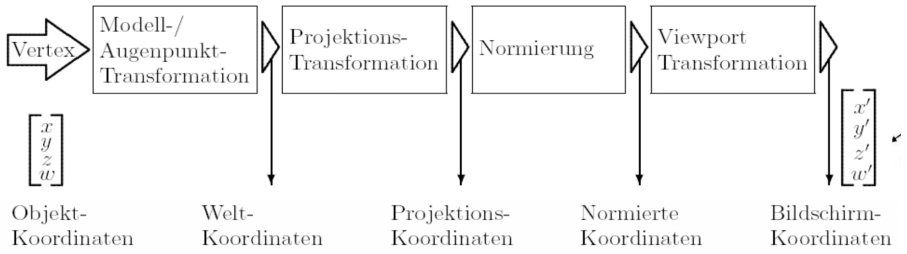
1 Transformationen

In der Computergrafik durchlaufen 3D-Daten verschiedene Stufen (die Grafikpipeline), um schließlich als 2D-Bild auf dem Bildschirm zu erscheinen. Dabei spielen **Koordinatensysteme** und **Transformationen** eine zentrale Rolle.

1.1 Koordinatensysteme in der Pipeline

Um ein Bild zu erzeugen, müssen Objekte durch verschiedene Koordinatenräume transformiert werden. Die Reihenfolge ist essenziell:

1. **Objektkoordinaten (Object Coordinates)**: Lokales Koordinatensystem eines einzelnen 3D-Modells (z.B. Mittelpunkt einer Kugel im Ursprung).
2. **Weltkoordinaten (World Coordinates)**: Beschreiben die gesamte Szene. Alle Objekte werden hier relativ zueinander platziert.
3. **Augenkoordinaten (Eye/Camera Coordinates)**: Die Szene aus der Sicht der Kamera. Die Kamera liegt im Ursprung, Blickrichtung oft entlang der negativen Z-Achse.
4. **Projektionskoordinaten (Clip Coordinates)**: Nach Anwendung der Projektion (perspektivisch oder parallel). Hier findet das *Clipping* statt.
5. **Normierte Gerätekoordinaten (Normalized Device Coordinates – NDC)**: Koordinatenbereich meist von $[-1, 1]$ oder $[0, 1]$.
6. **Bildschirmkoordinaten (Window Coordinates)**: Tatsächliche Pixelpositionen im Ausgabefenster.



1.1.1 Transformationstypen

- **Modelling Transformations:** Anordnung von Objekten in der Welt (Rotation, Translation, Skalierung).
- **Viewing Transformations:** Positionierung und Ausrichtung der Kamera.
- **Projection Transformations:** Definition des Sichtvolumens (Viewing Volume) und Projektion auf 2D.
- **Viewport Transformations:** Mapping auf die Pixelauflösung des Bildschirms.

1.2 Mathematische Grundlagen: Homogene Koordinaten

Um Transformationen effizient zu berechnen, nutzen wir Matrizen. Ein Problem im herkömmlichen euklidischen Raum (R^3) ist, dass lineare Abbildungen (Rotation, Skalierung) den Ursprung fix lassen müssen. Eine **Translation** (Verschiebung) ist jedoch eine affine Abbildung, keine lineare, und kann nicht durch eine 3×3-Matrix dargestellt werden.

Lösung: Homogene Koordinaten

Wir erweitern den n-dimensionalen Raum auf $n+1$ Dimensionen. Ein Punkt $(x, y, z)^T$ im 3D-Raum wird zu $(x, y, z, w)^T$ im homogenen Raum (meist ist $w = 1$).
Die Rückumwandlung erfolgt durch Division durch w:

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

Vorteile:

- Einheitliche Darstellung aller affinen Transformationen (inkl. Translation) als 4×4-Matrizen.
- Hintereinanderausführung von Transformationen entspricht einfacher Matrixmultiplikation.
- Ermöglicht perspektivische Projektionen (durch Manipulation von w).

1.3 Affine Abbildungen

Eine **Affine Abbildung** setzt sich aus einem linearen Teil (Matrix A) und einer Translation (Vektor b) zusammen: $\Phi(v) = A \cdot v + b$.

Eigenschaften affiner Abbildungen

1. Geraden werden auf Geraden abgebildet.
2. Parallelität von Linien und Ebenen bleibt erhalten.
3. Verhältnisse von Längen, Flächen und Volumina bleiben erhalten (relative Abstände).

In homogenen Koordinaten sieht die allgemeine Matrixform so aus:

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Der 3×3-Block oben links (a_{ij}) steuert Rotation, Skalierung und Scherung. Die letzte Spalte (t) steuert die Translation.

1.3.1 Translation (Verschiebung)

Verschiebung eines Punktes um den Vektor $d = (x_0, y_0, z_0)^T$. Der lineare Teil ist die Einheitsmatrix.

$$T(d) = \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1.3.2 Skalierung

Skalierung verändert die Größe entlang der Achsen. Sie wird durch eine Diagonalmatrix beschrieben. Mit den Faktoren s_1, s_2, s_3 für die x-, y- und z-Achse:

$$S(s_1, s_2, s_3) = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ist $s_1 = s_2 = s_3 = s$, spricht man von einer *uniformen Skalierung*.

1.3.3 Scherung (Shearing)

Verzerrt das Objekt, indem Koordinaten in Abhängigkeit von anderen Koordinaten verschoben werden. Beispiel einer Scherung entlang der x-Achse in Abhängigkeit von y:

$$SH_{xy} = \begin{pmatrix} 1 & s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1.3.4 Rotation

Wir verwenden ein **Rechtssystem** (Rechte-Hand-Regel). Positive Drehwinkel drehen gegen den Uhrzeigersinn, wenn man entlang der positiven Rotationsachse auf den Ursprung blickt.

Rotation um die Z-Achse (Winkel α):

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um die X-Achse (Winkel α):

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um die Y-Achse (Winkel α):

$$R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hinweis: Bei der Y-Rotation scheinen die Vorzeichen des Sinus im Vergleich zu X und Z vertauscht. Das liegt an der zyklischen Permutation der Achsen ($x \to y \to z \to x$).

1.4 Komplexe Transformationen & Komposition

1.4.1 Matrix-Multiplikation und Reihenfolge

Die Matrix-Multiplikation ist **nicht kommutativ** ($A \cdot B \neq B \cdot A$). Das bedeutet: **Die Reihenfolge der Transformationen ist wichtig!** Eine Rotation gefolgt von einer Translation ist etwas anderes als eine Translation gefolgt von einer Rotation.

Mathematisch werden Transformationen von rechts nach links auf den Punktvektor p angewendet:

$$p' = M_n \cdot \cdots \cdot M_2 \cdot M_1 \cdot p$$

Hier wird zuerst M_1, dann M_2, usw. ausgeführt.

1.4.2 Rotation um beliebigen Punkt / Achse

Die Standard-Rotationsmatrizen drehen immer um den Ursprung. Um ein Objekt um einen beliebigen Punkt P_f (Fixed Point) zu drehen:

1. Verschiebe das Objekt so, dass P_f im Ursprung liegt ($T(-P_f)$).
2. Rotiere um den Ursprung (R).
3. Verschiebe das Objekt zurück ($T(P_f)$).

Gesamtmatrix: $M = T(P_f) \cdot R \cdot T(-P_f)$.

Rotation um eine beliebige Achse r (durch den Ursprung): Man konstruiert eine orthonormale Basis (r, s, t), wobei r die normierte Rotationsachse ist.

1. Bilde eine Basiswechselmatrix R_{basis}, die die Achse r auf eine Hauptachse (z.B. x-Achse) abbildet. Dies geschieht, indem man die Basisvektoren in die Zeilen der Matrix schreibt.
2. Rotiere um diese Hauptachse (R_α^x).
3. Wende die inverse Basiswechselmatrix an ($R_{basis}^{-1} = R_{basis}^T$).

Dies ist oft effizienter als das Ausrechnen über Euler-Winkel.

1.5 Projektionen

Projektionen bilden den 3D-Raum auf eine 2D-Ebene ab. Dabei gehen Informationen (Tiefe) verloren.

1.5.1 Eigenschaften projektiver Abbildungen

Projektive Abbildungen sind eine allgemeinere Klasse von Transformationen als die affinen Abbildungen. Sie können ebenfalls durch homogene 4×4-Matrizen beschrieben werden, jedoch ist die letzte Zeile der Matrix nicht mehr zwingend $(0, 0, 0, 1)$.

Eigenschaften im Vergleich

Im Gegensatz zu affinen Abbildungen gelten bei projektiven Abbildungen folgende Besonderheiten:

- **Geradentreue:** Geraden werden weiterhin auf Geraden abgebildet.
- **Schnittpunkte:** Schnitte von Geraden bleiben erhalten.
- **Reihenfolge:** Die Reihenfolge von Punkten auf projektiven Geraden bleibt erhalten.
- **Parallelität geht verloren:** Parallele Geraden schneiden sich nach der Transformation oft in sogenannten **Fluchtpunkten**.
- **Winkel und Längen:** Winkel werden verändert und Längenverhältnisse bleiben *nicht* erhalten. Rechtecke werden zu allgemeinen Vierecken verzerrt.

Fluchtpunkte: Bei der perspektivischen Projektion treffen sich alle Geraden, die im 3D-Raum parallel sind (aber nicht parallel zur Bildebene liegen), in einem Punkt auf der Bildebene, dem Fluchtpunkt. Je nach Ausrichtung des Objekts zur Bildebene spricht man von einer 1-, 2- oder 3-Punkt-Perspektive.

1.5.2 Vergleich: Parallel vs. Perspektivisch

Parallelprojektion (Orthografisch)	**Perspektivische Projektion**
Projektionsstrahlen sind parallel.	Projektionsstrahlen treffen sich im Augpunkt (Center of Projection, COP).
Keine perspektivische Verkürzung (Größe unabhängig von Distanz).	Objekte, die weiter weg sind, erscheinen kleiner.
Winkel bleiben meist nicht erhalten, aber parallele Linien bleiben parallel.	Parallele Linien schneiden sich in Fluchtpunkten (außer sie sind parallel zur Bildebene).
Anwendung: Technische Zeichnungen, Medizin, Architektur (Grundriss).	Anwendung: Realistische Darstellung, menschliches Sehen.

1.5.3 Perspektivische Transformation

Mathematisch wird die perspektivische Verkürzung durch den Strahlensatz hergeleitet. Ein Punkt (x, y, z) wird auf die Bildebene im Abstand d projiziert. Die projizierte Koordinate ist $y_p = y \cdot \frac{d}{d+z}$ (wobei hier oft d als Position der Bildebene oder Abstand zum Augpunkt definiert wird).

In homogenen Koordinaten wird dies erreicht, indem die w-Komponente verändert wird. Eine typische Projektionsmatrix schreibt $-z$ (oder einen Faktor davon) in die w-Komponente.

Beispielmatrix für perspektivische Transformation (Zentrum im Ursprung, Bildebene bei $z = -d$):

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{pmatrix}$$

Das Ergebnisvektor ist $(x, y, z, -z/d)^T$. Nach der Homogenisierung (Division durch $w = -z/d$) erhalten wir die skalierten Koordinaten.

1.5.4 Kanonisches Sichtvolumen (Canonical View Volume)

In der Pipeline wird die komplexe Pyramide des Sichtvolumens (Frustum) durch die Projektionsmatrix in einen achsenparallelen Einheitswürfel (das kanonische Sichtvolumen) transformiert.

- **Parallelprojektion:** Das Sichtvolumen ist bereits ein Quader. Transformation ist Translation + Skalierung.
- **Perspektive:** Das Sichtvolumen ist ein Pyramidenstumpf. Die Matrix verformt diesen Stumpf in einen Würfel. Dadurch wird das spätere Clipping und die Rasterisierung vereinheitlicht.

1.6 3D-Interaktion

Ein fundamentales Problem bei der Interaktion mit 3D-Szenen ist die Nutzung von **2D-Eingabegeräten** (Maus, Touchscreen).

Das Problem der Mehrdeutigkeit

Eine Bewegung der Maus auf dem 2D-Bildschirm entspricht unendlich vielen möglichen Bewegungen im 3D-Raum (entlang des Sehstrahls).

1.6.1 Lösungsansätze

- **Mehrfachansichten (Quad-View):** Gleichzeitige Darstellung von Oben, Vorne, Seite und Perspektive. Interaktion findet in den 2D-Ansichten statt.
- **Modi-Umschaltung:** Mausbewegung wird je nach Taste/Modus unterschiedlich interpretiert (z. B. x/y Bewegung vs. z Bewegung).
- **3D-Widgets / Manipulatoren:**
  - Visuelle Hilfsobjekte, die in die Szene eingeblendet werden (z.B. Pfeile für Achsen, Ringe für Rotation).
  - Der Benutzer klickt auf einen bestimmten Teil des Manipulators (Handle).
  - Dadurch wird die Bewegungsfreiheit eingeschränkt (Constraint), z. B. “Rotation nur um die X-Achse”.
  - **Vorteil:** Intuitiv, reduziert Mauswege, löst Mehrdeutigkeit durch explizite Auswahl der Achse/Ebene.