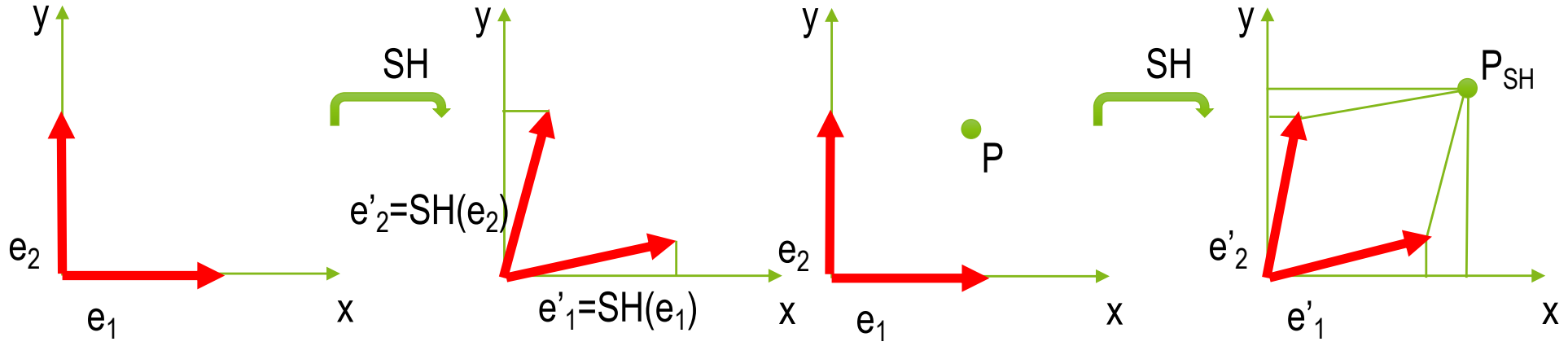


Scherung

Eine Scherung SH ergibt für die Basisvektoren:

- $SH\left(\begin{pmatrix}1\\0\\0\end{pmatrix}\right) = \begin{pmatrix}1\\s_1\\s_3\end{pmatrix}$
- $SH\left(\begin{pmatrix}0\\1\\0\end{pmatrix}\right) = \begin{pmatrix}s_2\\1\\s_4\end{pmatrix}$
- $SH\left(\begin{pmatrix}0\\0\\1\end{pmatrix}\right) = \begin{pmatrix}s_5\\s_6\\1\end{pmatrix}$

$$\begin{pmatrix}1 & s_2 & s_5\\ s_1 & 1 & s_6\\ s_3 & s_4 & 1\end{pmatrix}$$



Scherung

Die dazugehörige 3 x 3 Matrix wird daher zu

$$\begin{pmatrix} 1 & s_2 & s_5 \\ s_1 & 1 & s_6 \\ s_3 & s_4 & 1 \end{pmatrix}$$

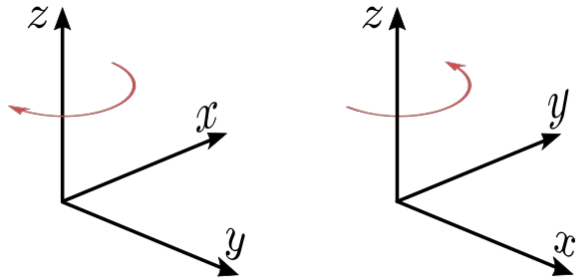
In homogenen Koordinaten schreibt man

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s_2 & s_5 & 0 \\ s_1 & 1 & s_6 & 0 \\ s_3 & s_4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

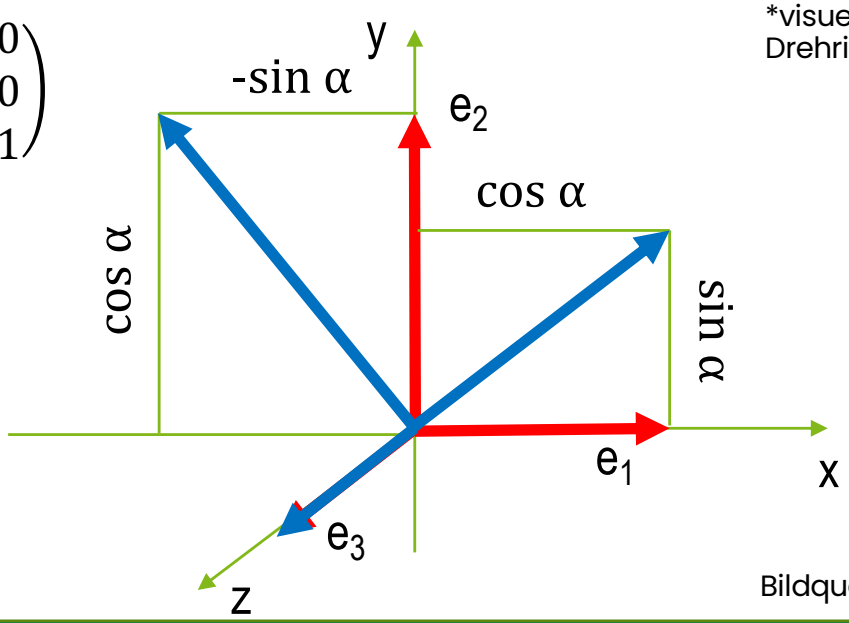
Rotation

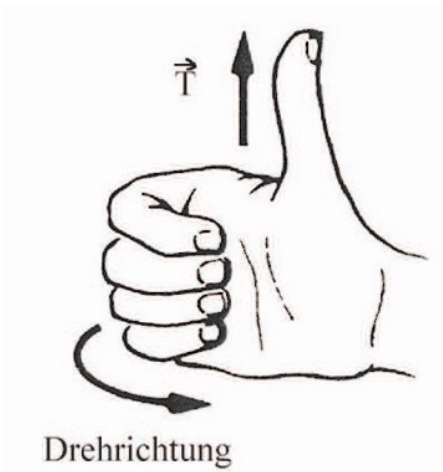
Eine Rotation R_α um den Winkel α um die z-Achse in *mathematisch positiver** Richtung ergibt für die Basisvektoren:

- $R_\alpha\left(\begin{pmatrix}1\\0\\0\end{pmatrix}\right) = \begin{pmatrix}\cos\alpha\\\sin\alpha\\0\end{pmatrix}$
- $R_\alpha\left(\begin{pmatrix}0\\1\\0\end{pmatrix}\right) = \begin{pmatrix}-\sin\alpha\\\cos\alpha\\0\end{pmatrix}$
- $R_\alpha\left(\begin{pmatrix}0\\0\\1\end{pmatrix}\right) = \begin{pmatrix}0\\0\\1\end{pmatrix}$



*visuelle Darstellung der mathematisch positiven Drehrichtung, Bildquelle: Wikipedia





Bildquelle: https://lp.uni-goettingen.de/get/text/4940

Rotation

Die zugehörige 3 x 3 Matrix für die Rotation R_α:

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In homogenen Koordinaten ergeben sich für die Rotationen R_α um die jeweiligen Achsen folgende Matrizen:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

z-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

y-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

x-Achse

Transformation zwischen Koordinatensystemen

Basisvektoren Translation

$$F(\mathbf{x}) = \begin{bmatrix} 2 & 0 & 0 & q_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{x}$$

Basisvektoren Translation

$$F(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 1 & p_x - p_z \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & p_x + p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{x}$$

Was kann ich der Transformationsmatrix ansehen?

- Ursprung des neuen Koord.-Systems
- Basisvektoren des neuen Koord.-Systems
- Länge der Basisvektoren ≠ 1
  → Skalierung entlang der entsprechenden Achse
- …

- Matrix-Dekomposition in Translation, Euler-Rotationen, Skalierung, … möglich, aber nicht immer trivial (siehe https://en.wikipedia.org/wiki/Matrix_decomposition)

Zusammengesetzte Transformationen

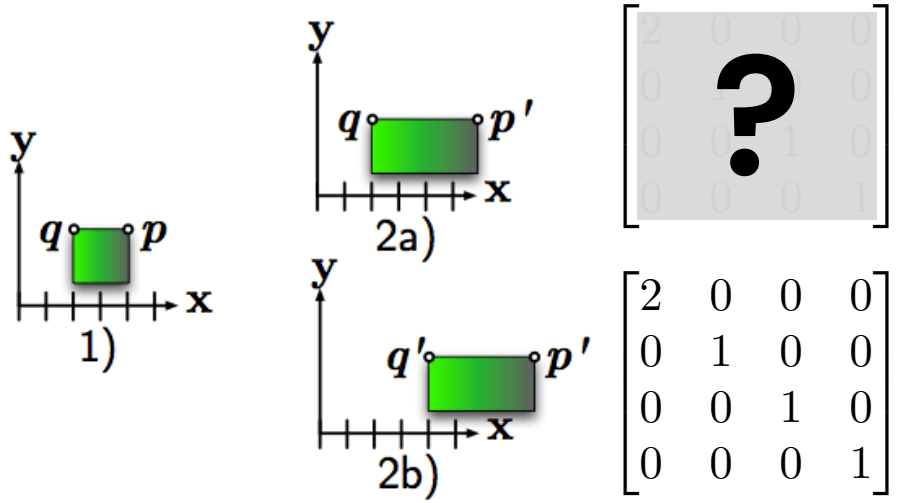
(englisch: Composite Transformations)

Was passiert, wenn mehrere Transformationen nacheinander ausgeführt werden?

Zusammengesetzte Transformationen

Motivation

- Einfache Transformationen beziehen sich immer auf den Ursprung.
- In sehr vielen Fällen möchte man jedoch ein Objekt bzgl. eines bestimmten Fixpunktes oder einer Achse transformieren.
- Dafür bedient man sich zusammengesetzter Transformationen.

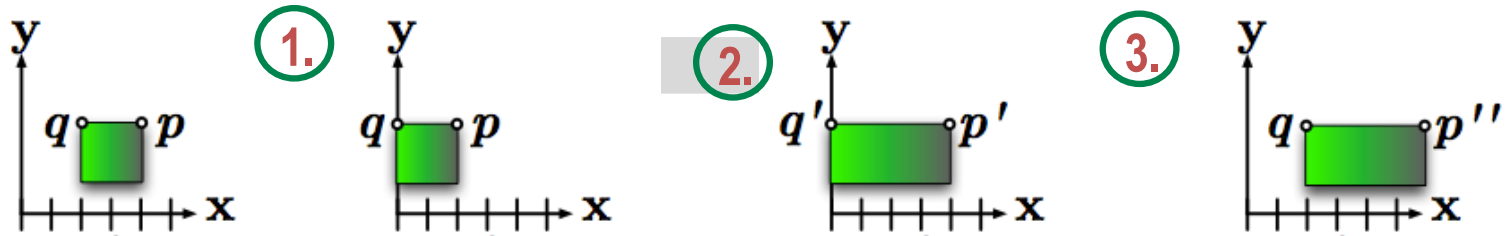


$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Objekt soll skaliert werden so wie in (2a) dargestellt.
Skalierung (2b) verschiebt jedoch das Objekt!

Skalierung um (2,1,1); obere linke Ecke q bleibt fest

1. Verschiebe alle Punkte um den Vektor $(\mathbf{0} - \mathbf{q})$
   - Der Vektor $(\mathbf{0} - \mathbf{q})$ schiebt den Fixpunkt in den Ursprung
   - In diesem Beispiel reicht es aus, nur die X-Koordinate zu verschieben.
2. Skaliere Objekt (alle Punkte) um gewünschte Größe
   - Bei Skalierung in x-Richtung bleiben Punkte mit x = 0 unverändert!
3. Verschiebe alle Punkte um die additive Inverse $(\mathbf{q} - \mathbf{0})$ des ursprünglichen Vektors



$$F(\mathbf{x}) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & q_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{3.} \cdot \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{2.} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 & -q_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{1.} \cdot \mathbf{x} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & -q_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{zusammen}} \cdot \mathbf{x}$$

„Lesart“ bei Matrix-Multiplikationen

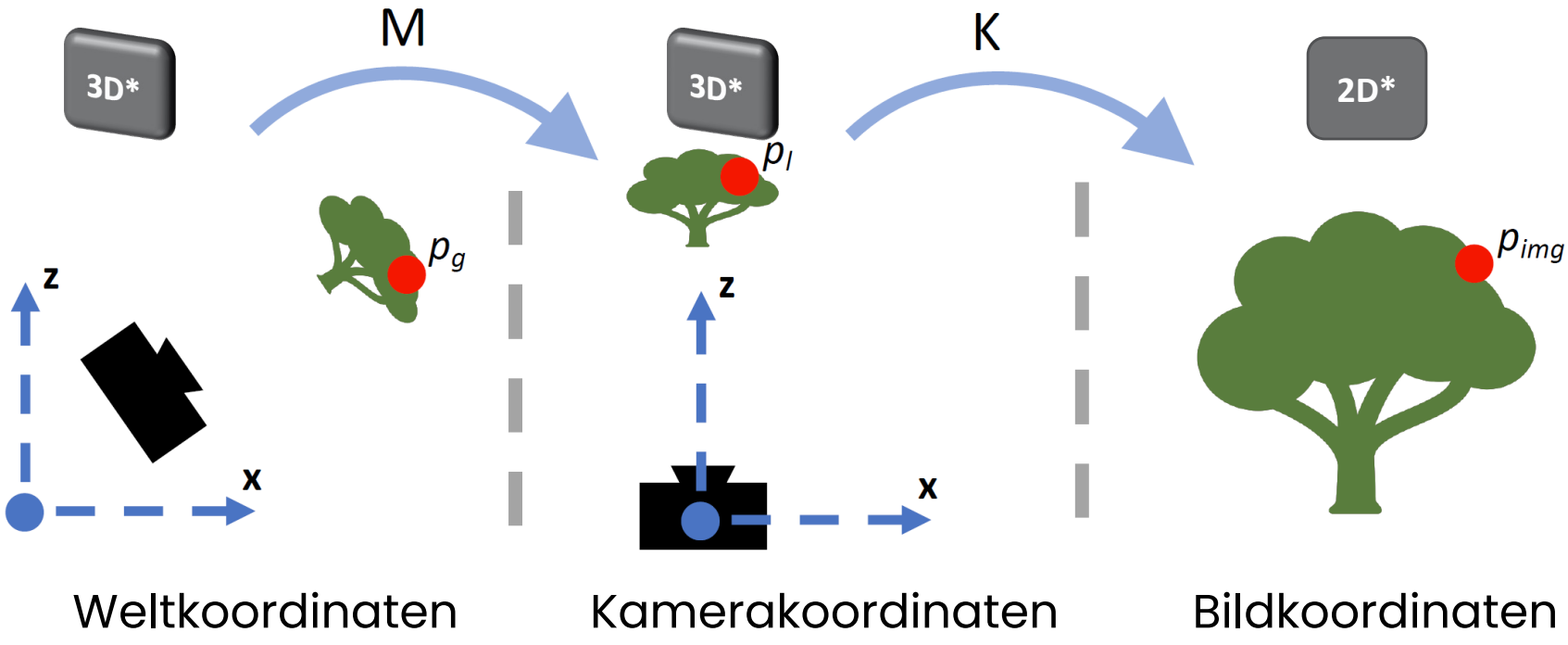
$$F(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \left(\begin{bmatrix} \cos\frac{\pi}{2} & 0 & \sin\frac{\pi}{2} & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\frac{\pi}{2} & 0 & \cos\frac{\pi}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{x} \right) \right)$$

Transformation eines Punktes: rechts-nach-links

- Punkt steht rechts, Transformationen kommen immer von links hinzu
- Zur Verdeutlichung können Klammern gedacht werden
- Äquivalent: geschachtelte Funktionsaufrufe denken
  - `F(x) = translate( rotate( translate(x,-p), Y_AXIS, Math.PI/2), p);`
- Vorsicht bei objektorientierter Programmierung:
  - `F(x) = x.translate(-p).rotate(Y_AXIS, Math.Pi/2).translate(p);`

Wie wendet man das in der Computergrafik an?

Koordinatensysteme in der Fotografie

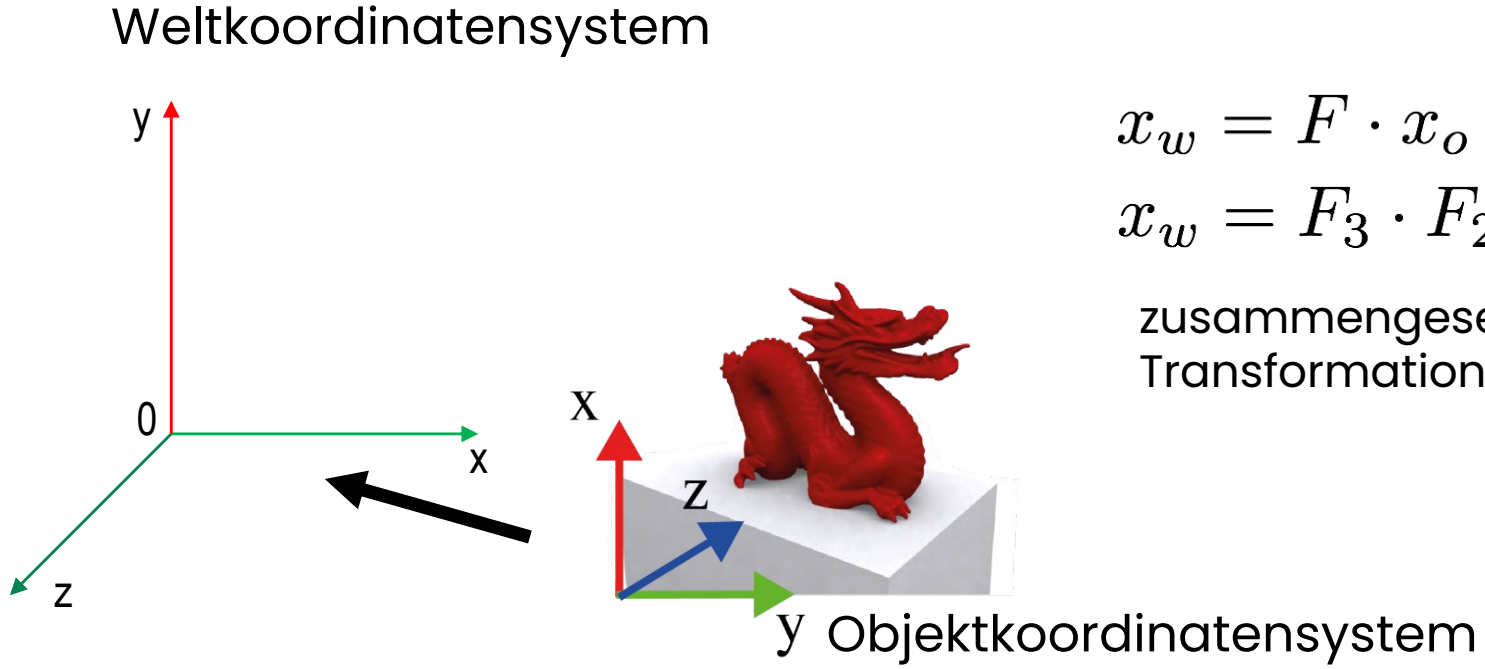


M und K sind Transformationsmatrizen

Frage: Warum das * bei den Dimensionen?

Figure credit: Peter Hedman

Modeling-Transformation in der Computergrafik



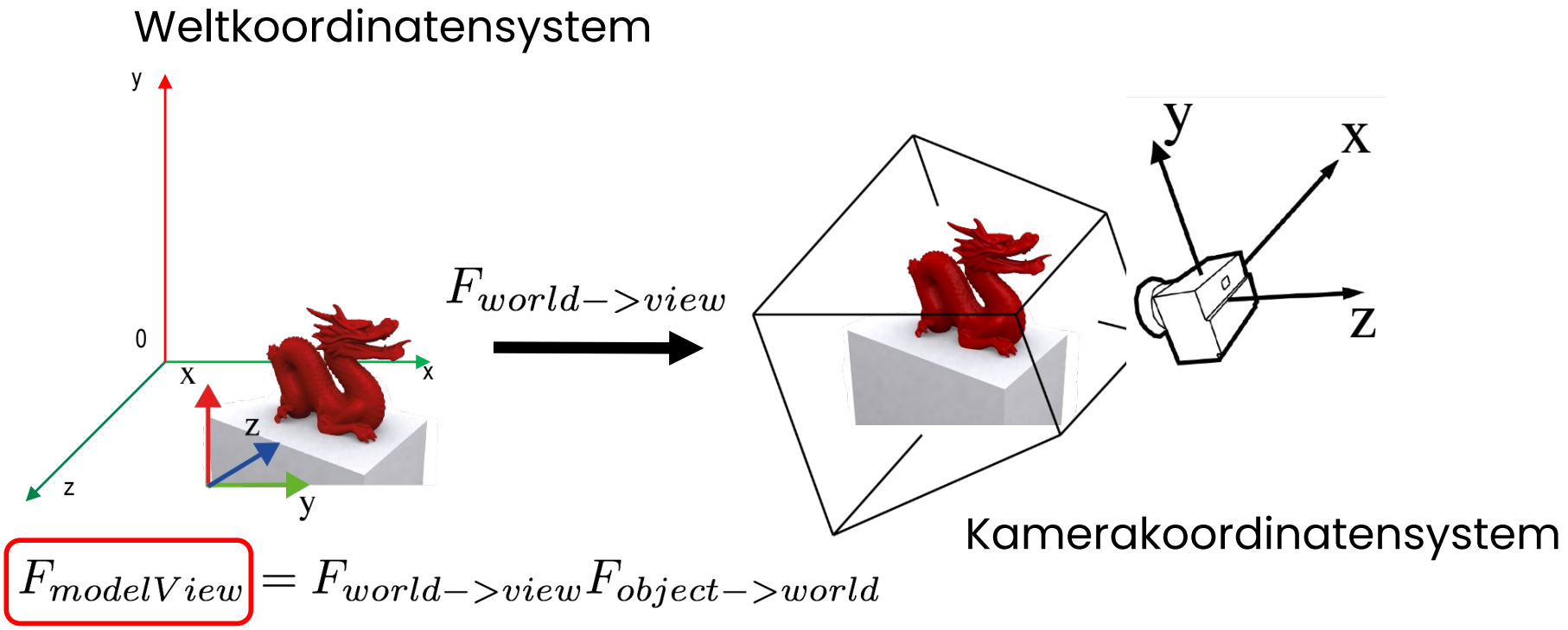
$$x_w = F \cdot x_o$$

$$x_w = F_3 \cdot F_2 \cdot F_1 \cdot x_o$$

zusammengesetzte Transformationen

$$F(x,y,z,w) = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

View-Transformation in der Computergrafik

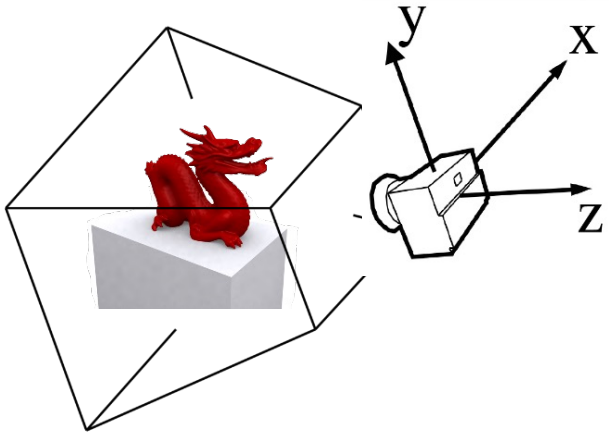


$$F_{modelView} = F_{world->view} F_{object->world}$$

Wird häufig zur *Model-View-Matrix* zusammengefasst.

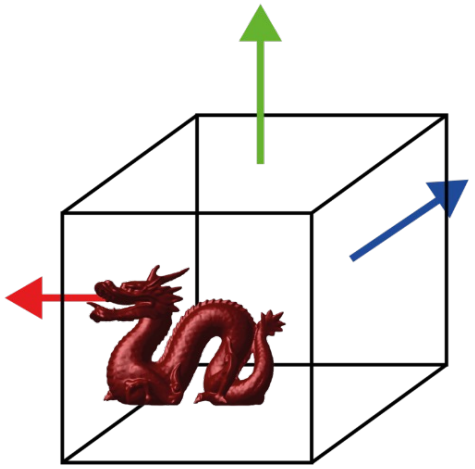
Projektionstransformation in der Computergrafik

Kamerakoordinatensystem

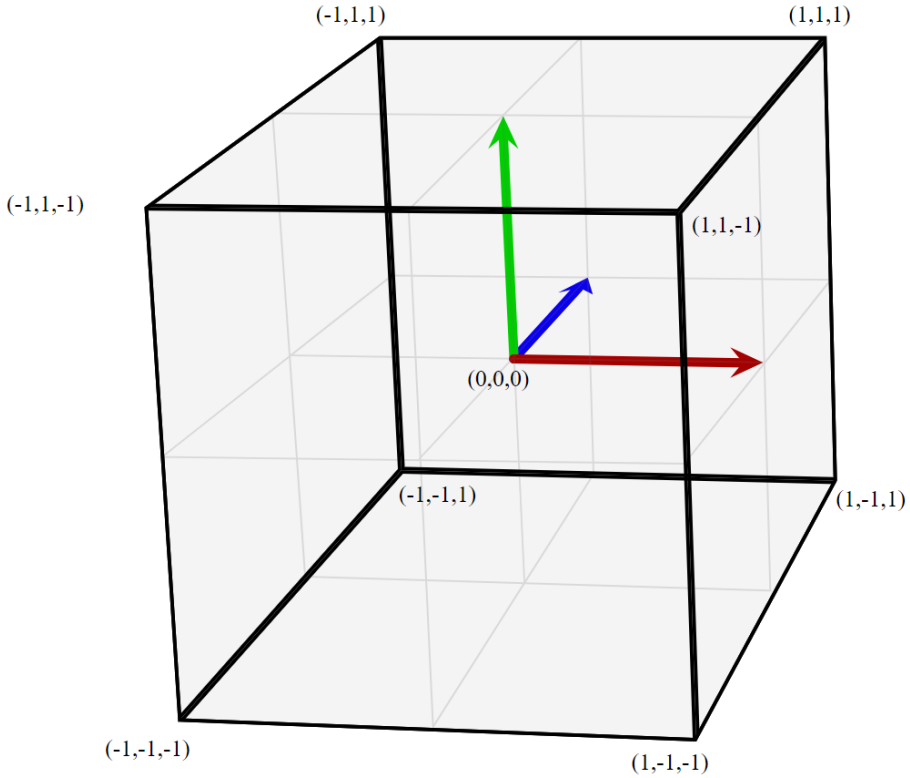


$F_{view->clip}$

Kanonisches Volumen



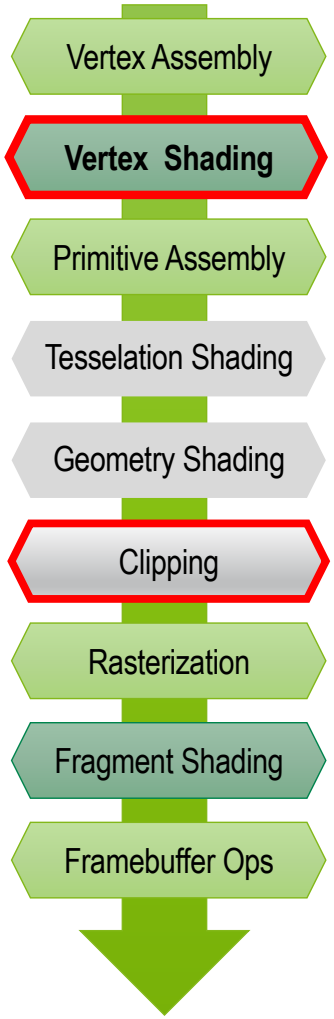
Details aus der Computergrafik: Clipspace



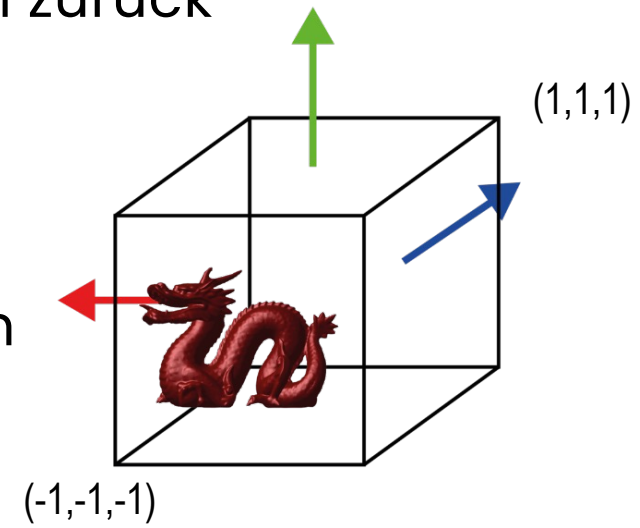
Clipspace:

Ein im Ursprung zentrierter Würfel der Kantenlänge 2

Details aus der Computergrafik: Viewport - Transformation



- Vertex-Shader liefert Position im kanonischen Volumen vor der perspektischen Division zurück

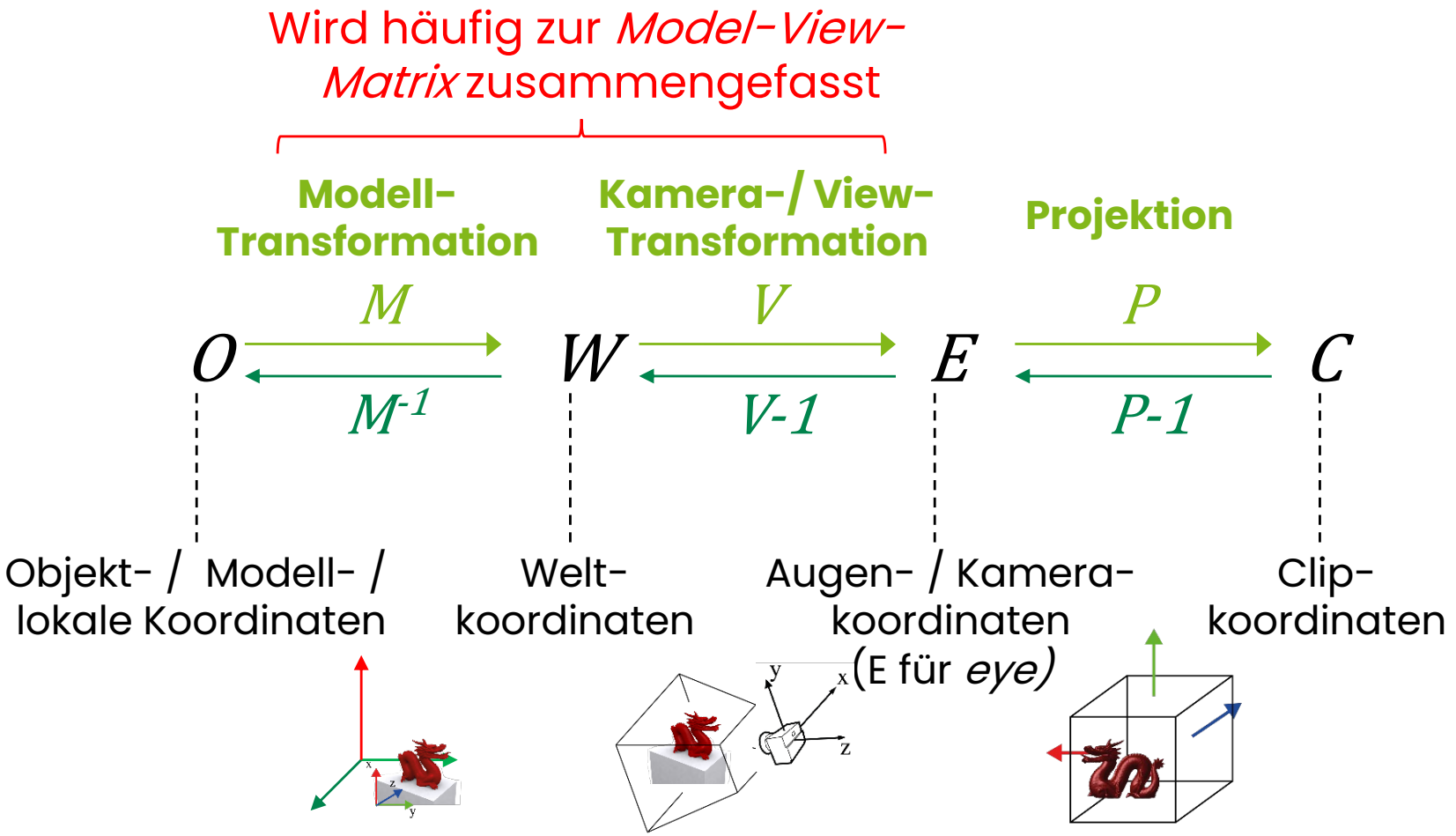


- Clipping
- Perspektivische Division

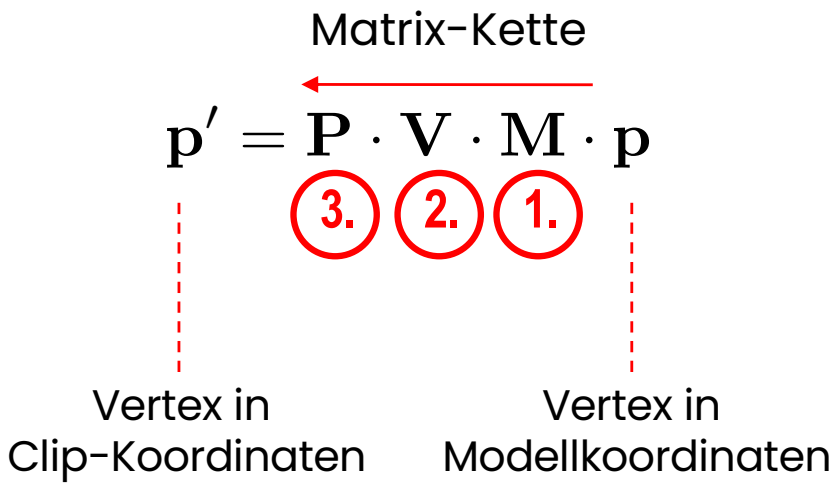
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \frac{1}{w_c} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

- Viewport-Transformation (Window-Koordinaten)

Die Transformationskette von Modell- in Clipkoordinaten



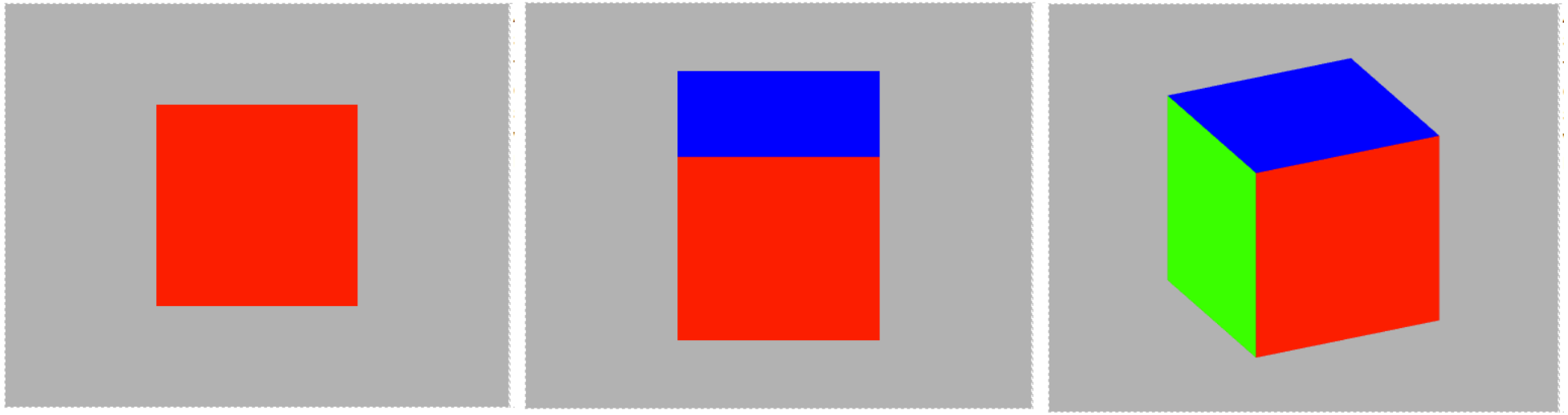
Die Transformationskette von Modell- in Clipkoordinaten



Achtung Reihenfolge!

- Der zu transformierende **Vertex steht rechts**
- Die Transformationen werden der Reihe nach **von links multipliziert**
  - → von rechts nach links lesen

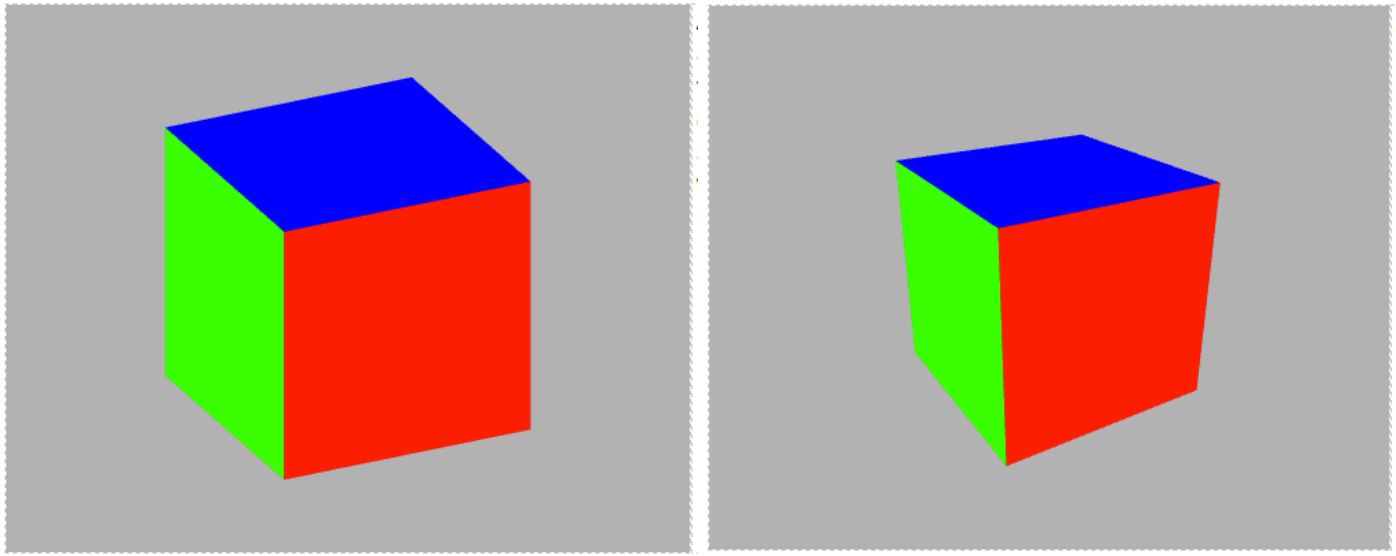
Effekt der Model-View-Transformationsmatrix



Wer rotiert hier eigentlich, die Kamera oder die Szene?

- Die Modelview-Matrix ist ***relative Transformation*** zwischen zwei Koordinatensystemen
- Rotation der Szene entspricht *entgegengesetzter* Rotation der Kamera - die eine Transformation ist die ***Inverse*** der anderen

Effekt der Projektionsmatrix



Orthographische Projektion bzw. **Parallelprojektion**

- Parallele Linien bleiben parallel
- Entspricht eher einem Tele-Objektiv

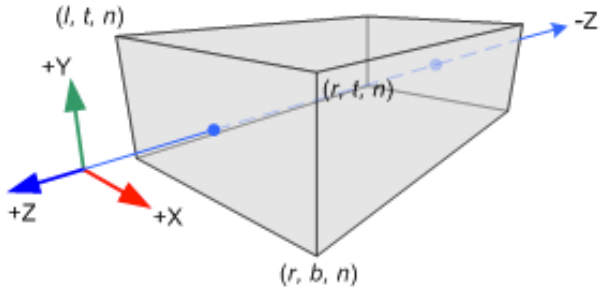
Perspektivische Projektion bzw. **Zentralprojektion**

- „Schräge“ Projektion mit einem wählbaren Öffnungswinkel
- Stürzende Linien, ferne Objekte erscheinen kleiner
- Entspricht eher einem Weitwinkel-Objektiv

Orthographische Projektion

Illustrationen von Song Ho Ahn

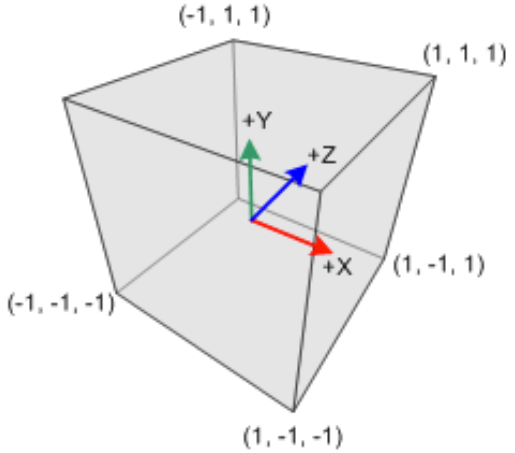
Kamera-Koordinaten



$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Projektionsmatrix

Clip-Koordinaten

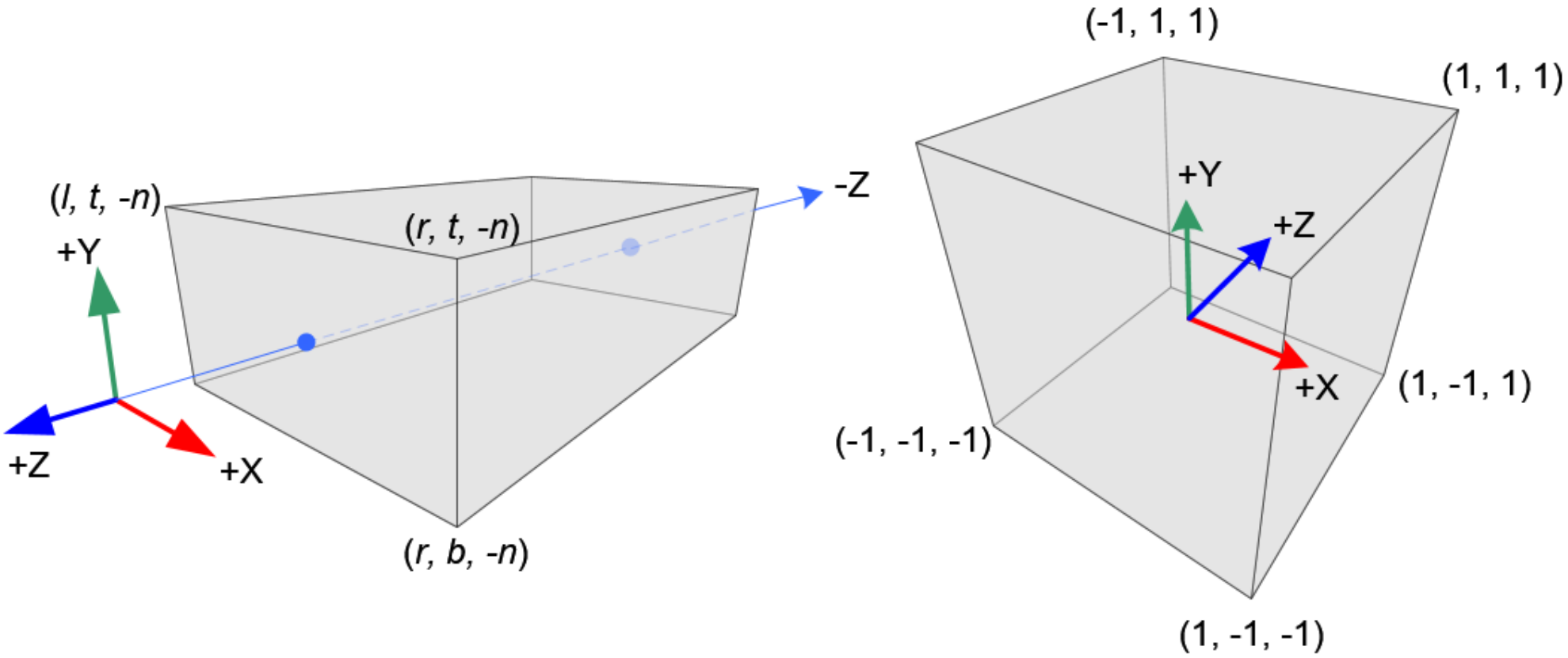


Im Code (hier three.js):

```
var camera = new THREE.OrthographicCamera(left,
right, top, bottom, near, far);
```

Orthografische Projektion - Herleitung

Wir müssen einen Quader in einen Würfel überführen:

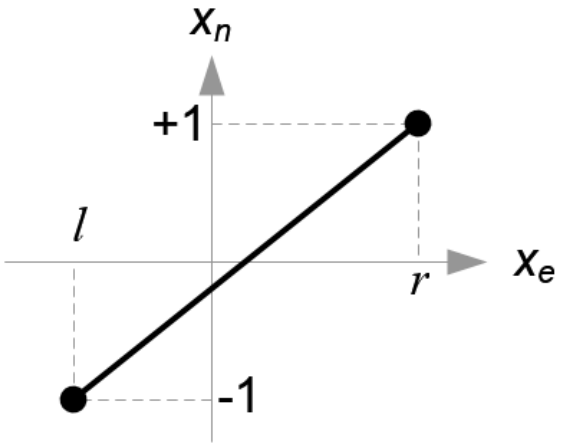


Also die Komponenten x_e, y_e, z_e (eye/camera space, links) in x_n, y_n, z_n (normalized clip space, rechts).

Illustrationen und Formeln von Song Ho Ahn

Orthografische Projektion - Herleitung

Mapping von x_e nach x_n:



$$x_n = \frac{1-(-1)}{r-l} \cdot x_e + \beta$$

$$1 = \frac{2r}{r-l} + \beta \qquad \text{(substitute } (r,1) \text{ for } (x_e, x_n)\text{)}$$

$$\beta = 1 - \frac{2r}{r-l} = -\frac{r+l}{r-l}$$

$$\therefore x_n = \frac{2}{r-l} \cdot x_e - \frac{r+l}{r-l}$$

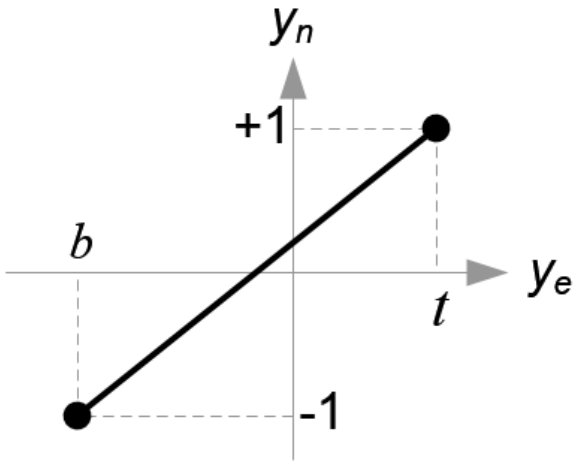
linearer Basisvektor $\overrightarrow{x_e} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow \overrightarrow{x_n} = \begin{pmatrix} \frac{2}{r-l} \\ 0 \\ 0 \end{pmatrix}$

additiver (homogener) Anteil: $-\frac{r+l}{r-l} \rightarrow P_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & ? & ? & ? \\ 0 & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$

Illustrationen und Formeln von Song Ho Ahn

Orthografische Projektion - Herleitung

Mapping von y_e nach y_n:



$$y_n = \frac{1-(-1)}{t-b} \cdot y_e + \beta$$

$$1 = \frac{2t}{t-b} + \beta \qquad \text{(substitute } (t,1) \text{ for } (y_e, y_n)\text{)}$$

$$\beta = 1 - \frac{2t}{t-b} = -\frac{t+b}{t-b}$$

$$\therefore y_n = \frac{2}{t-b} \cdot y_e - \frac{t+b}{t-b}$$

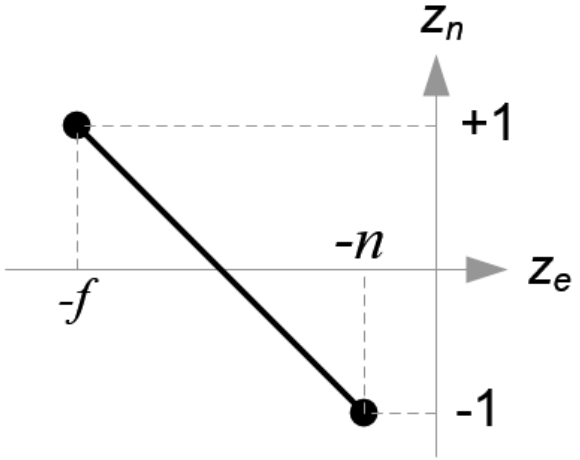
linearer Basisvektor $\overrightarrow{y_e} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow \overrightarrow{y_n} = \begin{pmatrix} 0 \\ \frac{2}{t-b} \\ 0 \end{pmatrix}$

additiver (homogener) Anteil: $-\frac{t+b}{t-b} \rightarrow P_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$

Illustrationen und Formeln von Song Ho Ahn

Orthografische Projektion - Herleitung

Mapping von z_e nach z_n:



$$z_n = \frac{1-(-1)}{-f-(-n)} \cdot z_e + \beta$$

$$1 = \frac{2f}{f-n} + \beta \qquad \text{(substitute } (-f, 1) \text{ for } (z_e, z_n))$$

$$\beta = 1 - \frac{2f}{f-n} = -\frac{f+n}{f-n}$$

$$\therefore z_n = \frac{-2}{f-n} \cdot z_e - \frac{f+n}{f-n}$$

linearer Basisvektor $\overrightarrow{z_e} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow \overrightarrow{z_n} = \begin{pmatrix} 0 \\ 0 \\ \frac{-2}{f-n} \end{pmatrix}$

additiver (homogener) Anteil: $-\frac{t+b}{t-b} \rightarrow P_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ ? & ? & ? & ? \end{bmatrix}$

Illustrationen und Formeln von Song Ho Ahn

Orthografische Projektion - Herleitung

Mapping von w_e nach w_n:

Es handelt sich um eine affine Transformation. Der homogene Teil bleibt also unverändert.

$$P_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Illustrationen und Formeln von Song Ho Ahn

Orthografische Projektion in three.js

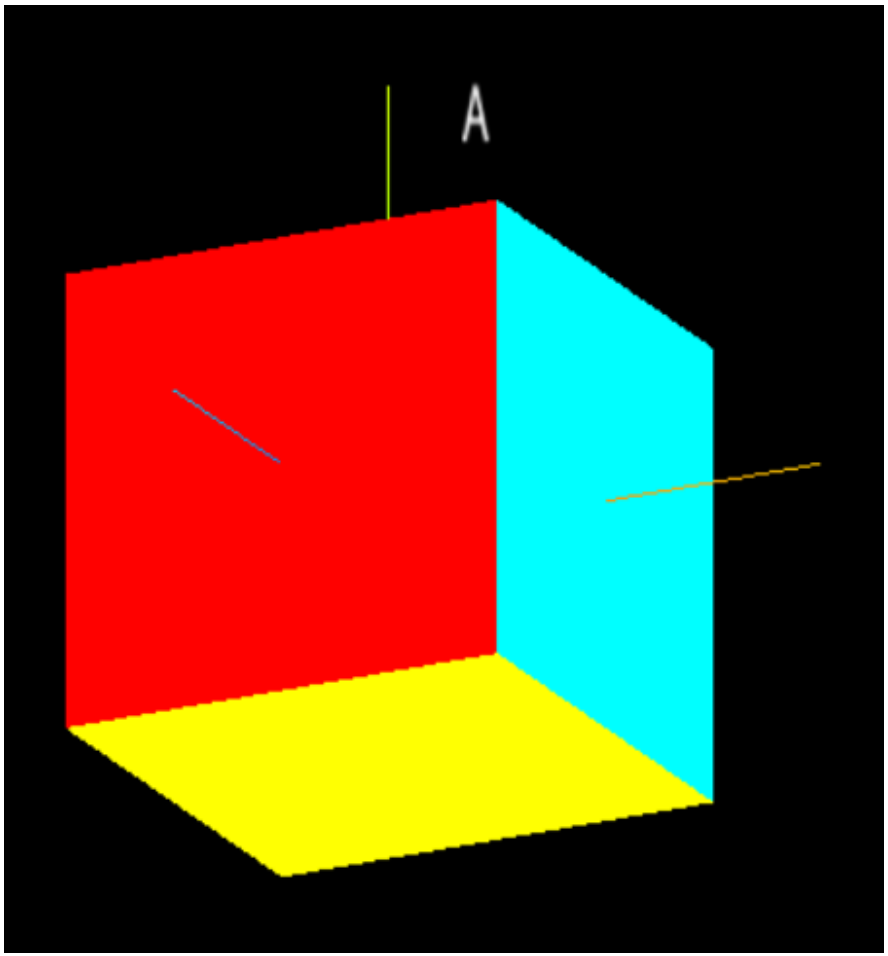
Was ergibt sich für die Projektionsmatrix?

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
const left = -1;
const right = 1;
const top = 1;
const bottom = -1;
const near = 1;
const far = 4;
```

```
cam.rotation.set(0, 0, 0);
cam.position.x = 0;
cam.position.y = 0;
cam.position.z = 3;
```

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{2}{3} & -\frac{5}{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



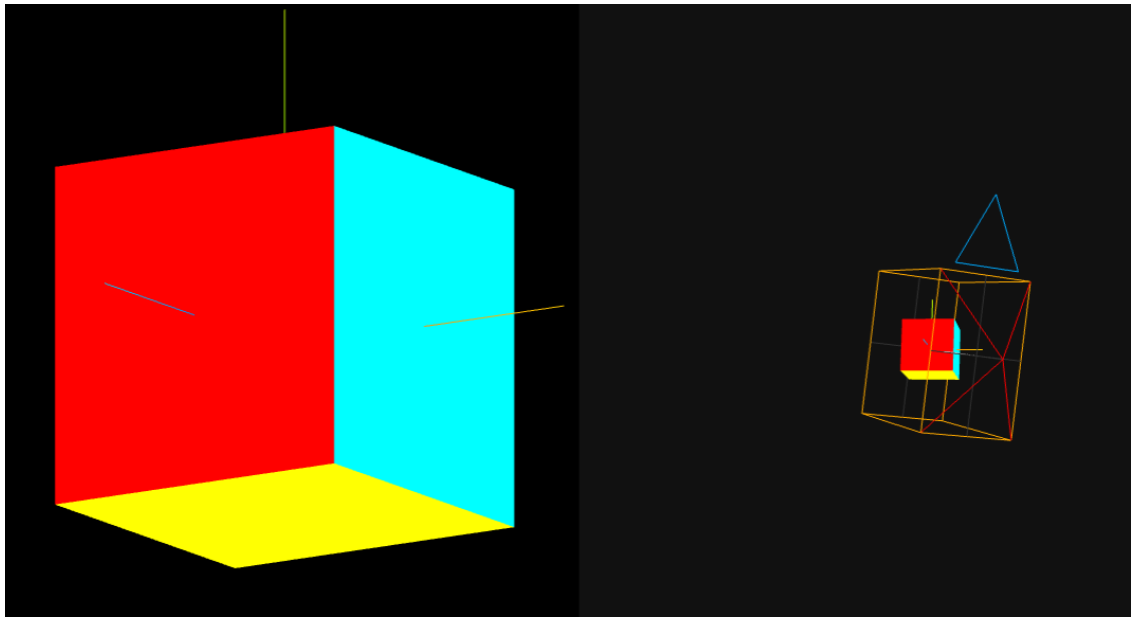
Orthografische Projektion in three.js

Wo landet die Ecke eines Würfels auf dem Bildschirm?

$$A = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}, P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{2}{3} & -\frac{5}{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

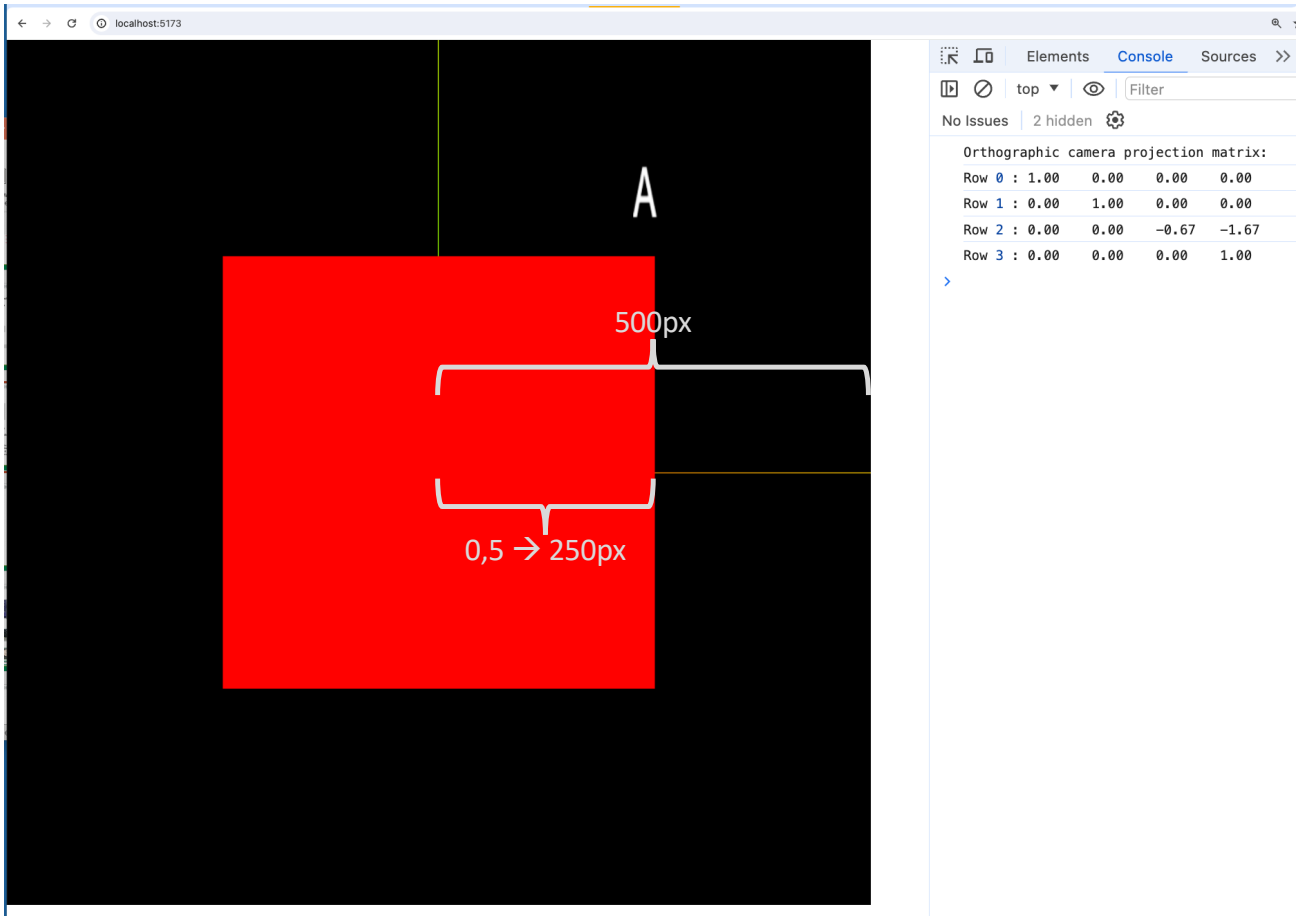
Der Würfel ist um drei Einheiten in z-Richtung relativ zur Kamera verschoben.

$$A_{ortho} = PA = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 1 \end{pmatrix}$$



Orthografische Projektion des Eckpunkts A

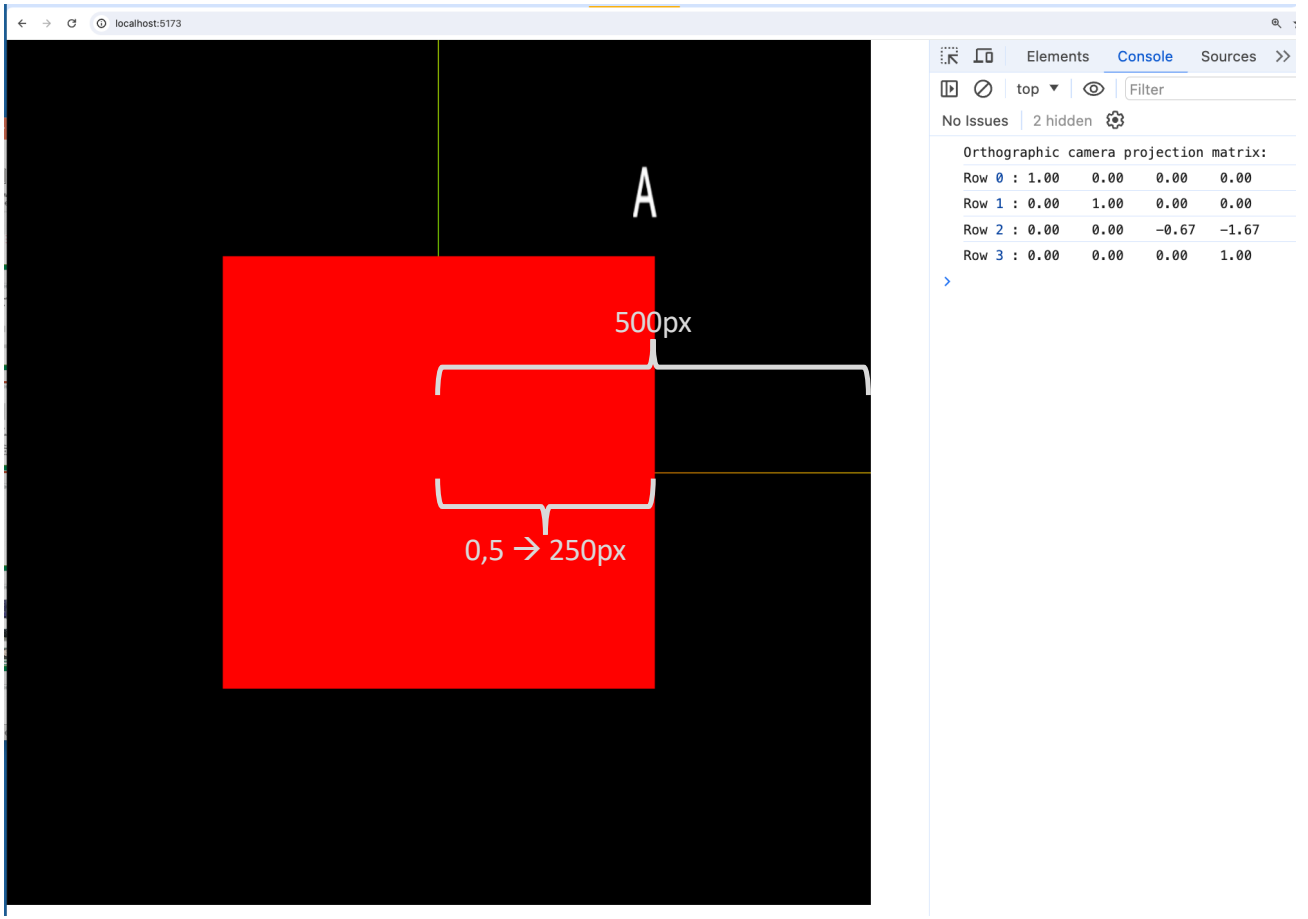
$$A_{ortho} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 1 \end{pmatrix}$$



Im Bild ist A_{ortho} bei $\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$, wenn man das Koordinatensystem in der Mitte nimmt.

Orthografische Projektion des Eckpunkts A

$$A_{ortho} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ 1 \end{pmatrix}$$

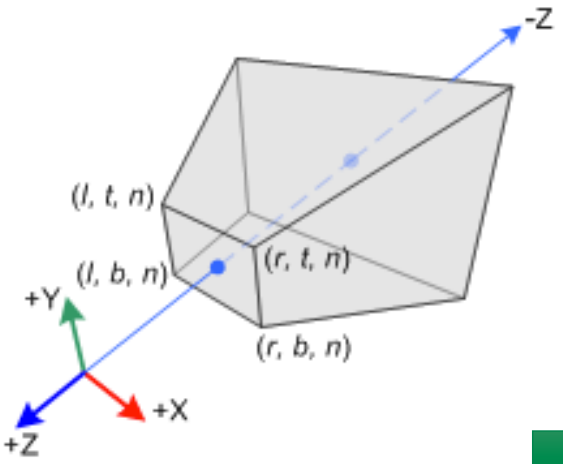


Der Z-Wert wird verwendet um zu überprüfen ob der Punkt im Clipspace [-1,1] liegt und zur Sortierung sich überdeckender Punkte bzw. Flächen.

Perspektivische Projektion

Illustrationen von Song Ho Ahn

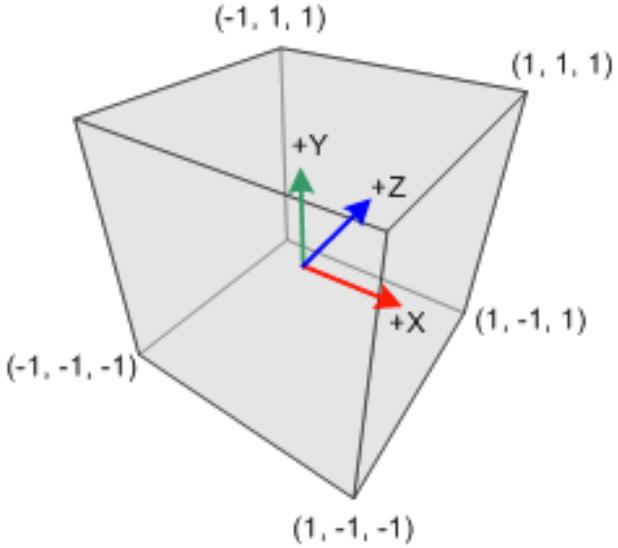
Kamera-Koordinaten



$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Projektionsmatrix

Clip-Koordinaten

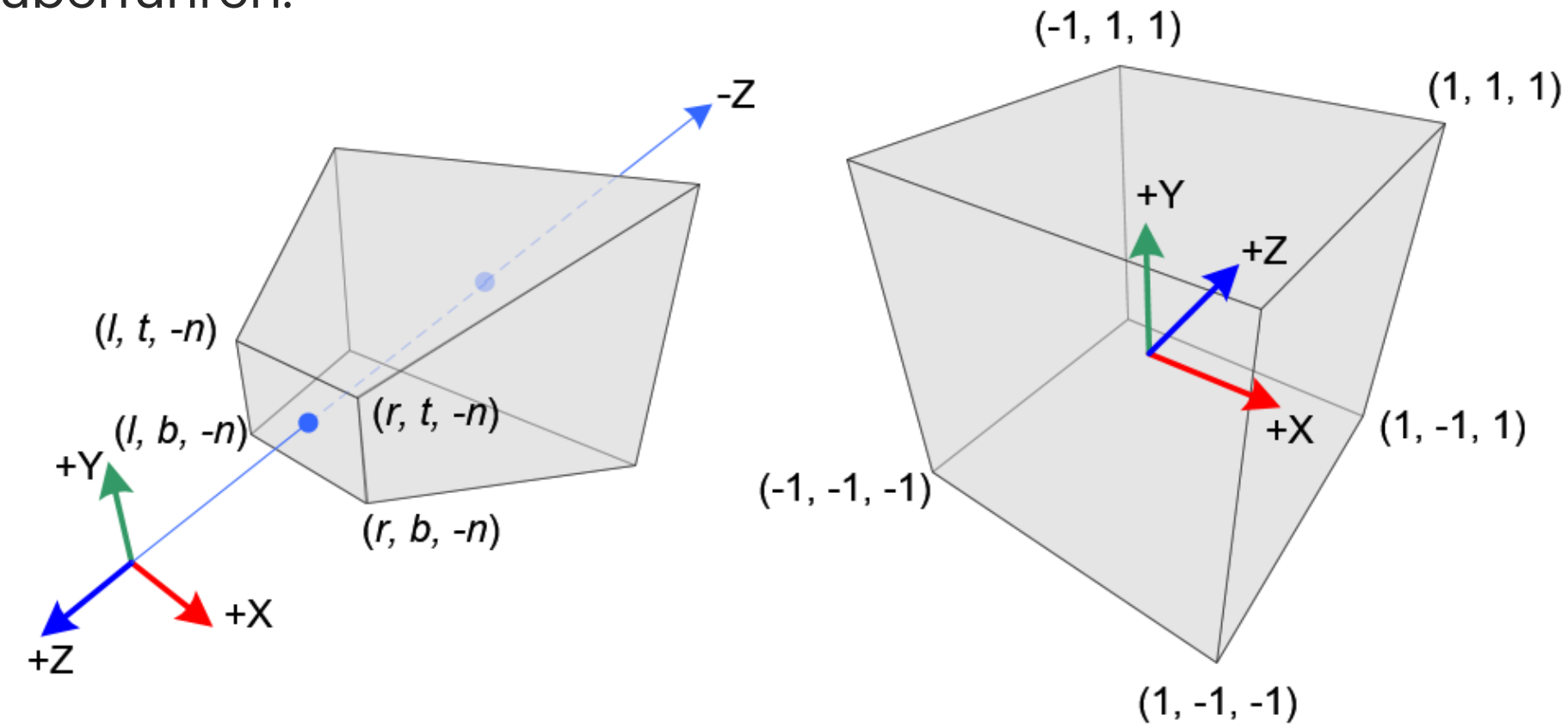


Im Code (hier three.js):

```
var camera = new THREE.PerspectiveCamera(fovy,
aspect, near, far);
```

Perspektivische Projektion – Herleitung

Wir müssen einen Pyramidenstumpf (Frustum) in einen Würfel überführen:

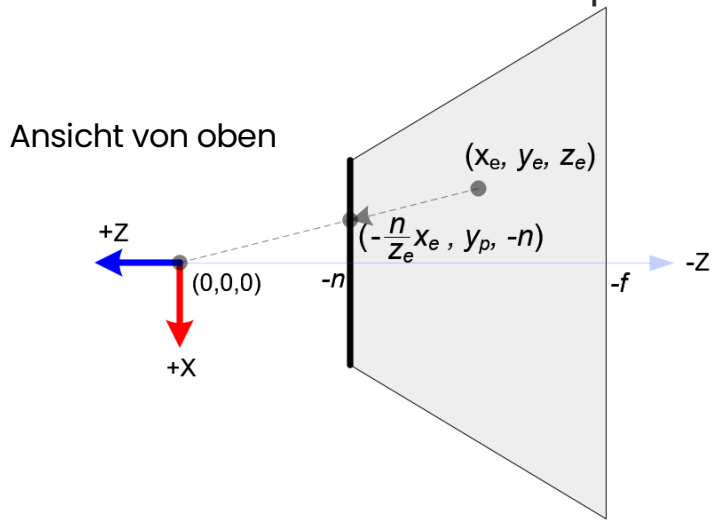


Also die Komponenten x_e, y_e, z_e (eye/camera space, links) in x_n, y_n, z_n (normalized clip space, rechts).

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

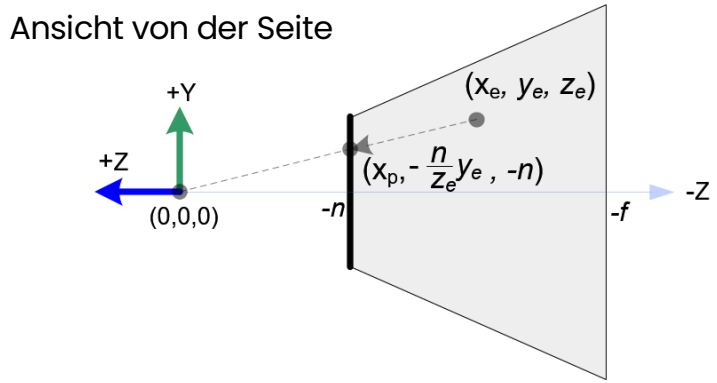
Mapping von x_e nach x_p, also auf die z_p=-n Ebene (near plane)



$$\frac{x_p}{x_e} = \frac{-n}{z_e}$$

$$x_p = \frac{-n \cdot x_e}{z_e} = \frac{n \cdot x_e}{-z_e}$$

Mapping von y_e nach y_p,also auf die z_p=-n Ebene (near plane)



$$\frac{y_p}{y_e} = \frac{-n}{z_e}$$

$$y_p = \frac{-n \cdot y_e}{z_e} = \frac{n \cdot y_e}{-z_e}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

Man beachte, dass sowohl x_p als auch y_p von z_e abhängen. Sie sind invers propotional zu $-z_e$. Anders ausgedrückt: sie wurden beide jeweils durch $-z_e$ geteilt.

Der Clipspace ist auch ein homogener (4D) Raum. Es wird also am Ende durch die w_c Komponente geteilt. Dies nutzen wir aus und setzen:

$w_c = -z_e$

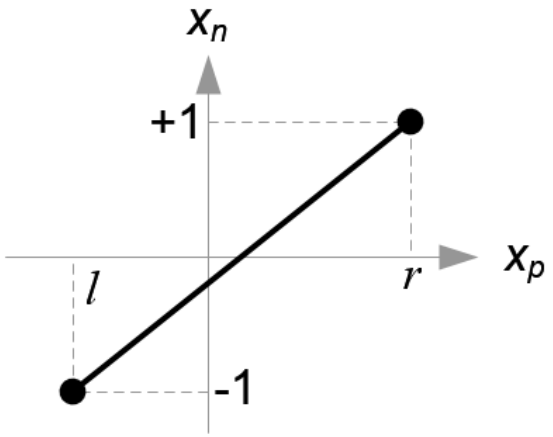
$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \\ z_c \\ -z_e \end{pmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}$$

$$P_{persp} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

Mapping von x_p nach x_n:



$$x_n = \frac{1-(-1)}{r-l} \cdot x_p + \beta$$

$$1 = \frac{2r}{r-l} + \beta \qquad \text{(substitute } (r,1) \text{ for } (x_p, x_n)\text{)}$$

$$\beta = 1 - \frac{2r}{r-l} = \frac{r-l}{r-l} - \frac{2r}{r-l}$$

$$= \frac{r-l-2r}{r-l} = \frac{-r-l}{r-l} = -\frac{r+l}{r-l}$$

$$\therefore x_n = \frac{2x_p}{r-l} - \frac{r+l}{r-l}$$

Dann für x_p Formel mit x_e einsetzen:

$$x_n = \frac{2x_p}{r-l} - \frac{r+l}{r-l} \qquad \left(x_p = \frac{nx_e}{-z_e}\right)$$

$$= \frac{2 \cdot \frac{n \cdot x_e}{-z_e}}{r-l} - \frac{r+l}{r-l}$$

$$= \frac{2n \cdot x_e}{(r-l)(-z_e)} - \frac{r+l}{r-l}$$

$$= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} - \frac{r+l}{r-l}$$

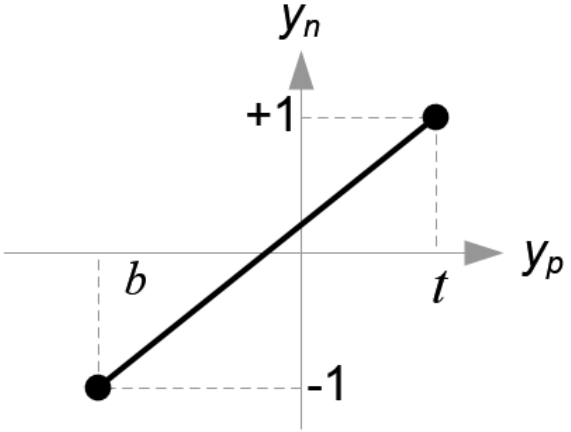
$$= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} + \frac{\frac{r+l}{r-l} \cdot z_e}{-z_e}$$

$$= \left(\underbrace{\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e}_{x_c} \right) \Big/ - z_e$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

Mapping von y_p nach y_n:



$$y_n = \frac{1-(-1)}{t-b} \cdot y_p + \beta$$

$$1 = \frac{2t}{t-b} + \beta \qquad \text{(substitute } (t,1) \text{ for } (y_p, y_n)\text{)}$$

$$\beta = 1 - \frac{2t}{t-b} = \frac{t-b}{t-b} - \frac{2t}{t-b}$$

$$= \frac{t-b-2t}{t-b} = \frac{-t-b}{t-b} = -\frac{t+b}{t-b}$$

$$\therefore y_n = \frac{2y_p}{t-b} - \frac{t+b}{t-b}$$

Dann für y_p Formel mit y_e einsetzen:

$$y_n = \frac{2y_p}{t-b} - \frac{t+b}{t-b} \qquad (y_p = \frac{ny_e}{-z_e})$$

$$= \frac{2 \cdot \frac{n \cdot y_e}{-z_e}}{t-b} - \frac{t+b}{t-b}$$

$$= \frac{2n \cdot y_e}{(t-b)(-z_e)} - \frac{t+b}{t-b}$$

$$= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} - \frac{t+b}{t-b}$$

$$= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} + \frac{\frac{t+b}{t-b} \cdot z_e}{-z_e}$$

$$= \Big(\underbrace{\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e}_{y_c} \Big) \Big/ -z_e$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

Aus den Ergebnissen lassen sich zwei weitere Zeilen der Projektionsmatrix aufstellen:

$$= \left(\underbrace{\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e}_{x_c} \right) \Big/ -z_e \qquad = \left(\underbrace{\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e}_{y_c} \right) \Big/ -z_e$$

$$P_{persp} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ ? & ? & ? & ? \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

Die Suche nach z_n unterscheidet sich ein wenig von den anderen, da z_e im Kamerakoordinatensystem immer auf die z= -n Eben (near plane) projiziert wird.

Wir benötigen aber einen eindeutigen z-Wert für das Clipping und den Tiefenvergleich. Außerdem sollten wir in der Lage sein, ihn zurück zu projizieren (invers zu transformieren).

Da wir wissen, dass z nicht vom x- oder y-Wert abhängt, nehmen wir die w-Komponente, um die Beziehung zwischen z_n und z_e zu finden. Daher können wir die 3. Zeile der Projektionsmatrix wie folgt angeben:

$$P_{persp} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$z_n = z_c/w_c = \frac{Az_e + Bw_e}{-z_e}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

Im Kamerakoordinatensystem (eye space) gilt w_e=1. Daher gilt:

$$z_n = \frac{Az_e + B}{-z_e}$$

Um die Koeffizienten A und B zu ermitteln, verwenden wir die Beziehung von z_e zu z_n für (-n, -1) und (-f, 1). Wir setzen sie in die obige Gleichung ein.

$$\begin{cases} \dfrac{-An + B}{n} = -1 \\ \dfrac{-Af + B}{f} = 1 \end{cases} \rightarrow \begin{cases} -An + B = -n & (1) \\ -Af + B = f & (2) \end{cases}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

To solve the equations for *A* and *B*, rewrite eq.(1) for B;

$$B = An - n \qquad (1')$$

Substitute eq.(1') to B in eq.(2), then solve for A;

$$-Af + (An - n) = f \qquad (2)$$

$$-(f - n)A = f + n$$

$$A = -\frac{f+n}{f-n}$$

Put A into eq.(1) to find B;

$$\left(\frac{f+n}{f-n}\right) n + B = -n \qquad (1)$$

$$B = -n - \left(\frac{f+n}{f-n}\right) n = -\left(1 + \frac{f+n}{f-n}\right) n = -\left(\frac{f-n+f+n}{f-n}\right) n$$

$$= -\frac{2fn}{f-n}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion - Herleitung

We found A and B. Therefore, the relation between z_e and z_n becomes;

$$z_n = \frac{-\frac{f+n}{f-n} z_e - \frac{2fn}{f-n}}{-z_e} \qquad (3)$$

$$P_{persp} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Illustrationen und Formeln von Song Ho Ahn

Perspektivische Projektion in three.js

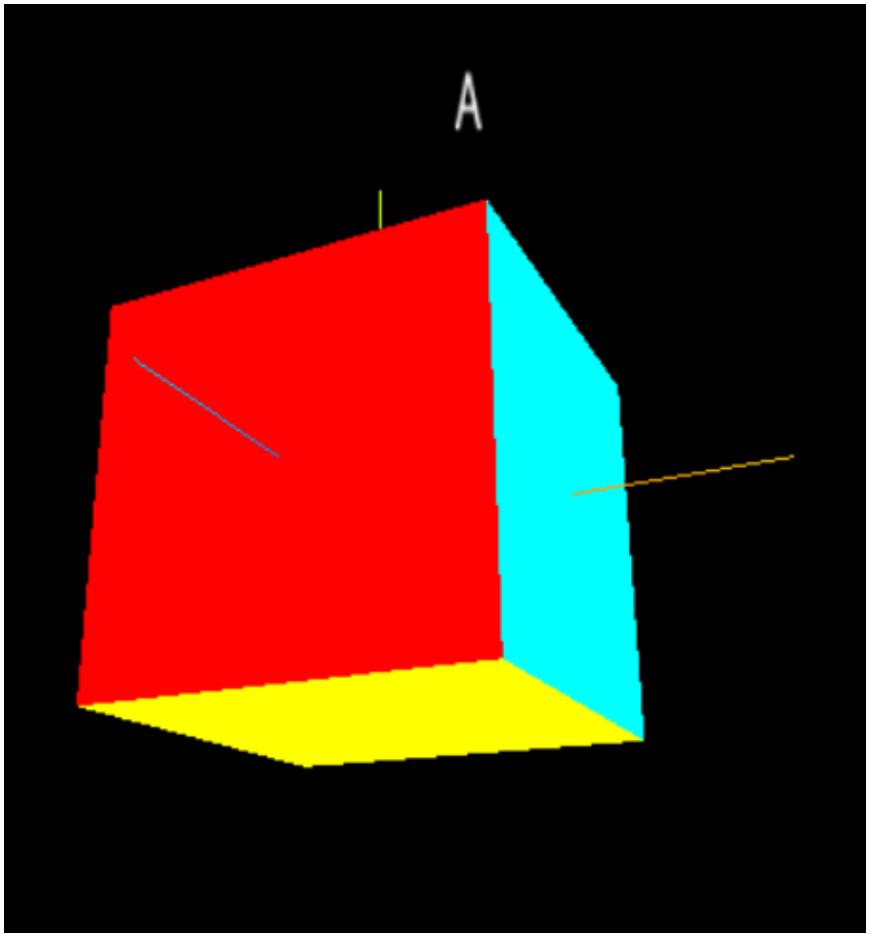
Was ergibt sich für die Projektionsmatrix?

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

```
const left = -1;
const right = 1;
const top = 1;
const bottom = -1;
const near = 1;
const far = 4;
```

```
cam.rotation.set(0, 0, 0);
cam.position.x = 0;
cam.position.y = 0;
cam.position.z = 3;
```

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{5}{3} & -\frac{8}{3} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



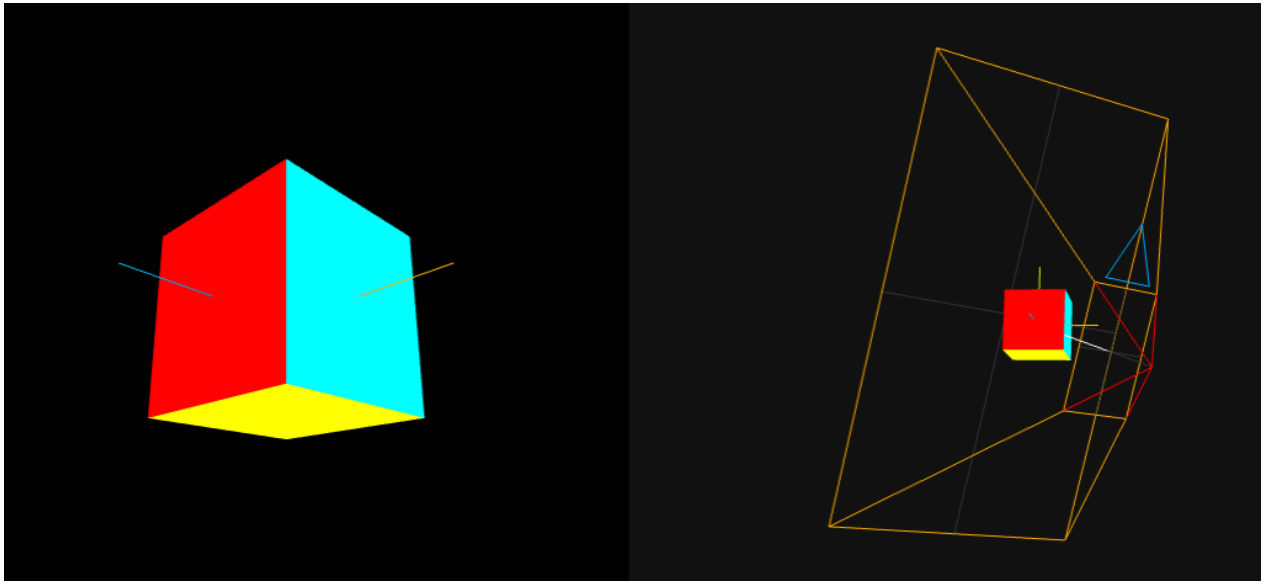
Perspektivische Projektion in three.js

Wo landet die Ecke eines Würfels auf dem Bildschirm?

$$A = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} - 3 \\ 1 \end{pmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{5}{3} & -\frac{8}{3} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

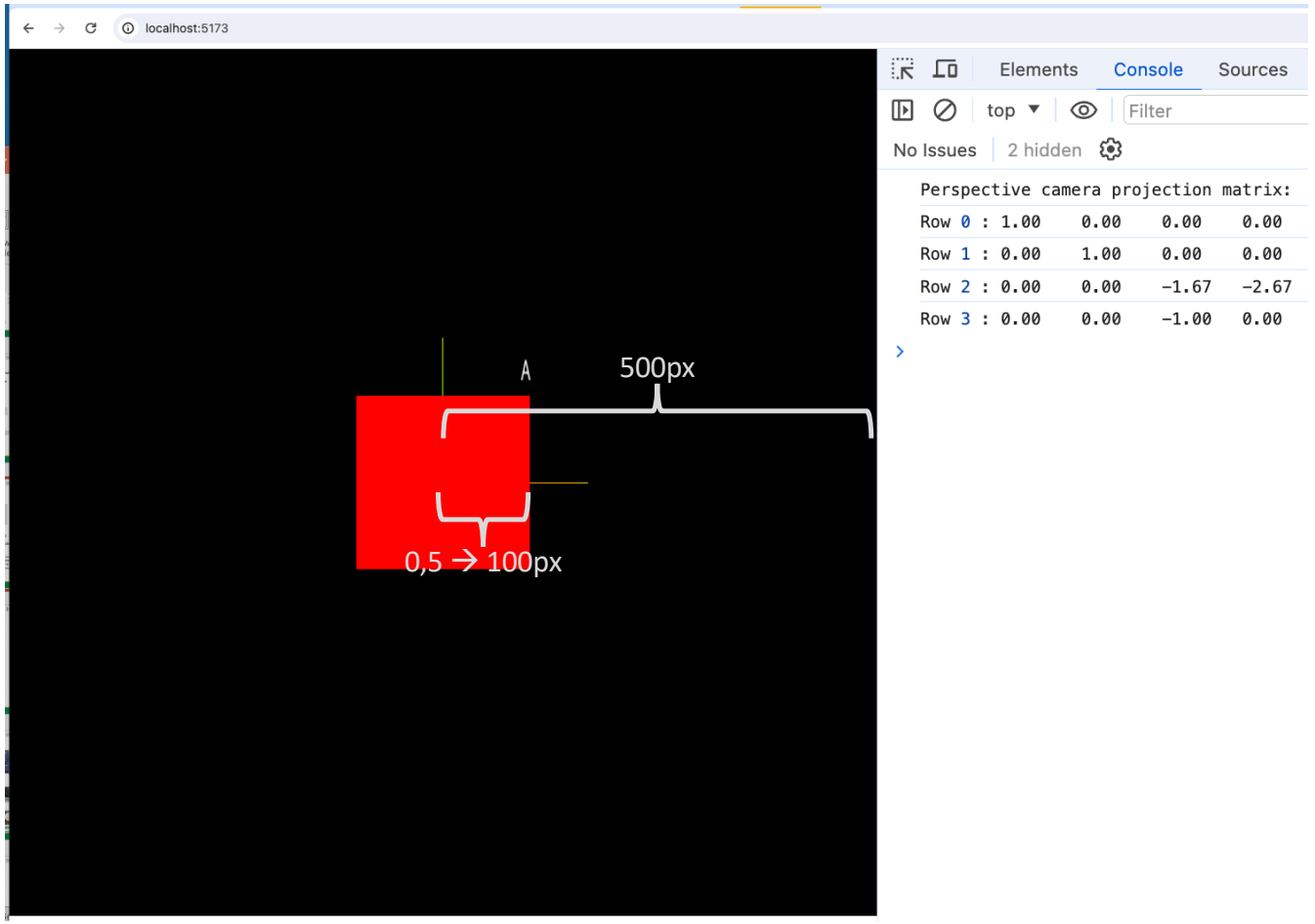
Der Würfel ist um drei Einheiten in z-Richtung relativ zur Kamera verschoben.

$$A_{persp} = PA = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{3}{2} \\ \frac{5}{2} \end{pmatrix}$$



Perspektivische Projektion des Eckpunkts A

$$A_{persp} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{3}{2} \\ \frac{5}{2} \end{pmatrix} \text{ !?}$$

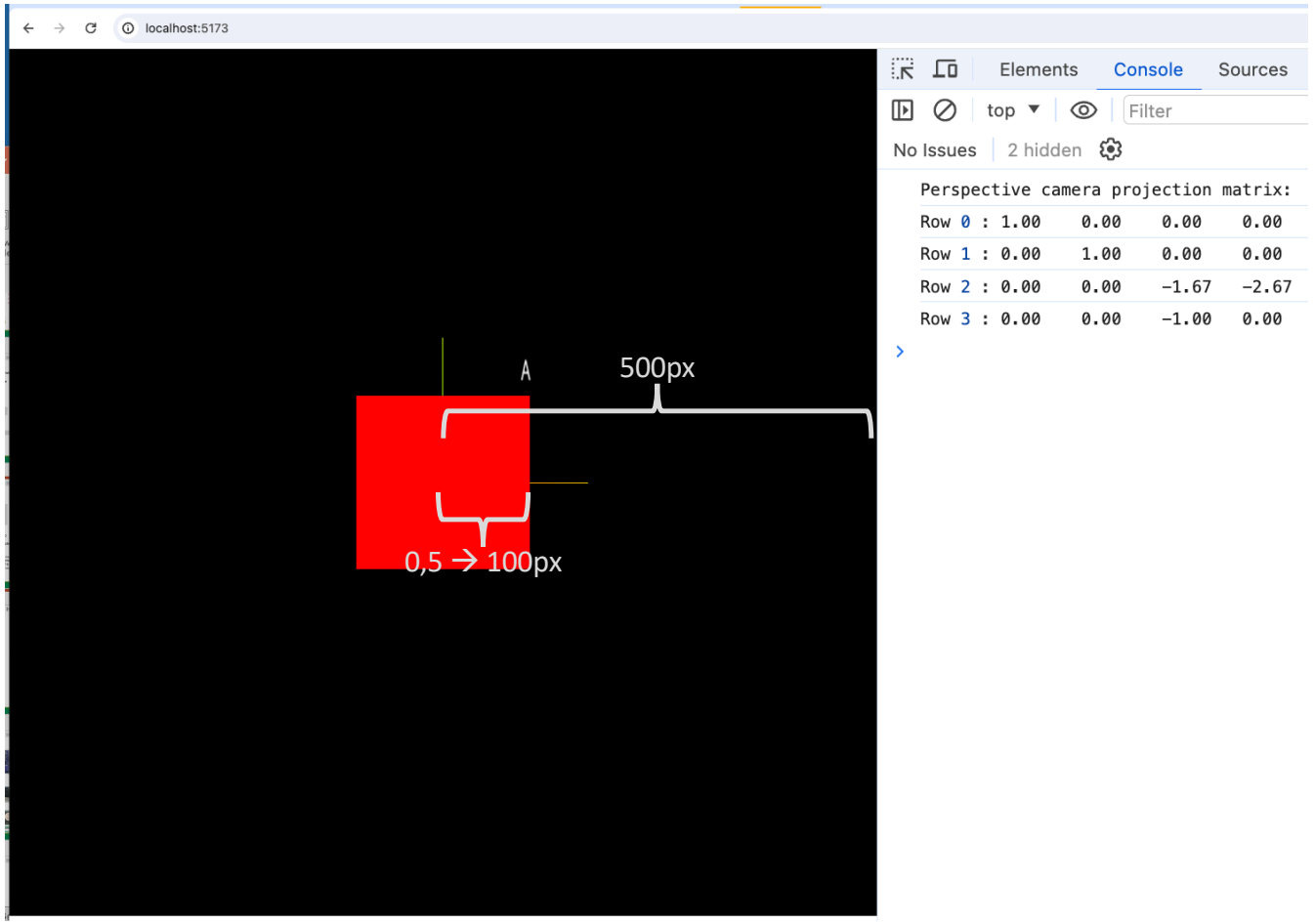


Im Bild ist A_{persp} bei $\begin{pmatrix} \frac{1}{5} \\ \frac{1}{5} \end{pmatrix}$, wenn man das Koordinatensystem in der Mitte nimmt.

Wieso!?

Perspektivische Projektion des Eckpunkts A

$$A_{persp} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{3}{2} & \frac{5}{2} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

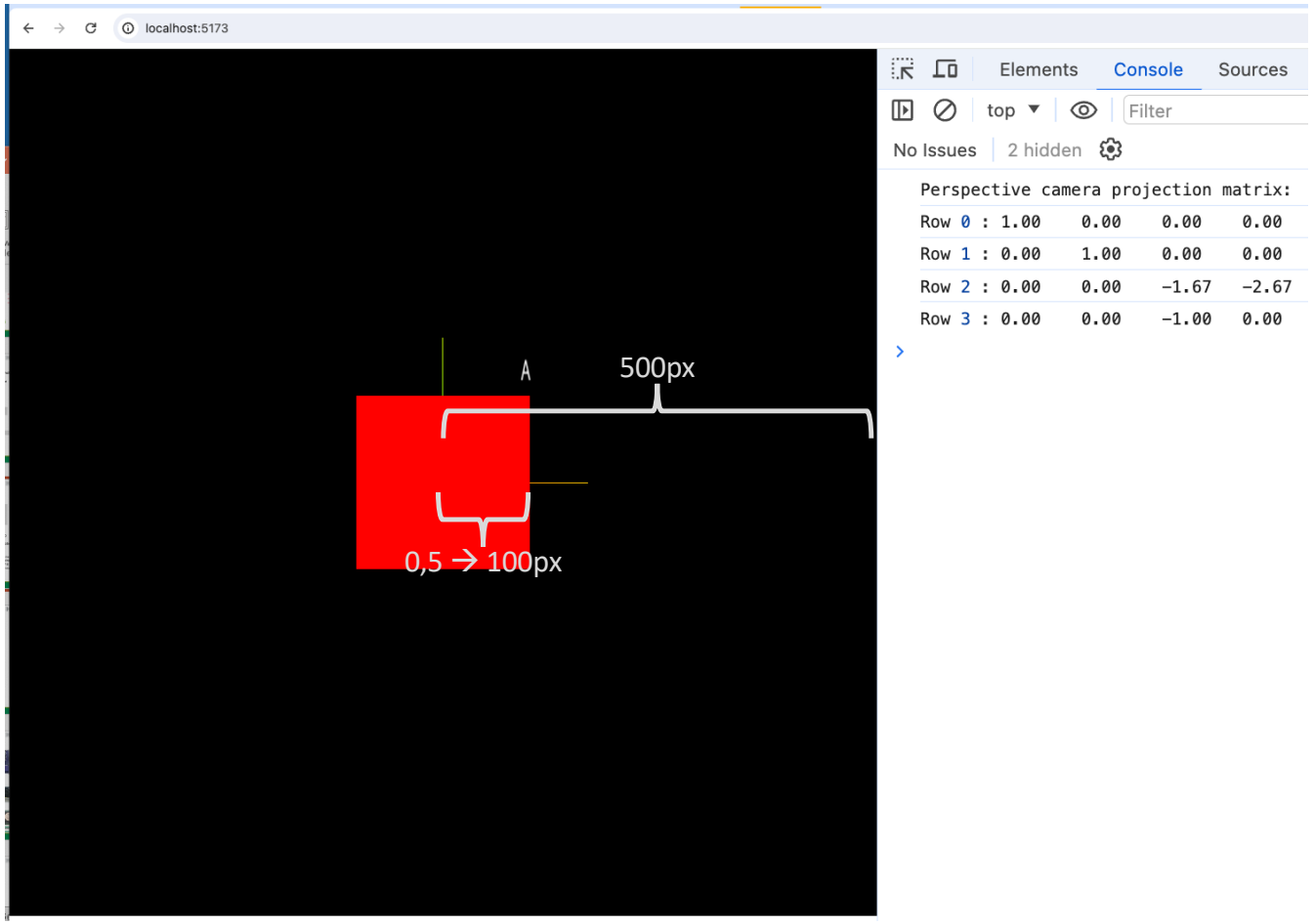


Wir teilen durch die homogene Koordinate:

$$w = \frac{5}{2}$$

Perspektivische Projektion des Eckpunkts A

$$A_{persp} = \begin{pmatrix} 1 & & & \\ & \frac{1}{5} & & \\ & & \frac{1}{5} & \\ & & \frac{3}{5} & \\ & & & 1 \end{pmatrix}$$

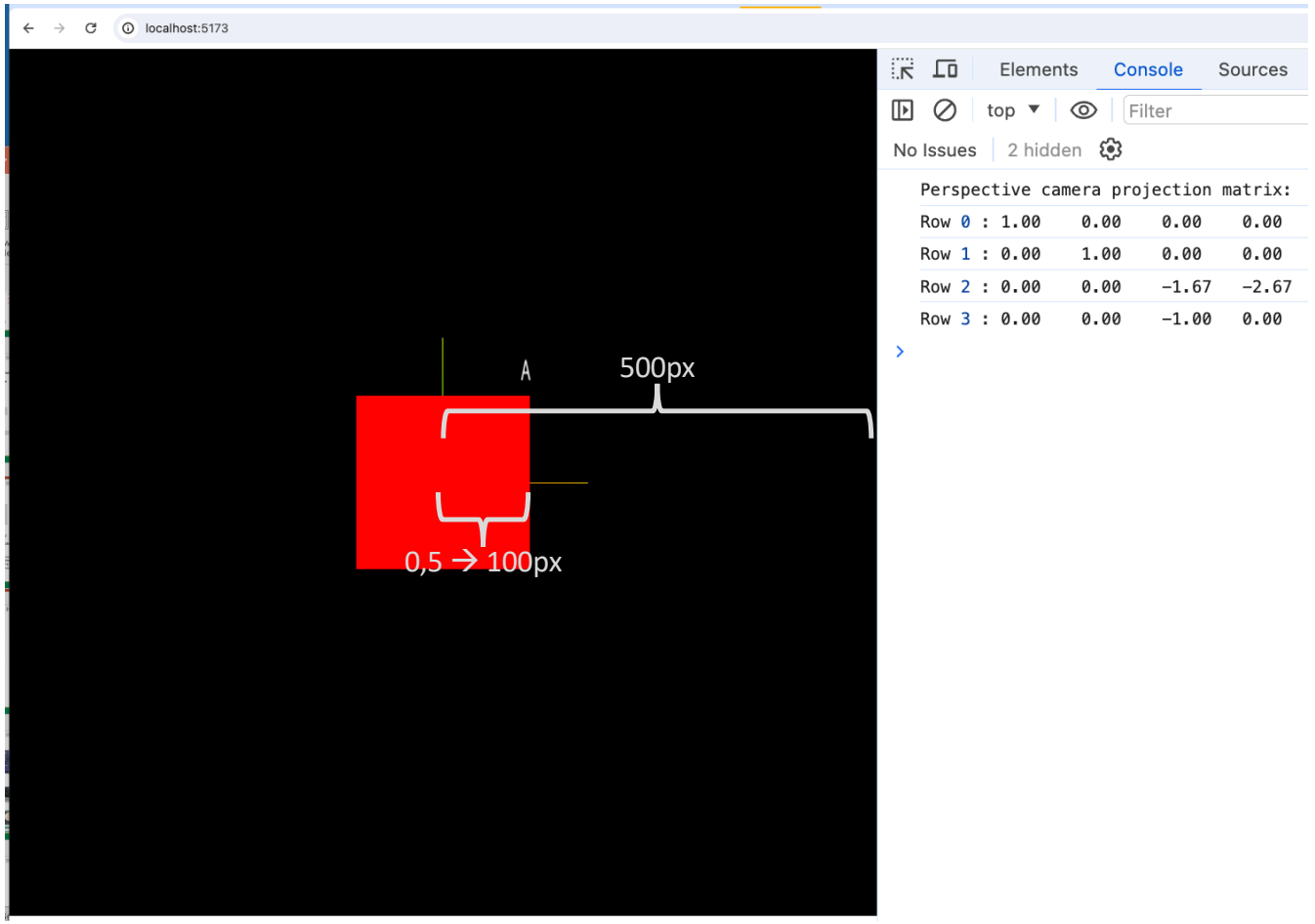


Wir teilen durch die homogene Koordinate:

Für x und y: $\frac{1}{2} / \frac{5}{2} = \frac{1}{2} \cdot \frac{2}{5} = \frac{1}{5}$

Perspektivische Projektion des Eckpunkts A

$$A_{persp} = \begin{pmatrix} 1 & & & \\ & \frac{1}{5} & & \\ & & \frac{1}{5} & \\ & & \frac{3}{5} & \\ & & & 1 \end{pmatrix}$$



Der Z-Wert wird verwendet um zu überprüfen ob der Punkt im Clipspace [-1,1] liegt und zur Sortierung sich überdeckender Punkte bzw. Flächen.