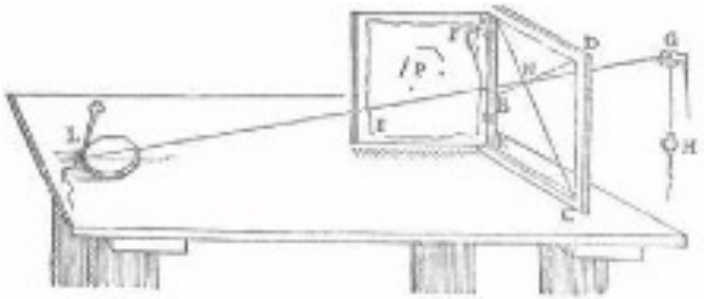
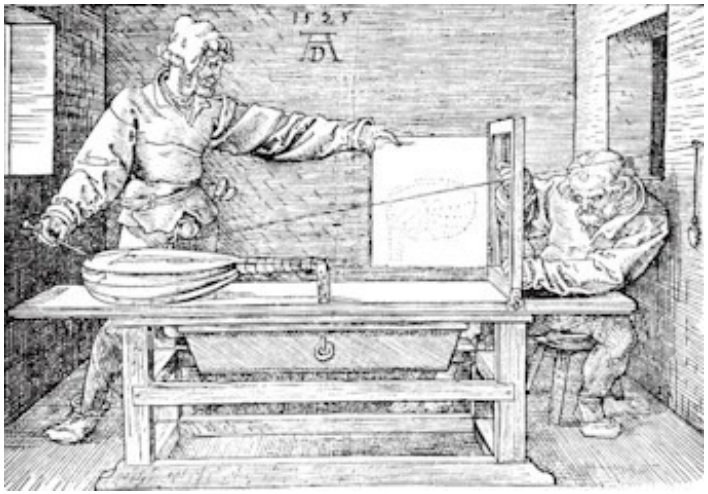
Bremen

CG VR

Computer-Graphik I
Projektionen, Perspektive & Viewing Transformation

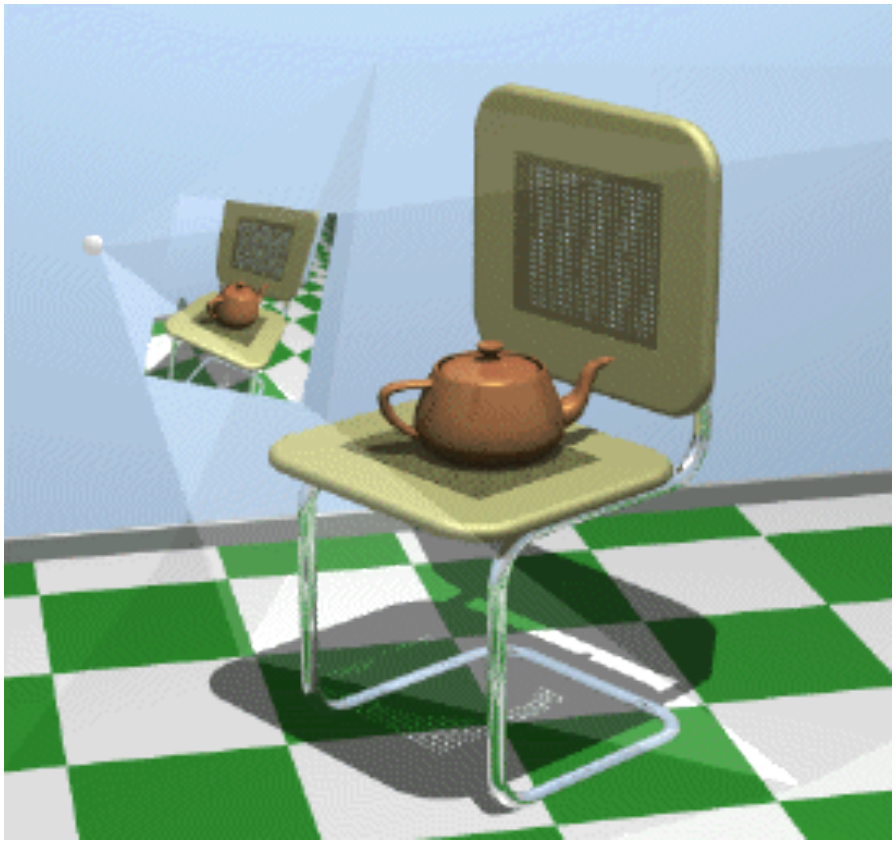




G. Zachmann
University of Bremen, Germany
cgvr.informatik.uni-bremen.de

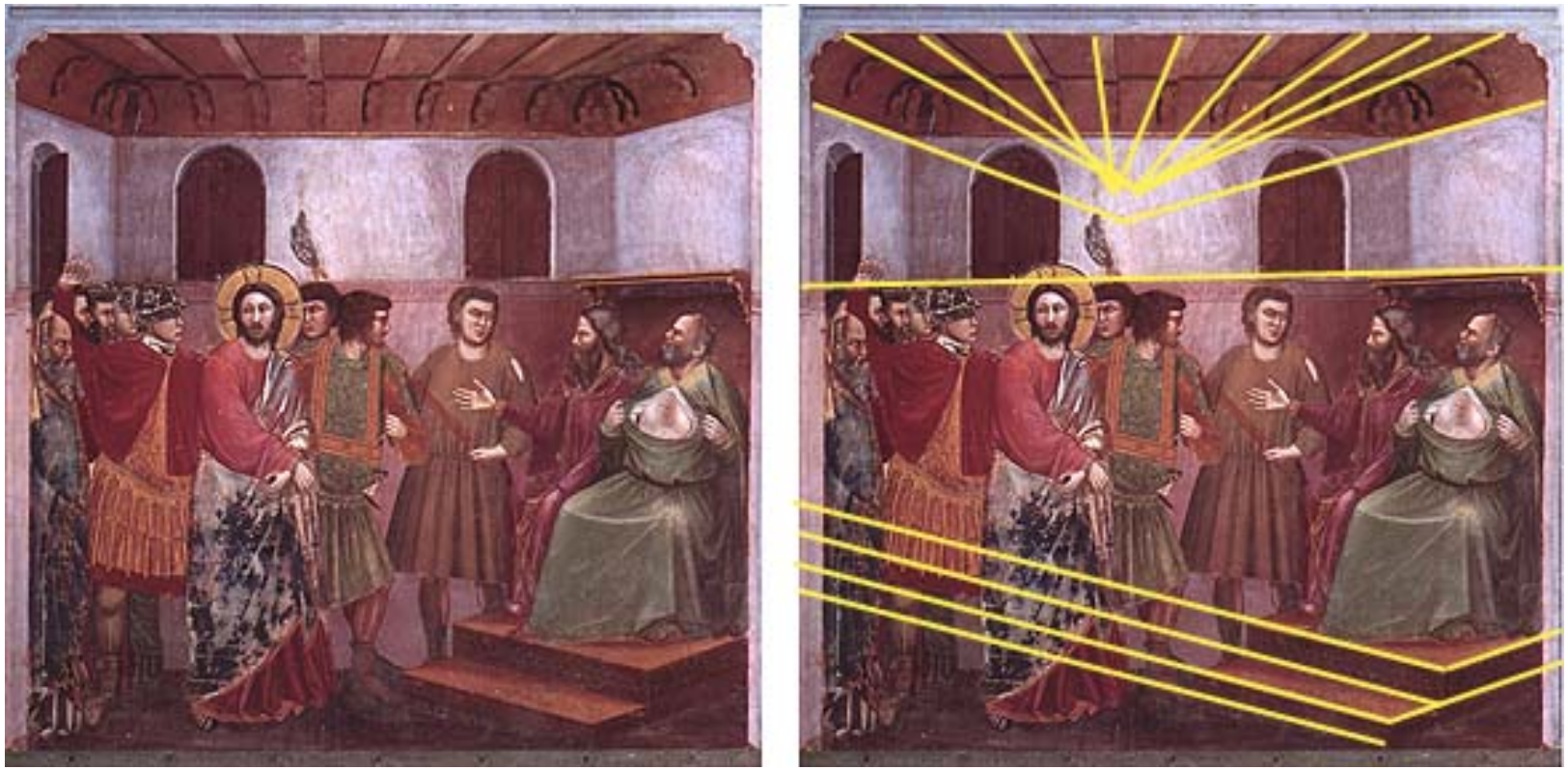
Motivation

- Man möchte die virtuelle 3D Welt auf einem 2D Display darstellen
- Dito in der Malerei (reale Welt → Leinwand)



Perspektive in der Geschichte der Malerei

- Erste Ansätze:



Giotto: Jesus vor Kaiphas (1305)

Brunelleschi's "Peep show" in Florenz, ca. 1410-1420

The Baptistry, San Giovanni, Florence

Duomo and Piazza del Duomo, Firenze

Es dauerte eine Weile bis das Know-How europaweit bekannt war ...

Reconstruction of the temple of Jerusalem.
From William of Tyre: *Histoire d'Outremer. France, Rouen, XVe siècle*
Artiste: Maître de l'Échevinage

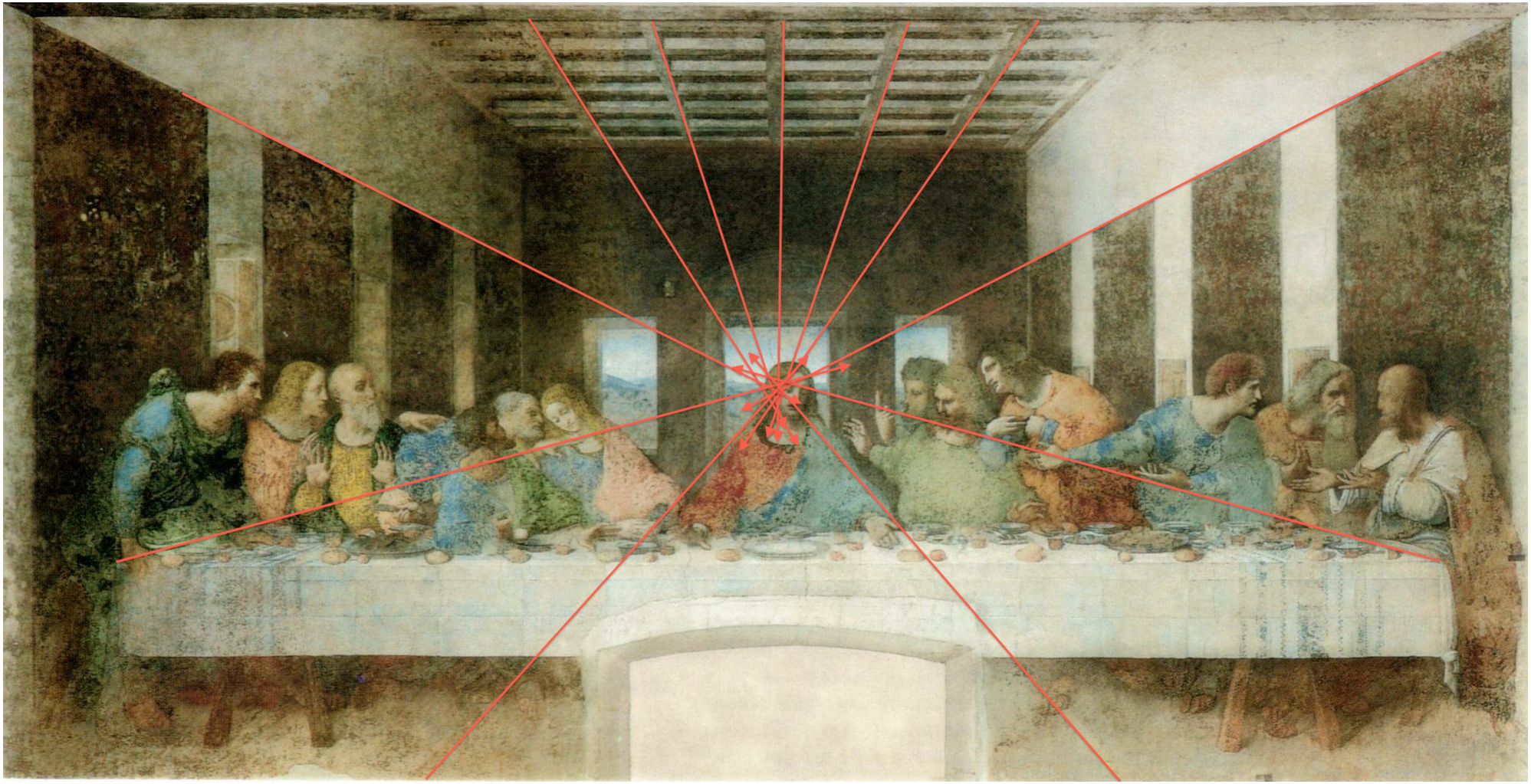
Ca. 1460-1470

Schachbrettmuster wurden sehr beliebt

Christ Handing the Keys to St. Peter

Pietro Perugino (1481-82), Fresco, Sixtinische Kapelle, Vatikan

Der gezielte Einsatz der Perspektive: da Vinci's Abendmahl



1494 — 1498

Erste perspektivische "Rätsel"

Die Gesandten
Hans Holbein
der Jüngere
(1533)

Hat Vermeer eine Camera Obscura (mit Linse) benutzt?

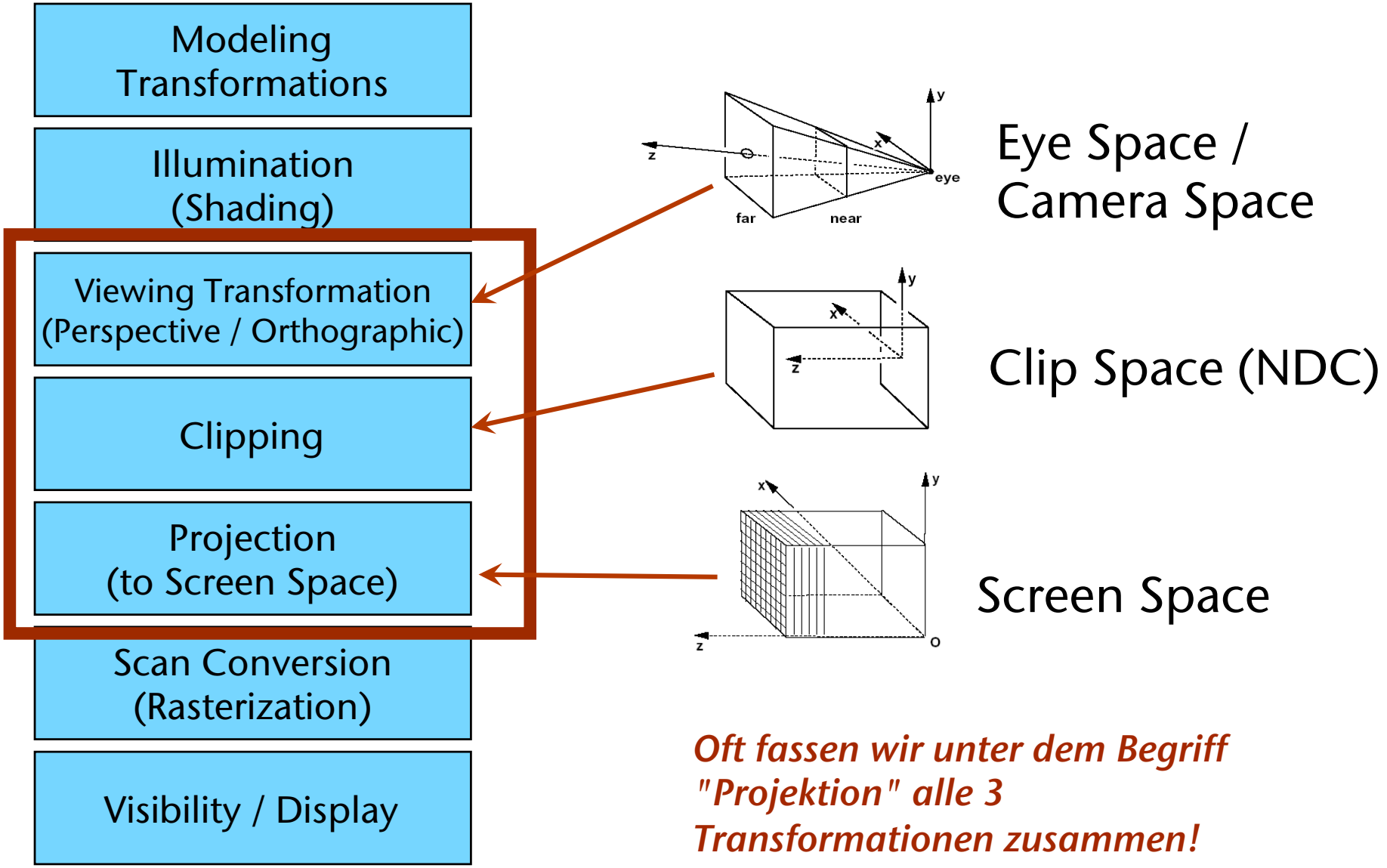
Eine Satire über Perspektive

"Satire on
False Perspective"
by William Hogarth,
1753

Bildunterschrift:

Whoever makes a DESIGN without the Knowledge of *PERSPECTIVE* will be liable to such Absurdities as are shewn in this **Frontispiece**.

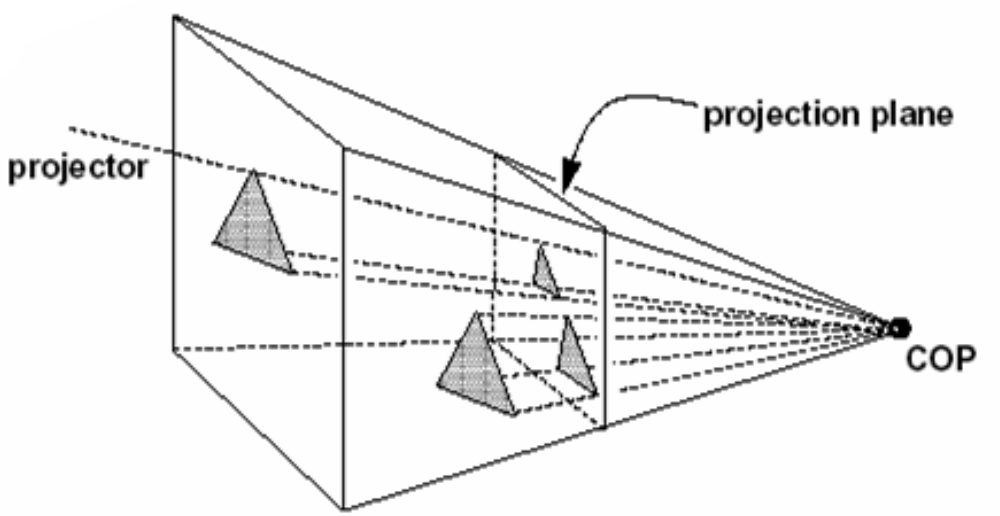
Projektionen in der Pipeline



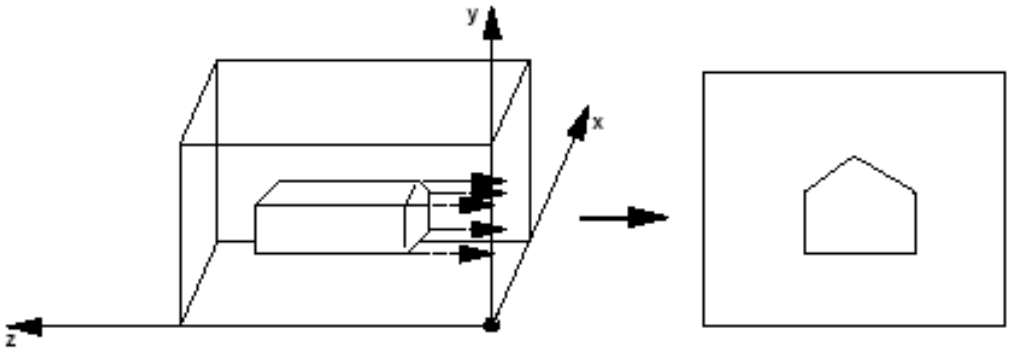
Oft fassen wir unter dem Begriff "Projektion" alle 3 Transformationen zusammen!

Orthographische vs. Perspektivische Projektion

- Perspektivisch = alle Projektionsstrahlen laufen im Projektionszentrum (COP) zusammen

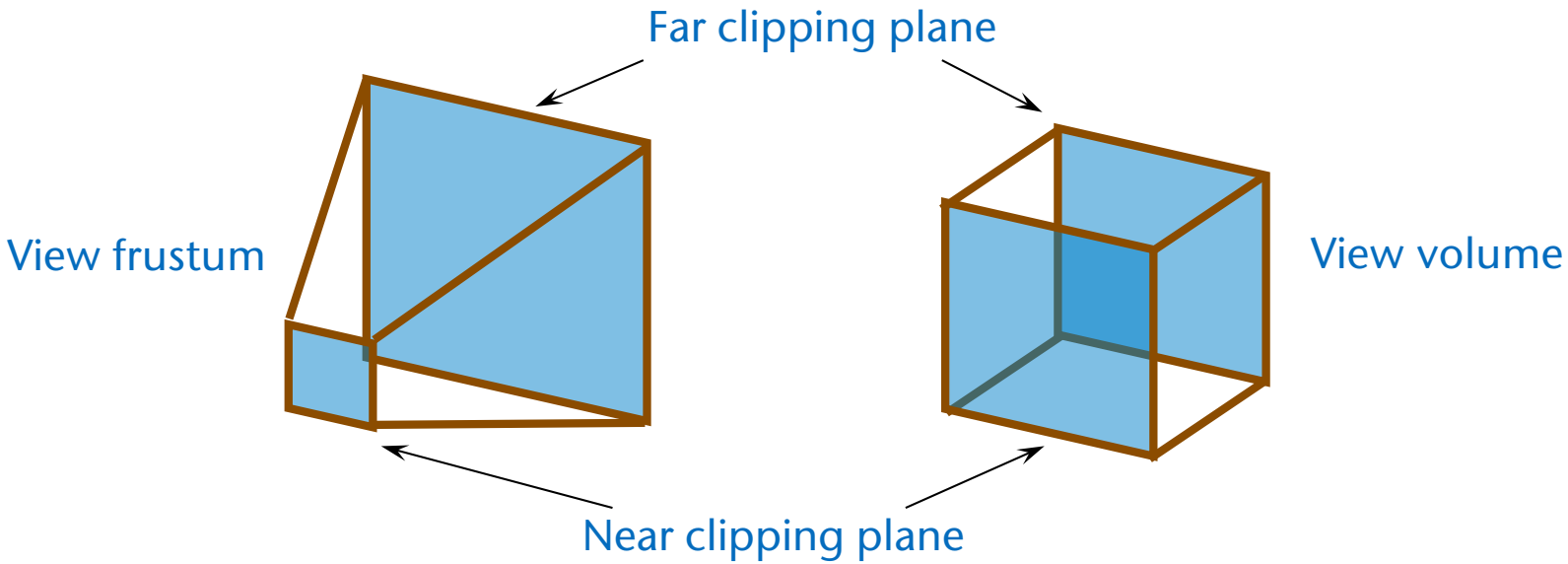


- Orthographisch = parallele Projektionsstrahlen
  - Kann man als Spezialfall der perspektivischen Projektion betrachten



Terminologie

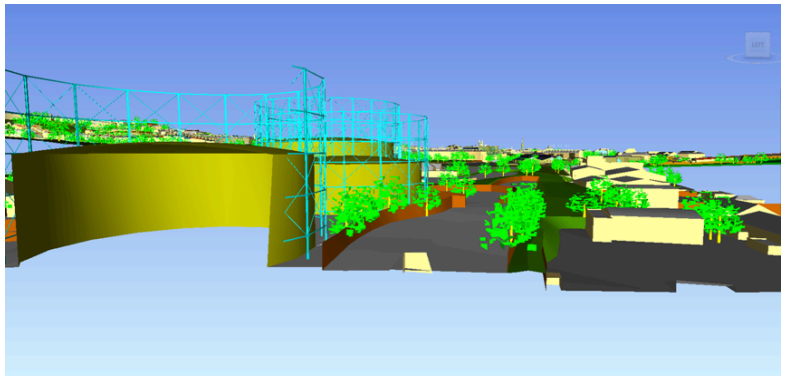
- Der Bereich des 3D-Raumes, der auf den Bildschirm projiziert wird, heißt View Volume, oder Viewing Volume
- Bei perspektivischer Projektion heißt er auch View Frustum
  - Lat. "frustum" = (abgebrochener) Brocken



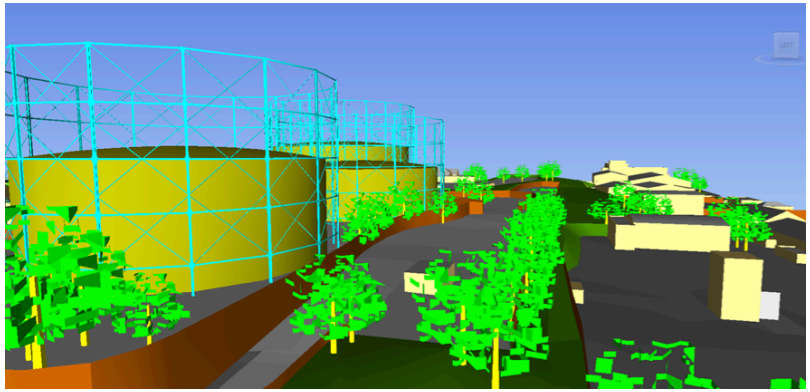
Die richtige Wahl der Near- und Far-Plane

Szene von oben

Near-Plane zu weit weg

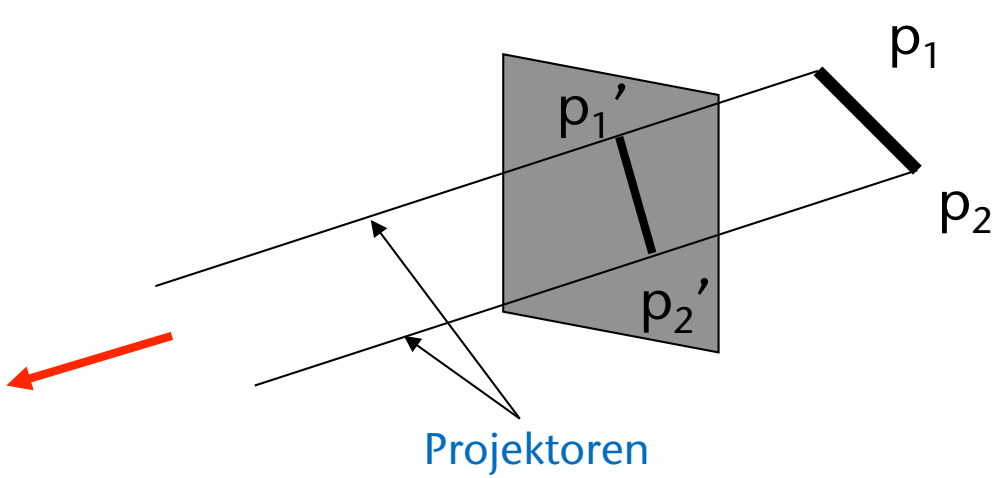


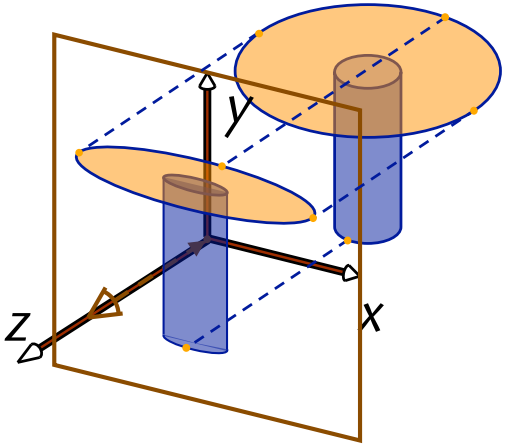
Far-Plane zu nah



Orthographische Projektion

- Punkte werden orthogonal auf die Projektionsebene (*viewing plane*) projiziert
  - Projektionslinien verlaufen senkrecht zur Projektionsebene
- Eigenschaften:
  - Parallele Linien bleiben parallel
  - Winkelverhältnisse bleiben erhalten, aufgrund der parallelen Verschiebung zu Projektionsebene
- Es gibt noch andere (schiefe) Parallel-Projektionen

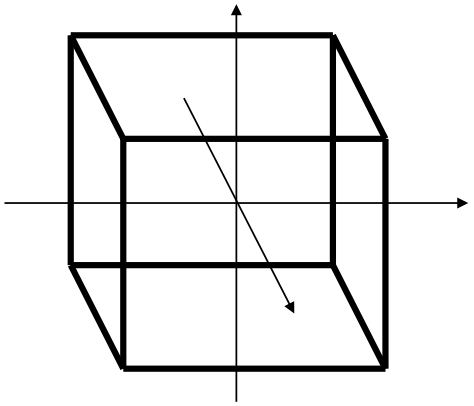




Die Projektionsmatrix

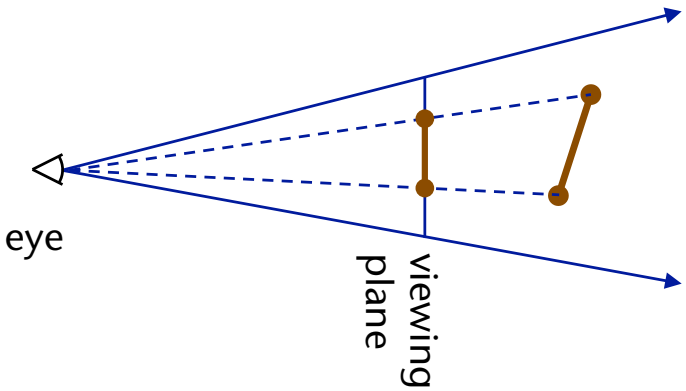
- Annahme: die komplette "virtuelle Welt" befindet sich im (kanonischen) Viewing-Volume (-1, 1, -1, 1, -1, 1)
- Die x- und y-Komponente bleiben unverändert
- Projektionsmatrix setzt z-Komponente auf 0:

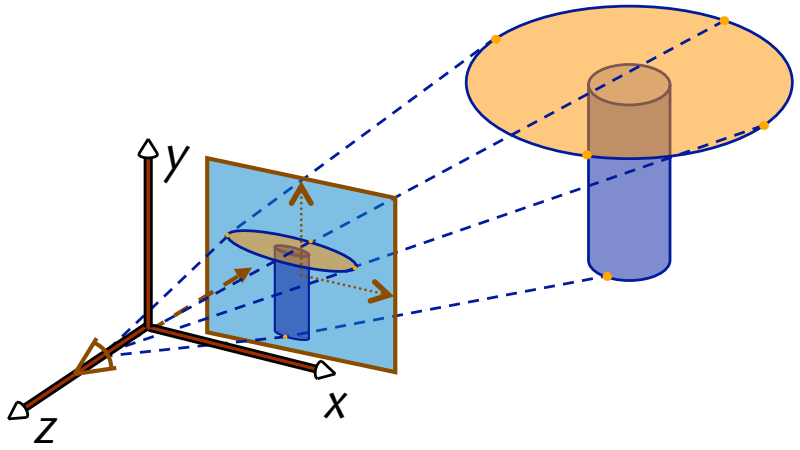
$$P_{\text{ortho}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



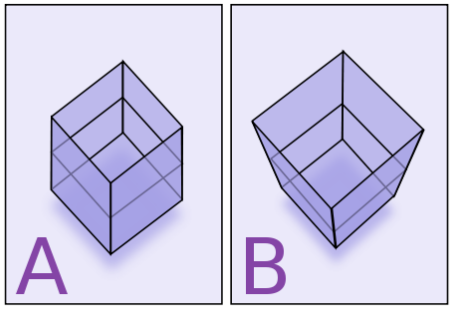
Perspektivische Projektion

- Wird am häufigsten verwendet in der Computergraphik & Malerei
- Unser Auge führt eine Zentralperspektive durch ("Lochkamera")
- Punkte werden entlang einer Gerade zum Zentrum der Projektion (COP; z.B. Mittelpunkt der Augenlinse) auf die Bild-Ebene (*viewing plane*) projiziert

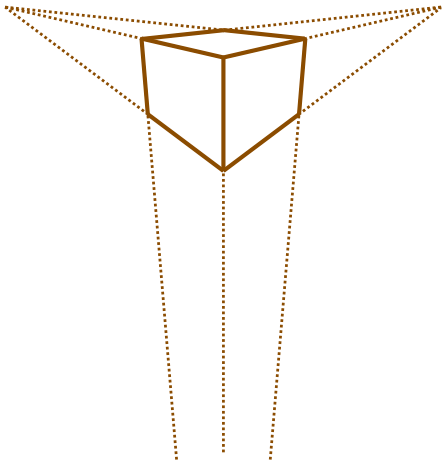


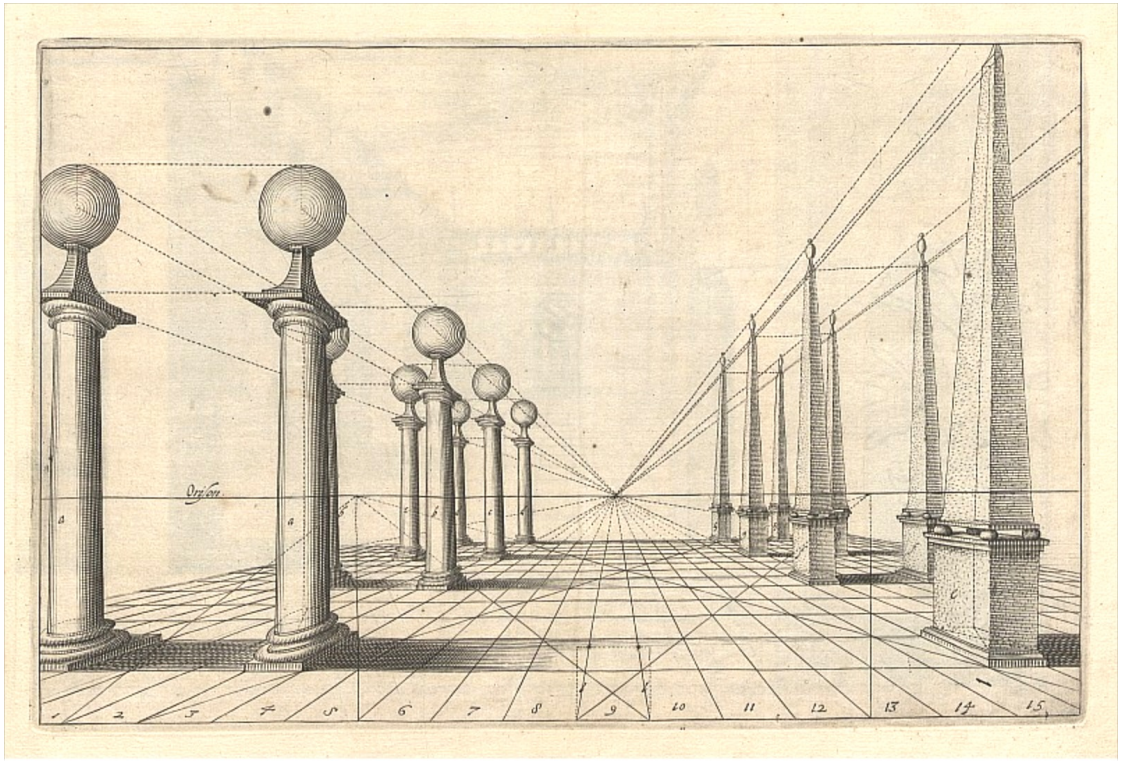


Eigenschaften



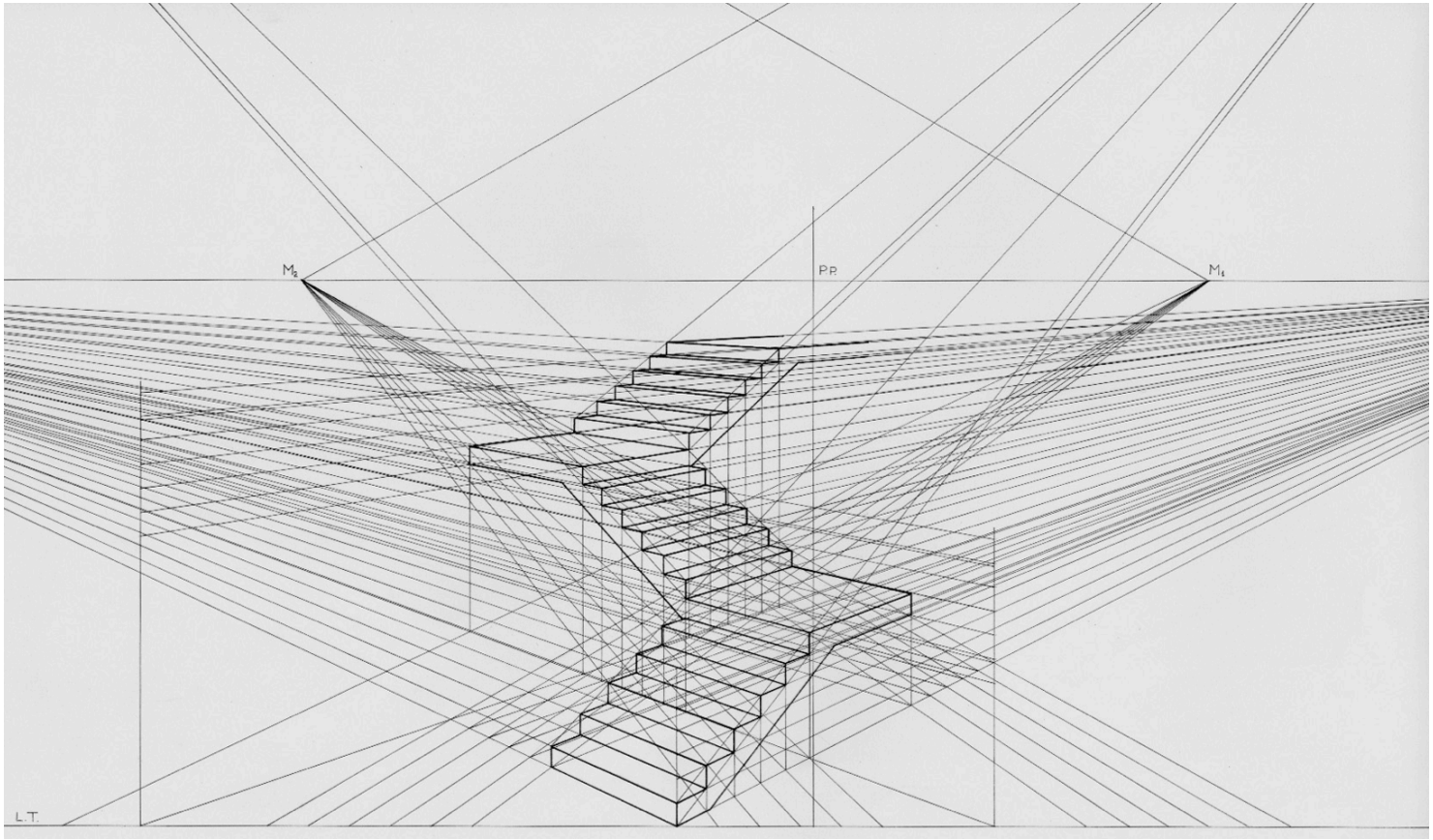
- Entfernte Objekte sind kleiner (perspektivische Verzerrung)
- Parallele Linien werden nicht auf parallele Linien abgebildet, sondern laufen scheinbar in einem gemeinsamen Punkt zusammen
  - Solch ein Punkt heißt Fluchtpunkt





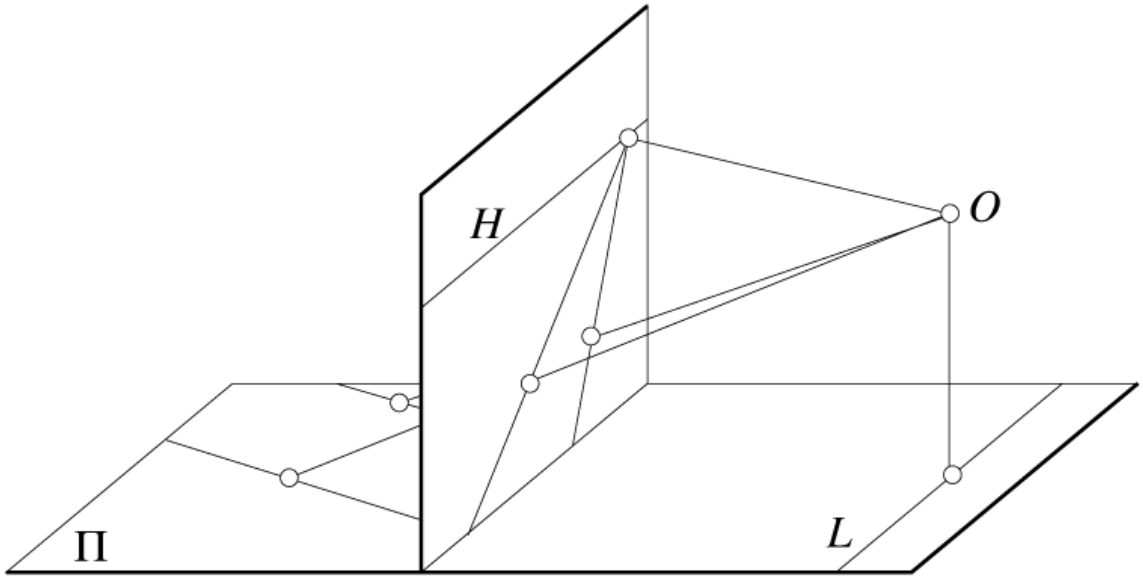
Quelle: Deutsche Fotothek

- Alle Bündel von parallelen Linien, die horizontal sind, haben einen Fluchtpunkt, der auf der Horizontlinie im Bild liegt
- Zu jeder Ebene im 3D gehört eine solche "Fluchtpunktlinie" im Bild

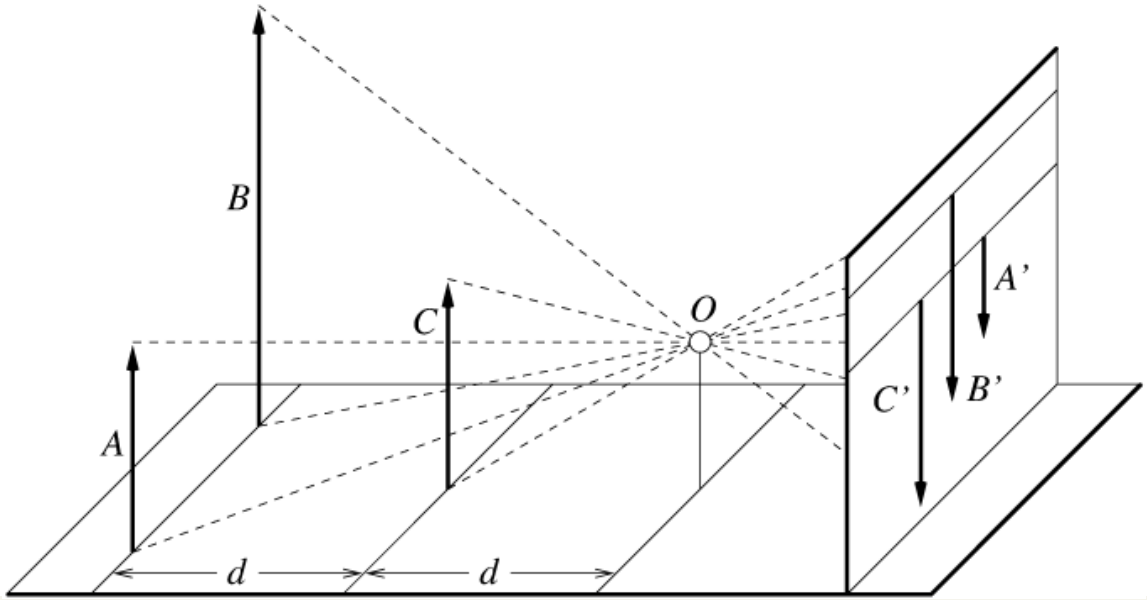


Veranschaulichung der Eigenschaften der perspektivischen Projektion

- Parallele Linien bleiben *nicht* parallel



- Längen hängen von der Tiefe ab

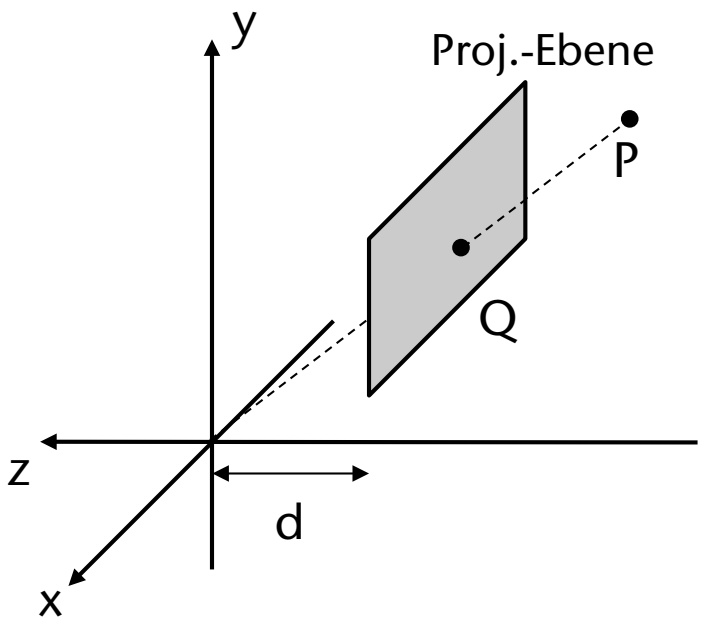


Die Projektionsmatrix

- Ann.: Kamera im Ursprung, schaut in Richtung negative z-Achse
- Projektion auf eine Ebene z = -d, d > 0

$$Q = \begin{pmatrix} -p_x \frac{d}{p_z} \\ -p_y \frac{d}{p_z} \\ -d \end{pmatrix} \cong \begin{pmatrix} -p_x \frac{d}{p_z} \\ -p_y \frac{d}{p_z} \\ -d \\ 1 \end{pmatrix} \cong \begin{pmatrix} p_x \\ p_y \\ p_z \\ -\frac{p_z}{d} \end{pmatrix}$$

$$Q = P_{\text{persp}} \cdot P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$



- Mit anschließender Projektion auf Ebene z = 0

$$P'_{\text{persp}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 0 \end{pmatrix}$$

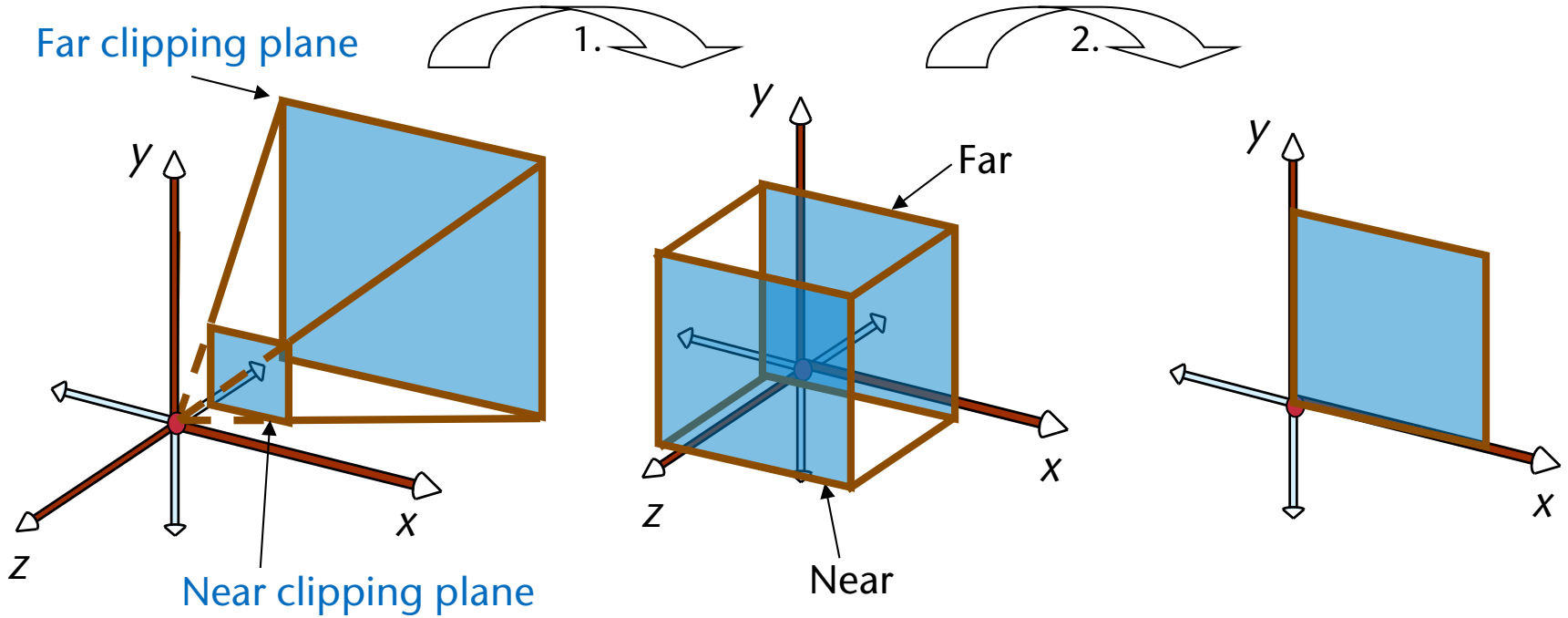
- Beachte:
  - Wenn d $\rightarrow \infty$, dann entspricht P_{persp} der orthographischen Projektion
  - Wenn d $\rightarrow 0$, dann wird

$$Q = \begin{pmatrix} p_x \\ p_y \\ p_z \\ -\frac{p_z}{d} \end{pmatrix}$$

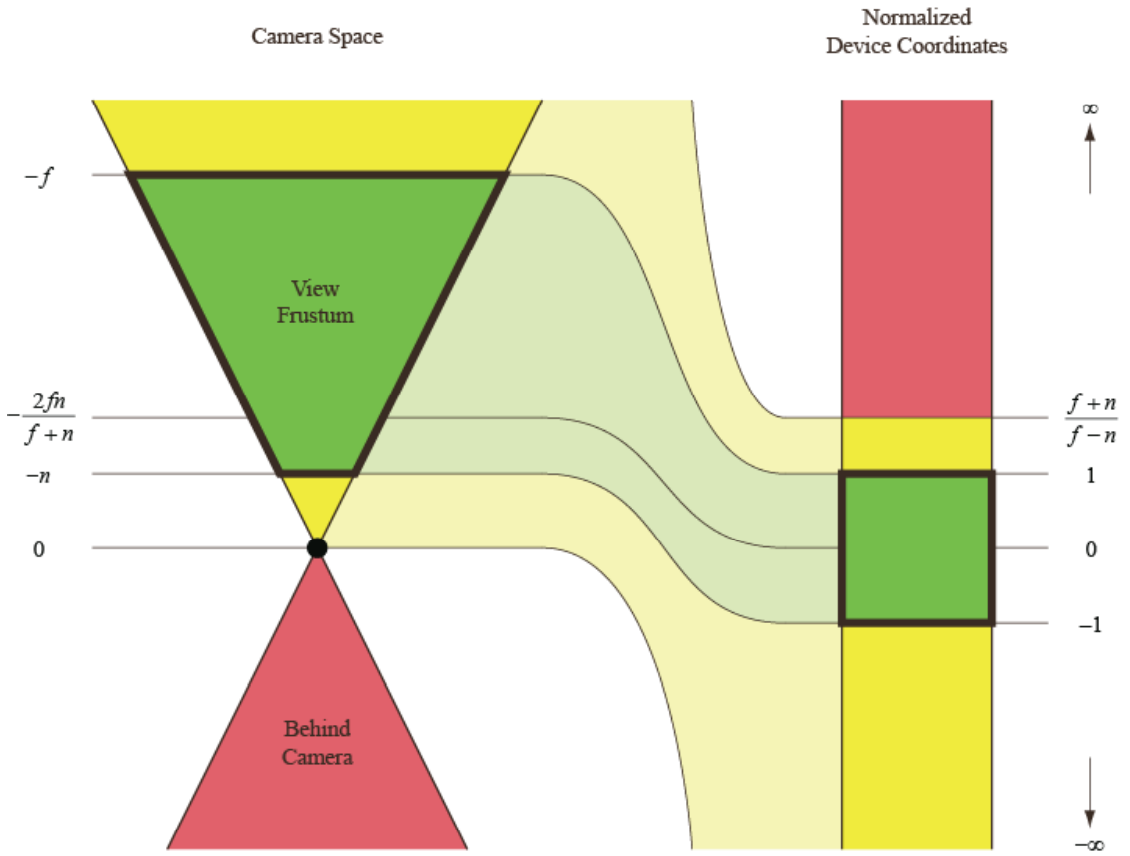
instabil

Perspektivische Projektion in 2 Schritten

- Wegen Clipping kann man die perspektivische Projektion in 2 Schritten machen:
  1. Perspektivische Abbildung (nicht Projektion!)
  2. Projektion auf Ebene (jetzt orthographisch)

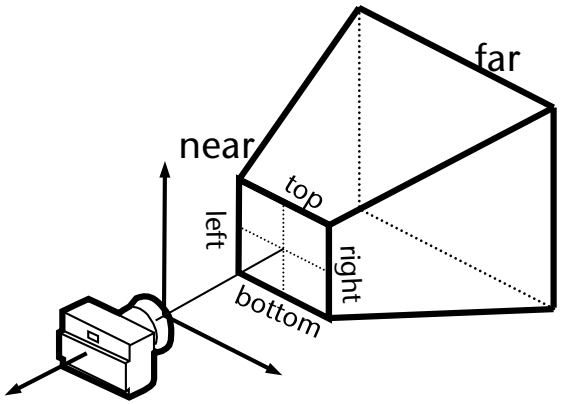


- Klassischerweise werden z-Werte im Kamera-Koordinatensystem auf einen Standard-z-Wertebereich in NDC abgebildet, der dann sehr einfach auf Integer-Z-Werte abgebildet werden kann:

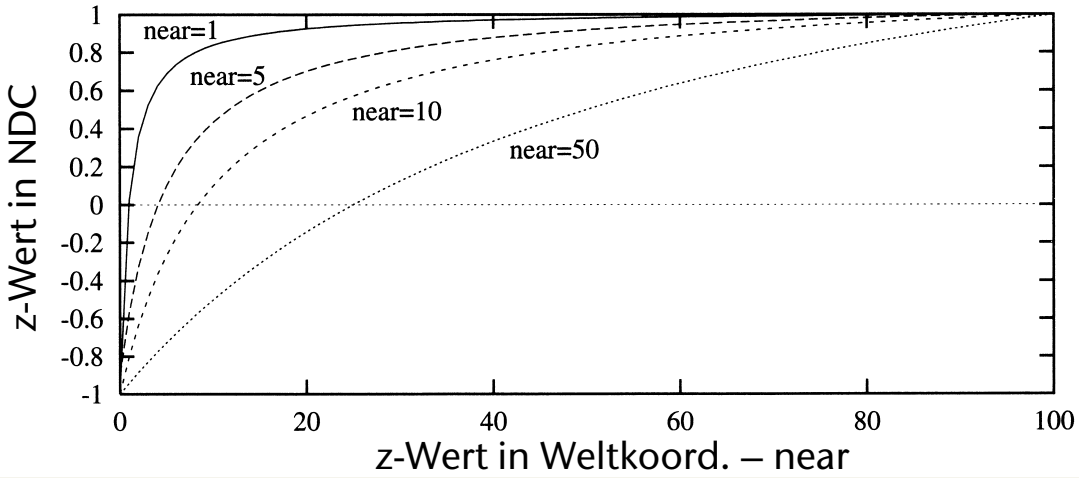


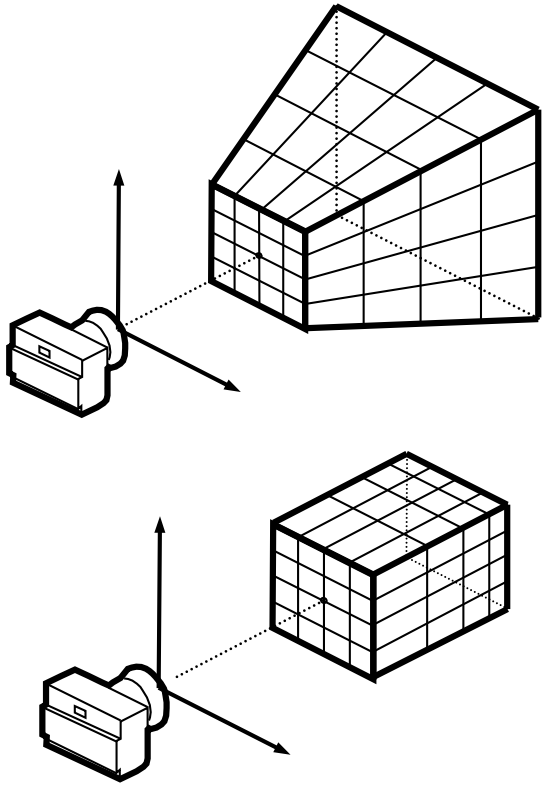
- Die Matrix für Schritt 1 (o.Bew.):

$$P_1 = \begin{pmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



- Achtung: der z-Wert in NDC hängt **nicht linear** vom z-Wert in Weltkoordinaten ab!





Eigenschaften von projektiven Transformationen

- Die allgemeine Matrix:

$$B = \begin{pmatrix} A & b \\ q^T & w \end{pmatrix}, \quad A \in \mathbb{R}^{3\times 3}, \quad b, q \in \mathbb{R}^3, \quad w \in \mathbb{R}$$

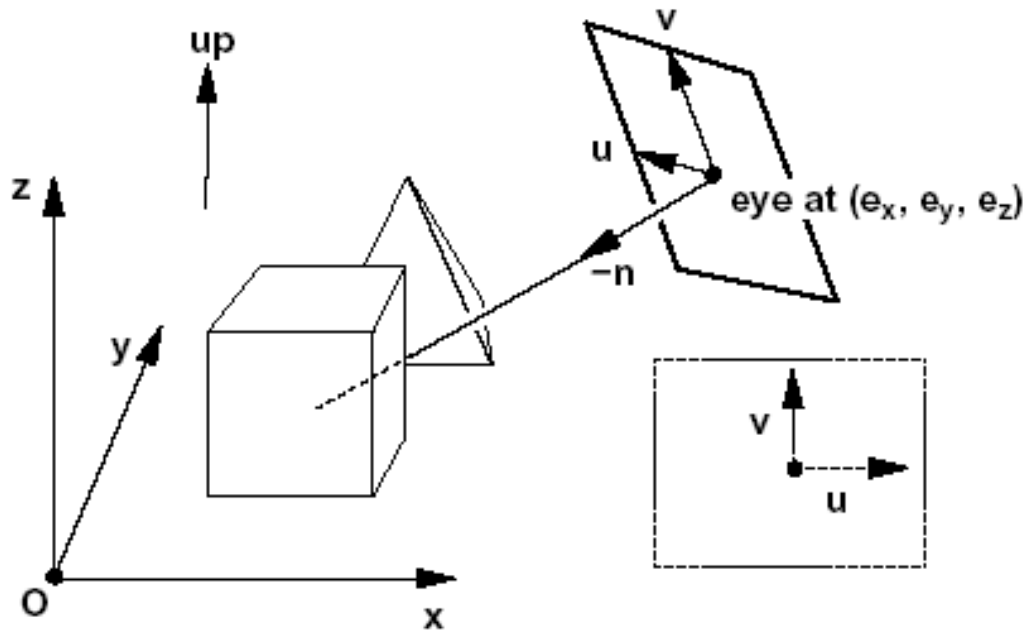
  - Entspricht für $y = \begin{pmatrix} x \\ 1 \end{pmatrix}$, $x \in \mathbb{R}^3$ der Abbildung

$$\psi : x \mapsto \frac{Ax + b}{q^T x + w}$$

- B und λB beschreiben dieselbe Abbildung ($\lambda \neq 0$)
- Bildet Geraden auf Geraden ab
- Erhält i. A. ***weder*** Parallelität ***noch*** Teilungsverhältnisse
- Erhält aber Doppelverhältnisse!

Transformation von World Space → Eye Space

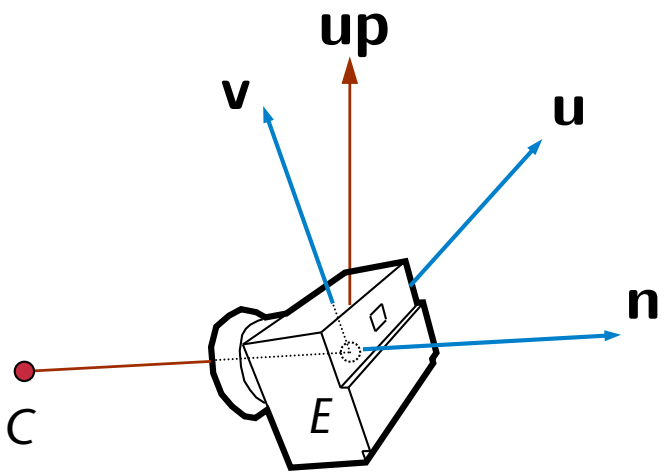
- Heißt Viewing Transformation oder Camera Transformation
- Parameter zum Positionieren der Kamera:
  - Augpunkt **E** = (e_x, e_y, e_z)
  - "Up"-Vektor in Weltkoordinaten: dieser Vektor soll senkrecht auf dem Bildschirm erscheinen, also parallel zu **v**
  - Punkt C in Weltkoordinaten, der in der Mitte des Bildes erscheinen soll (heißt auch *Look-At*)
- Aufgabe: daraus das Kamerakoordinaten-system **u,v,n** berechnen (*eye space*)



$$\mathbf{n} = \frac{E - C}{\|E - C\|}$$

$$\mathbf{u} = \frac{\mathbf{up} \times \mathbf{n}}{|\mathbf{up} \times \mathbf{n}|}$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u}$$

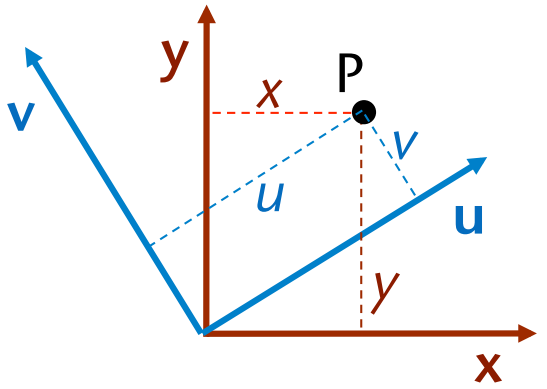


- Transformation von Weltkoord. in Kamerakoord. = Translation + Wechsel der Orthonormalbasis
  - Gegeben: Koord.achsen **x,y,z** & **u,v,n** und der Punkt P = (*x*,*y*,*z*)
  - Bestimme P in **u,v,n**-Koord., also P' = (*u*,*v*,*n*)

- Wechsel der Orthonormalbasis:

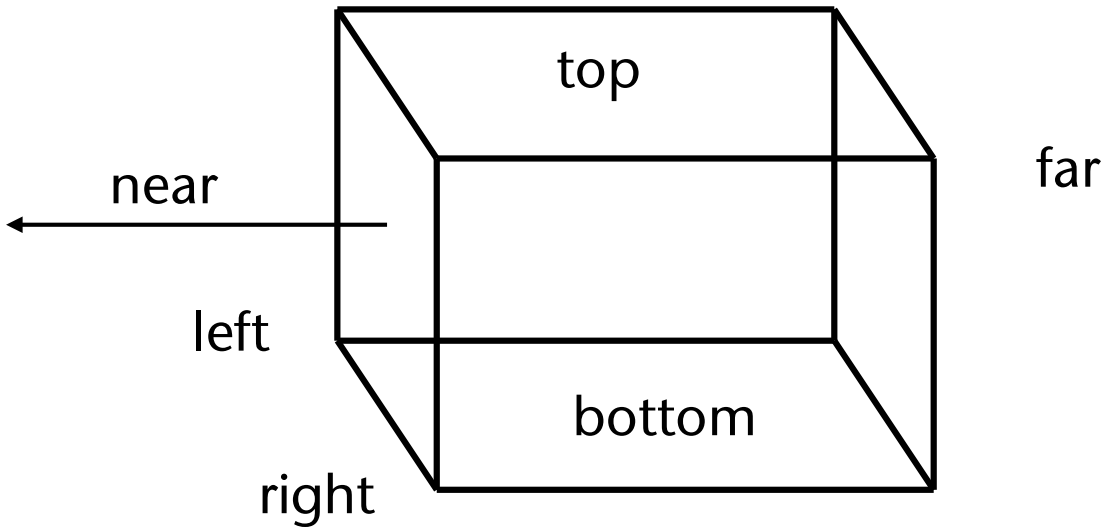
$$\begin{pmatrix} u \\ v \\ n \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

mit $\mathbf{u} = (u_x, u_y, u_z)$ etc.



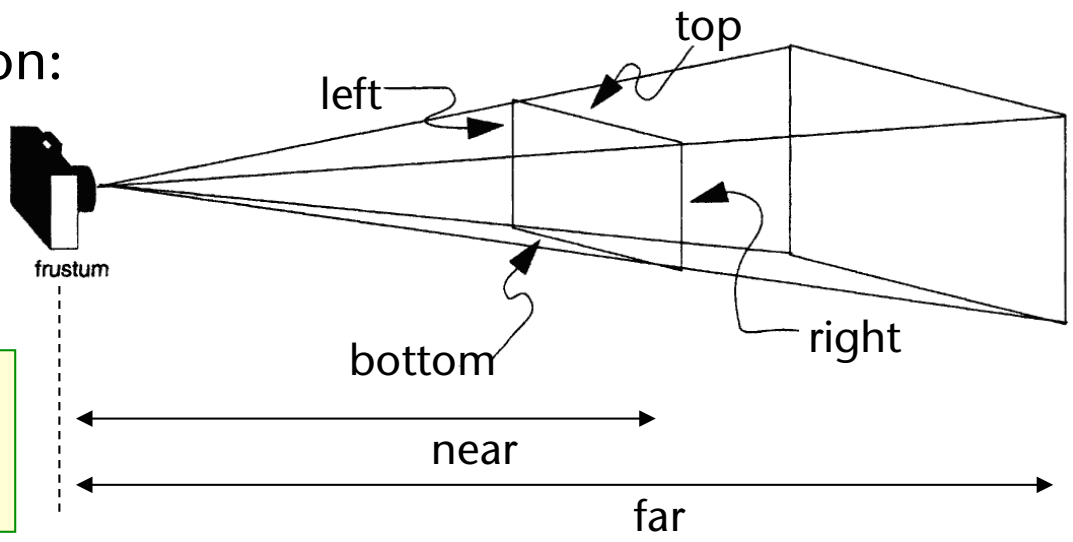
Projektion in OpenGL

- Orthographische Projektion

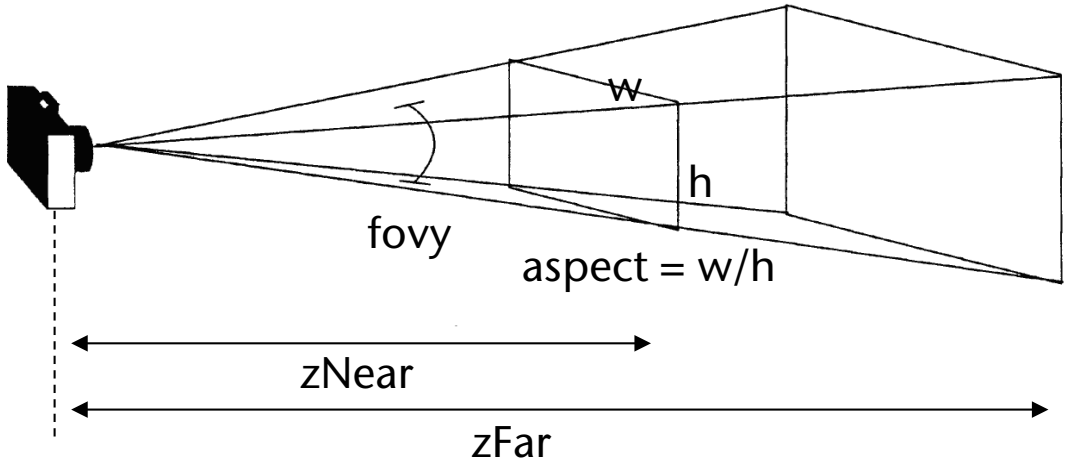



```
glOrtho(left, right, bottom, top, near, far);
```

- Perspektivische Projektion:

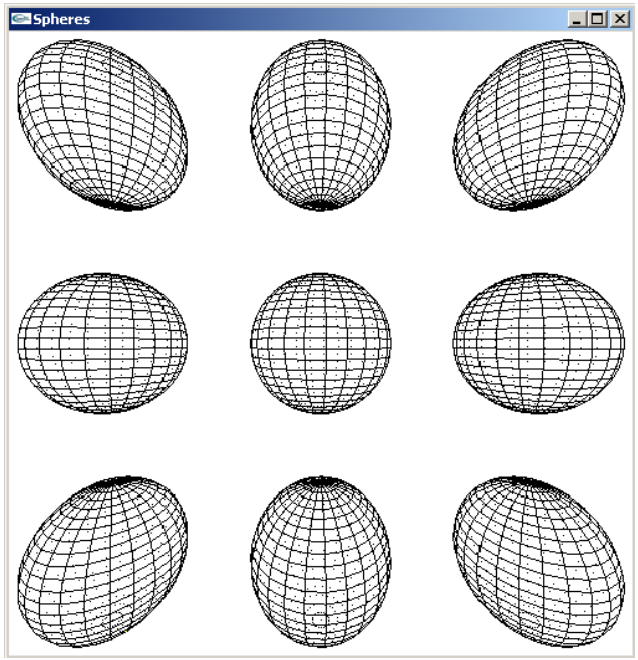



```
glFrustum( left, right,
           bottom, top,
           near, far );
```

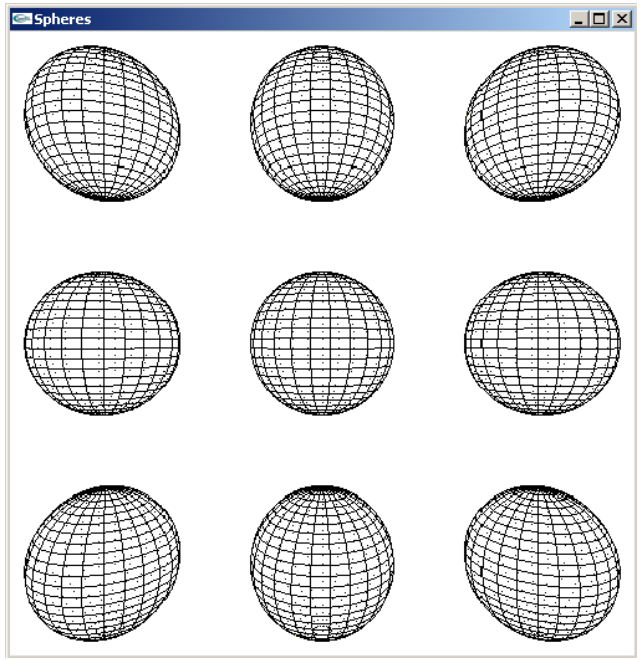



```
gluPerspective( fovy,
                aspect,
                zNear,
                zFar );
```

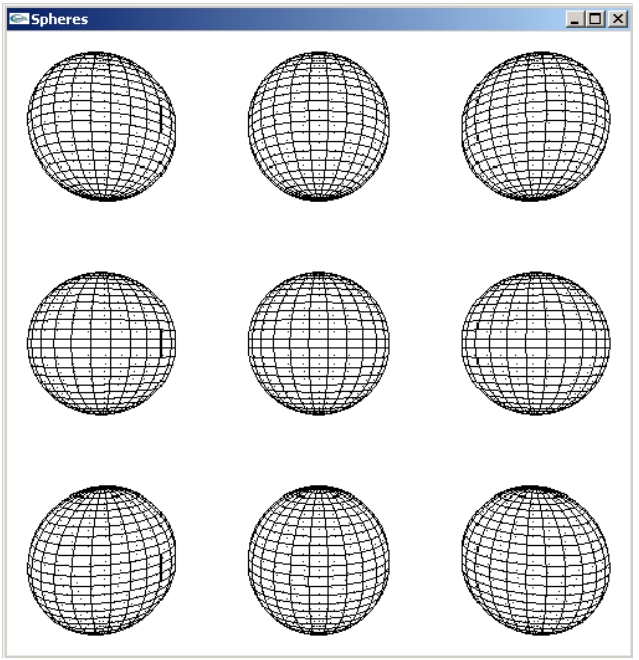
- Vorsicht bei Perspektive: Öffnungswinkel nicht zu groß wählen!



90°



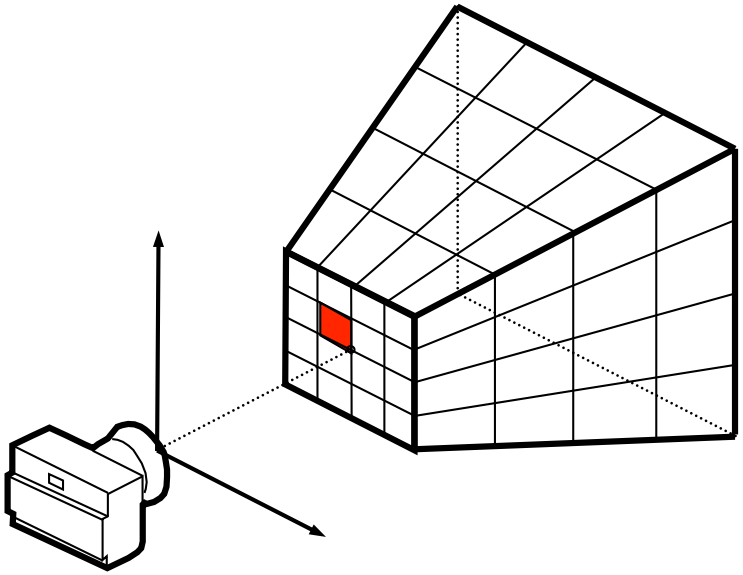
60°



45°

Asymmetrisches Frustum

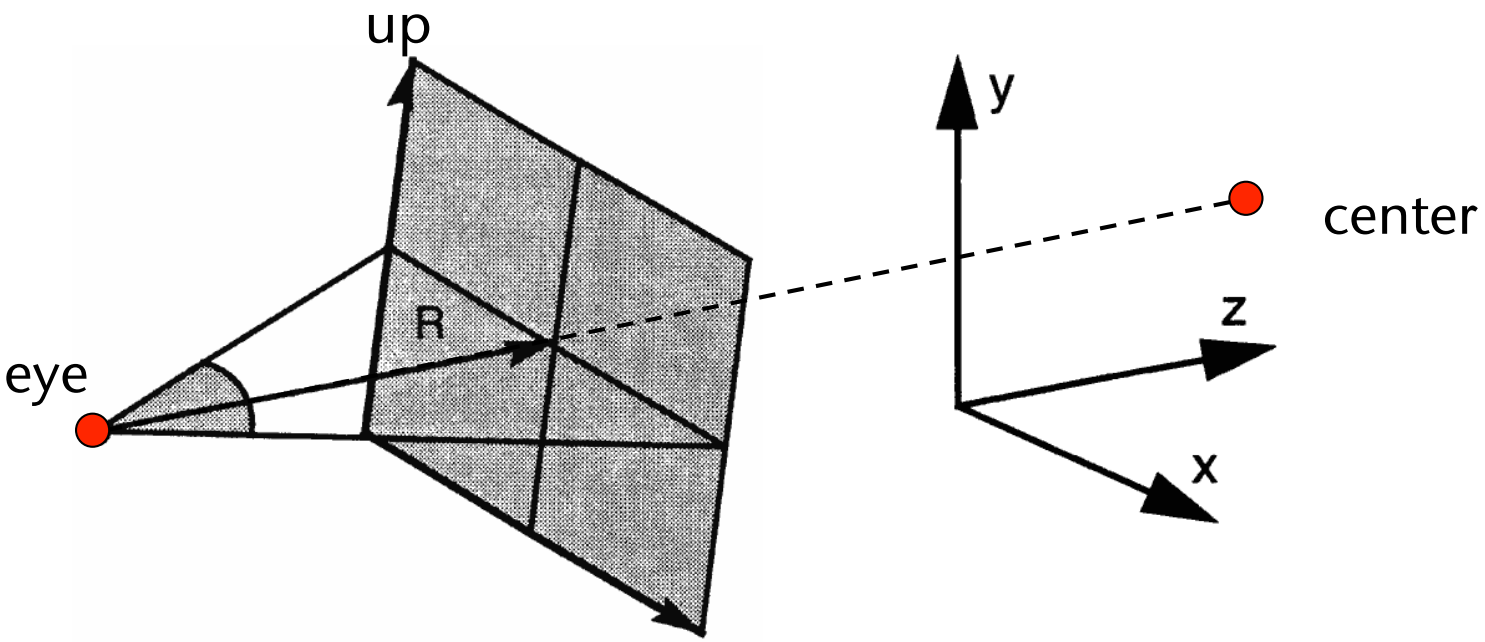
- Kommt manchmal vor, z.B.
  - Stereo-Projektion
  - Rendern eines Posters mit 10000 x 10000 Pixel (Framebuffer zu klein)
  - Mit gluPerspective() nicht möglich
- Poster: Zerlegen in viele Teilbilder mit voller Auflösung



```
glFrustum( left, right,
           bottom, top,
           near, far );
```

Hier ist left ≠ right, top ≠ bottom!

Festlegen der Viewing Transform mittels GL Utility




```
gluLookAt( eyeX, eyeY, eyeZ,
           centerX, centerY, centerZ,
           upX, upY, upZ );
```

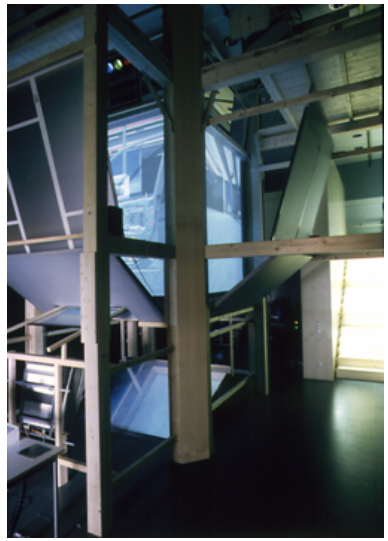
Die Projektionsmatrix in OpenGL

- Projektionen werden *auch* durch Matrizen realisiert
- Achtung: für diese existiert eine eigene "globale" Matrix!
- Achtung: alle Transformations-Operationen (**glTranslate**, ..., **glFrustum**, ...) multiplizieren immer mit der aktuell "eingeschalteten" Matrix!!
- Umschalten mittels

```
glMatrixMode( { GL_MODELVIEW, GL_PROJECTION } );
```

  - Nach dem Umschalten beziehen sich alle Matrixbefehle auf die entsprechende Matrix
- Zurückschalten auf **GL_MODELVIEW** nicht vergessen!

- Sollte man die Transformation von Welt- in Kamerakoord. in die MODELVIEW-Matrix oder in die PROJECTION-Matrix multiplizieren?
- Antwort: allg. üblich ist es, sie in die MODELVIEW-Matrix zu stecken, aber es geht genauso die PROJECTION-Matrix
  - In manchen Fällen, z.B. in einer Cave, muß man es sogar in der PROJECTION-Matrix machen



- Gesamtansicht aller Matrizen:

$$p' = \underbrace{M_{\text{proj}}}_{\text{GL\_PROJECTION}} \cdot \underbrace{V \cdot T \cdot \ldots \cdot S \cdot R}_{\text{GL\_MODELVIEW}} \cdot p$$

Zum Schluss: Transformation ins Kamerakoordinatensystem, z.B. mit gluLookAt als erstes Kommando

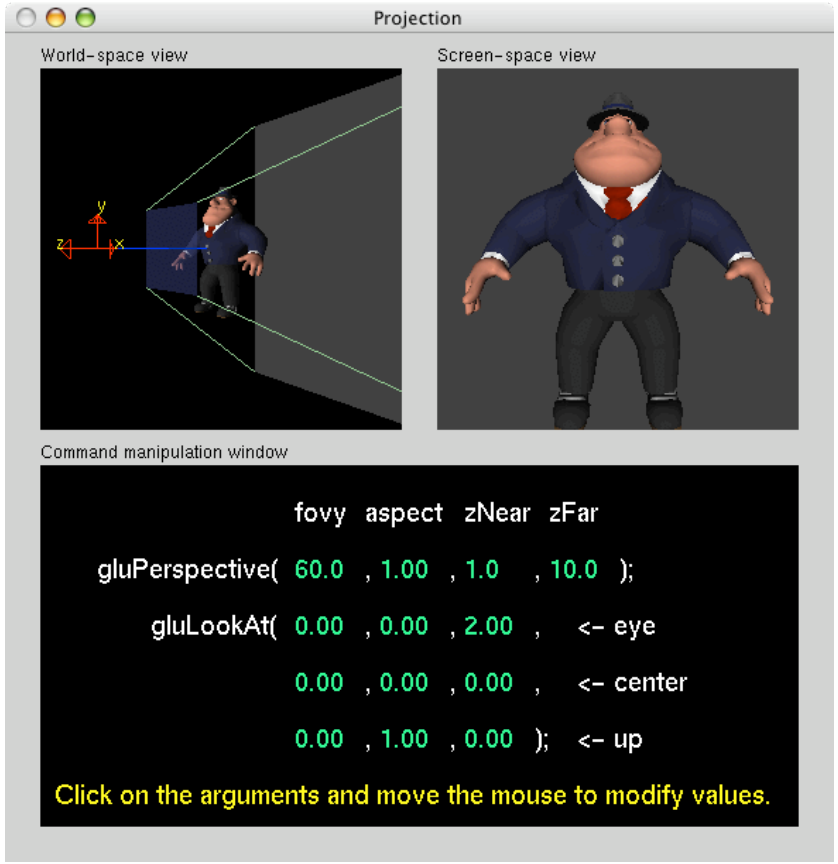
Beliebige Transformationen

OpenGL-Beispiel

- Ein typisches OpenGL-Programm sieht dann ungefähr so aus:

```
glColor3f( 0.0, 0.0, 0.0 );
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

glMatrixMode( GL_PROJECTION );     // set up projection
glLoadIdentity();

glOrtho( -1.0,1.0, -1.0,1.0, 1.0,10.0 );
oder
glFrustum( -1.0,1.0, -1.0,1.0, 1.0,10.0 );

glMatrixMode( GL_MODELVIEW );      // set up camera trf
glLoadIdentity();
gluLookAt( 0,0,1, 0,0,0, 0,1,0 );

glTranslatef( 0,0,tz );            // set up model-view
glRotatef( alpha, 1.0, 0.0, 0.0 );

render geometry …
```

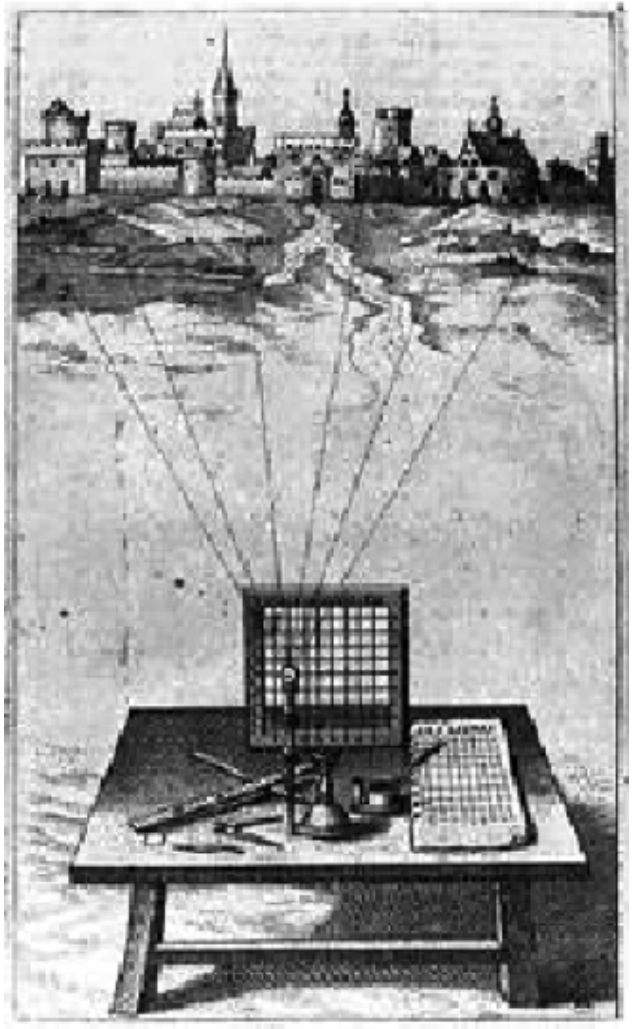
Demo



http://www.xmission.com/~nate/tutors.html

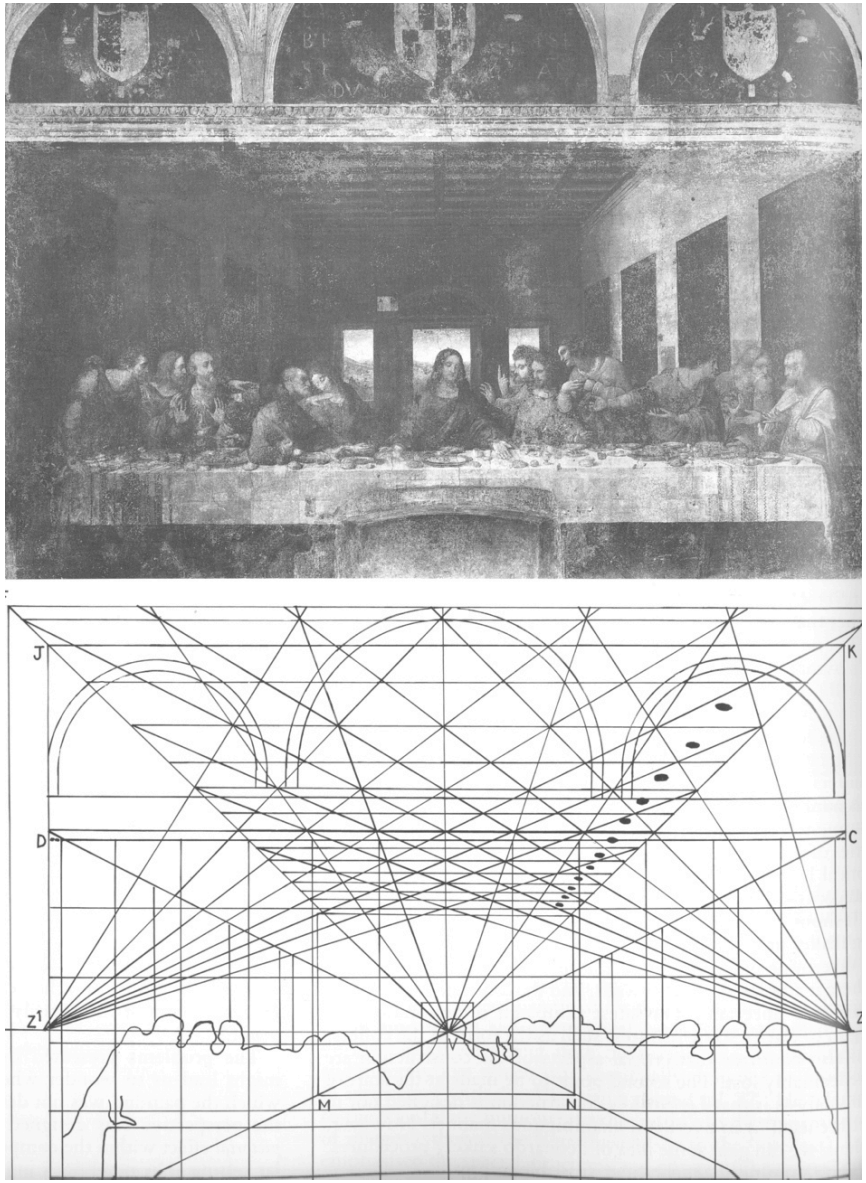
Noch einige Beispiele aus der Kunst

- Theoretisch wurde die Lösung des Problems der Perspektive von Leon Battista Alberti in seinem Buch *Della Pittura*, 1435-1436, beschrieben
- Brunelleschi löste es als erster praktisch 1410-1420



Alberti's *reticolato*

- Leonardo da Vinci sagte:
  There are some who look at the things produced by nature through glass, or other surfaces, or transparent veils. They trace outlines on the surface of the transparent medium... But such an invention is to be condemned in those who do not know how to portray things without it, how to reason about nature with their minds... They are always poor and mean in every invention and in the composition of narratives, which is the final aim of this science



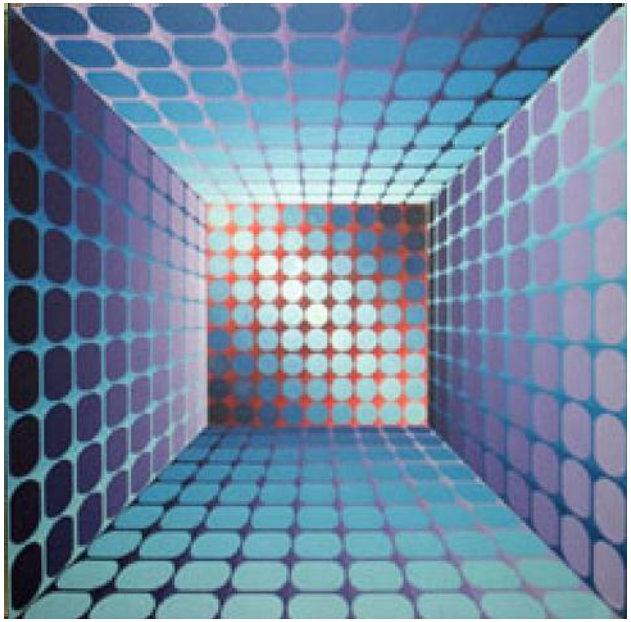
Erste gezielte Multi-Perspektive

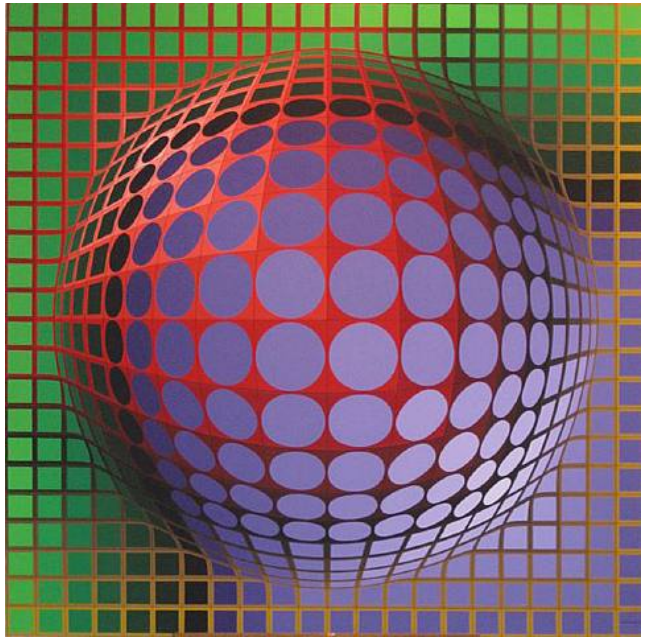
Raffael: *Die Schule von Athen*

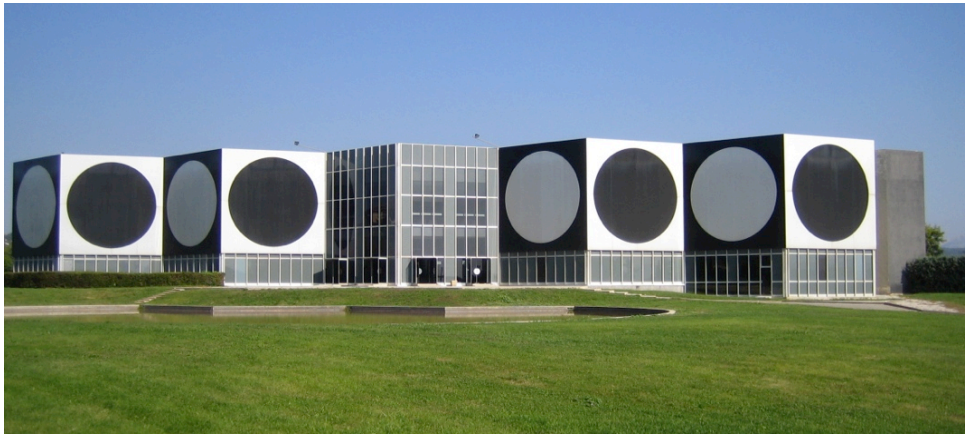
Multi-Perspektive, um den mystischen Eindruck zu erhöhen

De Chirico:

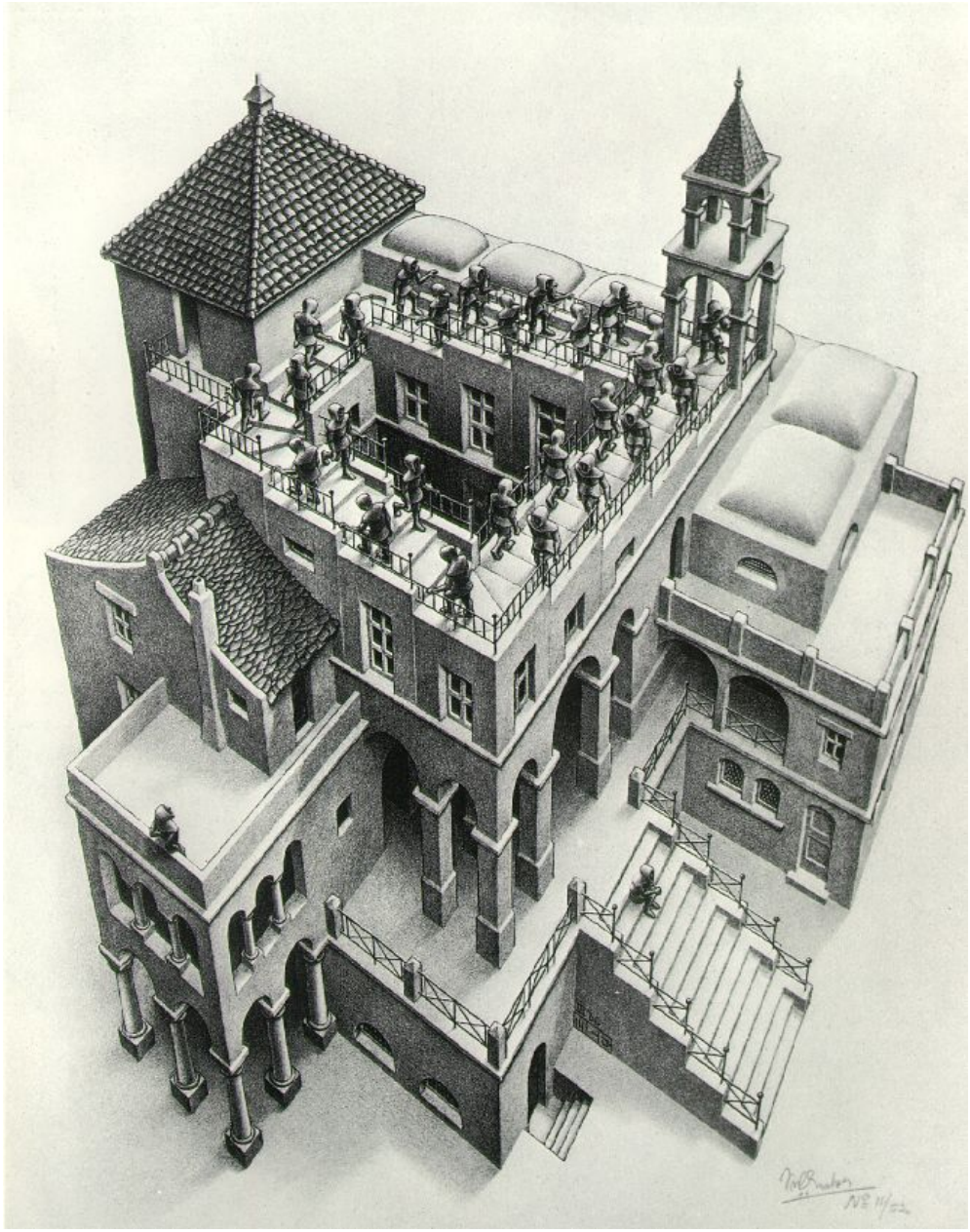
Viktor Vasarely: Perspektive in der abstrakten Kunst





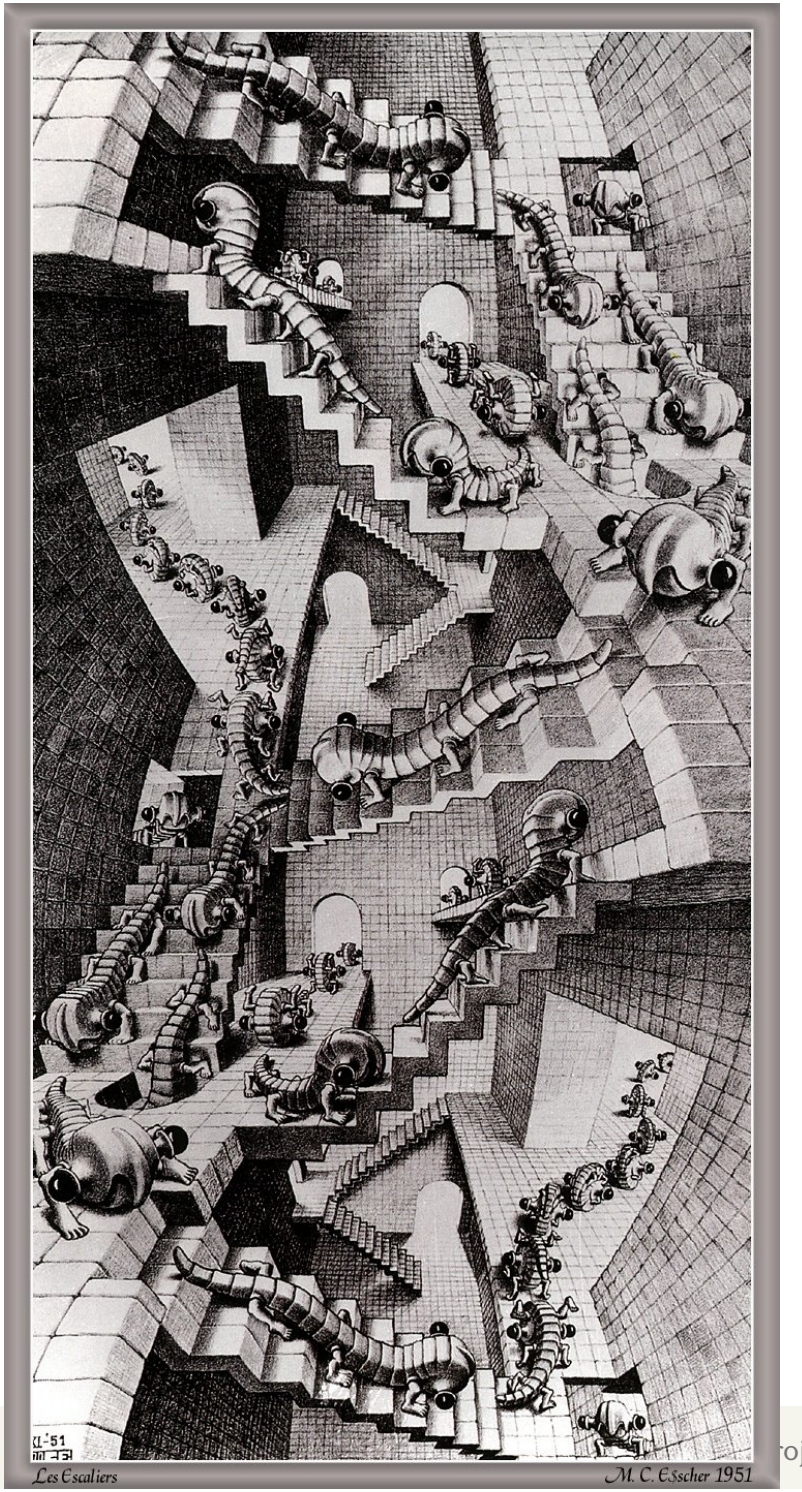


Einsatz der (korrekten) Perspektive zur Irritation des Betrachters



Maurits Cornelis ESCHER:
Ascending and Descending
1960, Lithograph

Nicht-lineare Perspektive



M. C. Escher

Bremen

calvin and hobbes

by WATTERSON