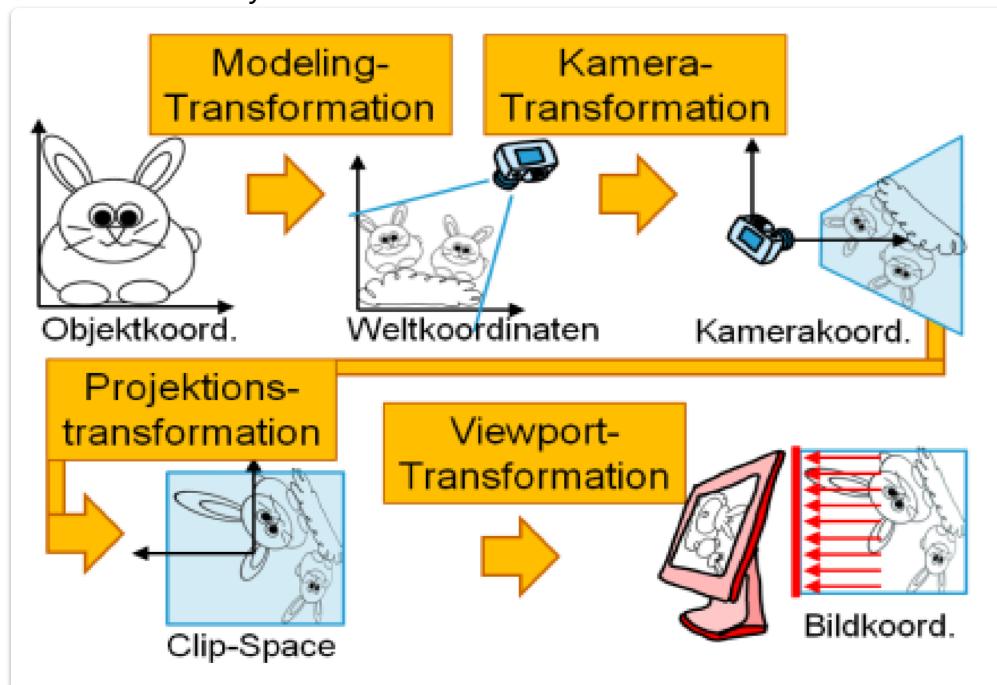


6. Viewing

Viewing in der Graphik-Pipeline

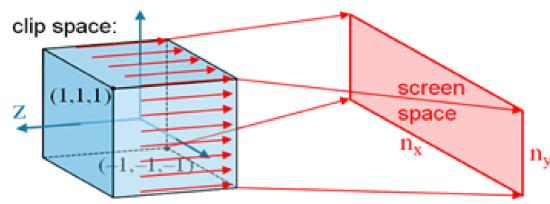
- Szenenmodellierung erfolgt in **Weltkoordinaten**
- Objekte werden aus **lokalen Koordinatensystemen** über **geometrische Transformationen (Matrizen)** in Weltkoordinaten überführt
- Nach Festlegung der **Kameraparameter**:
 - Transformation der Koordinaten in **Kamerakoordinaten**
 - Anschließende **Projektion** der Kamerakoordinaten
- Ergebnis der Projektion liegt in einem **normierten Würfel** (meist mit Seitenlänge 2)
- Danach erfolgt die **Viewport-Transformation**:
 - Überführung der normierten Koordinaten in **Gerätekoordinaten** des Ausgabemediums
- In der Geometrie existieren verschiedene Projektionen
- In der **Computergraphik** sind hauptsächlich zwei Projektionen relevant:
 - **Parallelprojektion**
 - **Perspektivische Projektion**
- Zuerst wird die **Parallelprojektion** angenommen
- Danach wird die **Integration der perspektivischen Projektion** in die Pipeline betrachtet
- Vorgehensweise erfolgt **von hinten nach vorne** (vom Endergebnis zurück), da dies einfacher zu analysieren ist



Viewport-Transformation

Das ist das was eben vom Clip-Space in die Bildkoordinaten des Ausgabegeräts umwandelt

Wir nehmen also an, dass die Szene bereits im Clip-Space vorhanden ist, d.h. alle relevanten Koordinaten befinden sich in einem achsenparallelen Würfel der Seitenlänge 2 mit Mittelpunkt (0,0,0). Wir wollen eine orthographische Abbildung (parallele Normalprojektion) mit Blickrichtung -z auf einen Bildschirm mit Abmessungen $n_x \times n_y$ (Pixel) durchführen. Es müssen also alle Punkte $(-1, -1, z)$ auf $(0, 0)$ abgebildet werden und alle Punkte $(1, 1, z)$ auf (n_x, n_y) . Diese lineare Abbildung wird durch die Matrix Mvp bewerkstelligt:

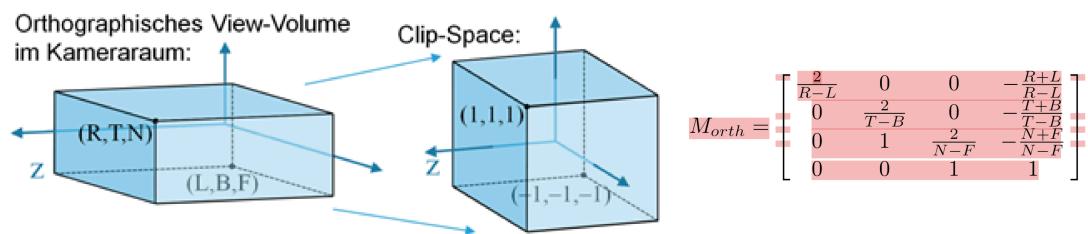


$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Deren Richtigkeit kann sofort nachgewiesen werden, indem man die Eckpunkte einsetzt. Die Matrix weist aber in den roten Zahlen noch eine Besonderheit auf: die z-Werte werden erhalten! Dies ist im Moment ohne Belang, wird aber bei späteren Schritten (vor allem bei der Berechnung der Sichtbarkeit) noch von großem Wert sein.

Projektionstransformation:

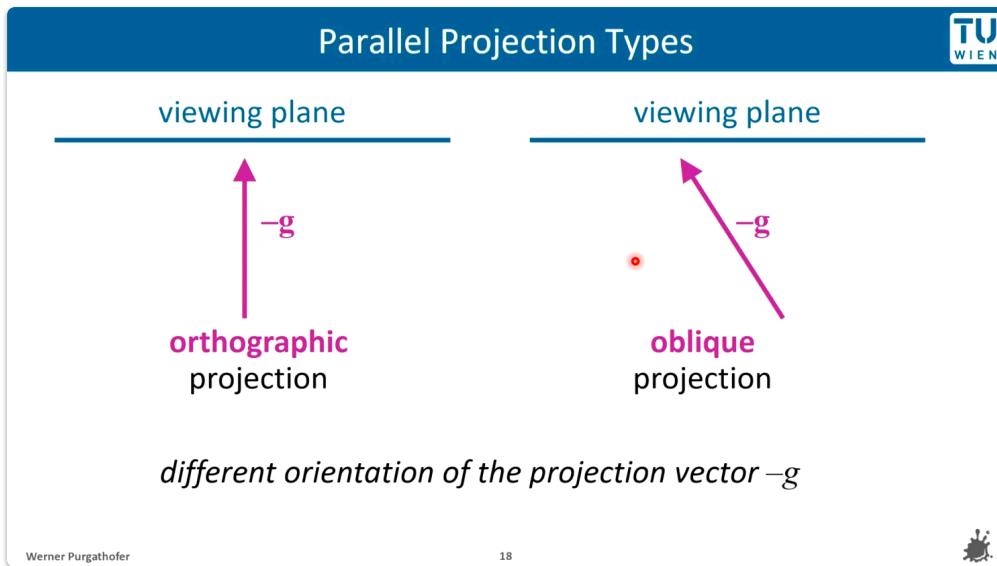
- Annahme: Orthographische Projektion
- Ziel: Vereinfachte Transformation durch achsenparallelen Quader
- Gegeben: Quader mit Grenzen:
 - L (Left)
 - R (Right)
 - B (Bottom)
 - T (Top)
 - N (Near)
 - F (Far)
- Transformation dieses Quaders in den normierten Würfel $[-1, 1]^3$
- Dabei gilt:
 - Punkt $(L, B, F) \rightarrow (-1, -1, -1)$
 - Punkt $(R, T, N) \rightarrow (1, 1, 1)$
- Umsetzung erfolgt über eine Transformationsmatrix



Bemerkung: Eine Parallelprojektion kann auch schräg auf eine Abbildungsebene erfolgen (zum Beispiel beim Schattenwurf), diese Variante wird hier nicht berücksichtigt.

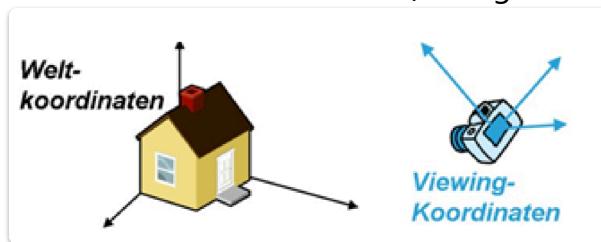
Warum ist das wichtig?

- **Anzeige auf 2D-Bildschirmen:** Unsere Monitore und Bildschirme sind zweidimensional. Um 3D-Grafiken darauf darstellen zu können, müssen die 3D-Objekte in 2D projiziert werden.



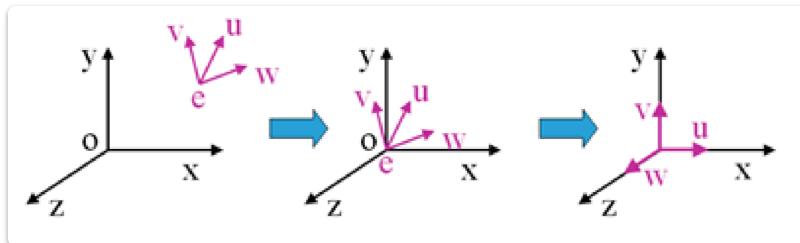
Kamera Transformation

- Beim Festlegen der Kamerawerte bestehen mehrere **Freiheitsgrade**:
 1. **Position der Kamera** im Raum
 2. **Blickrichtung** von dieser Position aus
 3. **Orientierung** der Kamera (Definition von „oben“)
 4. **Größe des Bildausschnittes** (analog zur Brennweite oder Zoomfaktor)



- Aus den ersten drei Werten ergibt sich das **Viewing-Koordinatensystem** mit den Achsen u, v, w
- Eigenschaften des Viewing-Koordinatensystems:
 - Die **uv-Ebene** ist **normal** zur Hauptblickrichtung
 - Die Blicke Richtung verläuft entlang der **negativen w-Achse**
- In **Animationen** wird die Kameradefinition oft automatisch aus Bedingungen berechnet:
 - z.B. Kamerafahrt um ein Objekt
 - z.B. Kamerasteuerung in Flugsimulationen
 - Ziel: **Unkomplizierte Erzeugung gewünschter Effekte**
- Umwandlung von **Weltkoordinaten** in **Viewingkoordinaten** erfolgt durch eine Kette von **einfachen Transformationen**:

- **Translation**, um Koordinatenursprünge aufeinander abzustimmen
- **Drei Rotationen**, um Koordinatenachsen zur Deckung zu bringen:
 - Zwei Drehungen zur Ausrichtung der ersten Achse
 - Eine weitere Drehung für die zweite Achse
 - Die dritte Achse ergibt sich automatisch korrekt
- Zusammensetzung dieser Transformationen in eine **Transformationsmatrix**:
 $M_{WC \rightarrow VC} = R_z \cdot R_y \cdot R_x \cdot T$
- Zur vollständigen **Projektionsbeschreibung** werden zusätzlich die **Grenzen des darzustellenden Bereichs** benötigt
- Mehr zu Transformations findet man hier: [3. Transformationen](#)



Ausgehend von der Kameraposition geht man prinzipiell folgendermaßen vor, um das Viewing-Koordinatensystem festzulegen:

1. Wahl einer Kameraposition (auch Augpunkt oder **Viewing-Punkt** genannt).
2. Wahl einer Blickrichtung, die **negative** Blickrichtung ergibt die **w-Achse**.
3. Wahl einer Richtung **t „nach oben“**; aus dieser lassen sich dann die **u- und v-Achsen** berechnen.
4. Da die **Abbildungsebene** normal auf die **Blickrichtung** liegt, ergibt das **Vektorprodukt $t \times w$** die **Richtung der u-Achse**.
5. Berechnung der **v-Achse** als **Vektorprodukt** der **w- und u-Achsen**: $v = w \times u$.
6. Die **Wahl von minimalen und maximalen u-, v- und w-Werten** zur Eingrenzung des Ausschnittes der Szene, der abgebildet wird: L(eft), R(ight), B(ottom), T(op), N(ear), F(ar).

Hier nochmal die Formeln:

e ... eye position

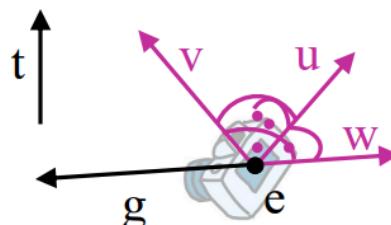
g ... gaze direction (**positive w-axis points to the viewer**)

t ... view-up vector

$$w = -\frac{g}{|g|}$$

$$u = \frac{t \times w}{|t \times w|}$$

$$v = w \times u$$



Orthographisches Viewing

Für die orthographische Projektion (Kamera bildet parallel ab) haben wir nun alle Schritte durch Matrizen beschrieben, diese können wir wie bei den geometrischen Transformationen zu einer einzigen Matrix zusammensetzen (multiplizieren), die dann die gesamte Viewing-Transformation durchführt:

$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ z \\ 1 \end{bmatrix} = (\mathbf{M}_{\text{vp}} * \mathbf{M}_{\text{orth}} * \mathbf{M}_{\text{cam}}) * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

formieren!

Man beachte: die rechteste Matrix wird zuerst mit dem Punkt (x,y,z) multipliziert. Wenn man das Assoziativgesetz anwendet, dann kann man aber auch zuerst die 3 Matrizen miteinander multiplizieren, und kann damit alle Punkte nur mit dieser einen Ergebnismatrix direkt von Weltkoordinaten in Gerätekordinaten transformieren!

Viewing: Camera + Projection + Viewport



$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ z \\ 1 \end{bmatrix} = \mathbf{M}_{\text{vp}} \cdot \mathbf{M}_{\text{orth}} \cdot \mathbf{M}_{\text{cam}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

projection transformation
 camera transformation
 viewport transformation

pixels on
the screen

world coordinates

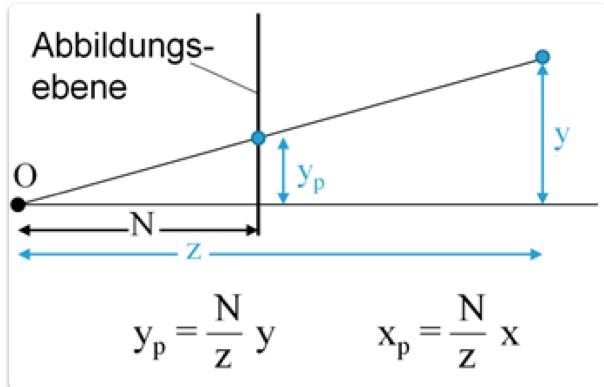
Perspektive

- Perspektivische Projektion ist **keine affine Transformation**:
 - Affine Eigenschaften wie Parallelität bleiben **nicht** erhalten
 - Kann **nicht** mit einer 3×3 -Matrix dargestellt werden
- Lösung: Verwendung von **homogenen Koordinaten**
 - Einziger Fall, in dem die **homogene Komponente $h \neq 1$**
 - Erfordert einen **Divisionsschritt** am Ende der Transformation: Koordinaten werden durch h geteilt
- Grundlagen der perspektivischen Transformation:
 - O : **Projektionszentrum**
 - **Blickrichtung**: entlang der **negativen z -Achse**
 - **Abbildungsebene**: normal zur z -Achse im Abstand N (Near)
- Abbildung eines Punktes (x, y, z) auf die Ebene:

$$(x, y, z) \rightarrow \left(\frac{x \cdot N}{z}, \frac{y \cdot N}{z}, N \right)$$

- Das lässt sich durch eine Matrix P darstellen:

$$P = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 1 & N + F & -F * N \\ 0 & 0 & N & 1 \end{bmatrix}$$



- Ein Punkt $(x, y, z, 1)$ wird mit der **Projektionsmatrix P** multipliziert
- Ergebnis der Multiplikation: $(x \cdot N, y \cdot N, z \cdot (N+F) - F \cdot N, z)$
- Danach erfolgt das **Homogenisieren** (Normalisieren): Division durch die letzte Komponente z
- Ergebnis nach Division:

$$\left(\frac{x \cdot N}{z}, \frac{y \cdot N}{z}, (N + F) - \frac{F \cdot N}{z}, 1 \right)$$

Hier der Inhalt der Folien zu dem:

Perspective Transformation

TU
WIEN

view plane

O

f

z

y_p

y

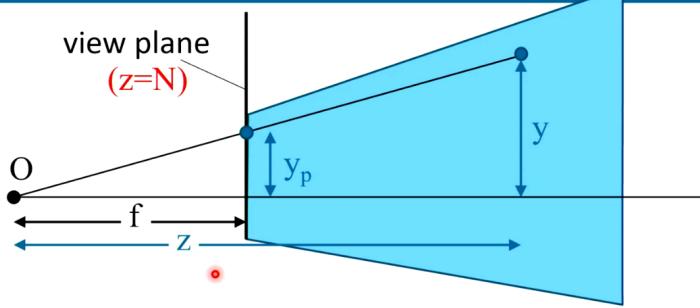
$\frac{y_p}{y} = \frac{f}{z}$

derivation of perspective transformation

$f \dots$ focal length

$y_p = \frac{f}{z} y$

Perspective Transformation



*derivation of
perspective
transformation*

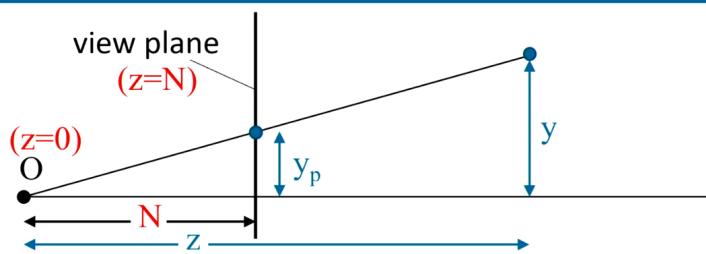
$$y_p = \frac{f}{z} y$$

Werner Purgathofer

38



Perspective Transformation



*derivation of
perspective
transformation*

$$\boxed{x_p = \frac{N}{z} x}$$

analogous:

$$\boxed{y_p = \frac{N}{z} y}$$

$$\begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & N+F & -F \cdot N \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Werner Purgathofer

38



Perspective Transformation

$$\mathbf{P} = \begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & N+F & -F \cdot N \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot N \\ y \cdot N \\ z \cdot (N+F) - F \cdot N \\ z \end{bmatrix}$$

*derivation of
perspective
transformation*

homogenization: divide by z

$$\boxed{x_p = \frac{N}{z} x}$$

$$\boxed{y_p = \frac{N}{z} y}$$

$$\leadsto \begin{bmatrix} x \cdot N/z \\ y \cdot N/z \\ (N+F) - F \cdot N/z \\ 1 \end{bmatrix}$$

Werner Purgathofer

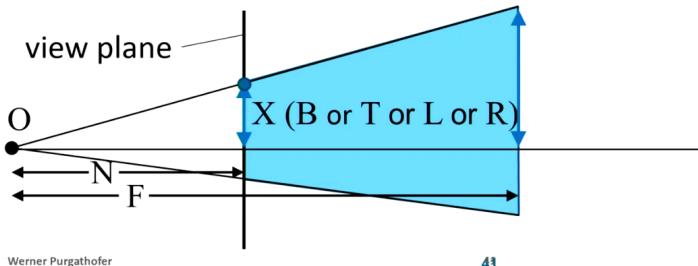
39



mit homogenization ist gemeint, dass man alles durch z dividiert, sodass die z -Koordinate = 1 ist.

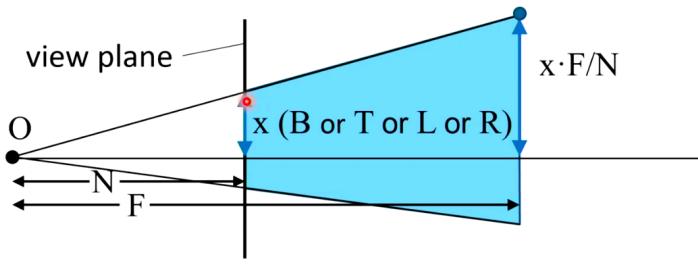
Example: (Right) Top Near Corner

$$\begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & N+F & -F \cdot N \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R \\ T \\ N \\ 1 \end{bmatrix} = \begin{bmatrix} R \cdot N \\ T \cdot N \\ N \cdot (N+F) - F \cdot N \\ N \end{bmatrix} \rightsquigarrow \begin{bmatrix} R \\ T \\ N \\ 1 \end{bmatrix}$$



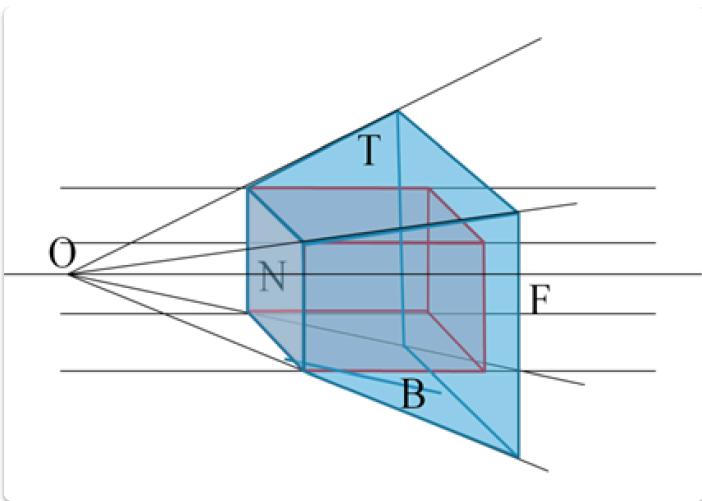
Example: (Left) Top Far Corner

$$\begin{bmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & N+F & -F \cdot N \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} L \cdot F/N \\ T \cdot F/N \\ F \\ 1 \end{bmatrix} = \begin{bmatrix} L \cdot F \\ T \cdot F \\ F \cdot (N+F) - F \cdot N \\ F \end{bmatrix} \rightsquigarrow \begin{bmatrix} L \\ T \\ F \\ 1 \end{bmatrix}$$



- Die perspektivische Projektion führt zu einer **Verzerrung des Szenebereichs**:
 - Dieser Bereich wird als „**View Frustum**“ bezeichnet
 - Das View Frustum ist ein **Pyramidenstumpf**, der auf einen **achsparallelen Quader** abgebildet wird
 - In diesem Quader liefert die **orthographische Projektion** dasselbe Bild wie die **perspektivische Projektion** im View Frustum
- Nach der Verzerrung kann die bereits erarbeitete **Parallelprojektion** angewendet werden, um die perspektivische Matrix M_{per} zu berechnen
- Alternative Methode:
 - **Einsetzen** der Projektionsmatrix P an der richtigen Stelle in der Gesamtviewingberechnung
 - Dadurch wird eine **Gesamtmatrix** erzeugt, die die Transformation von Modellkoordinaten (x, y, z) zu Gerätekordinaten (x_{screen}, y_{screen}) in einem Schritt ausführt

$$\begin{bmatrix} x_{screen} \\ y_{screen} \\ z \\ 1 \end{bmatrix} = M_{vp} * \overbrace{(M_{orth} * P * M_{cam} * M_{mod})}^{M_{per}} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Clip Space vs. NDC Space

TU
WIEN

- Clip Space: $[-w, w] \times [-w, w] \times [-w, w]$, $w > 0$
- Normalized Device Coordinates (NDC): $[-1,1] \times [-1,1] \times [-1,1]$

Clip Space	NDC Space
Constraints $x_{\min} = -w$ $x_{\max} = w$ $y_{\min} = -w$ $y_{\max} = w$ $z_{\min} = -w$ $z_{\max} = w$ $w > 0$	$(x_{\min}/w, y_{\max}/w, z_{\max}/w)$ $(x_{\max}/w, y_{\max}/w, z_{\max}/w)$ $(x_{\min}/w, y_{\max}/w, z_{\min}/w)$ $(x_{\max}/w, y_{\max}/w, z_{\min}/w)$ $(x_{\min}/w, y_{\min}/w, z_{\max}/w)$ $(x_{\max}/w, y_{\min}/w, z_{\max}/w)$ $(x_{\min}/w, y_{\min}/w, z_{\min}/w)$ $(x_{\max}/w, y_{\min}/w, z_{\min}/w)$
Pre-perspective divide puts the region surviving clipping within $-w \leq x \leq w, -w \leq y \leq w, -w \leq z \leq w$	Post-perspective divide puts the region surviving clipping within the $[-1,+1]^3$
40	Image Source: Mark Kilgard, University of Texas, CS 354 Graphics Math (2012)

- **Homogenisierung:**

- Wenn eine **perspektivische Abbildung** beteiligt ist, muss das Ergebnis am Ende durch die homogene Komponente z' dividiert werden.
- Im **Clipraum** wird nicht nur das **Clipping** durchgeführt, sondern auch die **Homogenisierung**.
- Nach der Homogenisierung wird die **Viewport-Matrix** als letzter Schritt angewendet.

- **Wichtige Eigenschaften der Projektionstransformation:**

1. **Gerade Strecken bleiben gerade Strecken:**

- Um eine Strecke (z.B. Seite eines Polygons) abzubilden, reicht es, nur die beiden Endpunkte zu transformieren.

2. **Relative Ordnung der z-Werte bleibt erhalten:**

- Der Abstand der Punkte von der Kamera bleibt relativ zueinander erhalten, jedoch nicht die **Abstandswerte selbst**.
- Diese Eigenschaft ist wichtig für die **Sichtbarkeitsberechnung**:

$$z_1, z_2, N, F < 0$$

$$z_1 < z_2$$

$$\frac{1}{z_1} > \frac{1}{z_2}$$

$$| * (-F * N)(< 0)$$

$$-F * \frac{N}{z_1} < -F * \frac{N}{z_2}$$

$$| + (N + F)$$

$$(N + F) - F * \frac{N}{z_1} < (N + F) - F * \frac{N}{z_2}$$

Fluchtpunkte:

Anzahl der Hauptfluchtpunkte hängt von der Lage der **Bildebene** zum **Koordinatensystem** ab:

1. **Einpunktperspektive**: Zwei Achsen sind parallel zur Bildebene.
2. **Zweipunktperspektive**: Eine Achse ist parallel zur Bildebene.
3. **Dreipunktperspektive**: Keine Achse ist parallel zur Bildebene, es gibt **3 Hauptfluchtpunkte**.

