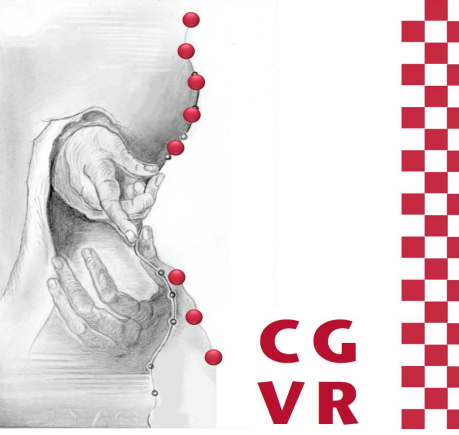
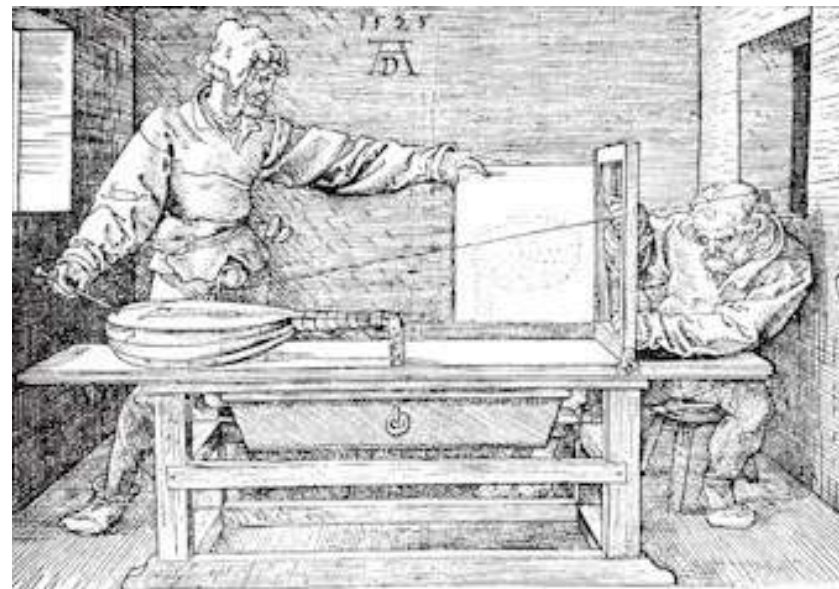


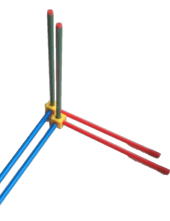


Computer-Graphik I

Projektionen, Perspektive & Viewing Transformation

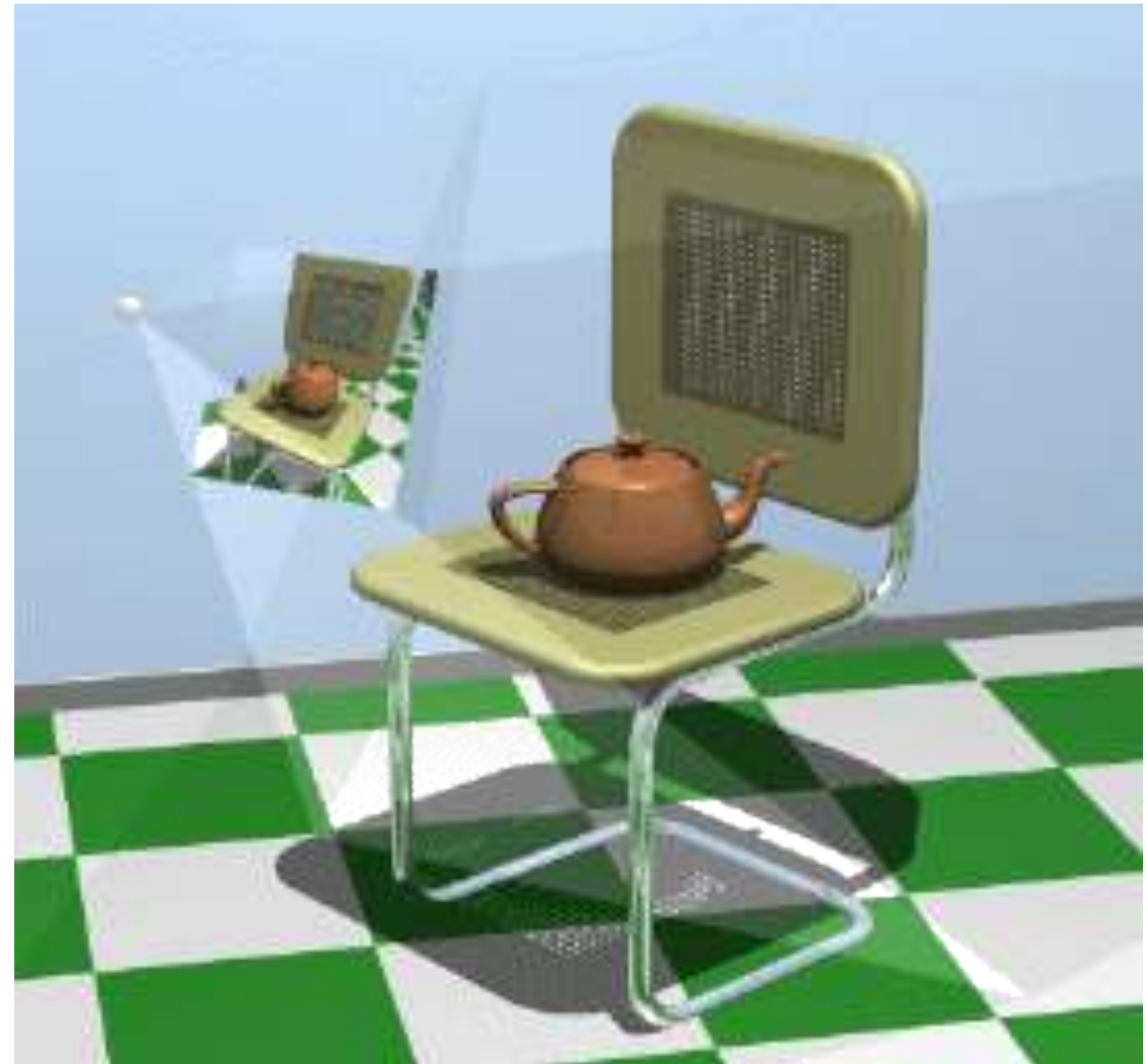


G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

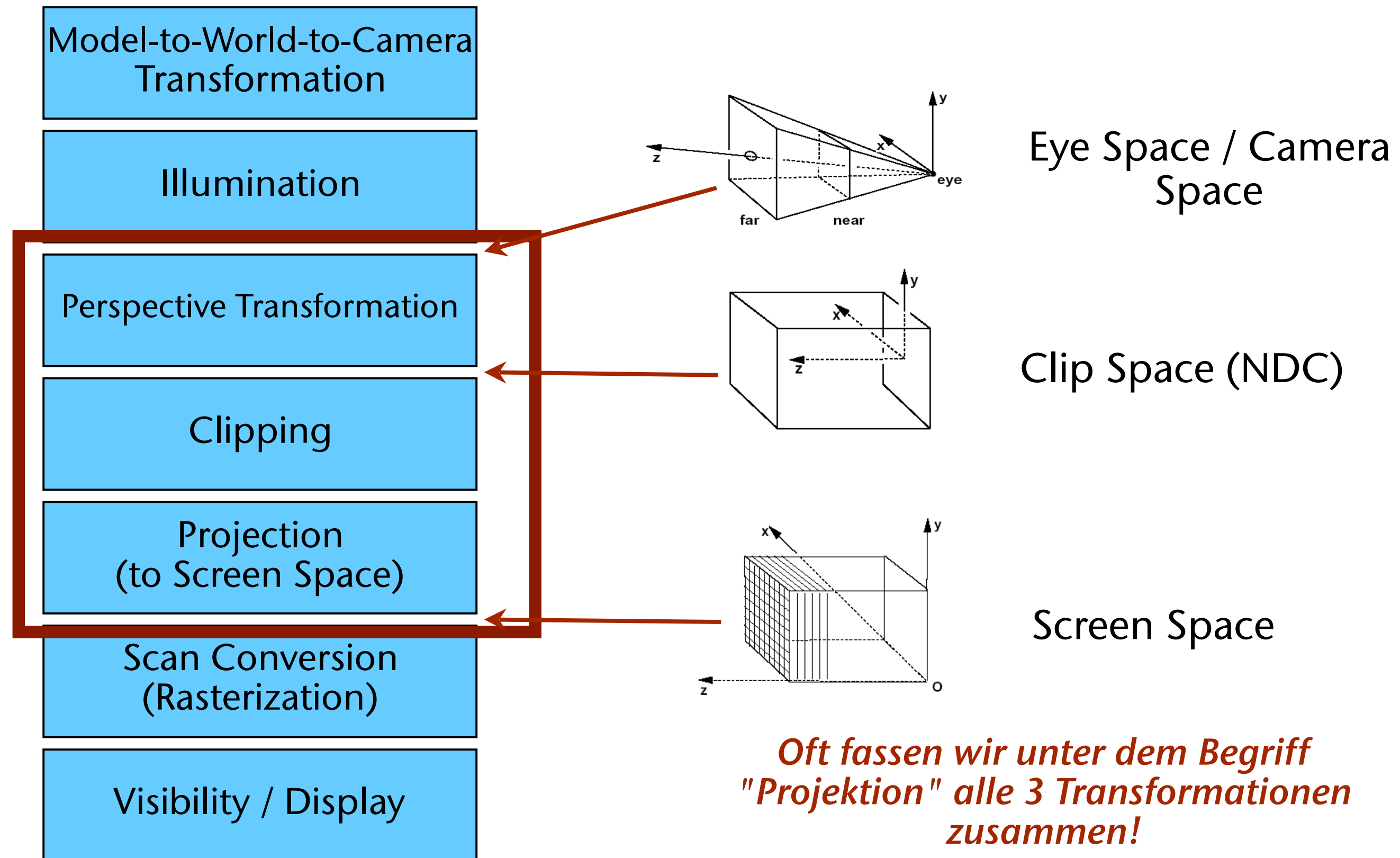


Prinzipielles Vorgehen

- Ziel: die virtuelle 3D Welt auf einem 2D Display darstellen
- 1. Transformiere die Welt (= World Space; einschließlich der gedachten Kamera) so, daß die Kamera im Ursprung landet und Richtung $-Z$ schaut → **Viewing Transformation**
- 2. Führe die Projektion (i.A. perspektivisch) auf die Bildebene durch → **Projection Transform**



Viewing & Projection Transformation in der Pipeline



Transformation von World Space $\rightarrow$ Eye Space

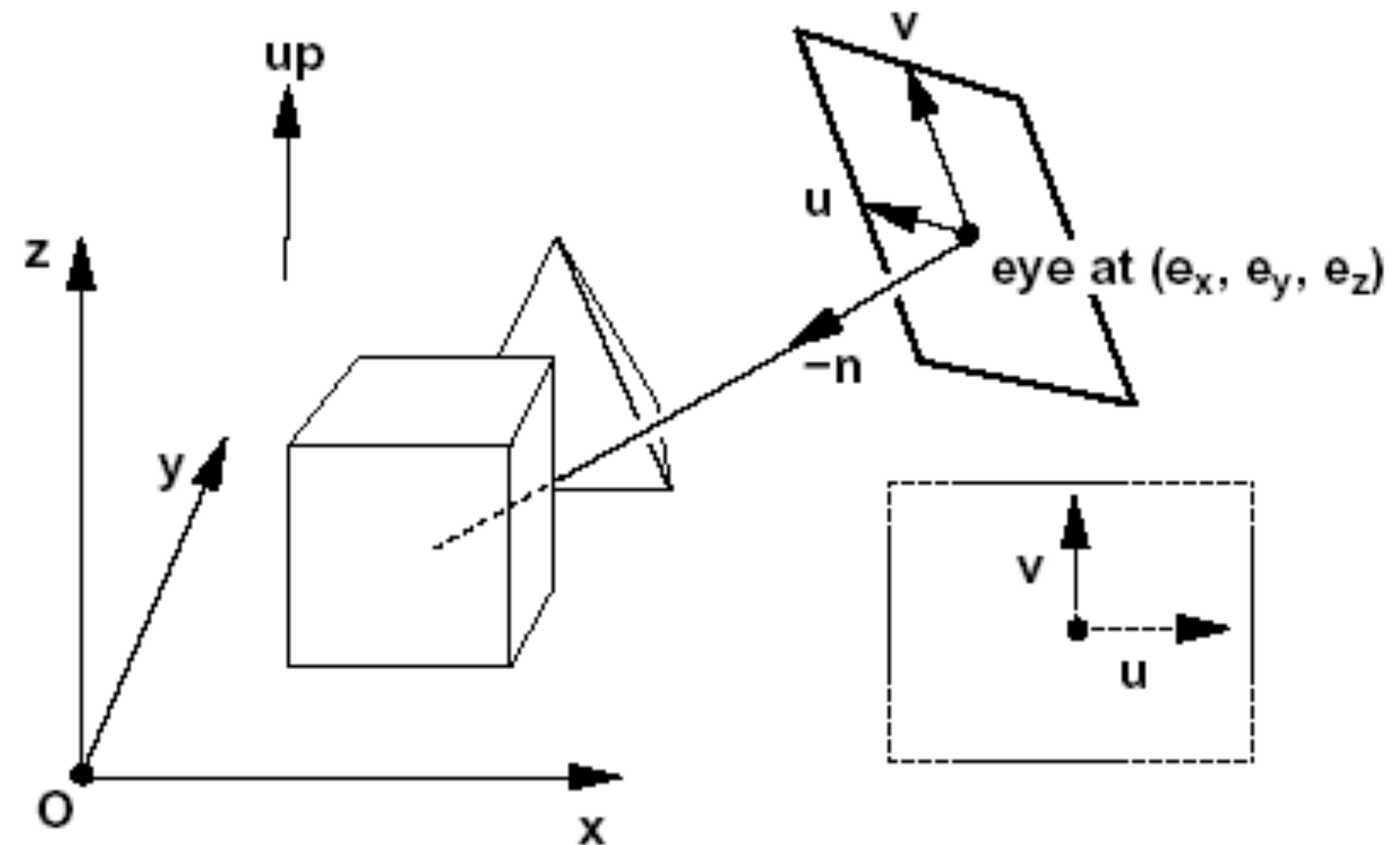
- Heißt **Viewing Transformation** oder **Camera Transformation**

- Manchmal auch kombiniert als **Modelview 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**)

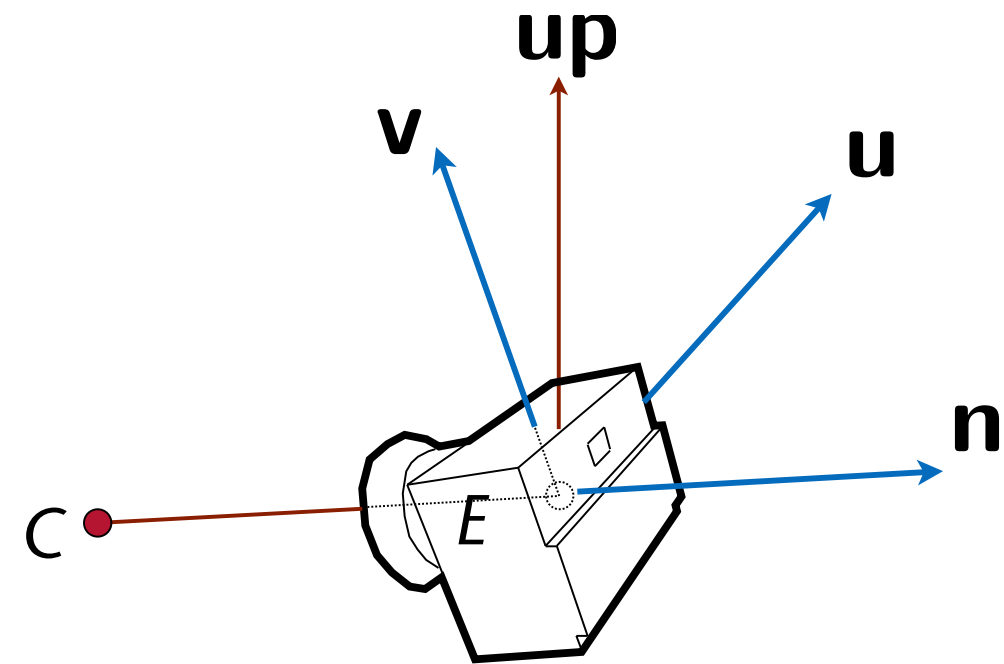
- Aufgabe: daraus das **Kamerakoordinatensystem** (u, v, n) berechnen (**camera space**)



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

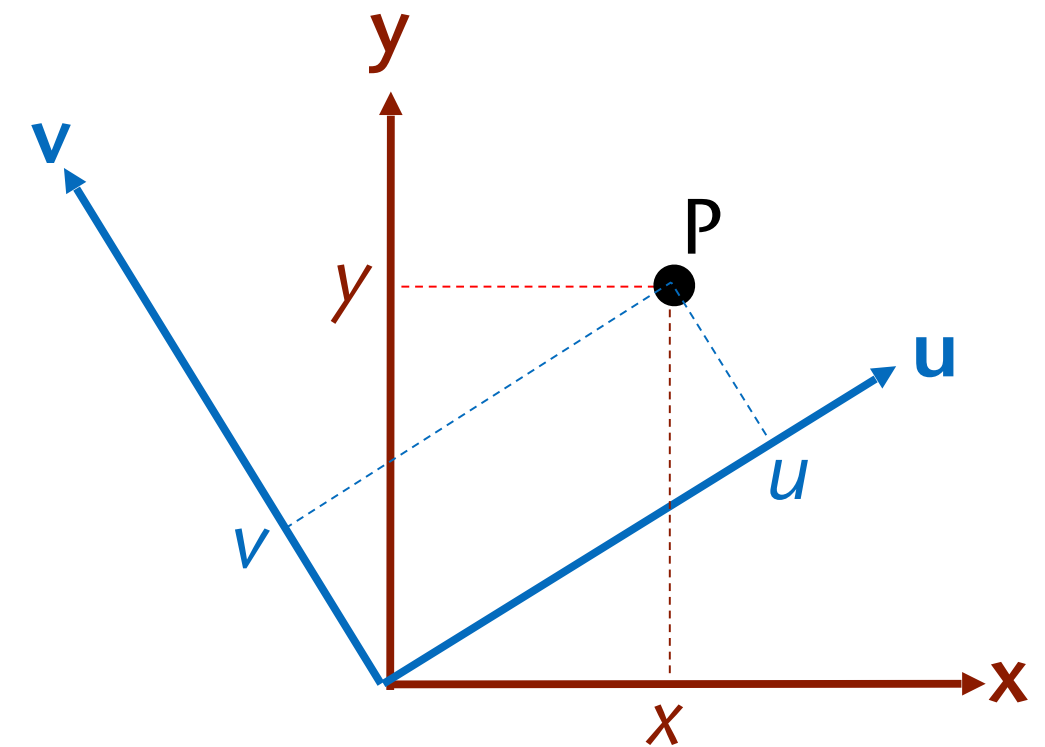
$$\mathbf{u} = \frac{\mathbf{up} \times \mathbf{n}}{|\mathbf{up} \times \mathbf{n}|}$$

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



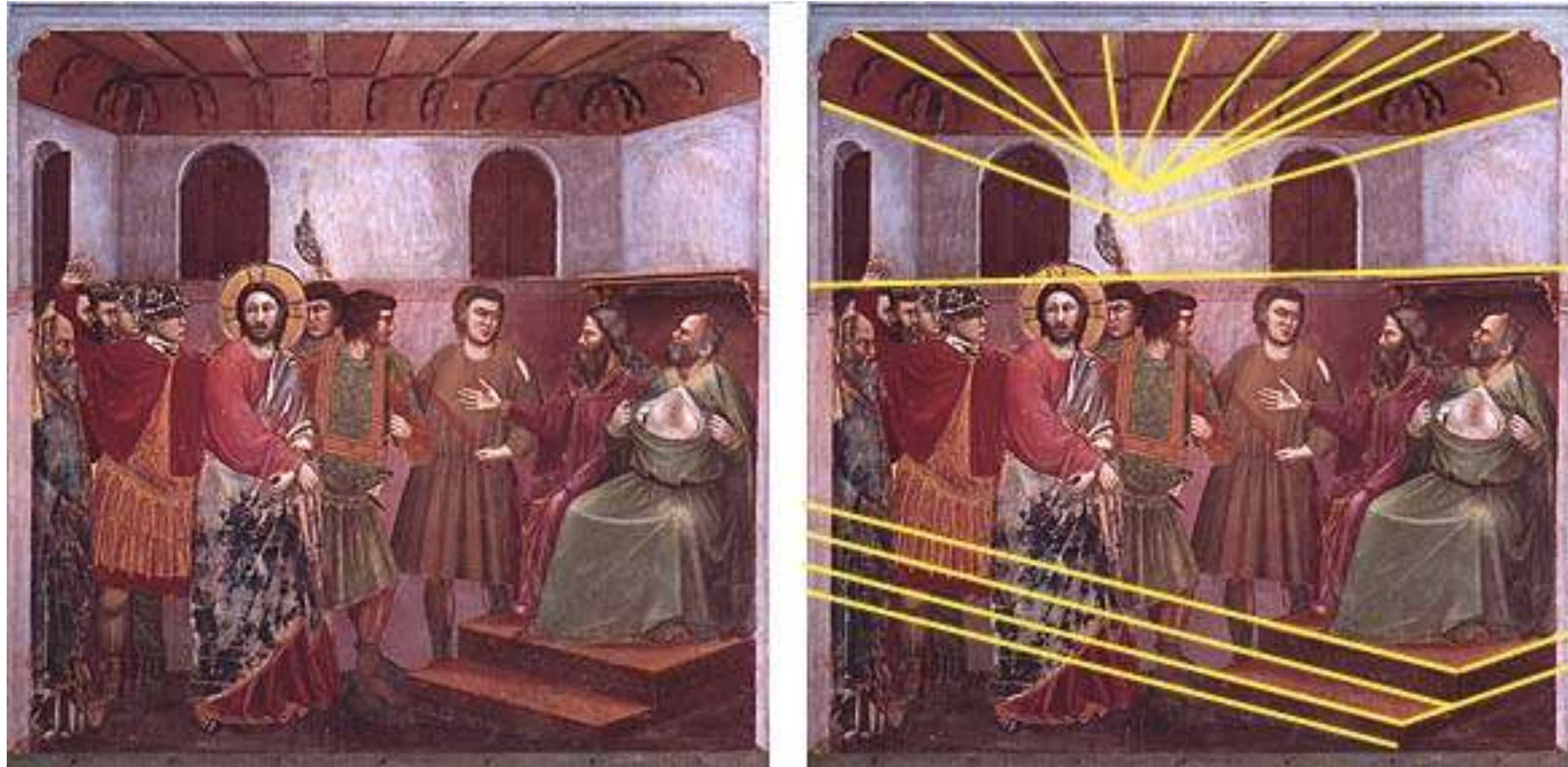
- Transformation von Weltkoord. nach Kamerakoord. = Translation + Wechsel der Basis
- Gegeben: Koord.achsen **x,y,z** & **u,v,n** und ein Vertex $P = (p_x, p_y, p_z)$
- Bestimme P in **u,v,n**-Koord., also $P' = (p_u, p_v, p_n)$
- Wechsel der Basis:

$$\begin{pmatrix} p_u \\ p_v \\ p_n \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$$



Perspektive in der Kunstgeschichte

- 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 ...



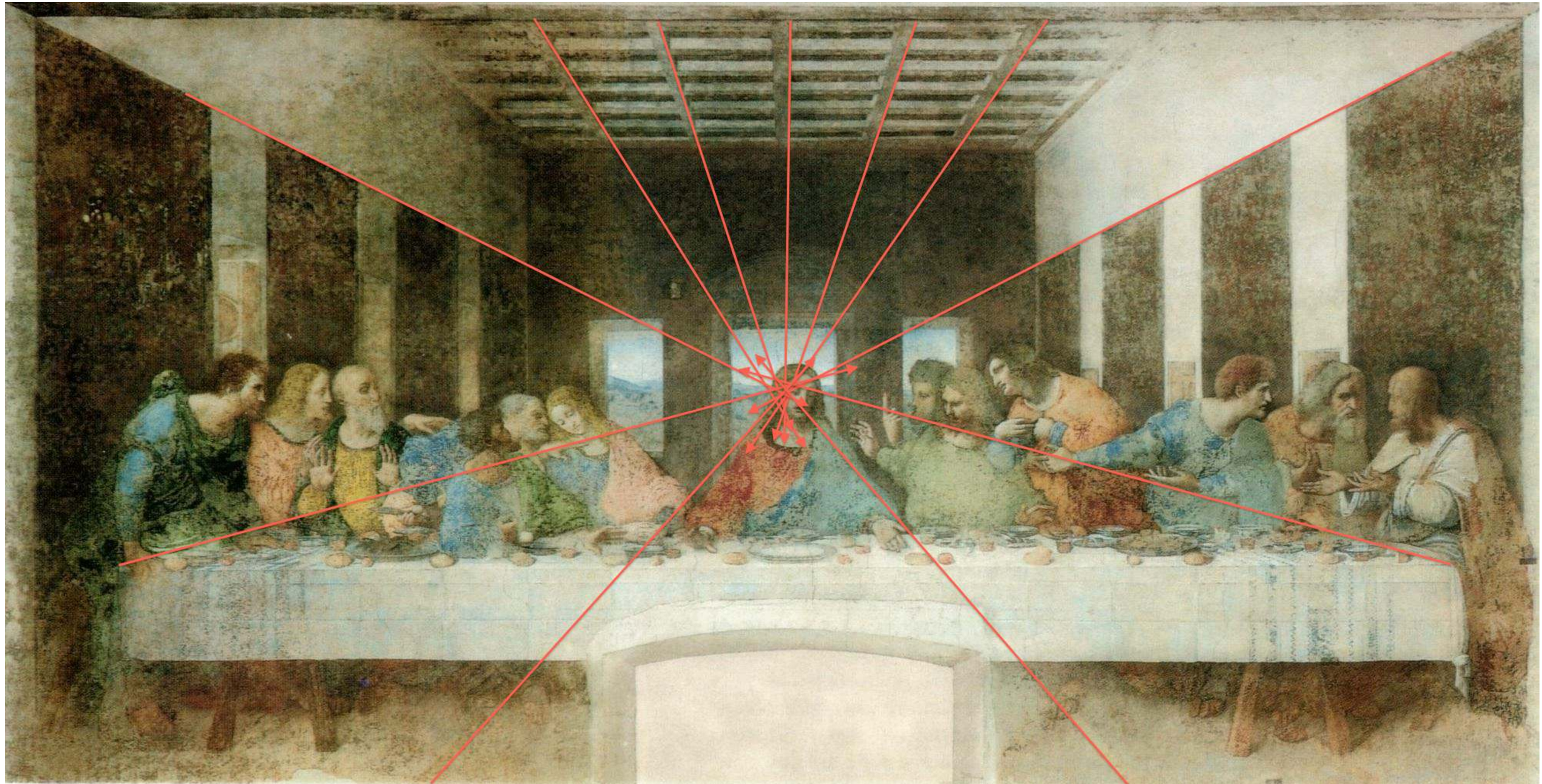
Reconstruction of the temple of Jerusalem.
From William of Tyre: Histoire d'Outremer.
France, Rouen, XVe siècle
Artist: Maître de l'Échevinage
Ca. 1460-1470

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

In der Werbung



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

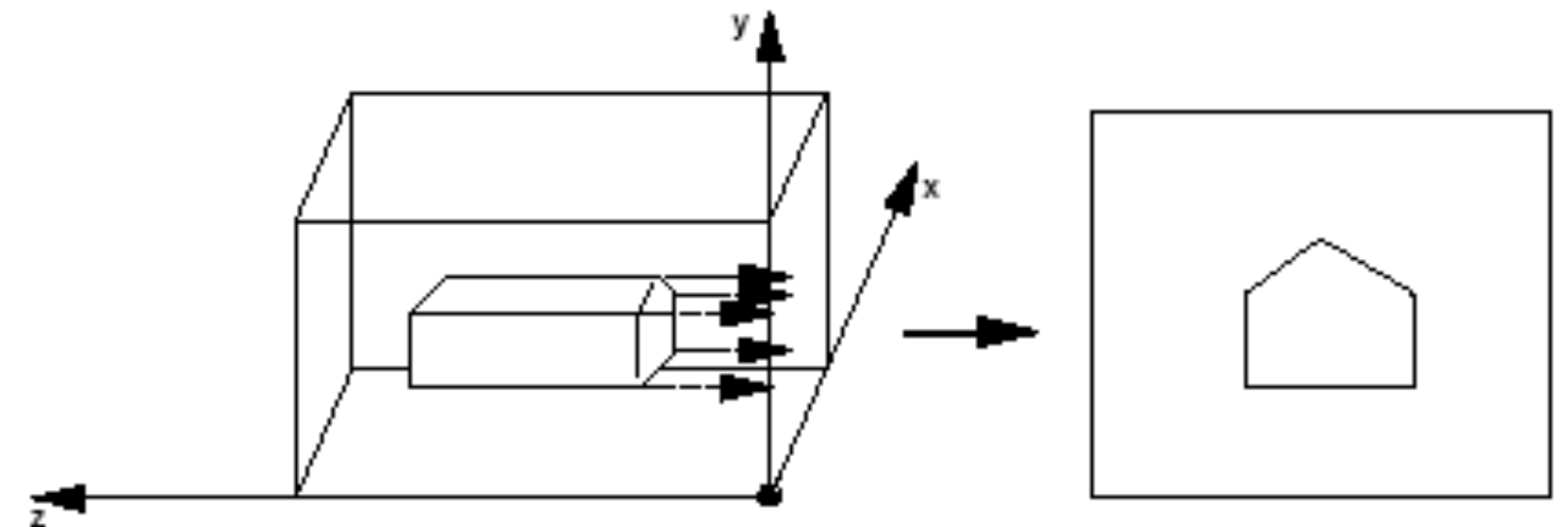
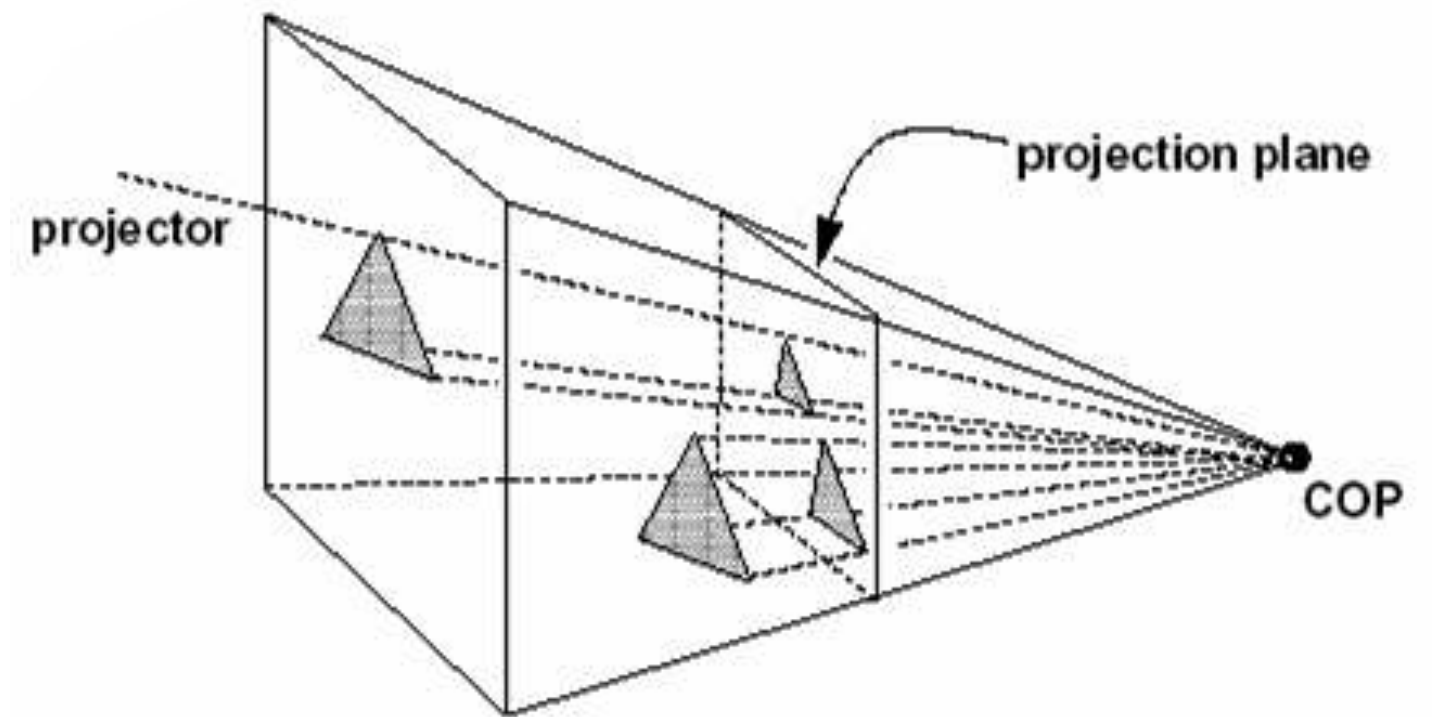
Bildunterschrift:

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



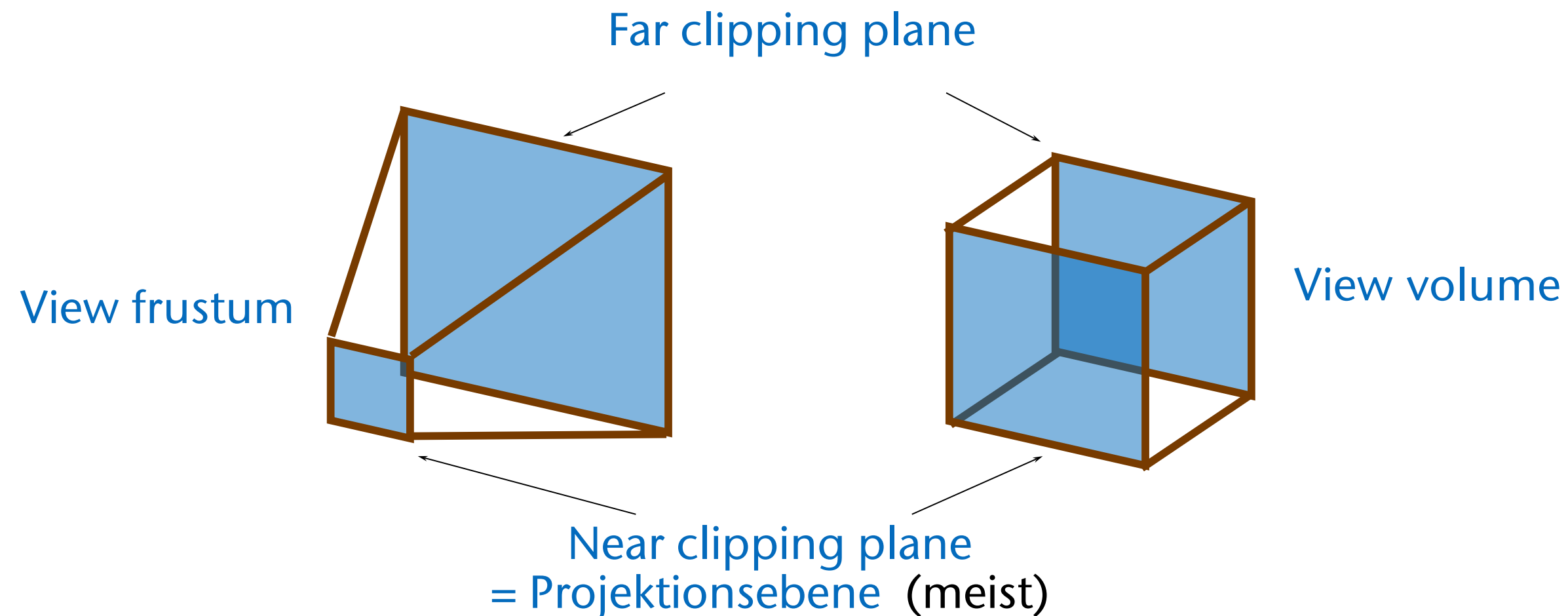
Orthographische vs. Perspektivische Projektion

- **Perspektivische Projektion** =
alle Projektionsstrahlen laufen im Projektionszentrum (COP) zusammen
- **Orthographische Projektion** =
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

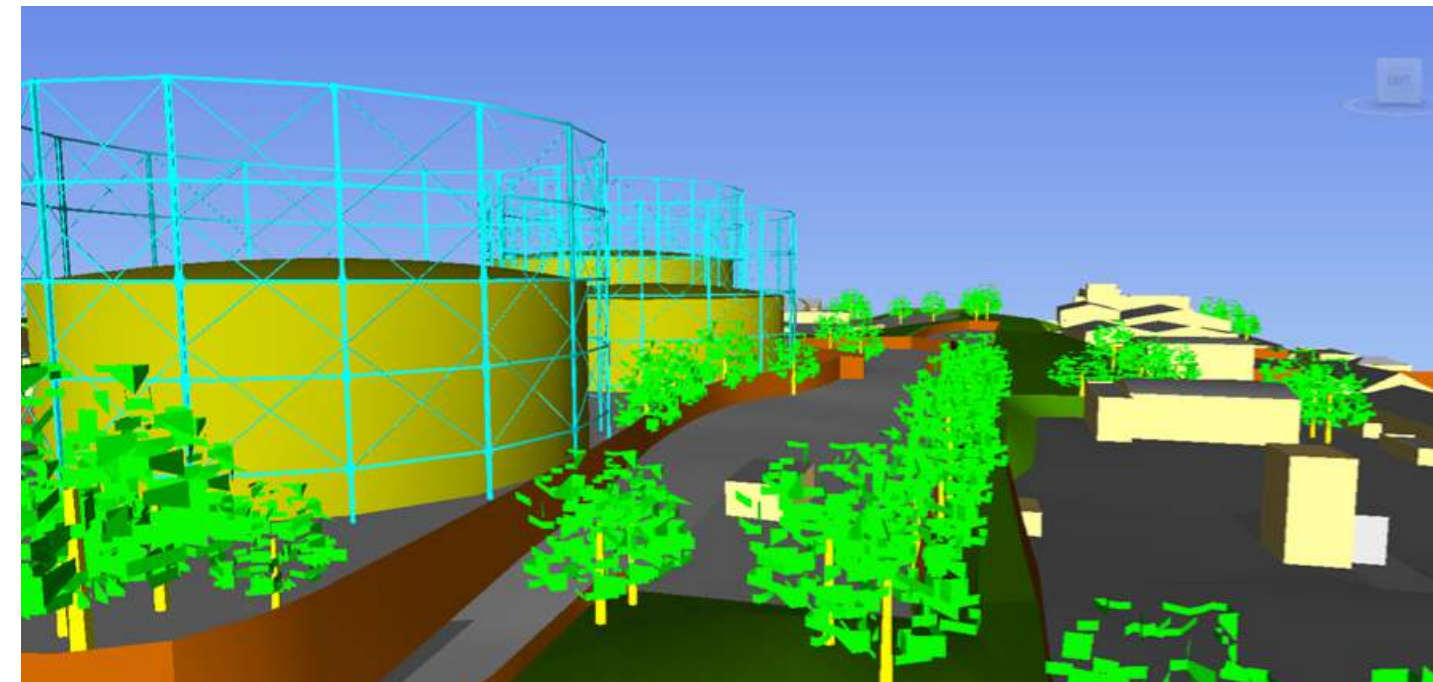
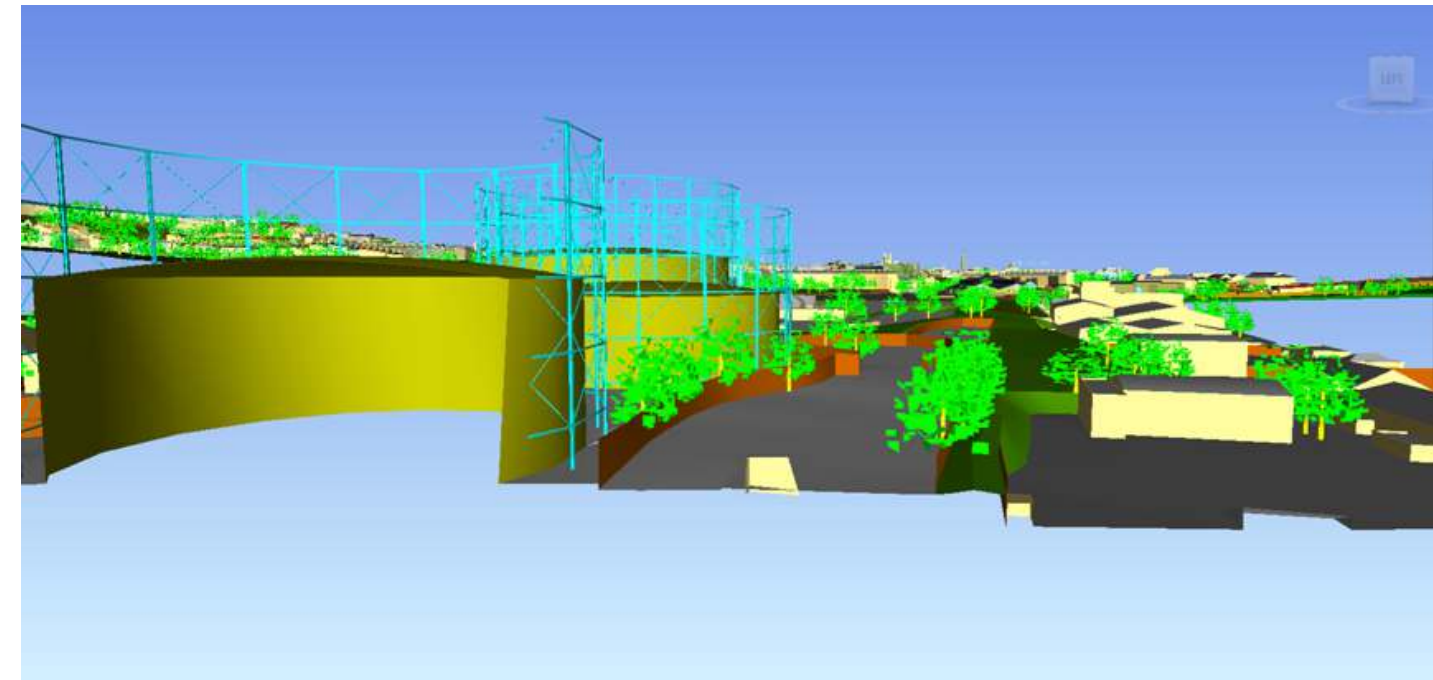


Die richtige Wahl der Near- und Far-Plane

Szene von oben



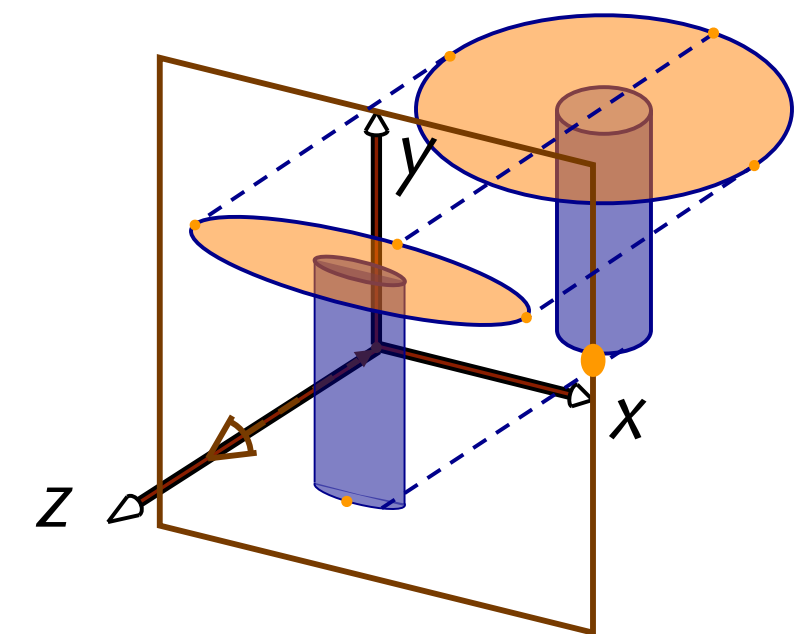
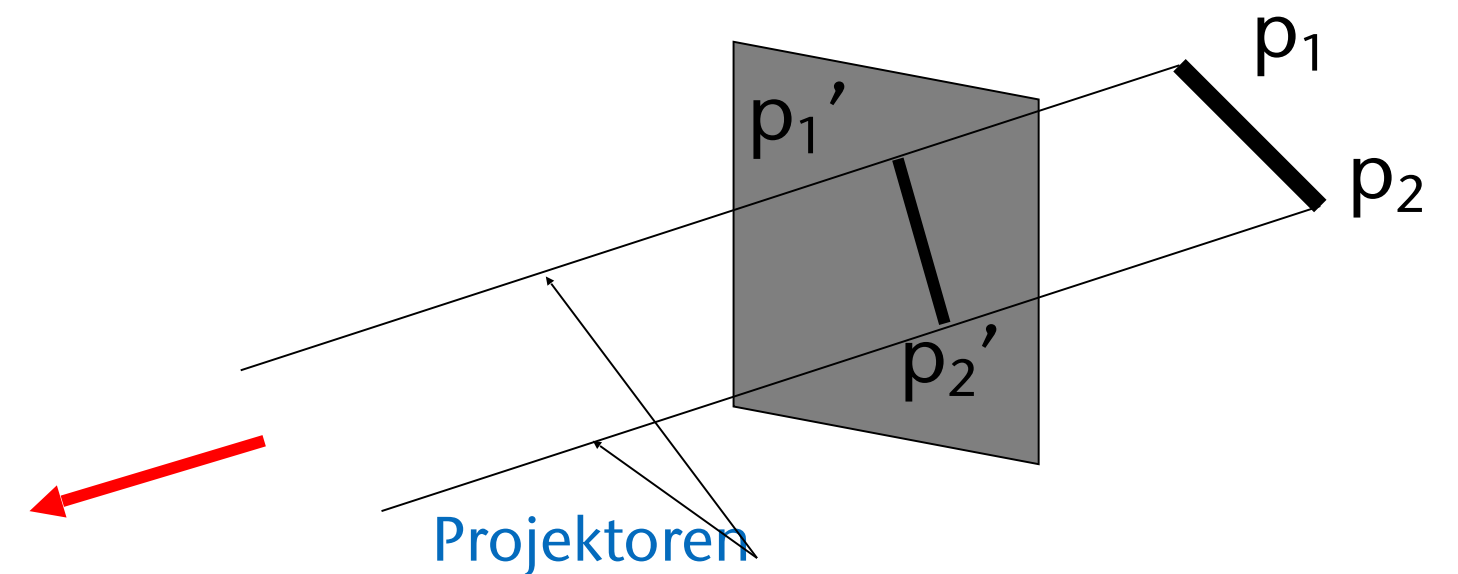
Near-Plane zu weit weg



Far-Plane zu nah

Orthographische Projektion

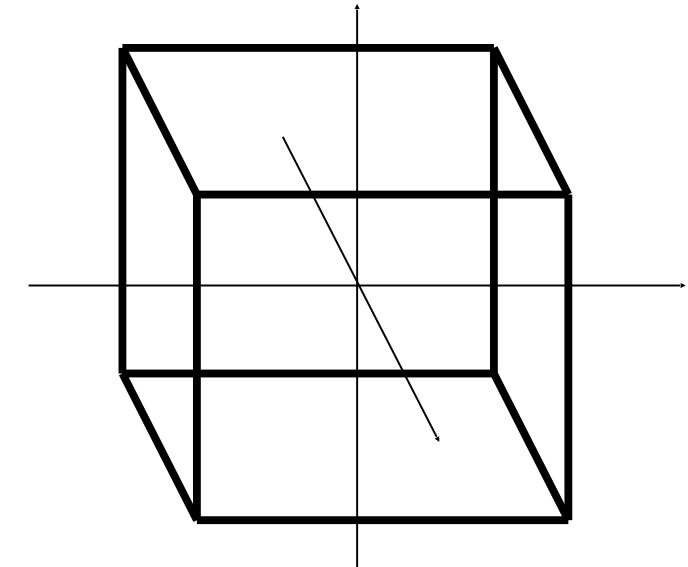
- Punkte werden orthogonal auf die Projektionsebene (*viewing plane*) projiziert
 - Projektionslinien verlaufen senkrecht zur Projektionsebene
- Eigenschaften:
 - Parallele Linien bleiben parallel
 - Winkelverhältnisse bleiben erhalten, aufgrund der parallelen Verschiebung zur Projektionsebene
- Es gibt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



Die Projektionsmatrix

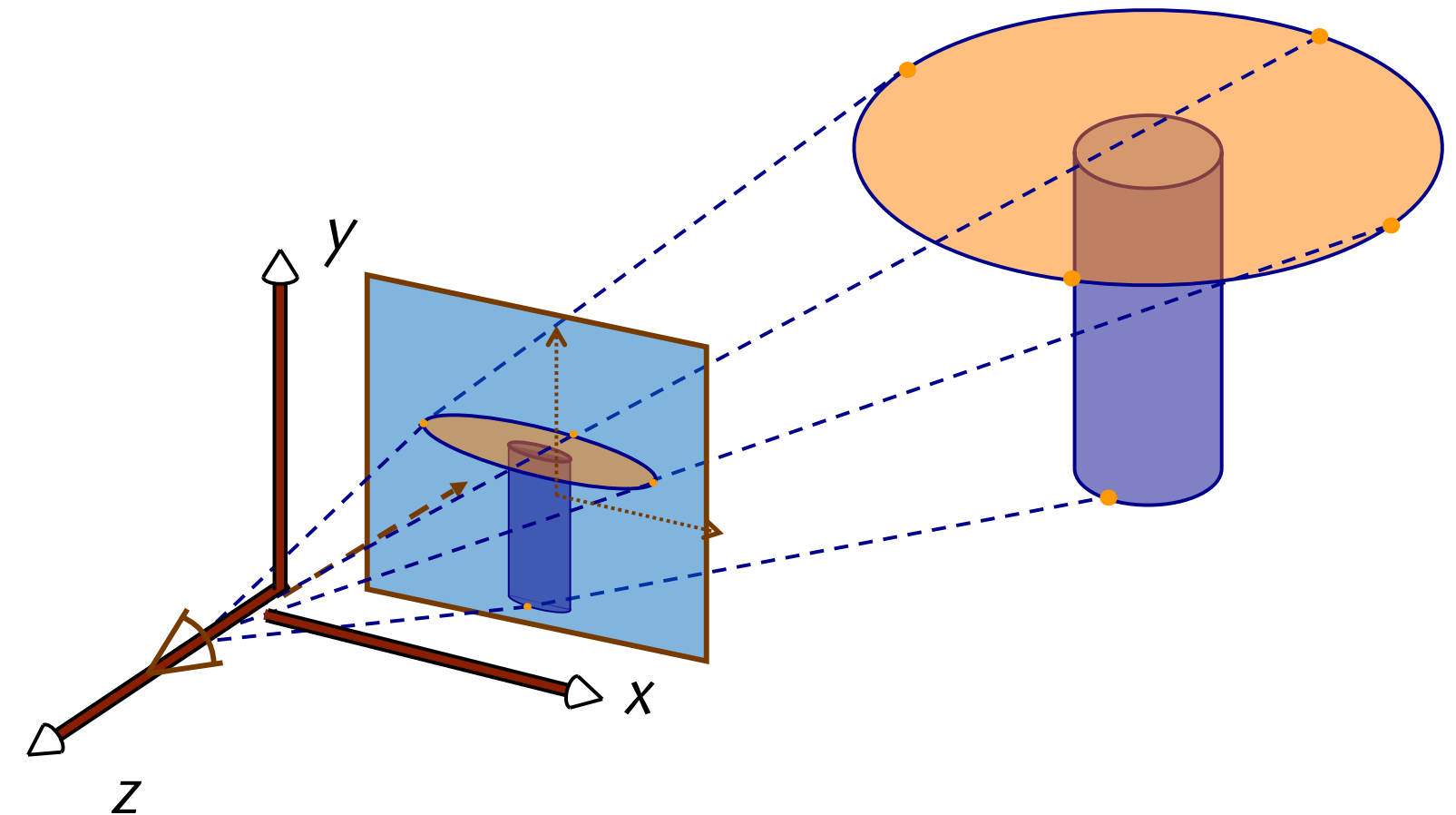
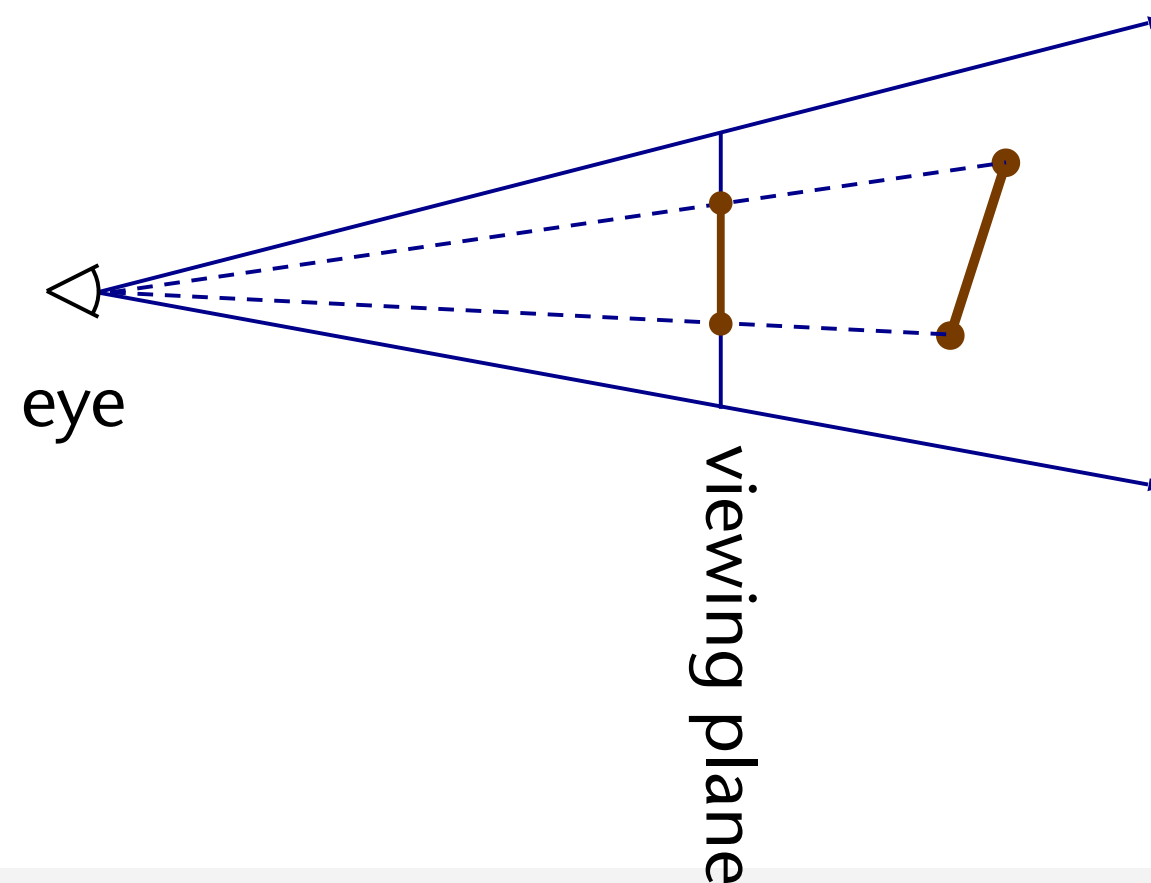
- Annahme: die komplette "virtuelle Welt" befindet sich im (kanonischen) Viewing-Volume (-1,-1,-1) bis (1,1,1)
- Die x- und y-Komponente bleiben unverändert
- Projektionsmatrix setzt z-Komponente auf 0:

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

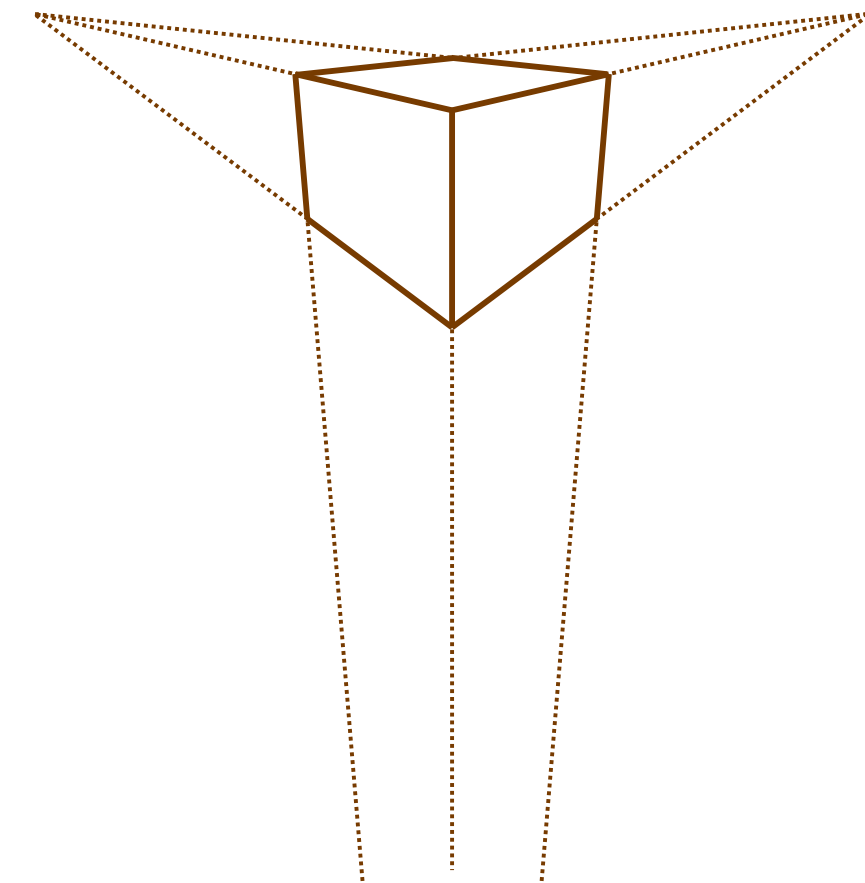
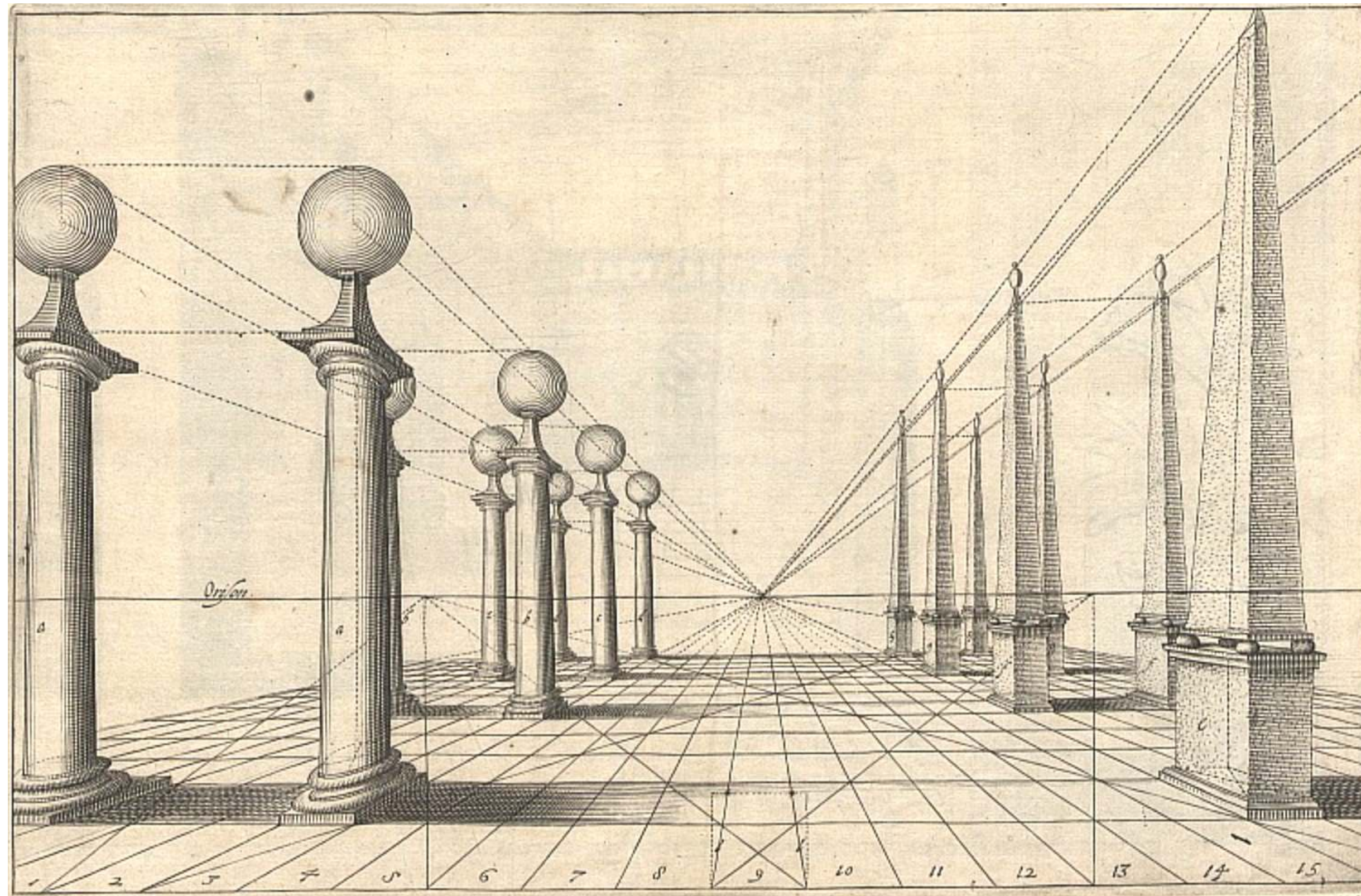


Perspektivische Projektion

- Wird am häufigsten verwendet in der Computergraphik & Malerei (früher)
- 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



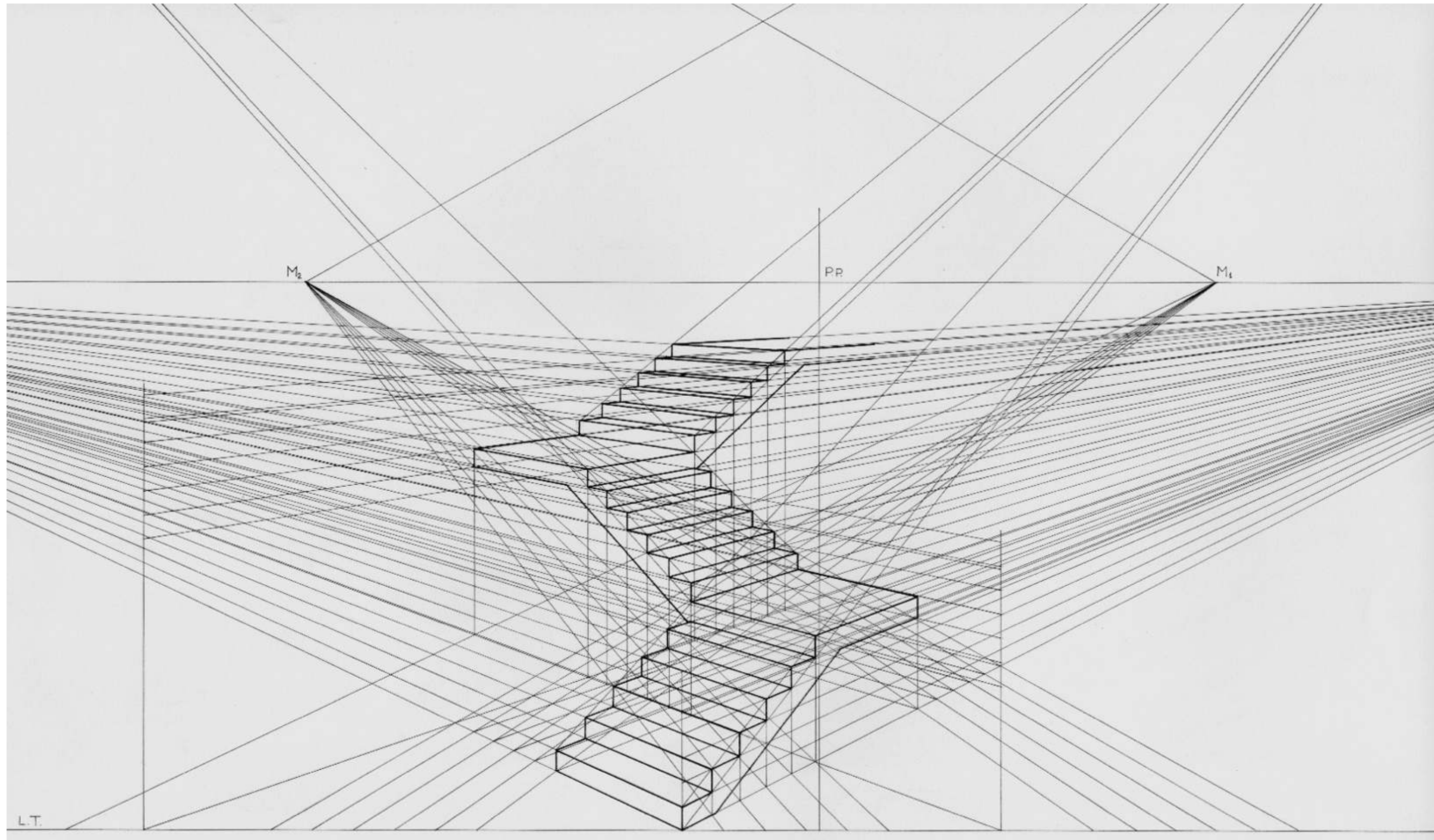
Konstruktion in der Malerei



3-Punkt-Perspektive

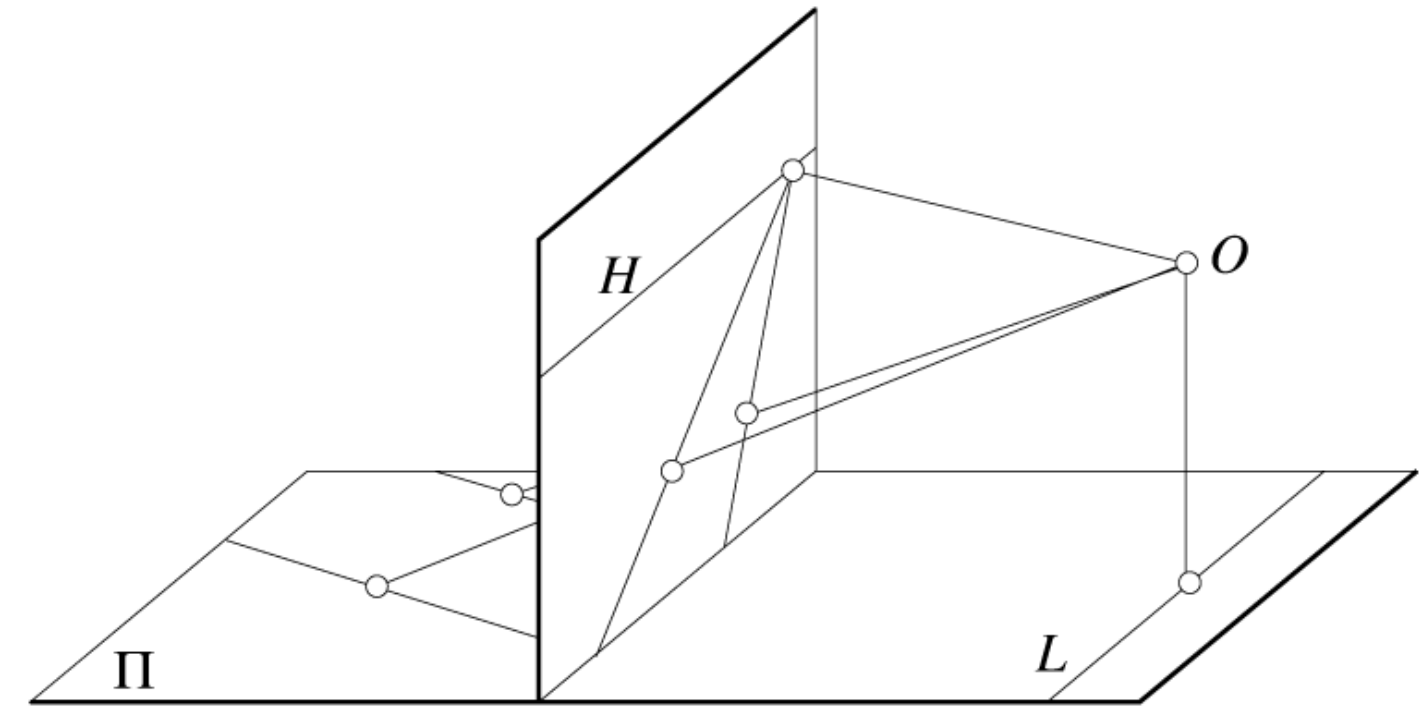
Eigenschaften

- 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**
- Alle Bündel von parallelen Linien, die horizontal sind (müssen nicht notw. in einer horizontalen Ebene liegen!), haben (jeweils) einen Fluchtpunkt, der auf der Horizontlinie im Bild liegt
- Zu jeder Schar von Ebenen im 3D gehört eine solche "Fluchtpunktlinie" im Bild

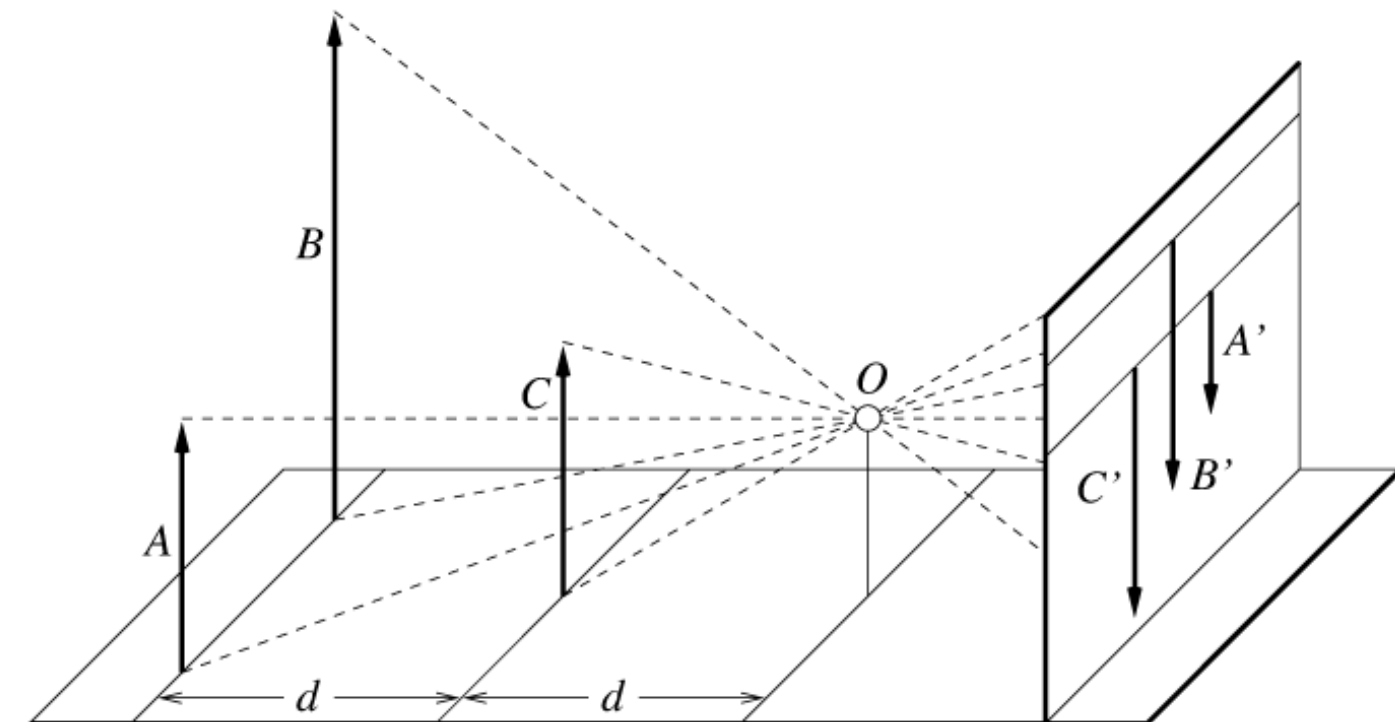


Veranschaulichung der Eigenschaften der perspektivischen Projektion

- Parallele Linien bleiben *nicht* parallel



- Längen im Bild hängen von der Tiefe ab

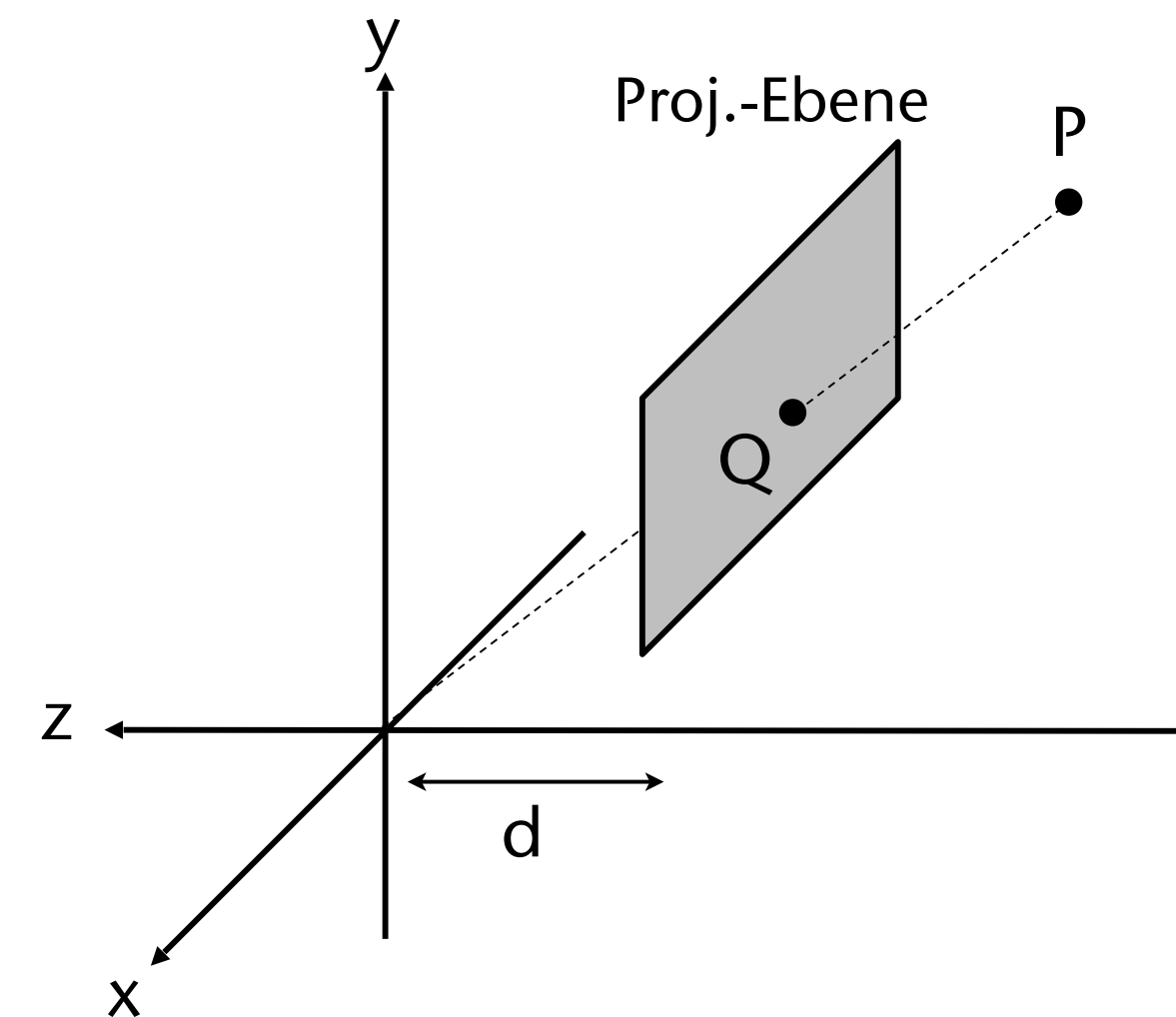


Die Projektionsmatrix

- 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

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

numerisch instabil

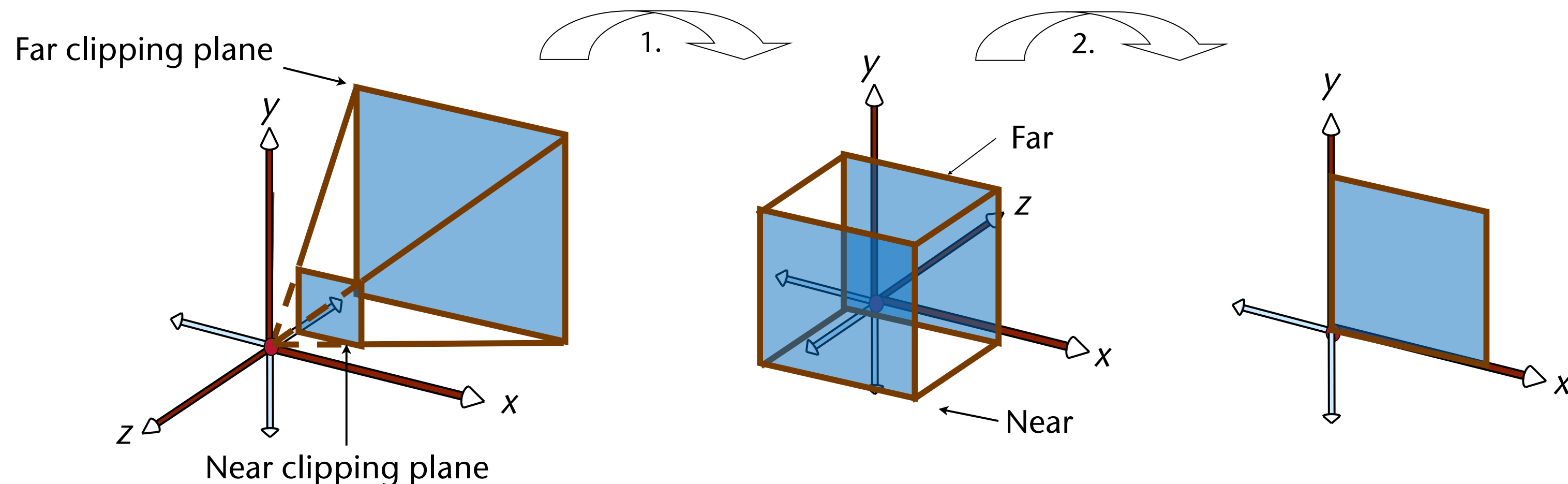
Perspektivische Projektion in 2 Schritten

- Wegen Clipping macht man die perspektivische Projektion in zwei Schritten

1. Perspektivische Abbildung (noch keine Projektion, im Vertex-Shader)

- Achtung: enthält Übergang von Recht- auf Links-System!!

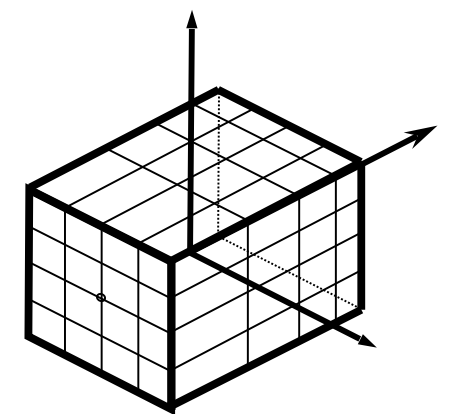
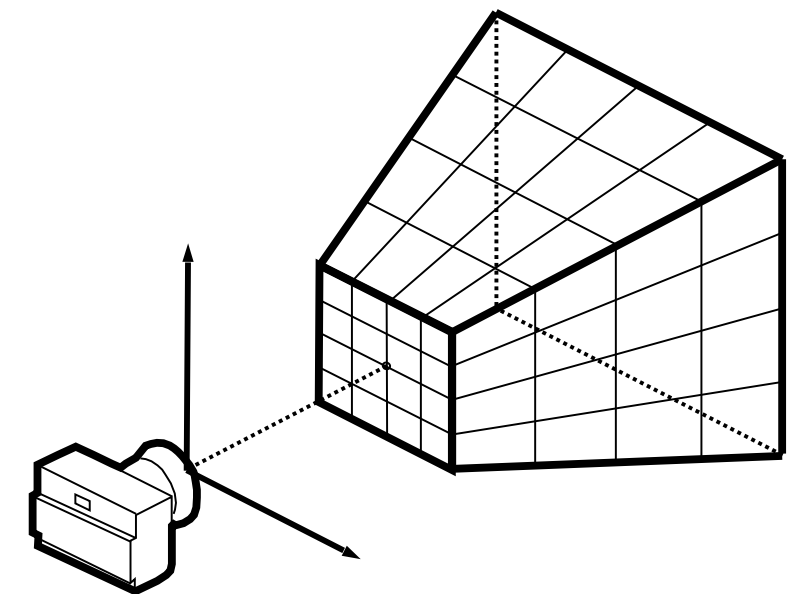
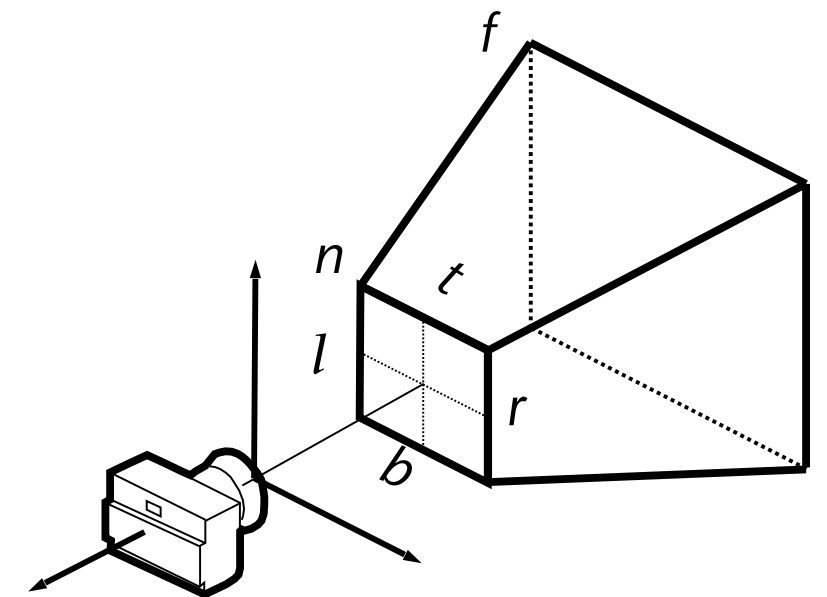
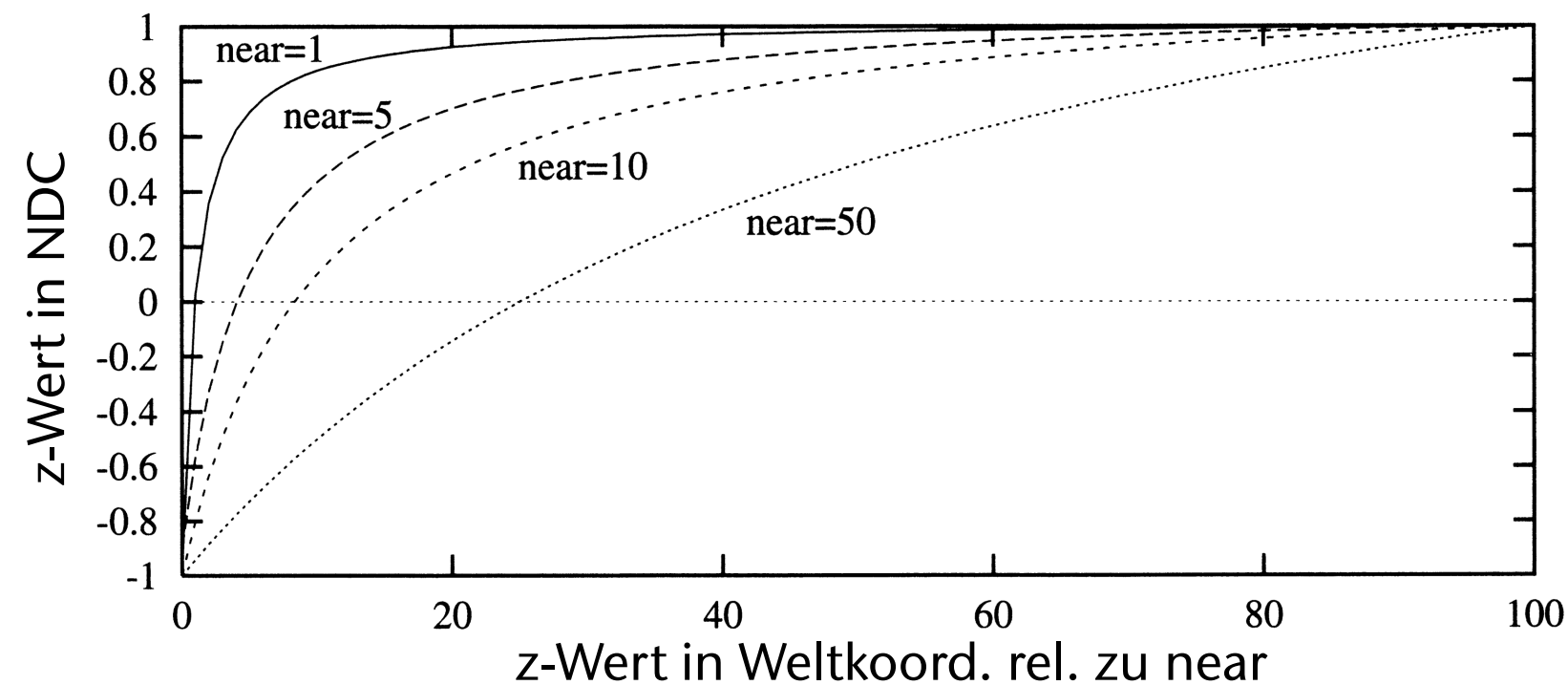
2. Projektion auf Ebene (einfach Z-Koord. weglassen, automatisch)



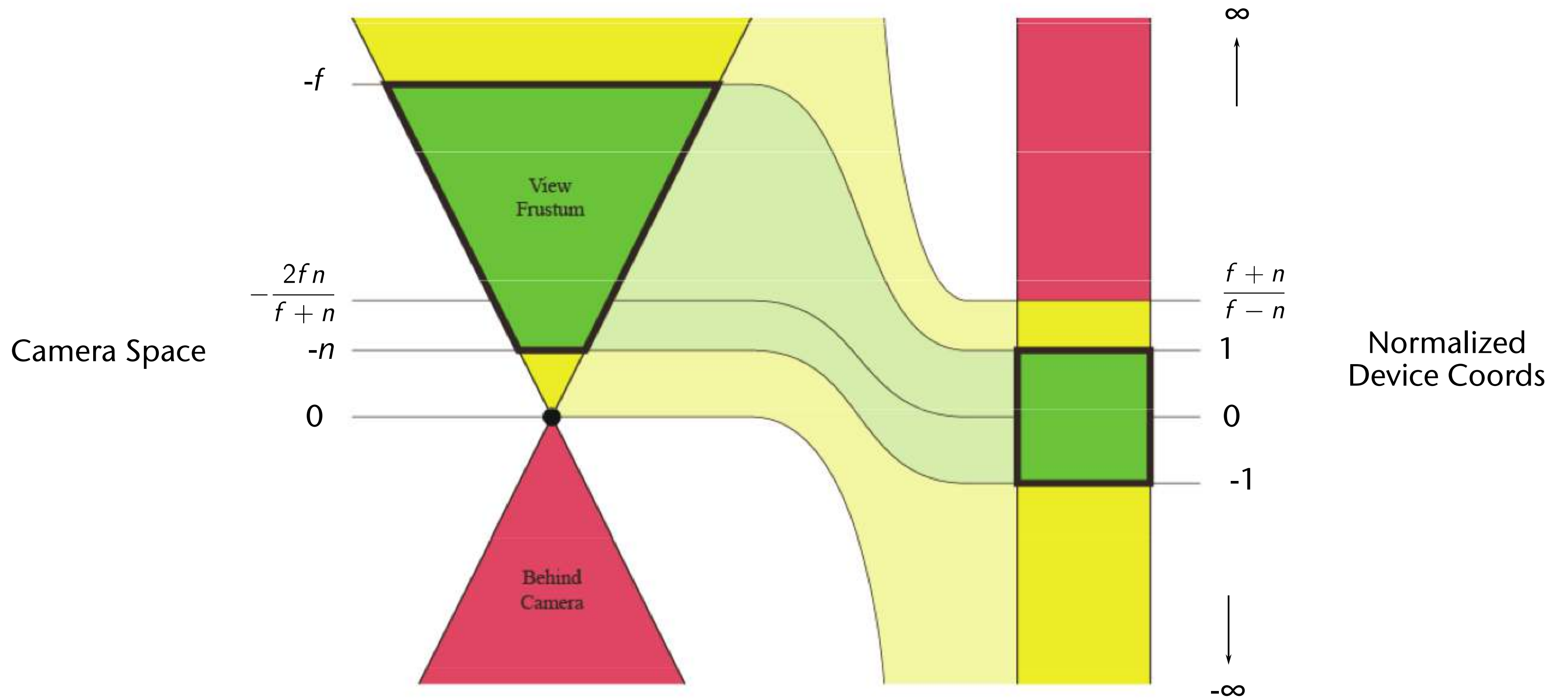
- Die Matrix für Schritt 1 (o. Bew.):

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

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



Visualisierung des Z-Werte-Bereiches



Allgemeine projektive Transformationen

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

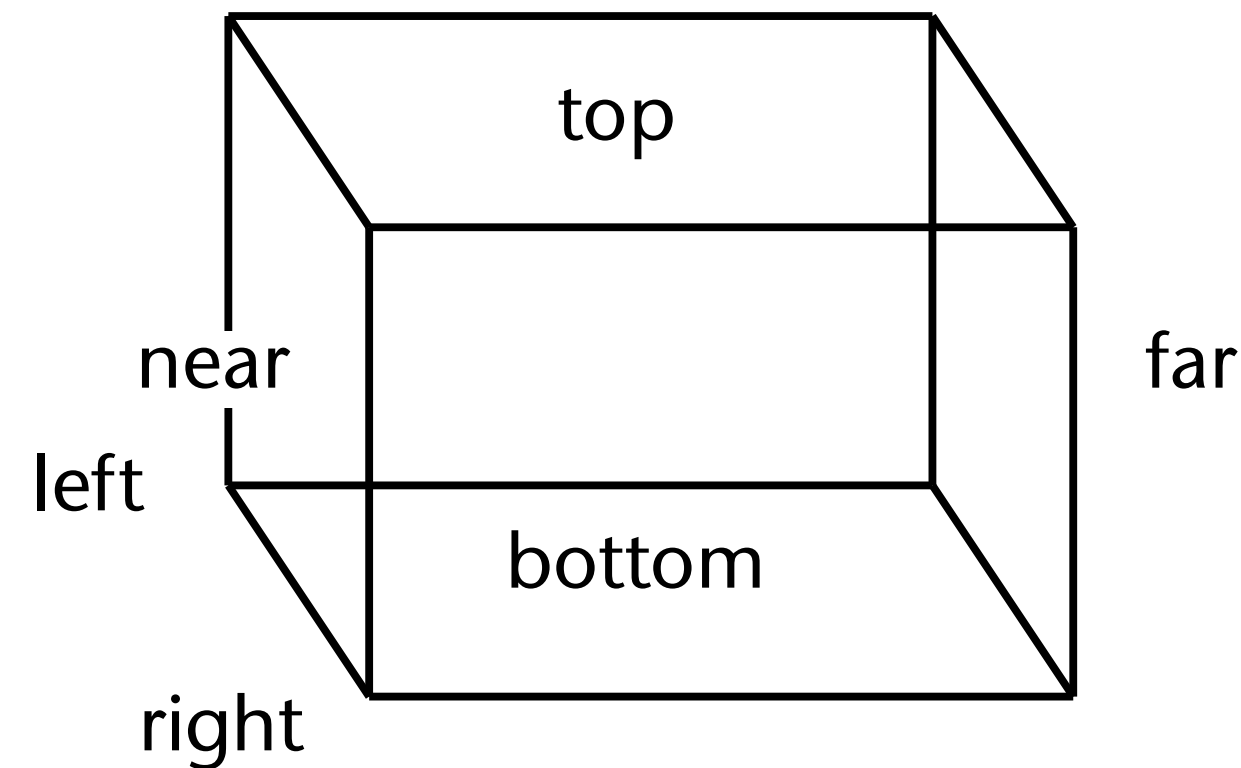
entspricht für $y = \begin{pmatrix} x \\ 1 \end{pmatrix}$, $x \in \mathbb{R}^3$, der Abbildung

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

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

Projektionsmatrizen mit Qt für OpenGL

- Orthographische Projektion



```
QMatrix4x4::ortho(left, right, bottom, top, near, far);
```

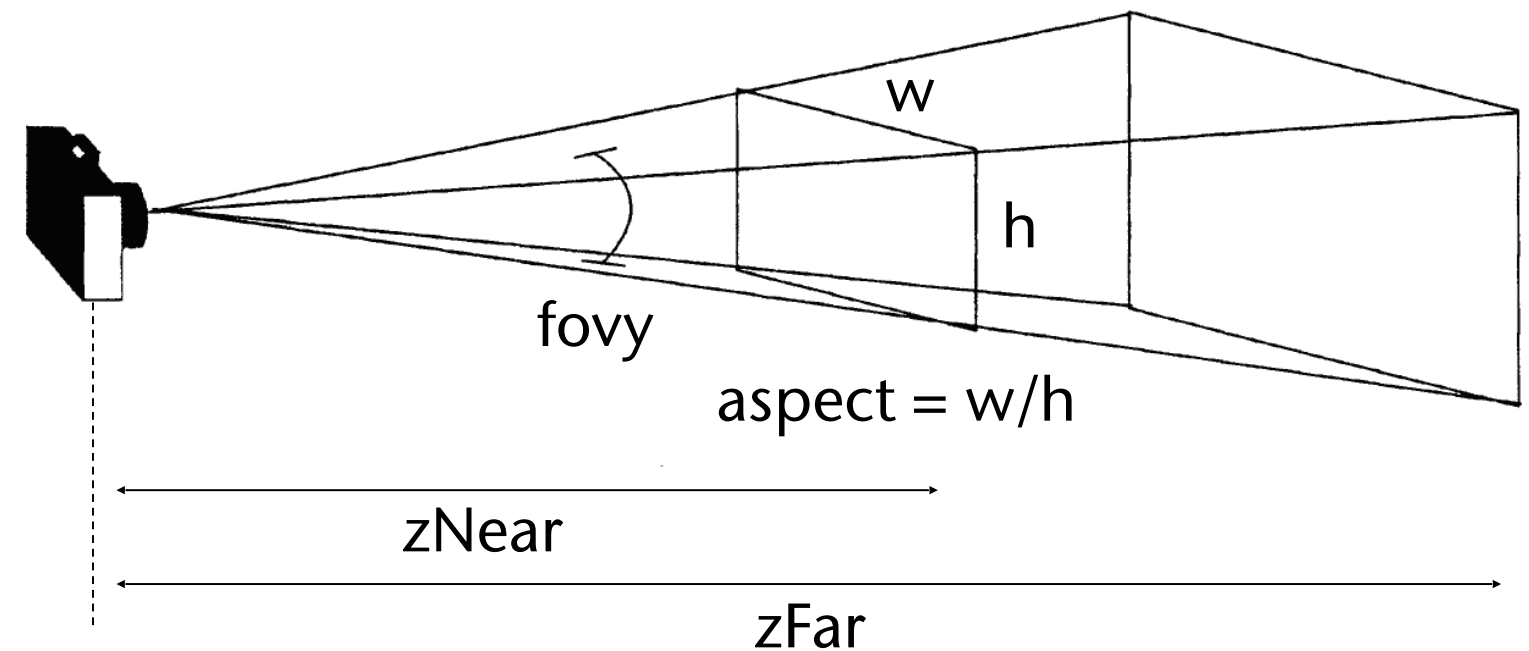
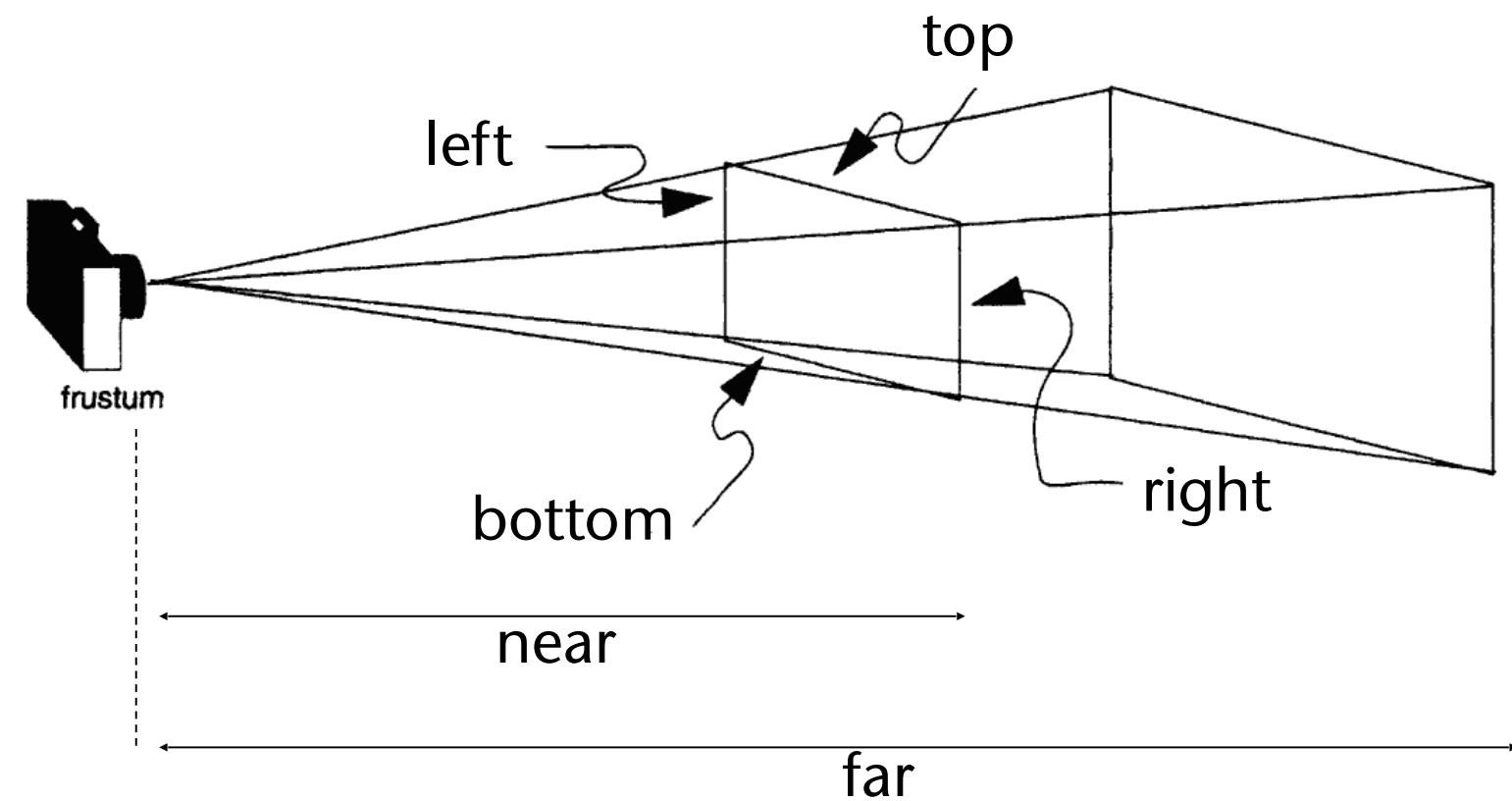
<http://doc.qt.io/qt-5/qmatrix4x4.html>

- Perspektivische Projektion:

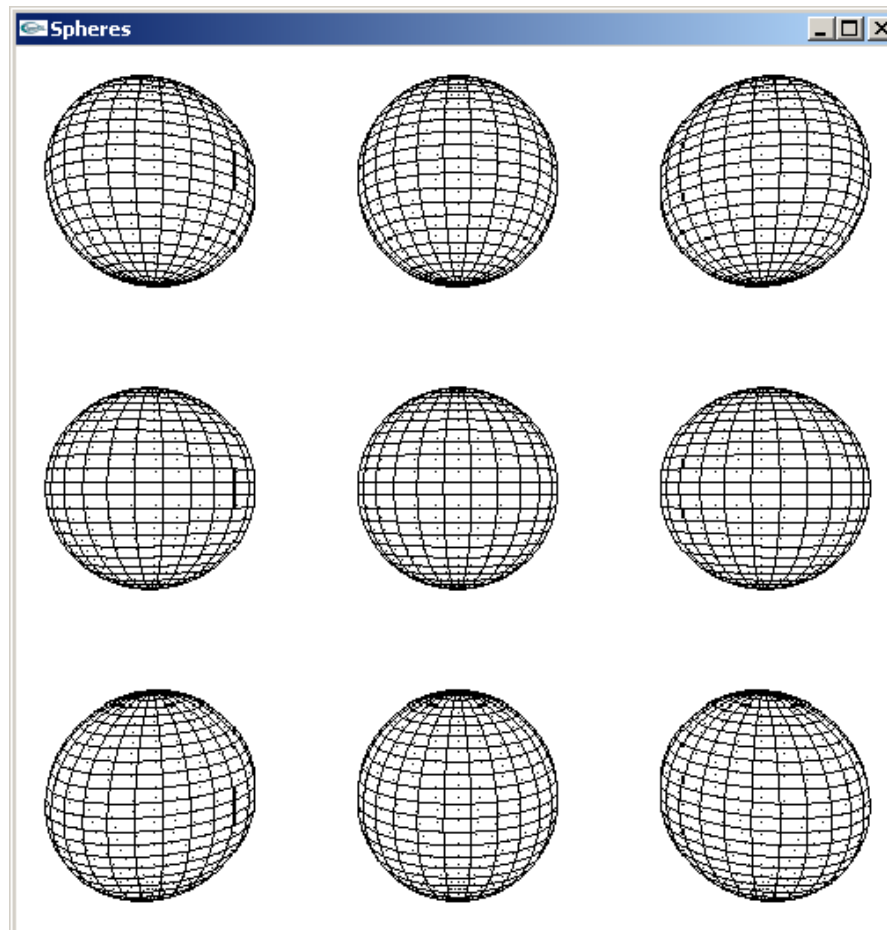
```
QMatrix4x4::frustum(
    left, right,
    bottom, top,
    near, far );
```

```
QMatrix4x4::perspective(
    fovy, aspect,
    zNear, zFar );
```

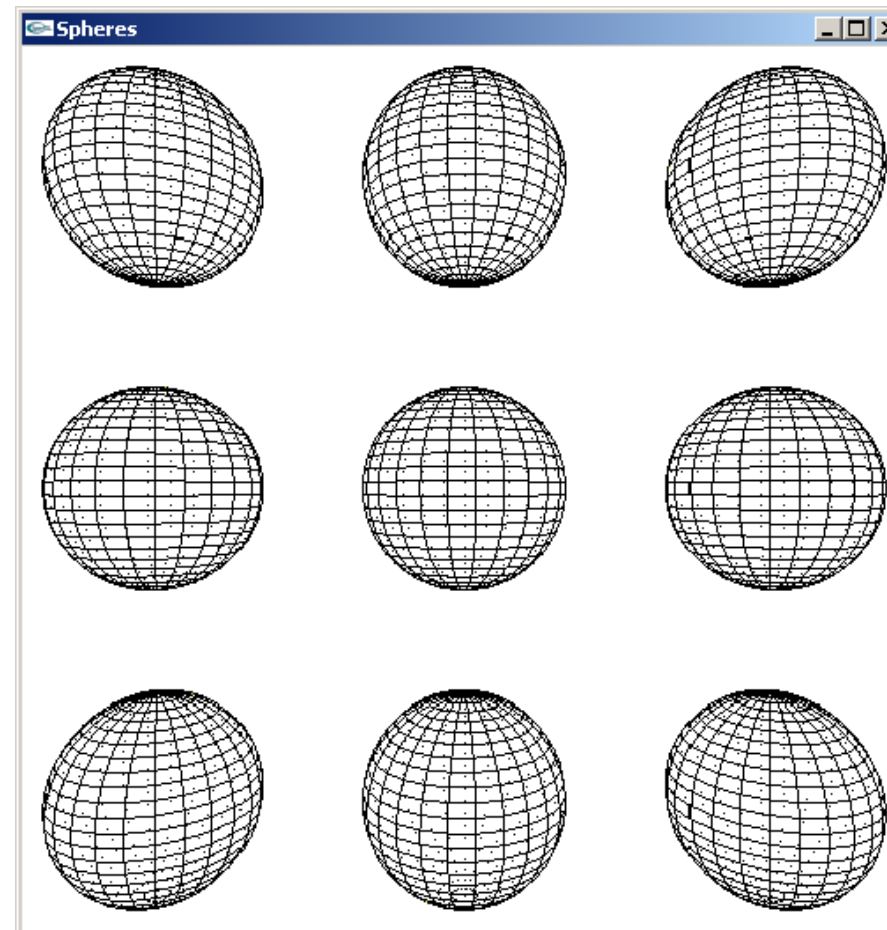
<http://doc.qt.io/qt-5/qmatrix4x4.html>



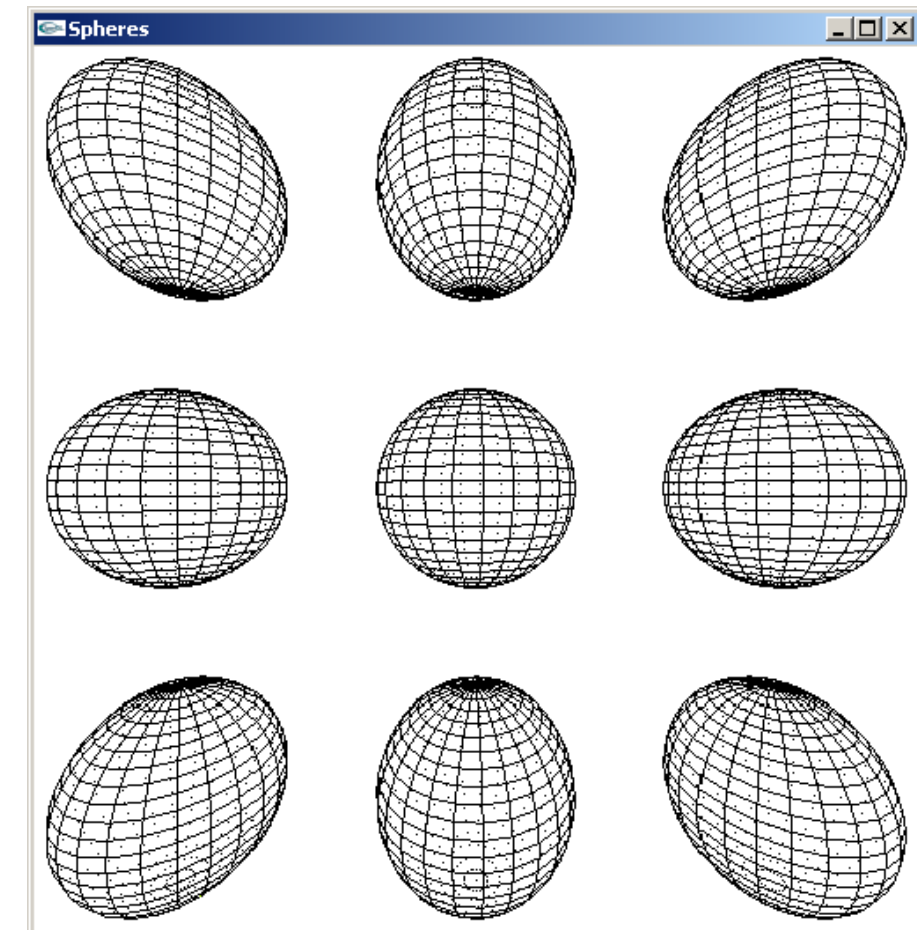
- Vorsicht bei Perspektive: Öffnungswinkel nicht zu groß wählen!



45°



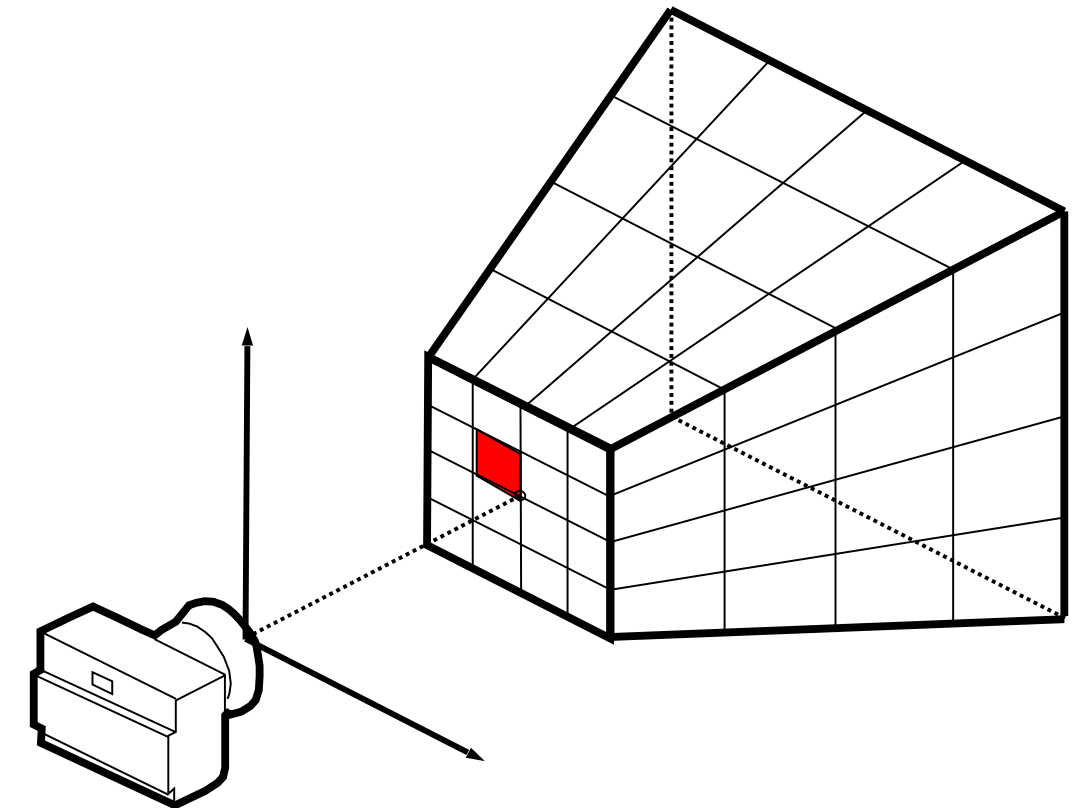
60°



90°

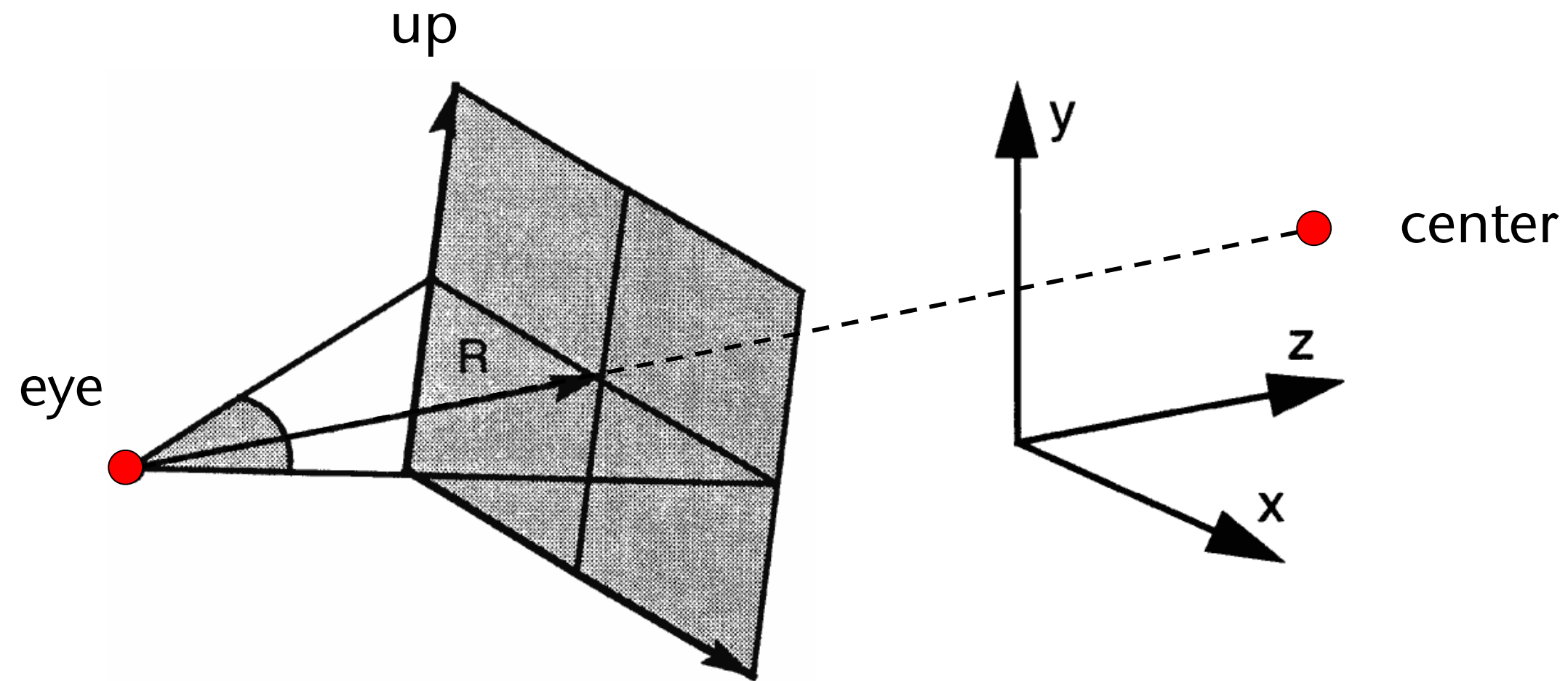
Asymmetrisches Frustum ("off-center perspective")

- Kommt manchmal vor, z.B.
 - Stereo-Projektion (s. VR-Vorlesung)
 - Rendern eines Posters mit 10000 x 10000 Pixel (Framebuffer zu klein)
- Projektionsmatrix muss "von Hand" generiert werden (mit `QMatrix4x4::perspective()` nicht möglich)
- Poster: Zerlegen in viele Teilbilder mit voller Auflösung



Hier ist left $\neq$ right, top $\neq$ bottom!

Festlegen der Kamera-Transformation mittels Qt



```
QMatrix4x4::lookAt( eyeX, eyeY, eyeZ,  
                    centerX, centerY, centerZ,  
                    upX, upY, upZ );
```

Die Projektionsmatrix in OpenGL

- Die Projektions-Matrix muss ab OpenGL 2+ manuell an das Shader-Programm übergeben werden
 - Man muss im Shader-Programm selbst die Transformation durchführen

```
glUniformMatrix4fv(
    // location of the uniforms in shader programm
    glGetUniformLocation(program_id, "matProjection"),
    1,                      // one 4x4 matrix
    GL_FALSE,              // do not transpose
    matProjection.data() ); // pointer to float values
```

Host
program
(in C)

```
uniform mat4 matProjection;
// ...
void main() {
    gl_Position = matProjection * matCamera
                  * matModel * vPosition; }
```

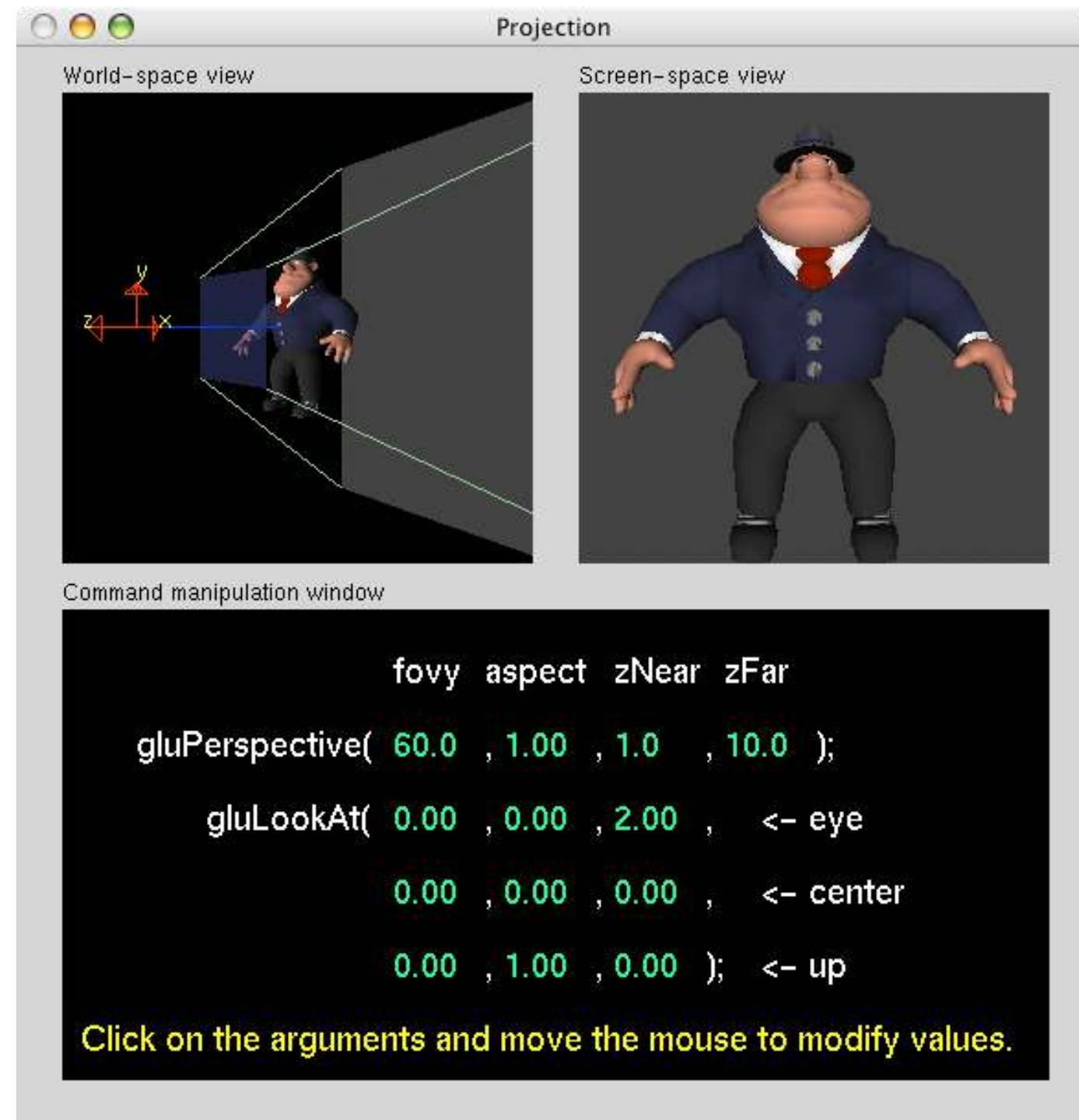
Shader
program
(in GLSL)

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

```
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
glUseProgram( program_id );                               // load shader
glBindVertexArray( vao );                                  // select 'object'

glUniformMatrix4fv( locMatProjection, 1, GL_FALSE,      matProjection.data() );
glUniformMatrix4fv( locMatCamera,      1, GL_FALSE,      matCamera.data() );
glUniformMatrix4fv( locMatModelToWorld, 1, GL_FALSE,      matModelToWorld.data() );

// render geometry ...
```



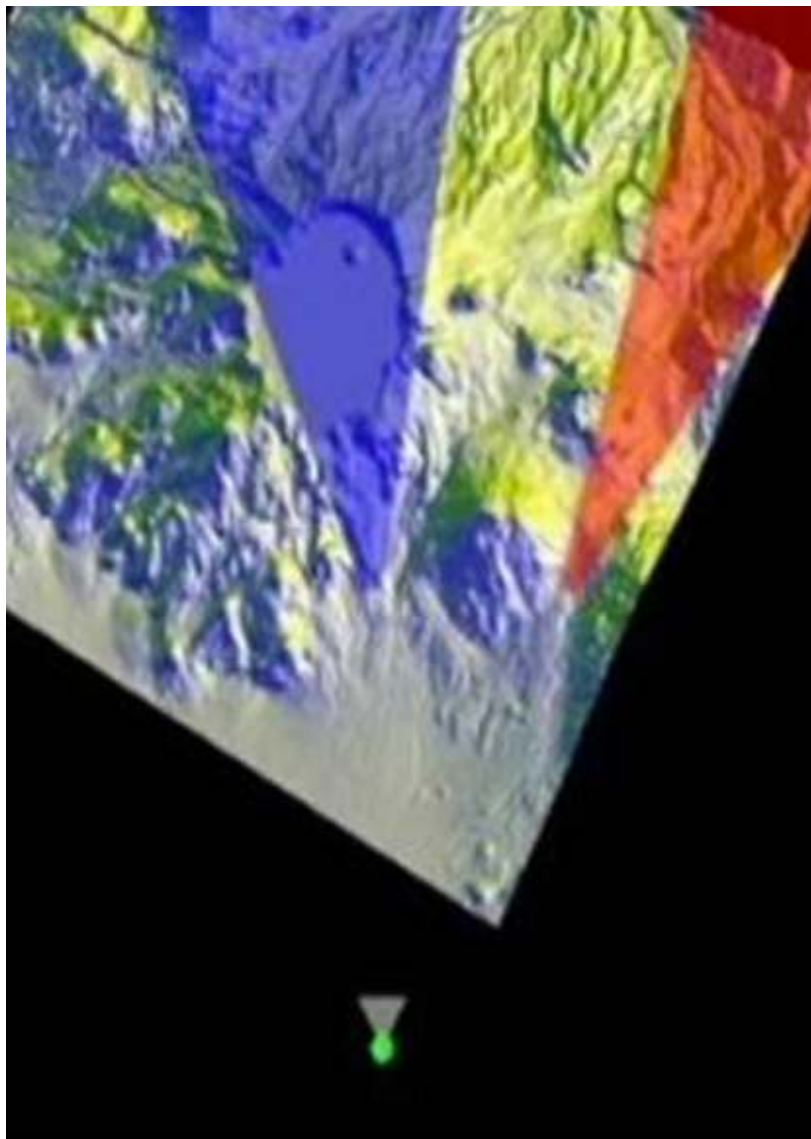
<http://www.xmission.com/~nate/tutors.html>

Bei Projektionen geht (zwangsläufig) Information verloren

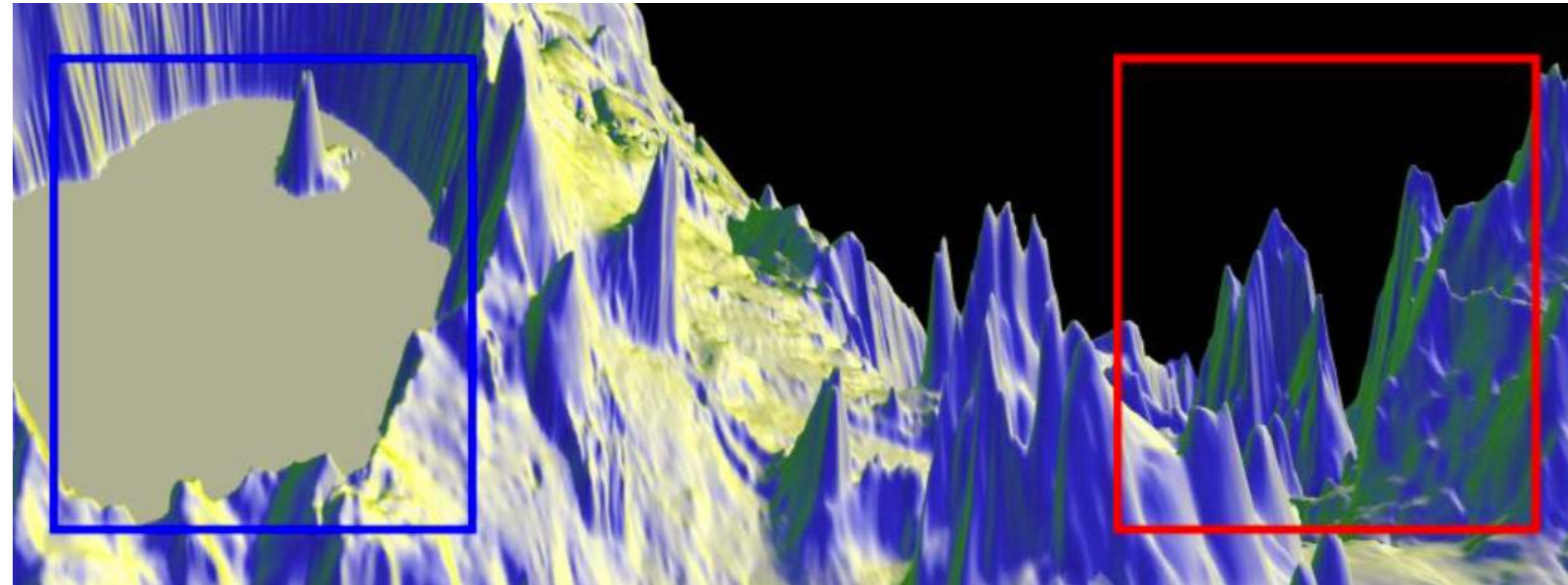


Multi-Perspective

- Eine Visualisierungstechnik aus der Kategorie Focus+Context
- Beispiel: Hervorhebung von bestimmter Details

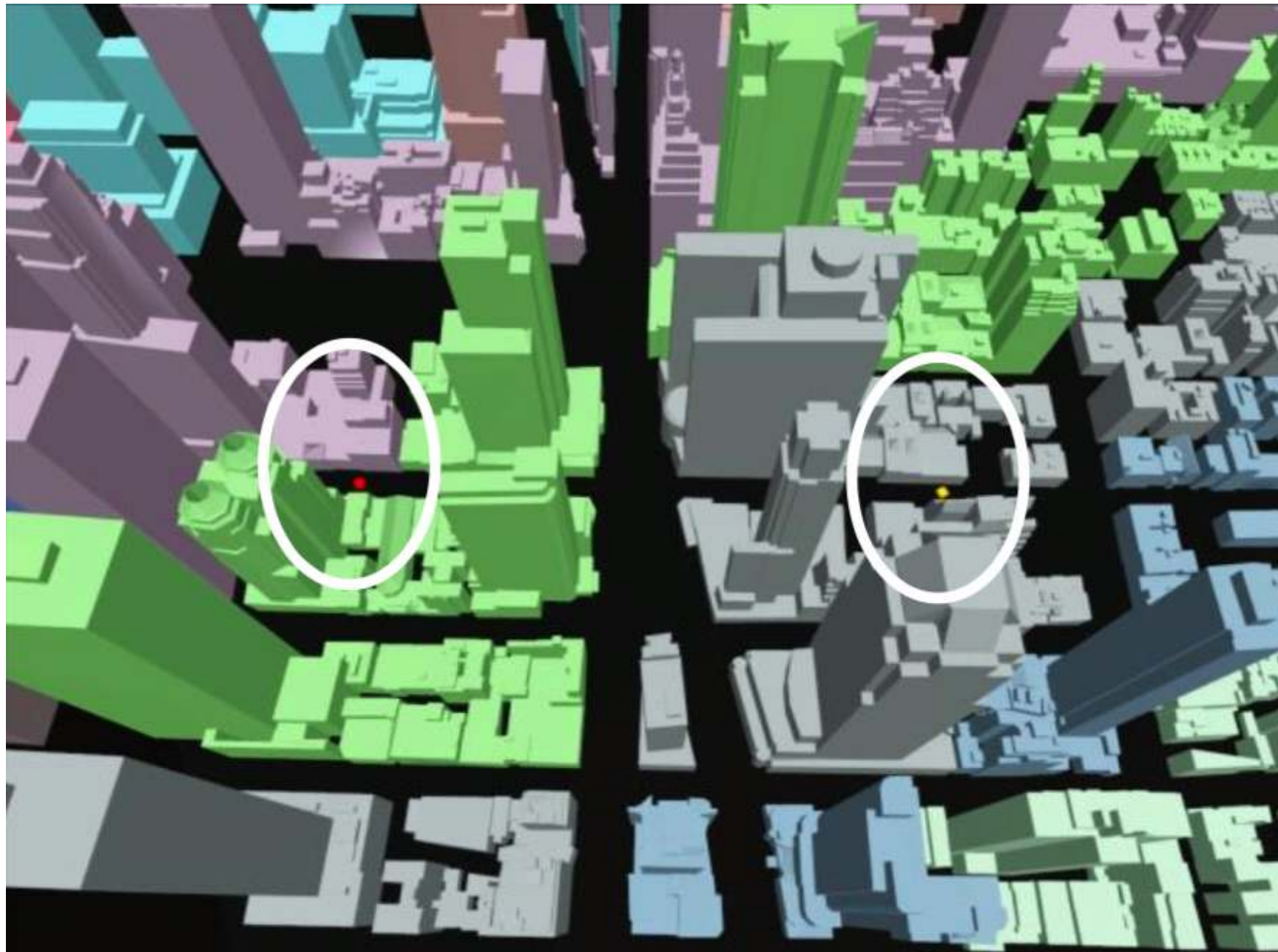


Vogelperspektive



Ansicht für den User

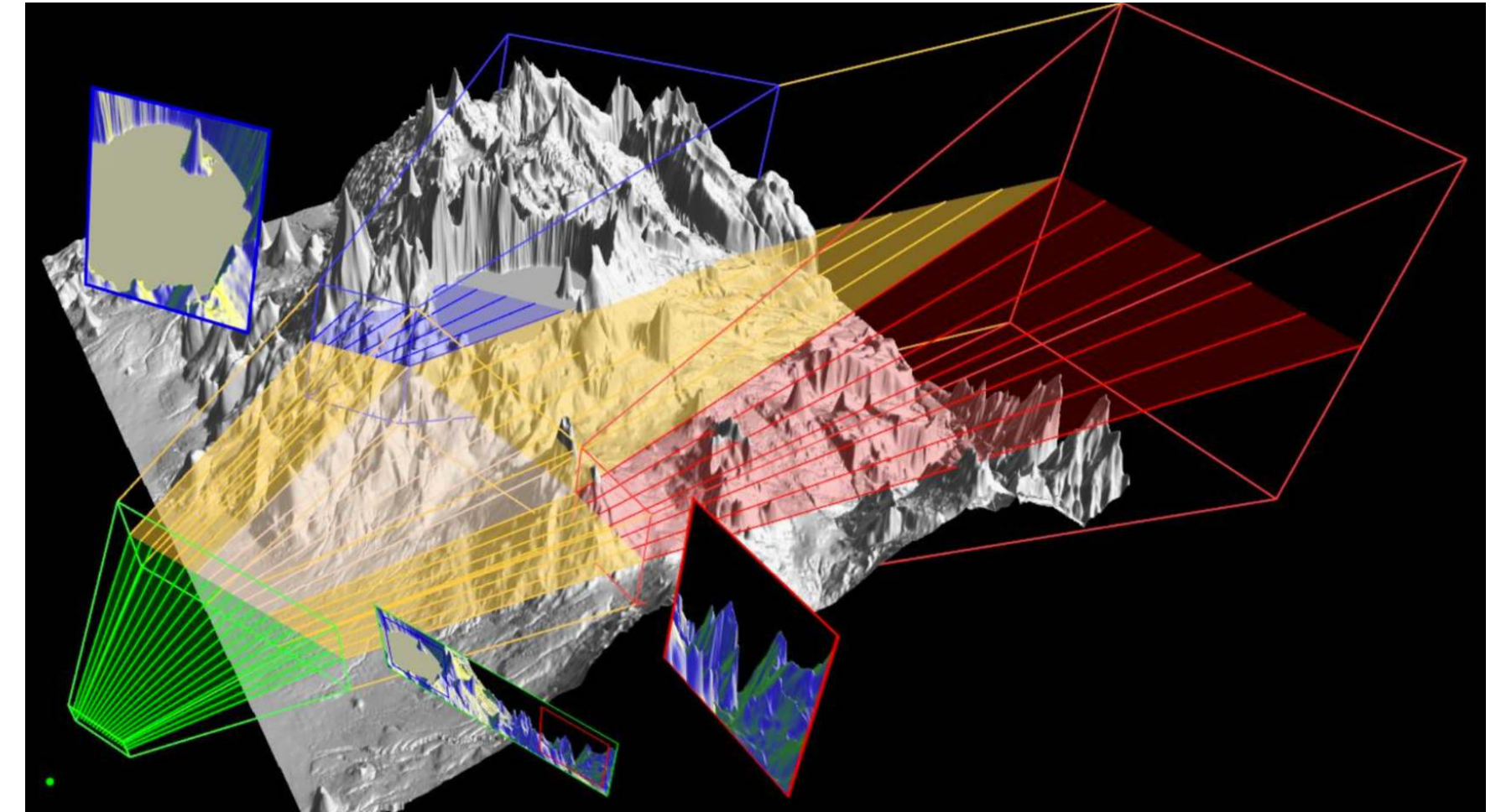
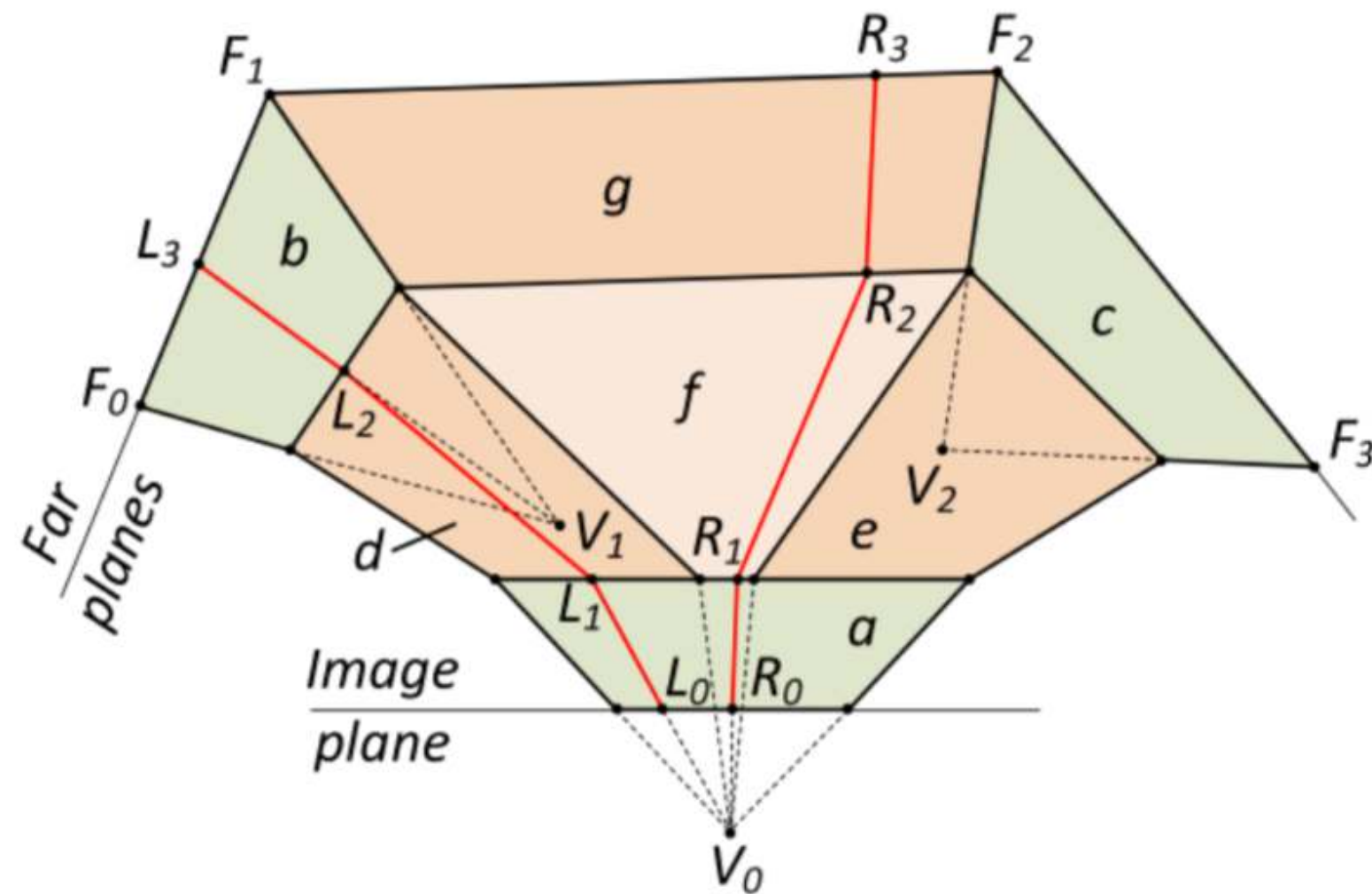
- Beispiel: Occlusion wichtiger Teile der Szene



Roter und gelber Punkt sollen sichtbar sein

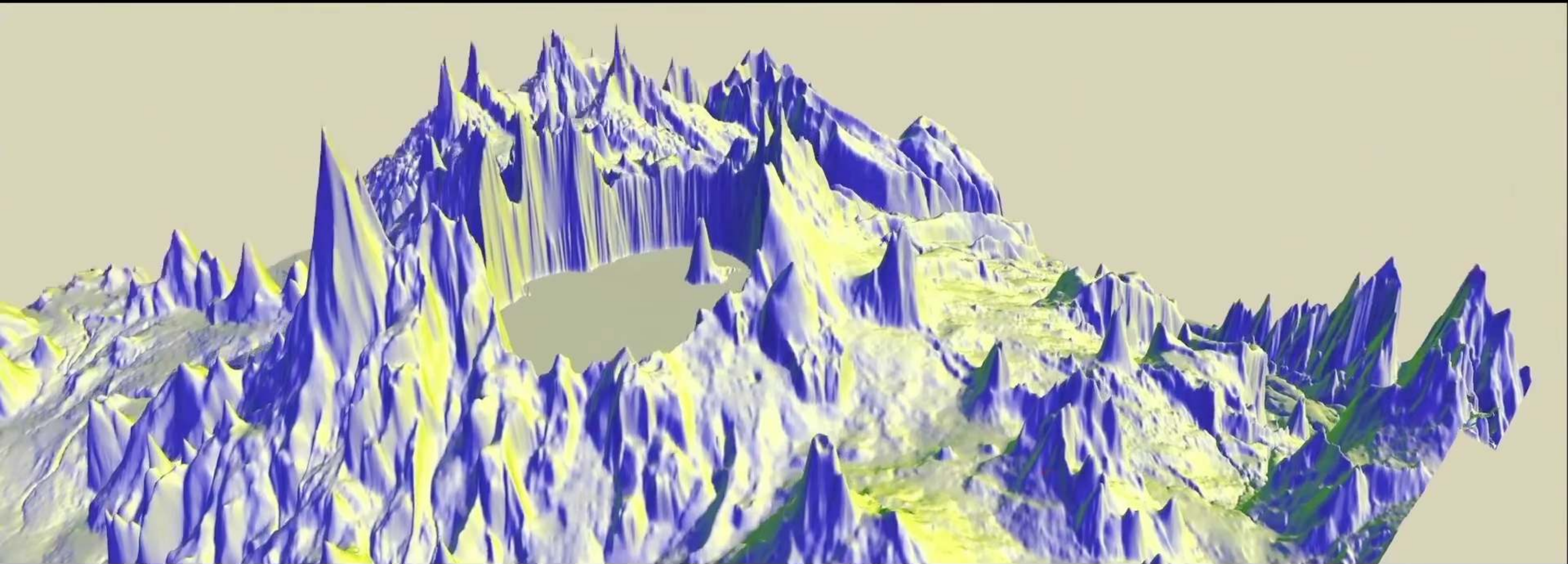
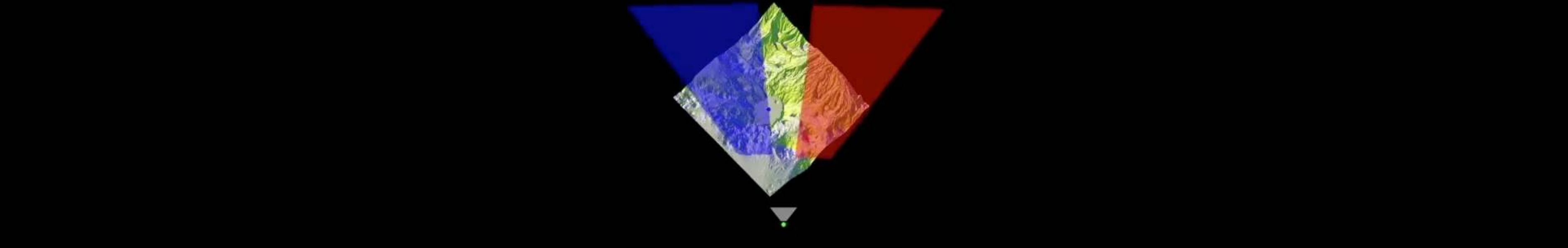


Bei normaler perspektivischer Projektion
wären diese verdeckt



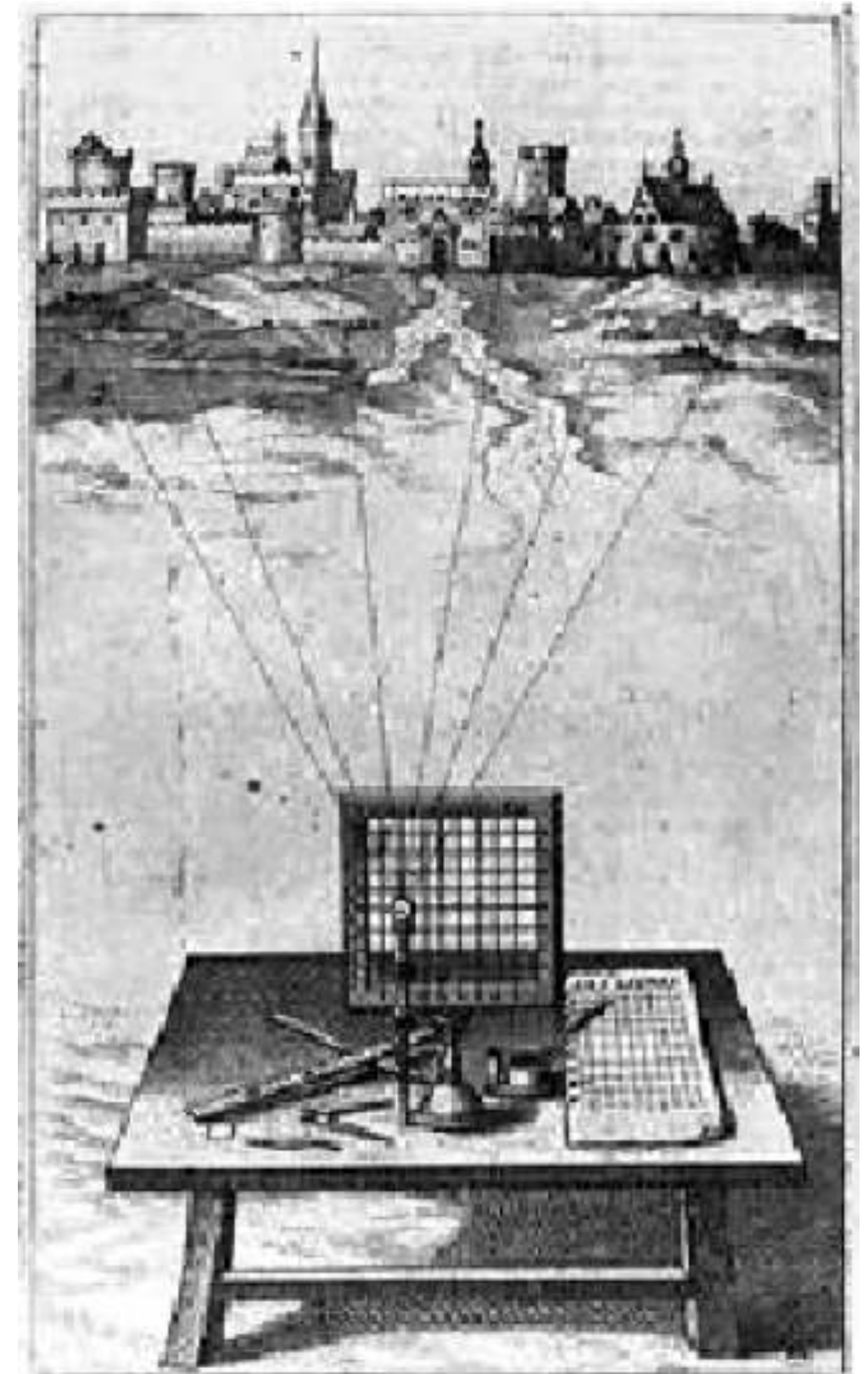
Die Projektion wird durch mehrere View-Frusta definiert. Jedem Pixel wird ein Strahl zugeordnet, der mehrfach "gebrochen" werden kann durch die verschiedenen View-Frusta.

Visualisierung mehrerer dieser Strahlen



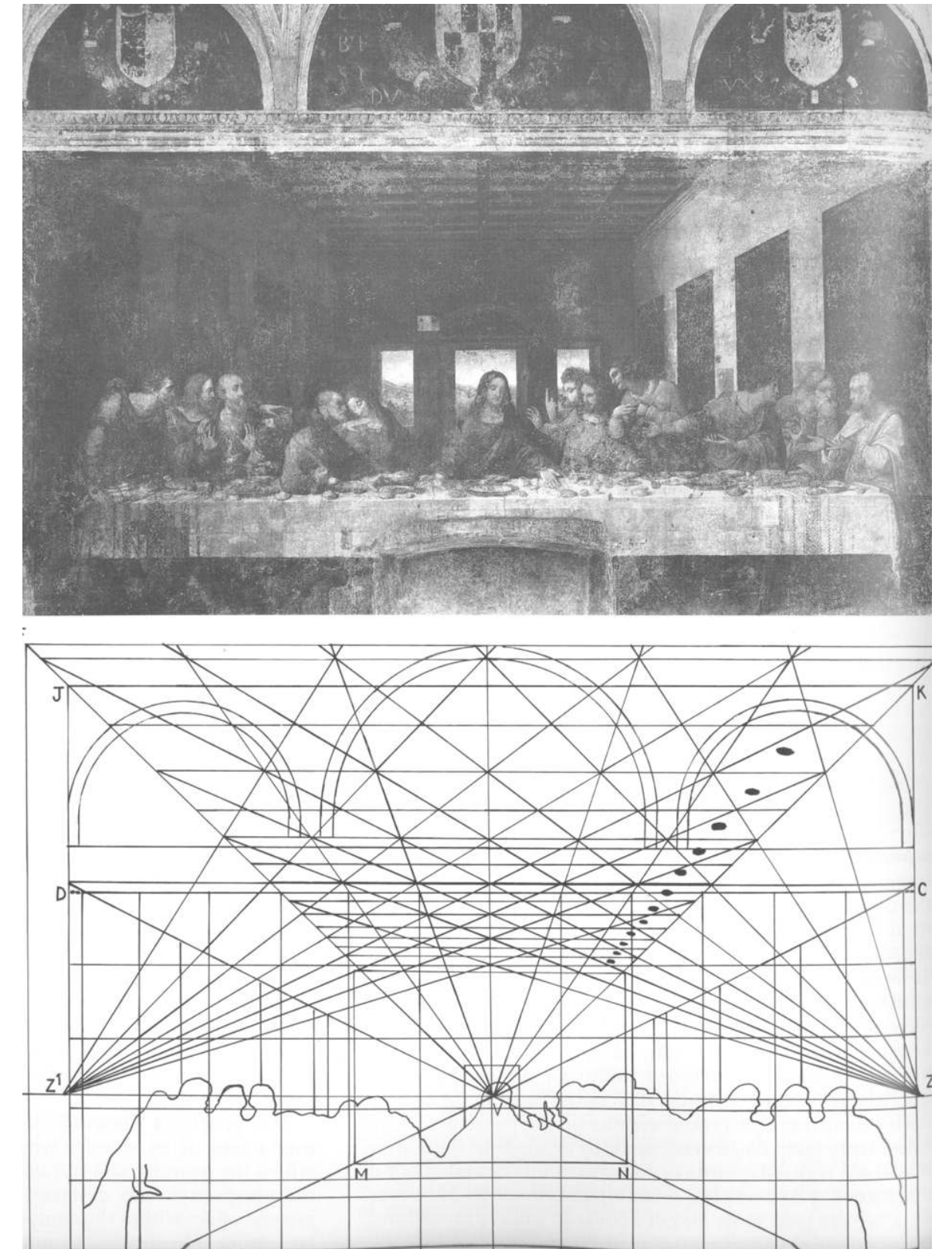
Noch einige Beispiele aus der Kunst

- 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*

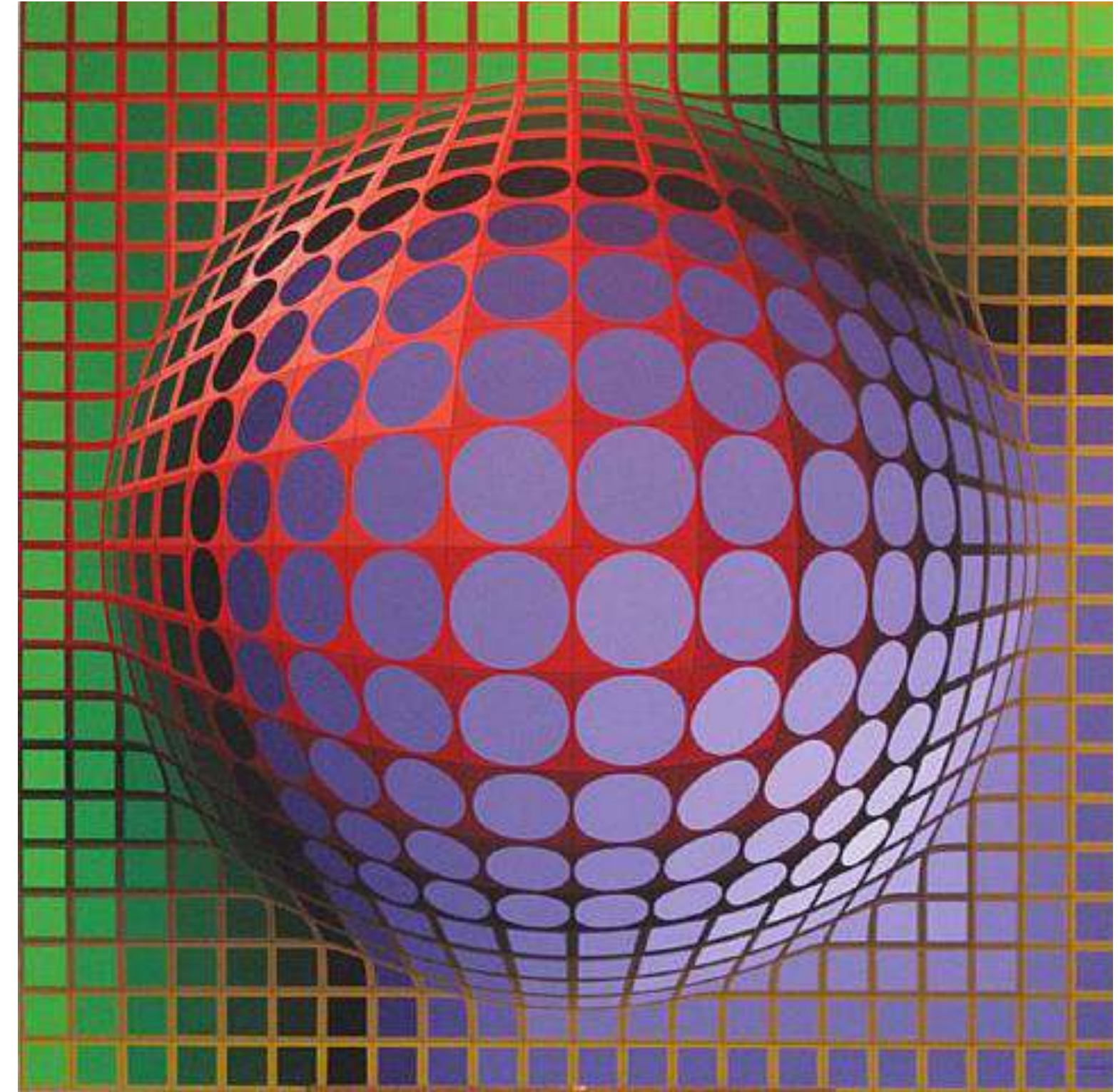
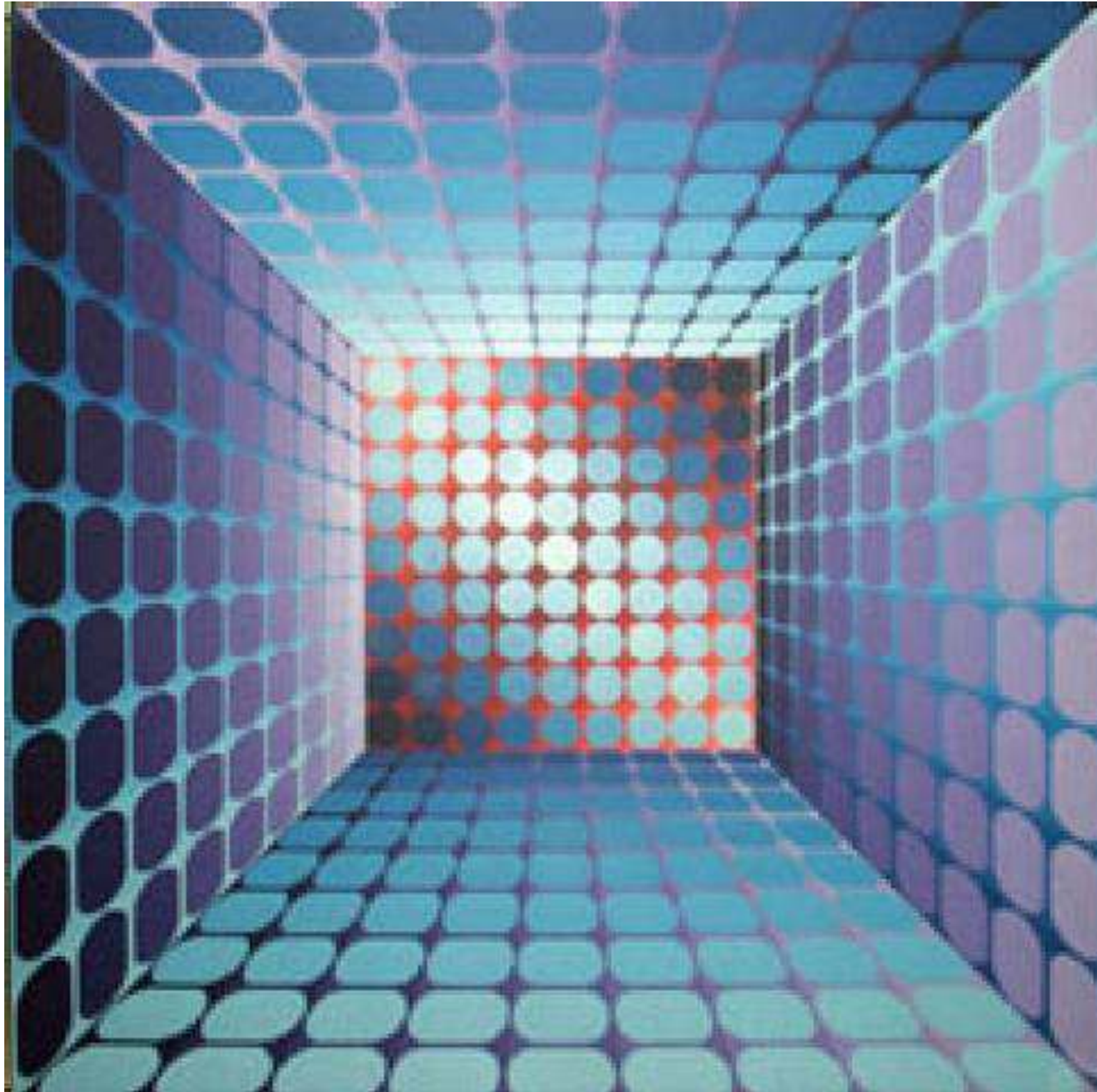


Multi-Perspektive, um den mystischen Eindruck zu erhöhen

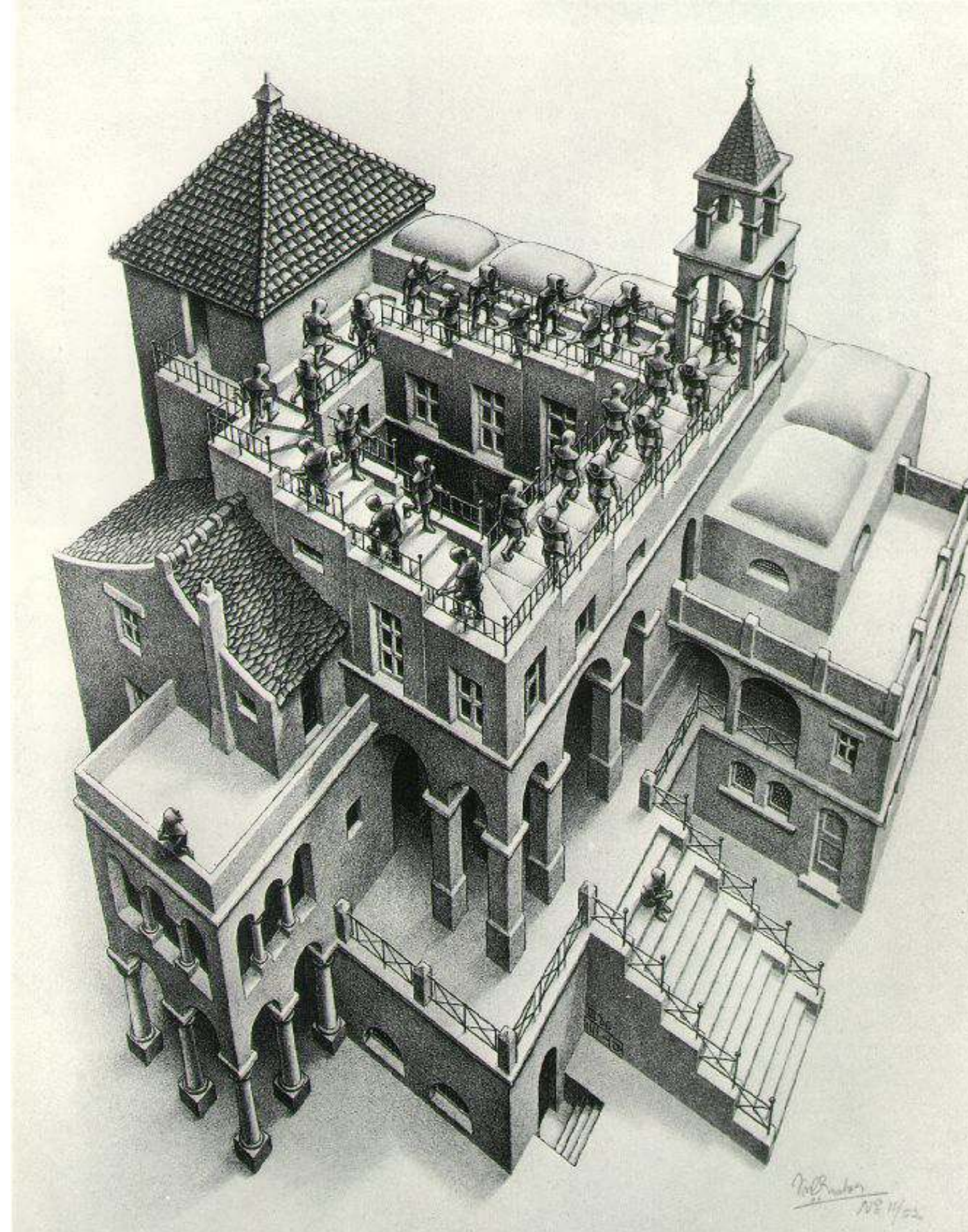


De Chirico

Viktor Vasarely: Perspektive in der abstrakten Kunst

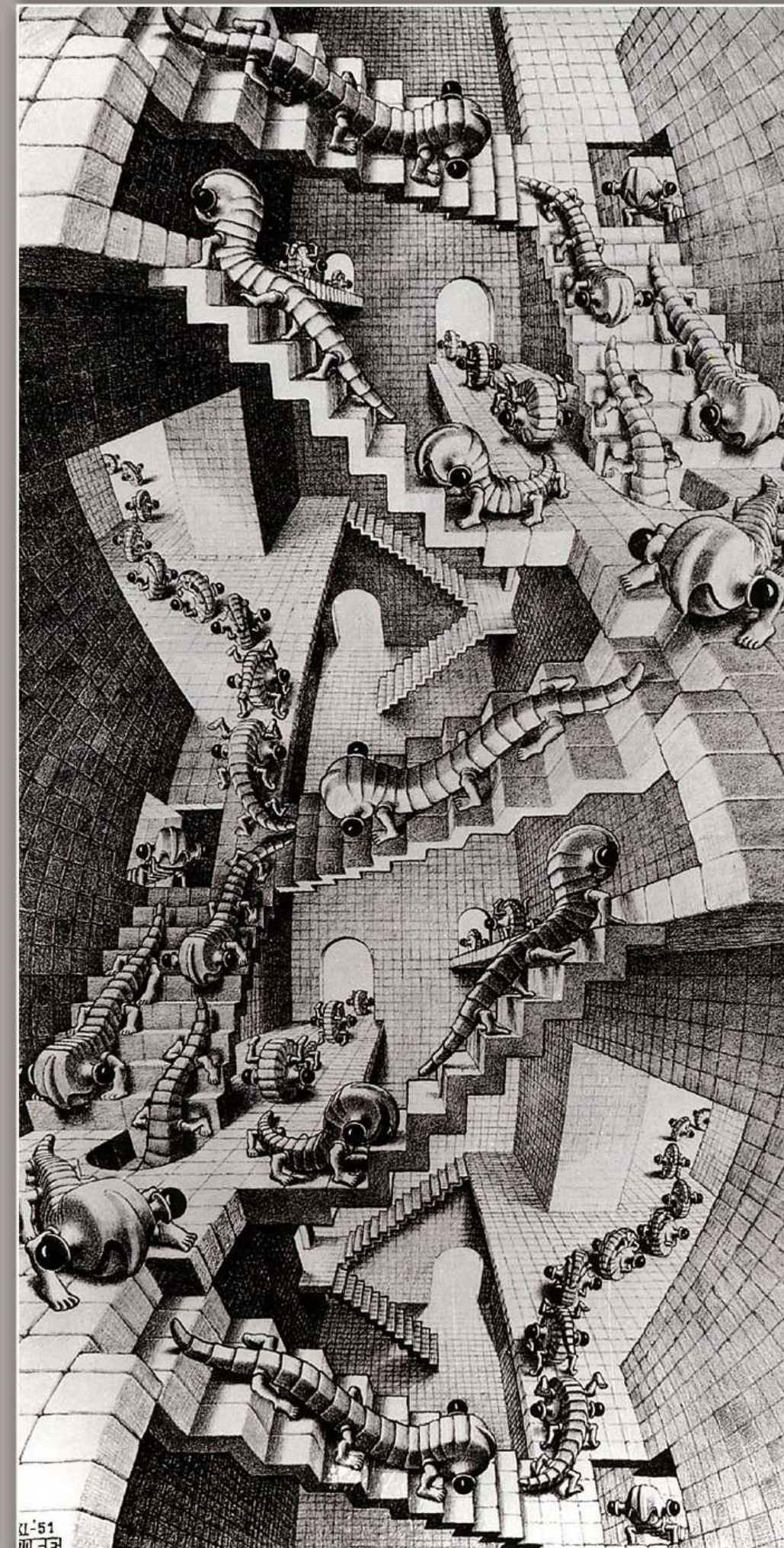


Einsatz der (korrekten) Perspektive zur Irritation des Betrachters



Maurits Cornelis Escher:
Ascending and Descending
1960, Lithograph

Nicht-lineare Perspektive



51
1951
Les Escaliers

M. C. Escher 1951

M. C. Escher

calvin and Hobbes

WATKINSON

