# Interaktive Computergrafik

**Prof. Dr. Frank Steinicke**
Human-Computer Interaction
Department of Computer Science
University of Hamburg

# Interaktive Computergrafik

## Lektion 6

**Prof. Dr. Frank Steinicke**
Human-Computer Interaction, Universität Hamburg

# Rückblick: Pipeline

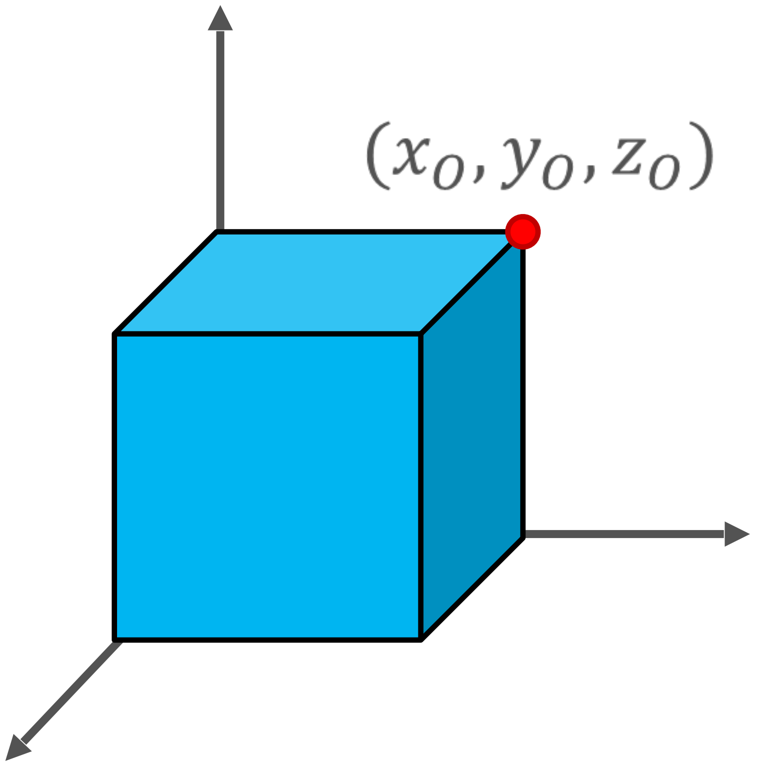
- Objekt-Koordinaten
  - Model Matrix
- Welt-Koordinaten
  - View Matrix
- Sicht-Koordinaten
  - Projection Matrix
- Clip-Koordinaten
  - Dividieren durch w
- Normalisierte Geräte-Koordinaten
  - Skalierung und Translation um Viewport-Parameter
- Screen-Koordinaten

*Definition in Anwendungsprogramm*

# Pipeline

## Ausgangszustand

Objekt-Koordinaten

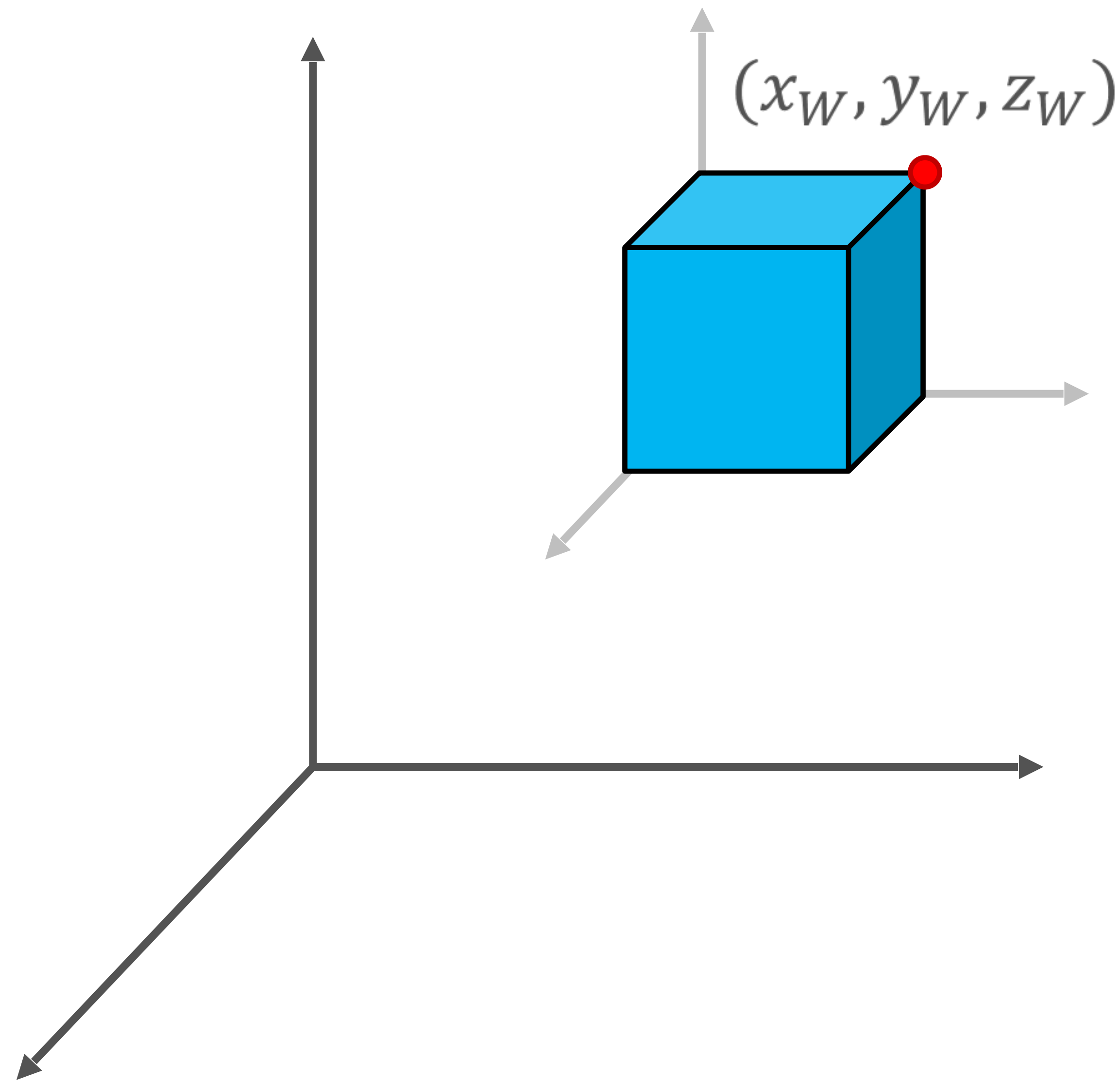


# Pipeline

## 1. Transformation

Objekt-Koordinaten

Welt-Koordinaten

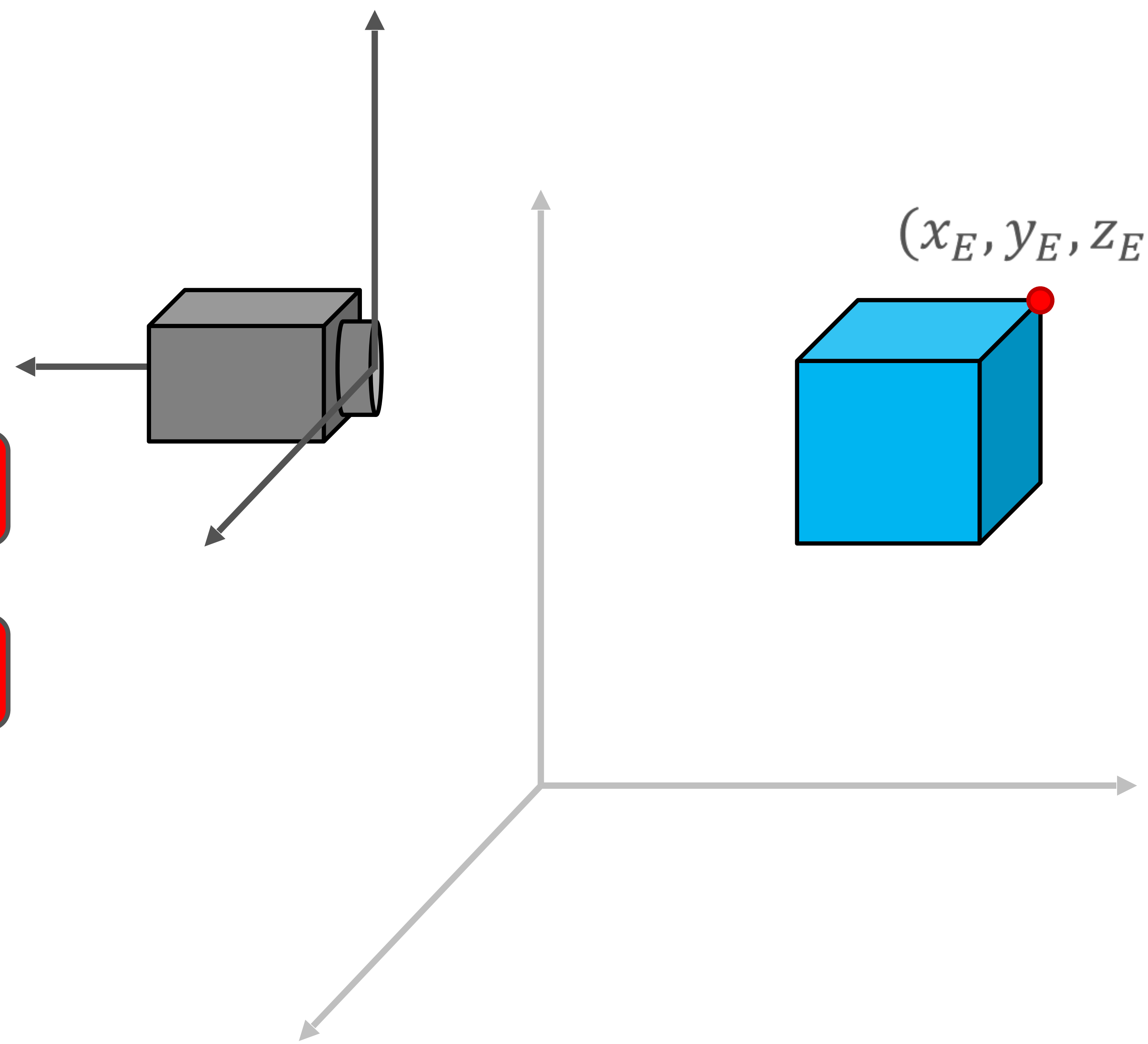


# Pipeline

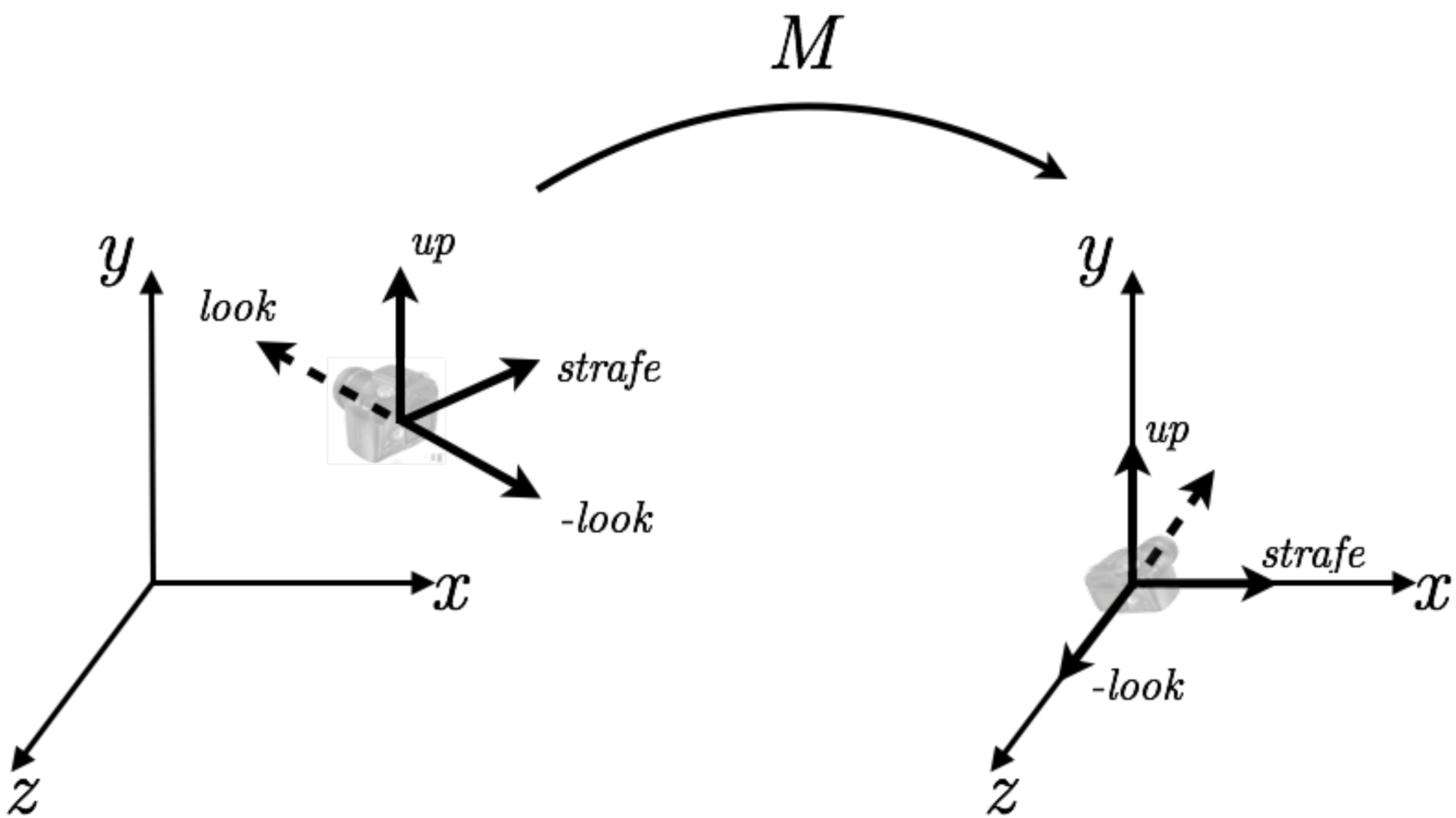
## 2. Transformation

Welt-Koordinaten

Sicht-Koordinaten



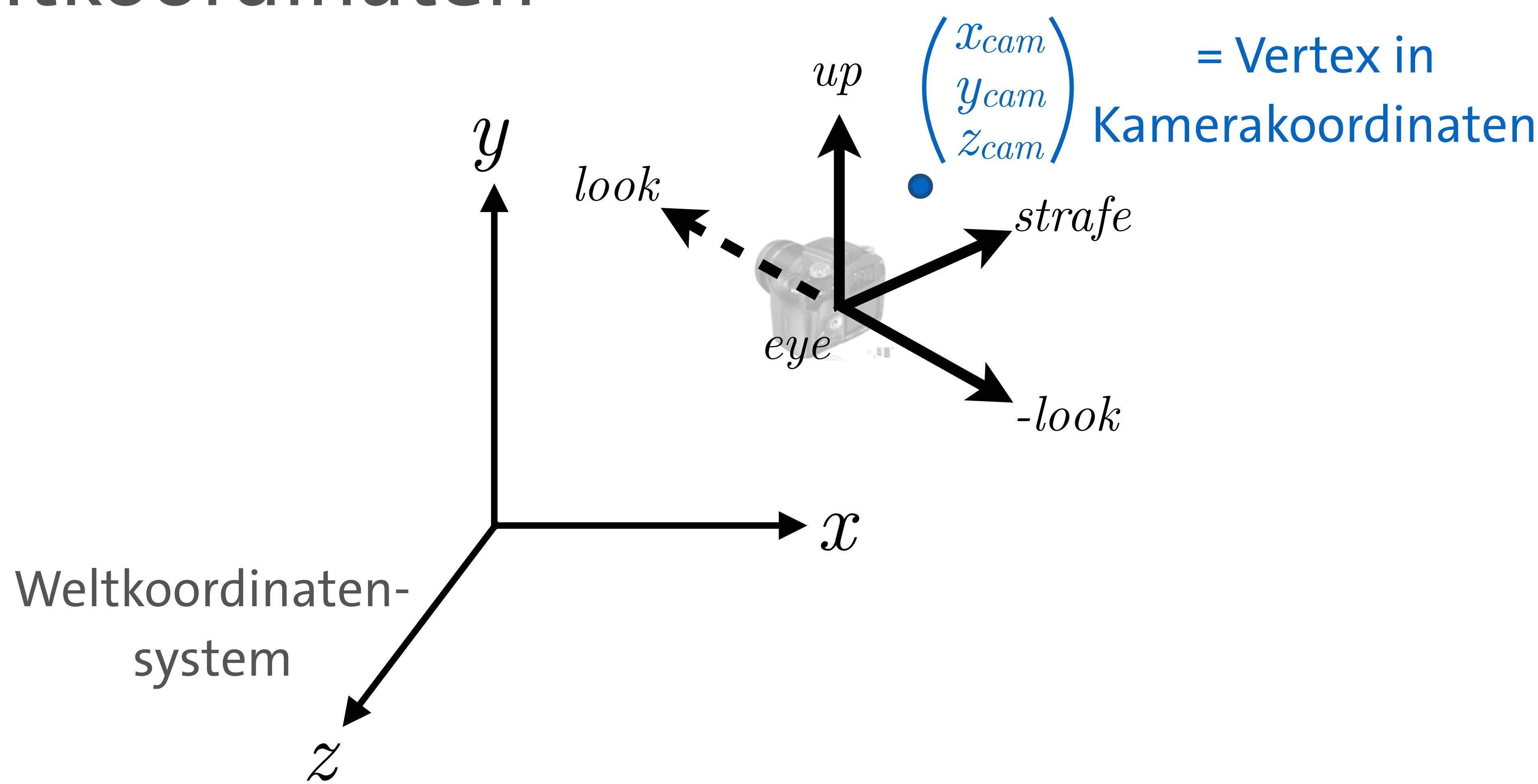
# View Matrix




**Wie berechnet sich die View-Matrix M?**

# View Matrix

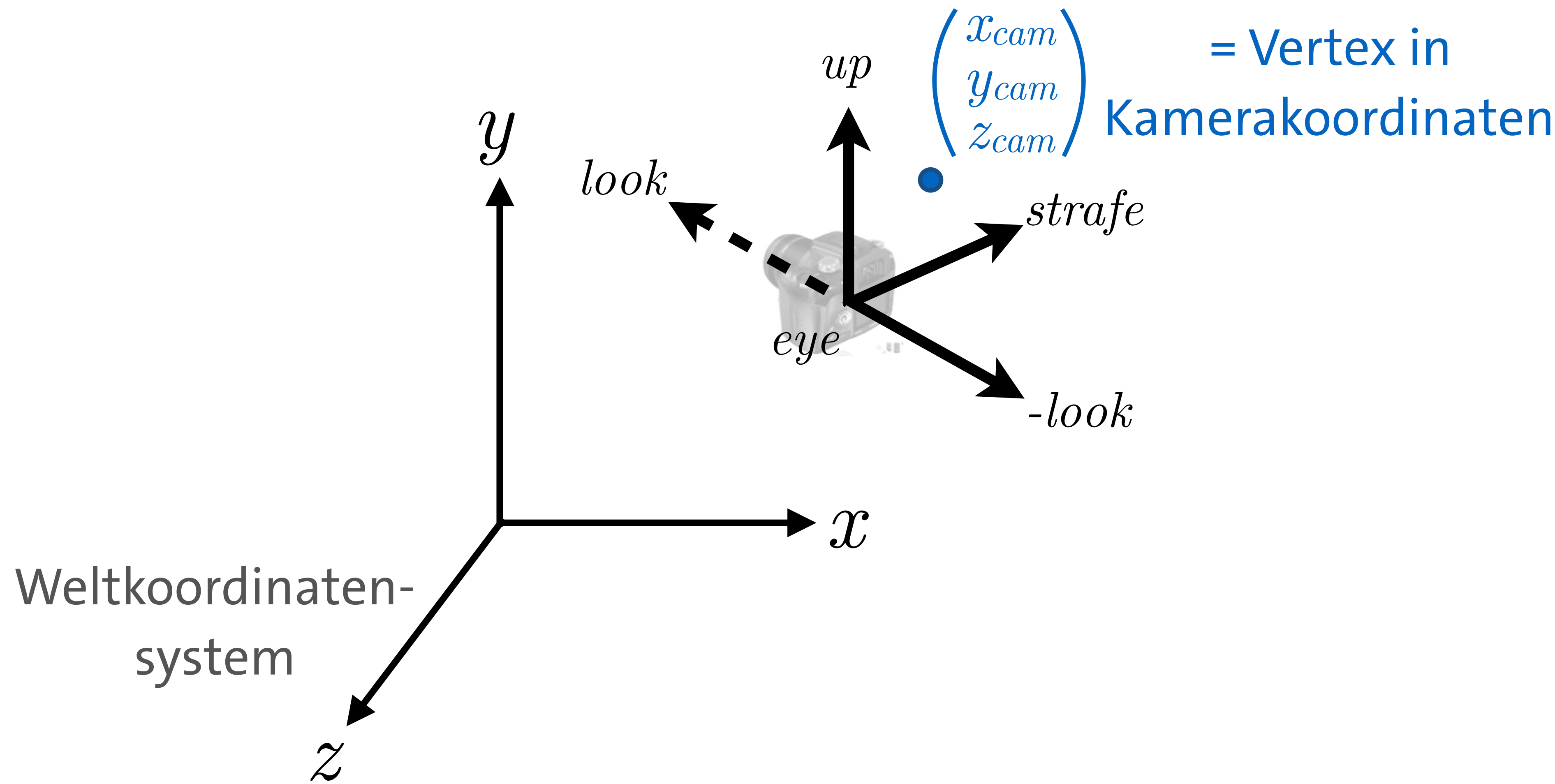
- Betrachten zunächst Gegenrichtung: Matrix $M^{-1}$ transformiert von Kamera- in Weltkoordinaten



# View Matrix

$$\begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \end{pmatrix} = strafe \cdot x_{cam} + up \cdot y_{cam} + (\text{-}look) \cdot z_{cam} + eye$$

= Vertex in Weltkoordinaten



# View Matrix

$$\begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \end{pmatrix} = \overbrace{strafe \cdot x_{cam} + up \cdot y_{cam} + (\text{-}look) \cdot z_{cam}}^{\text{Kamerakoordinaten zu Weltkoordinaten machen}} + \overbrace{eye}^{\text{Vektor zur Kameraposition}}$$

= Vertex in Weltkoordinaten

Vektoren in Weltkoordinatensystem

$$\begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} strafe_x & up_x & \text{-}look_x & eye_x \\ strafe_y & up_y & \text{-}look_y & eye_y \\ strafe_z & up_z & \text{-}look_z & eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{M^{-1}} \cdot \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{pmatrix}$$

= Vertex in homogenen Weltkoordinaten

$M^{-1}$ (inverse View-Matrix)

= Vertex in homogenen Kamerakoordinaten

# View Matrix

$$M^{-1} = \begin{pmatrix} strafe_x & up_x & -look_x & eye_x \\ strafe_y & up_y & -look_y & eye_y \\ strafe_z & up_z & -look_z & eye_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transponiert

$$\boxed{M = \begin{pmatrix} strafe_x & strafe_y & strafe_z & -eye \cdot strafe \\ up_x & up_y & up_z & -eye \cdot up \\ -look_x & -look_y & -look_z & eye \cdot look \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

= View-Matrix

# View Matrix in WebGL

## Library *glMatrix*

- Rückblick: Sammlung von JavaScript-Funktionen für Matrix- und Vektoroperationen

- Darf in Übungsaufgaben benutzt werden!

- Source + Dokumentation: *glmatrix.net*

# View Matrix in WebGL

## Library *glMatrix*

```
mat4.lookAt(
    outMatrix,
    eye,
    target,
    up);
```