Computergraphik

Prof. Dr.-Ing. Kerstin Müller

Kapitel 4

Transformationen und Projektionen

Geri‘s Game

Translationen

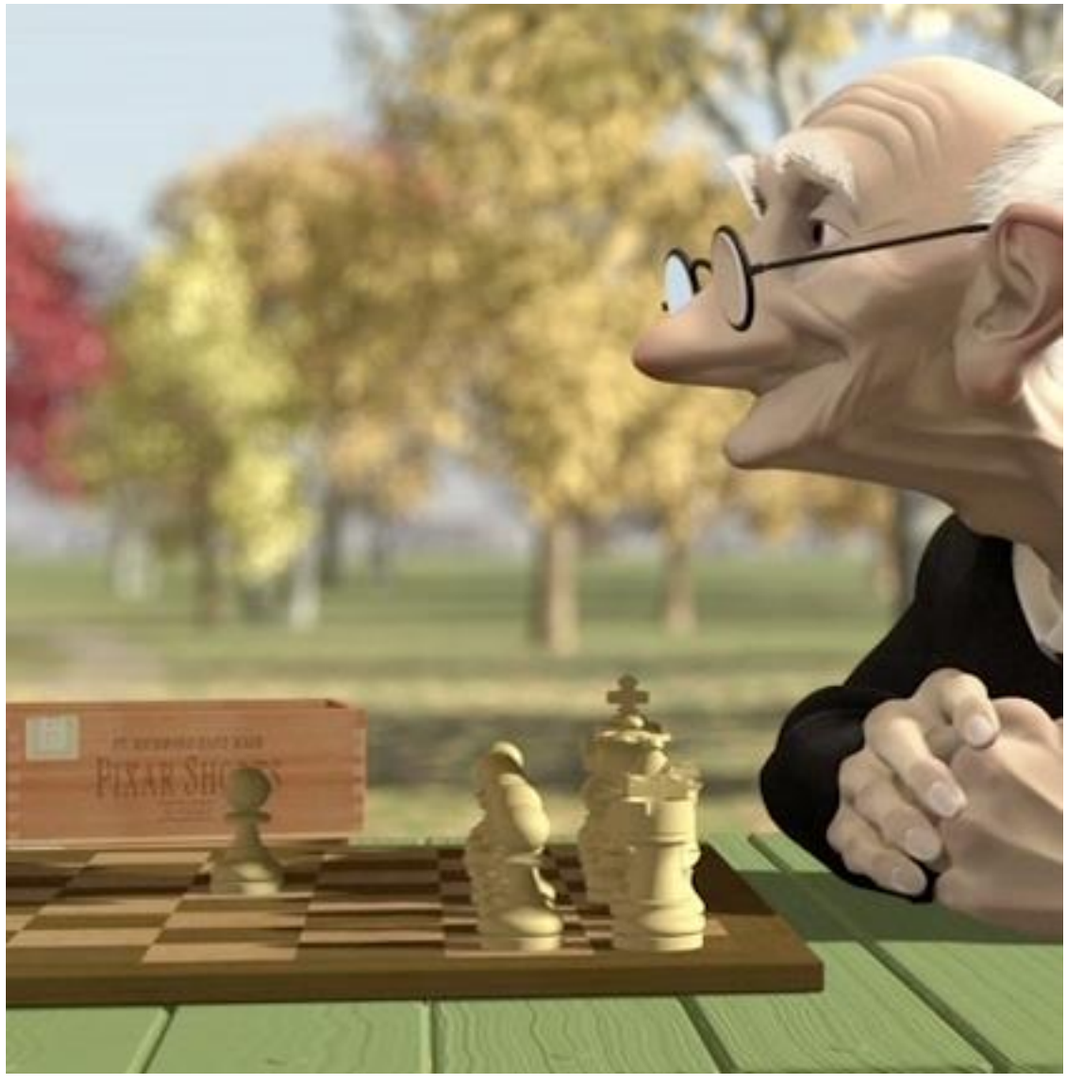
- Translationen müssen in 2D und in 3D durchgeführt werden.
  - Schieben der Schachfiguren auf dem Brett (2D).
  - Schachfigur bewegt sich von der Ursprungs-Position auf einer 3D - Raumkurve zum Zielfeld (3D).
  - Vereinfacht: Stückweise lineare Transformation bildet die Raumkurve.

Weitere Transformationen

- Neben Translationen müssen auch Rotationen durchgeführt werden, falls eine Schachfigur geschlagen wird und in der Kiste landet oder der König umgeworfen wird, wenn das Spiel verloren ist.
- Dieses Kapitel behandelt Translationen, Rotationen und Scherungen.

Transformation 3D zu 2D

- Integration eines Kamerakonzeptes in die Berechnung des 2D-Bildes aus dem 3D-Modell.
  - Einführung eines Kamerakoordinatensystems.
  - Welcher Teil der Gesamt-Szene ist überhaupt sichtbar? (Clipping)
  - Berechnung einer perspektivischen Projektion (weit entfernte Objekte sollen kleiner erscheinen)
  - Hinzufügen von Tiefenunschärfe.



Kapitelübersicht

- 4.1 Die Computergraphik Pipeline
- 4.2 Koordinatentransformationen
- 4.3 Transformationen in der Ebene
- 4.4 Transformationen im Raum
- 4.5 Projektionen
- 4.6 Windowing
- 4.7 Clipping

4.1 Die Computergraphik Pipeline

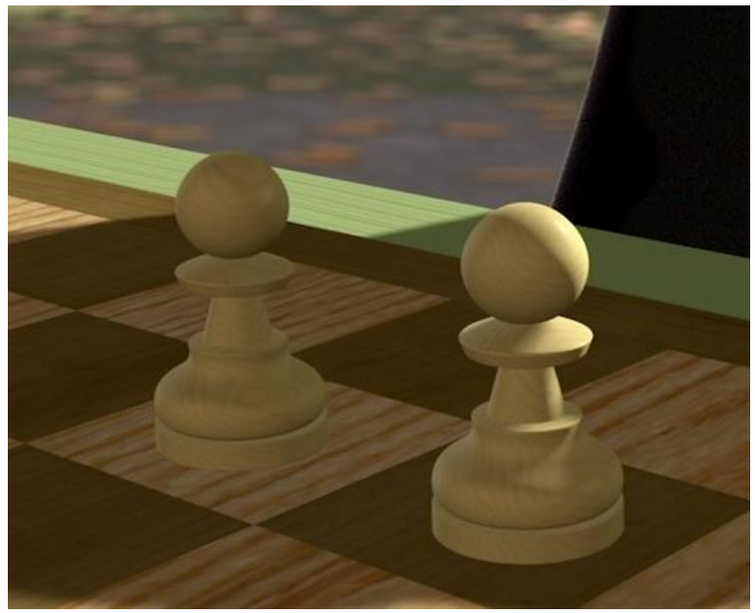
- Was ist prinzipiell nötig, um ein computergraphisch erzeugtes Bild auf dem Monitor/Drucker auszugeben?
- Definition (Computergraphik-Pipeline): Modellvorstellung, welche die einzelnen Verarbeitungsschritte vom Modell bis hin zum fertigen Bild verknüpft und zeitlich anordnet.
- Die tatsächliche Implementierung kann in Hardware, in Software oder teils-teils erfolgen.
- Das Konzept von Pipelining kann auch konkret umgesetzt werden.

Das Objekt-Koordinatensystem

- Schritt 1: Beschreibung der 3D-Objekte
  - Objektbeschreibungen können aus einer Datenbank geladen oder selbst beschrieben werden.
  - Dies geschieht aus Gründen der Einfachheit in einem eigenen Objekt Koordinatensystem oder lokalen Koordinatensystem.
  - Man kann so z.B. den Ursprung des lokalen Koordinatensystems in die Mitte eines symmetrischen Objekts legen.
  - Die Objekte werden nun mittels Transformationen, beispielsweise Translationen (d.h. Verschiebungen), Rotationen oder Skalierungen and die ihnen zugedachen Plätze in der Szene platziert.
  - Mathematisch wird dies durch Multplikationen der Transformations-matrizen mit den die Objekte definierenden Punkten durchgeführt.

Das Objekt-Koordinatensystem

- Beispiel: Modellierung der Schachfiguren
  - Jeder Typ von Schachfigur wird nur einmal in der Datenbank abgelegt, d.h. es gibt z.B. nur einen Bauern in der Datenbank obwohl er 16 mal auf dem Spielfeld (mit verschiedenen Positionen und Farben) vorkommt.
  - Das lokale Koordinatensystem ist definiert durch die Standebene und die Hochachse der Figur.
  - Für das verschiedene Auftreten der Figuren wird je eine Kopie mit der entsprechenden Transformation (zunächst Verschiebung) versehen.
  - Zunächst einfach: 2D-Verschiebung auf dem Brett.



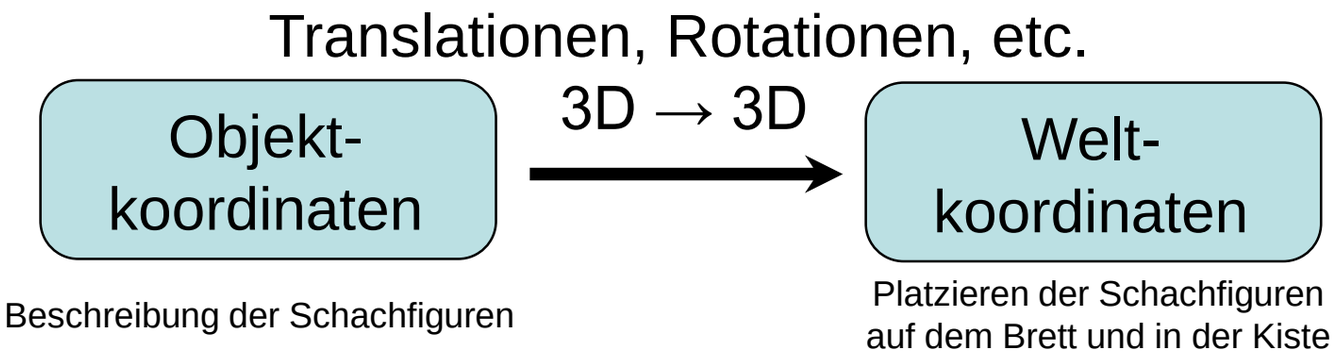
Das Objekt-Koordinatensystem

- Beispiel: Modellierung der Schachfiguren
  - Falls eine Figur geschlagen wird, muss sie in beliebiger Orientierung (z.B. liegend in der Holzkiste) dargestellt werden können.
  - Daher sind Rotationen und Verschiebungen im 3D Raum (die oben in der Kiste liegende Figur ist „höher“ positioniert als das Spielfeld) notwendig.
  - Ein lokales Koordinatensystem ist insbesondere bei spiegelsymmetrischen Objekten sinnvoll, wenn eine Koordinatenebene die Spiegelebene ist.
    - Hier kann bei interaktiver Veränderung des Objektes nur eine Seite modifiziert werden, und die Symmetrie durch Spiegelung an der Koordinatenebene (sehr einfach: nur das Vorzeichen einer Koordinate invertieren) beibehalten werden.

Das Welt Koordinatensystem

- Schritt 2: Zusammenstellung aller Objekte im Raum
  - Sind alle Objekte in der Szene platziert worden, ist das Ergebnis eine Beschreibung der Szene im Welt-Koordinatensystem oder globalen Koordinatensystem.
  - Technisch wird dieser Schritt in der heutigen Graphik-Hardware ausgeführt.
  - Die Regeln für den Zusammenbau der Szene , d.h. welche Transformationen und in welcher Reihenfolge werden in einem so genannten Szenengraphen gespeichert.
  - Auf diese Art und Weise werden nicht nur Objekte, sondern zum Beispiel auch virtuelle Lichtquellen plaziert.

Computergraphik-Pipeline Stufe 1



Das Kamera-Koordinatensystem

- Schritt 3: Platzierung einer virtuellen Kamera
  - Der Kamera wird ein eigenes lokales Koordinatensystem das Kamera-Koordinatensystem oder Eye-Koordinatensystem zugeordnet, aus dessen Sicht sich alle im Welt-Koordinatensystem beschriebene Objekte betrachten lassen.
  - Also Betrachtungsweise der Szene aus der Sicht einer Kamera.
  - Analogie: Der Pilot eines Flugzeugs schaut in eine Szene
    - Was er sieht hängt von seiner momentanen Position, der Flugrichtung und der Orientierung (Rotation bei Kurvenflug) ab.
    - Es wird hier noch davon ausgegangen dass er unendlich weit blicken kann.
  - Prinzip Flugzeug wird oft in virtuellen Rundflügen durch eine Szene verwendet:

Beispiel: Virtueller Flug durch eine Szene

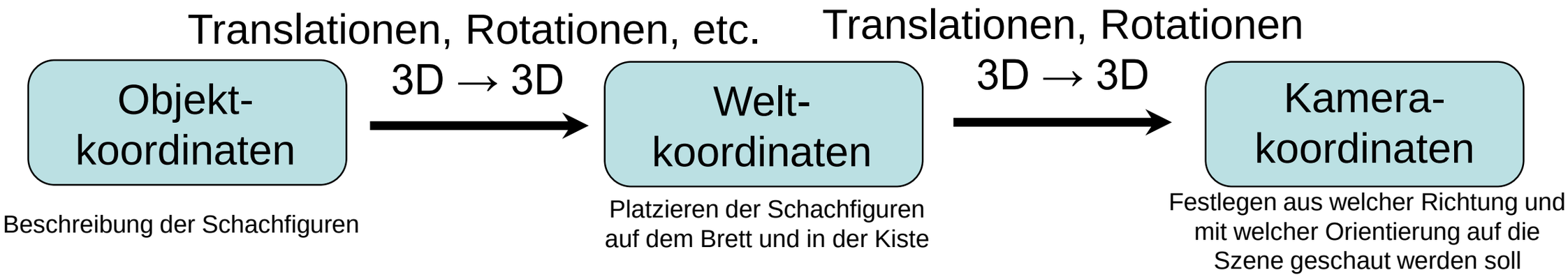
- Umsetzung in Computerspielen:
  - Die Szene (hier Mine) ist fix, die Sicht-weise ändert sich.

- Umsetzung in Animationen, hier z.B. in Ocean`s Eleven

Welt- und Kamera-Koordinaten kombiniert

- Wegen der Analogie
  Platzierung der Kamera in einer Szene und
  Szene vor der Kamera ausrichten
  wird die Transformation von Objekt in Weltkoordinaten und weiter in Kamerakoordinaten oft in einem Schritt ausgeführt.
- Ein Beispiel ist die Modelview-Matrix in OpenGL.
- Im Bereich dieser beiden Stufen ist dann auch das Konzept des Szenegraphen (nicht Teil von OpenGL) angesiedelt.

Computergraphik-Pipeline Stufe 1-2



Clipping-Koordinaten

- Stufe 3: Clipping Koordinaten
- Festlegen wie groß das Sichtbarkeitsvolumen ist.
- Üblicherweise wird ein Würfel oder ein Pyramidenstumpf (sogenanntes Frustum) verwendet.
- Unproblematisch falls Objekte ganz oder gar nicht innerhalb des Sichtbarkeitsvolumens liegen.
- Bei Objekten die teilweise sichtbar sind ist eine Sonder-behandlung notwendig.
  - Entweder Objekte als innen definieren und mitprojizieren, dann in 2D Ränder zuschneiden
  - Oder Schnittoperationen in 3D durchführen.

Computergraphik-Pipeline Stufe 1-3

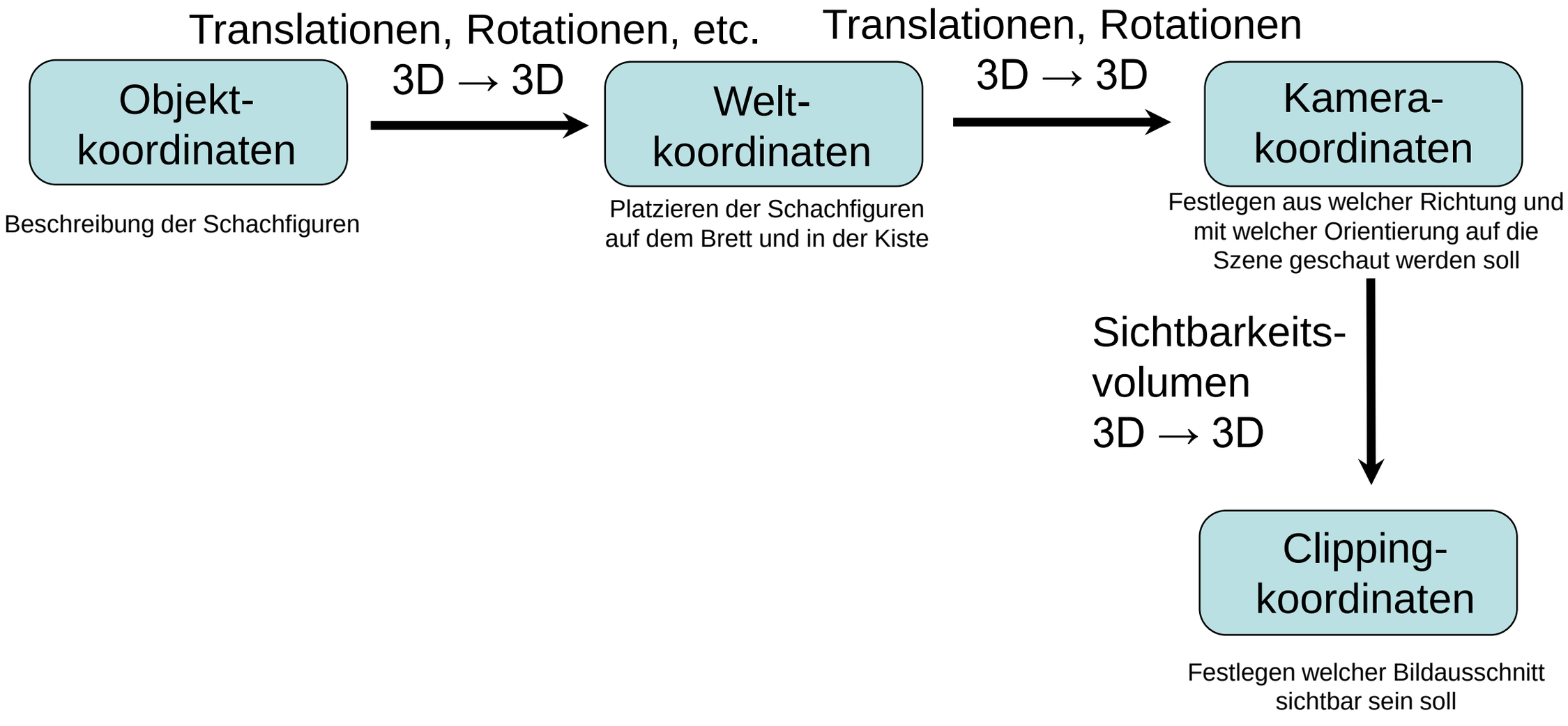
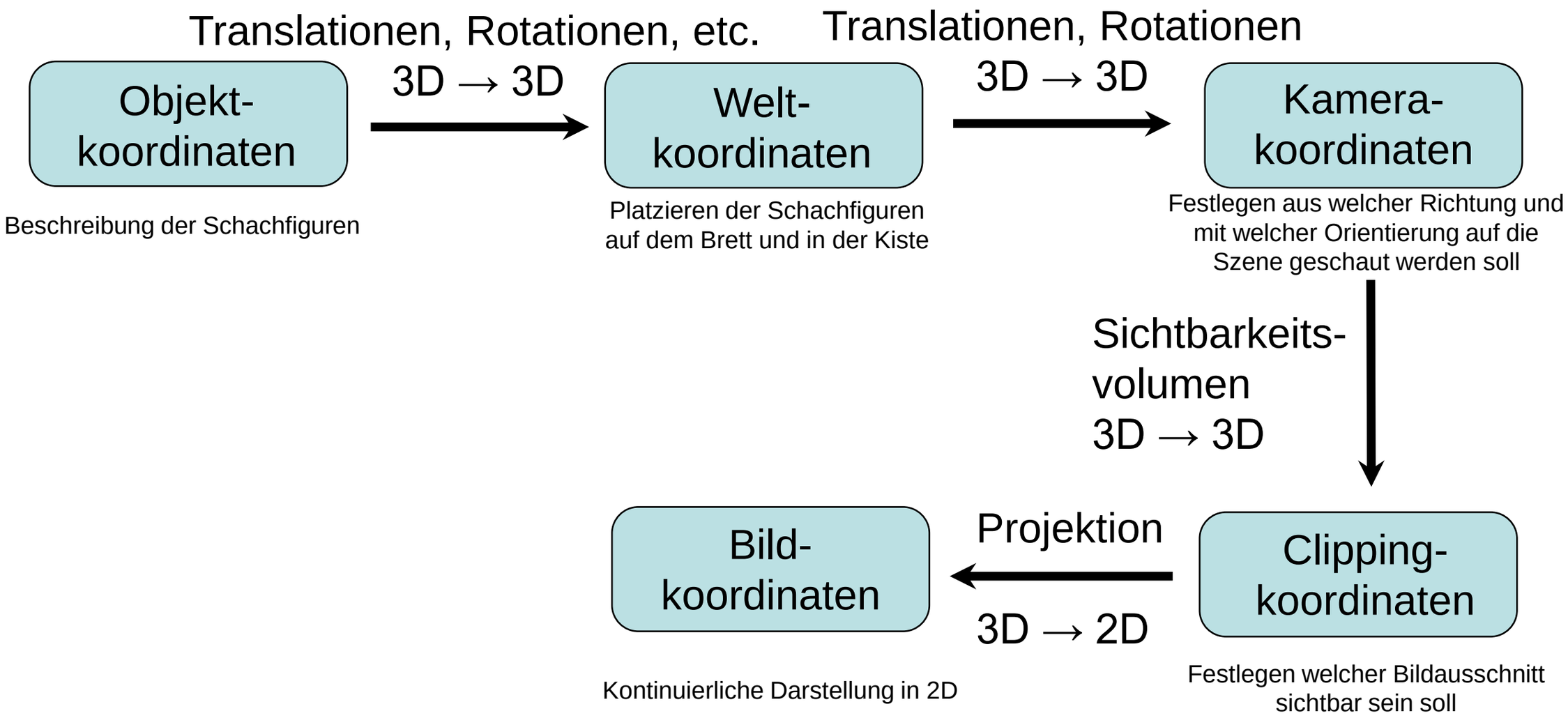


Bild Koordinaten

- Stufe 4: Beim Übergang von Clipping Koordinaten zu Bild Koordinaten wird der logische Übergang von 3D auf 2D Koordinaten vollzogen.
- Die Projektion wird ebenfalls durch eine Matrixmultiplikation durchgeführt (i.d.R. in Hardware realisiert).
- Es entstehen Koordinaten in der Projektions- oder Bildebene, im Bild-Koordinatensystem oder screen space. Die Darstellung ist noch immer kontinuierlich!!!
- Dabei interessieren allerdings nur Punkte innerhalb eines rechteckigen Bereichs, des Windows oder Fensters.
- Das Zuschneiden wird also auch in dieser Stufe realisiert.

Computergraphik-Pipeline Stufe 1-4



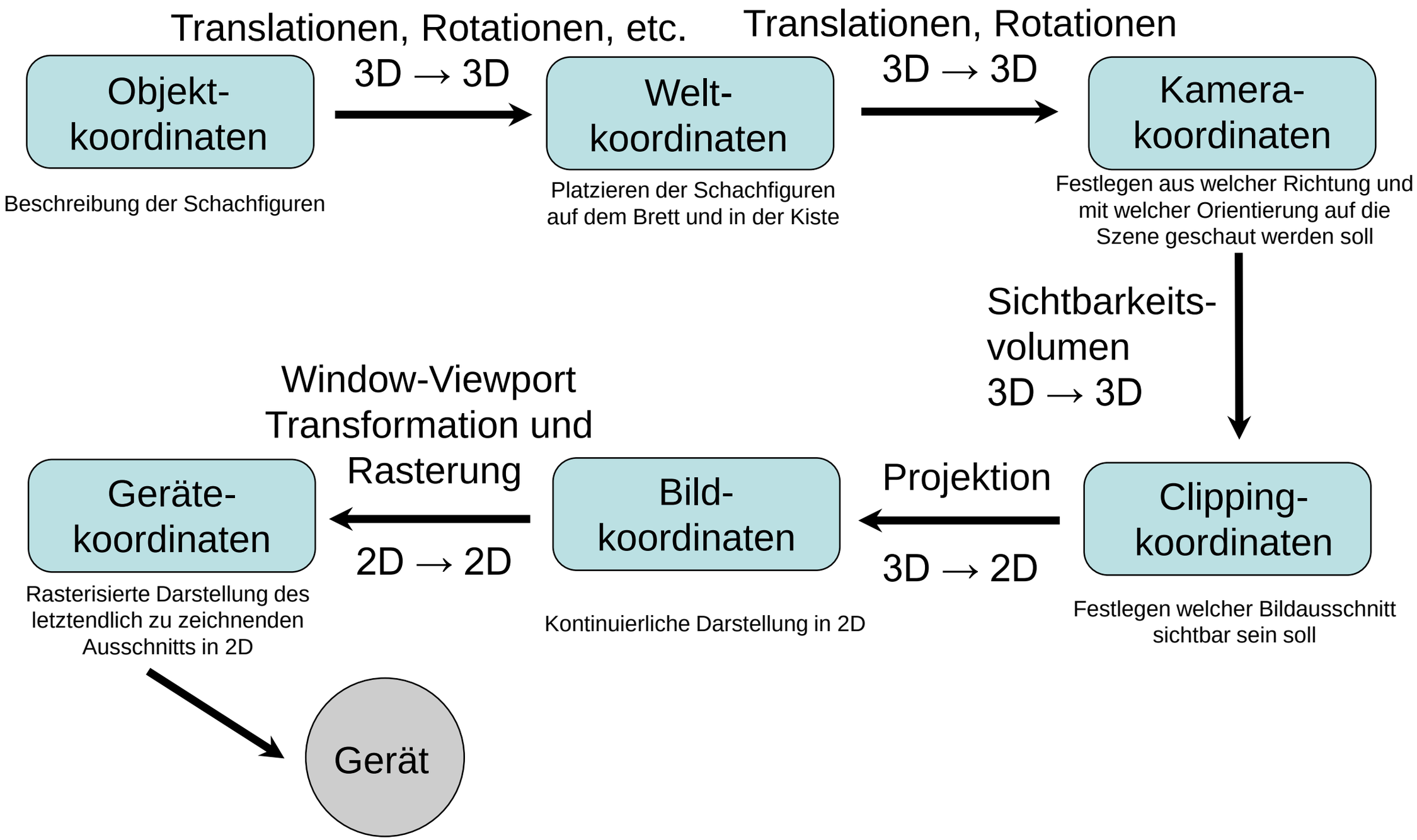
Geräte-Koordinaten

- Ausgabegräte (meist Bildschirm), verwenden ein diskretes 2D-Raster von einzelnen Bildpunkten, den Pixeln .
  - Die Bildpunkte sollen quadratisch sein und identische Größe haben.
- Als Organisation zur Speicherung des Rasterbildes dient eine zweidimensionale Speichermatrix, der Bildschirm-speicher oder frame buffer.
  - Die Größe (Höhe,Breite) des frame buffers bestimmt die virtuelle Bildschirmauflösung.
  - Diese kann von der festen physikalischen Auflösung des Ausgabegräts abweichen.
  - Die Größe des Framebuffers ist oft per Software einstellbar.

Geräte-Koordinaten

- Die Gerätekoordinaten beziehen sich auf die virtuelle Bilschirmauflösung nicht die physikalische!!!
- Stufe 5: Die Umrechnung der kontinuierlichen 2D Darstellung in eine Rasterdarstellung wird heute ebenfalls komplett in Hardware realisiert.
- Ebenfalls in dieser Stufe wird die so genannte Window-Viewport Transformation durchgeführt.
  - Der Viewport gibt den Bildschirmbereich an, in dem der Inhalt eines Fensters abgebildet werden soll. Window gibt an was gezeichnet werden soll.

Zusammenfassung: Graphik-Pipeline



4.2 Koordinatentransformationen

- Viele geometrische Objekte können über Punktkoordinaten beschrieben werden.
- z.B. ist ein Würfel eindeutig festgelegt durch die Angabe der Koordinaten seiner 8 Eckpunkte, eine Kugel durch die Angabe ihres Mittelpunktes und des Radius.
- Daraus folgt die Notwendigkeit zwischen absoluten Positionen (Mittelpunkt der Kugel, soll von der Transformation geändert werden) und relativen Grössen (Radius, soll nicht geändert werden) zu unterscheiden.

Berechnung einer Translation

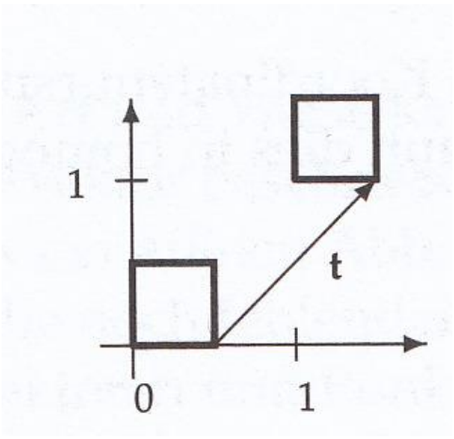
- Erster Ansatz:
  - Translation komponentenweise berechnen, Unterscheidung absolute Position vs. relative Position explizit.
    - `class Position`

```
{
    private:

        double x,y;
        bool isRelative;

    public:

        void operator + (Position const& arg);
};
```



- if-Abfrage in der Implementierung des Operators „+“.

Berechnung einer Translation

- Mathematisch gesehen handelt es sich bei der Translation um eine Vektor-Addition.
  - Ein Punkt P mit Koordinaten $\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$ wird um $T = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{pmatrix}$ verschoben.
  - Es entsteht einer neuer Punkt P` mit P` = T + P.
  - In Vektor-Schreibweise

$$\begin{pmatrix} \mathbf{x}' \\ \mathbf{y}' \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} + \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{x} + \mathbf{t}_1 \\ \mathbf{y} + \mathbf{t}_2 \end{pmatrix}$$

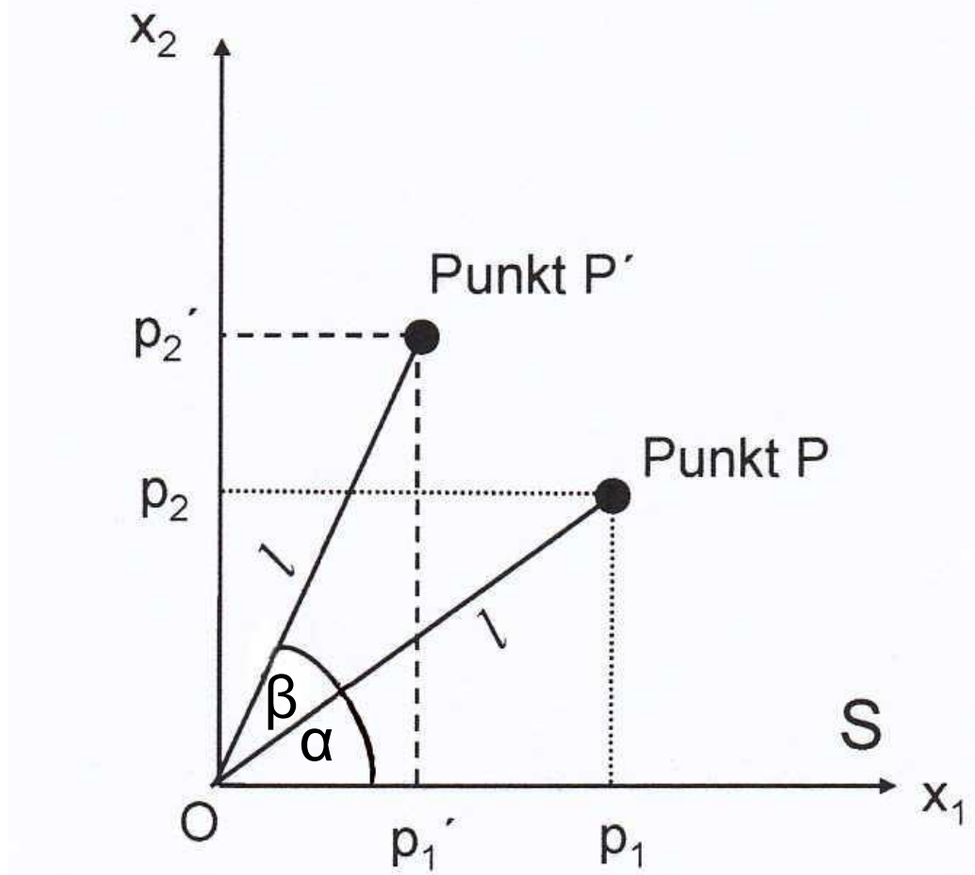
  - Die Tatsache ob es sich um eine relative oder absolute Koordinate handelt, ist nicht in der Berechnung abzulesen, sondern muss separat mitgeloggt werden.

Rotationen um den Ursprung

- Mathematische Herleitung der Rotationsformel

$$\mathbf{p}_1 = l \cdot \cos\alpha$$
$$\mathbf{p}_2 = l \cdot \sin\alpha$$

$$\mathbf{p}'_1 = l \cdot \cos(\alpha+\beta)$$
$$\mathbf{p}'_2 = l \cdot \sin(\alpha+\beta)$$



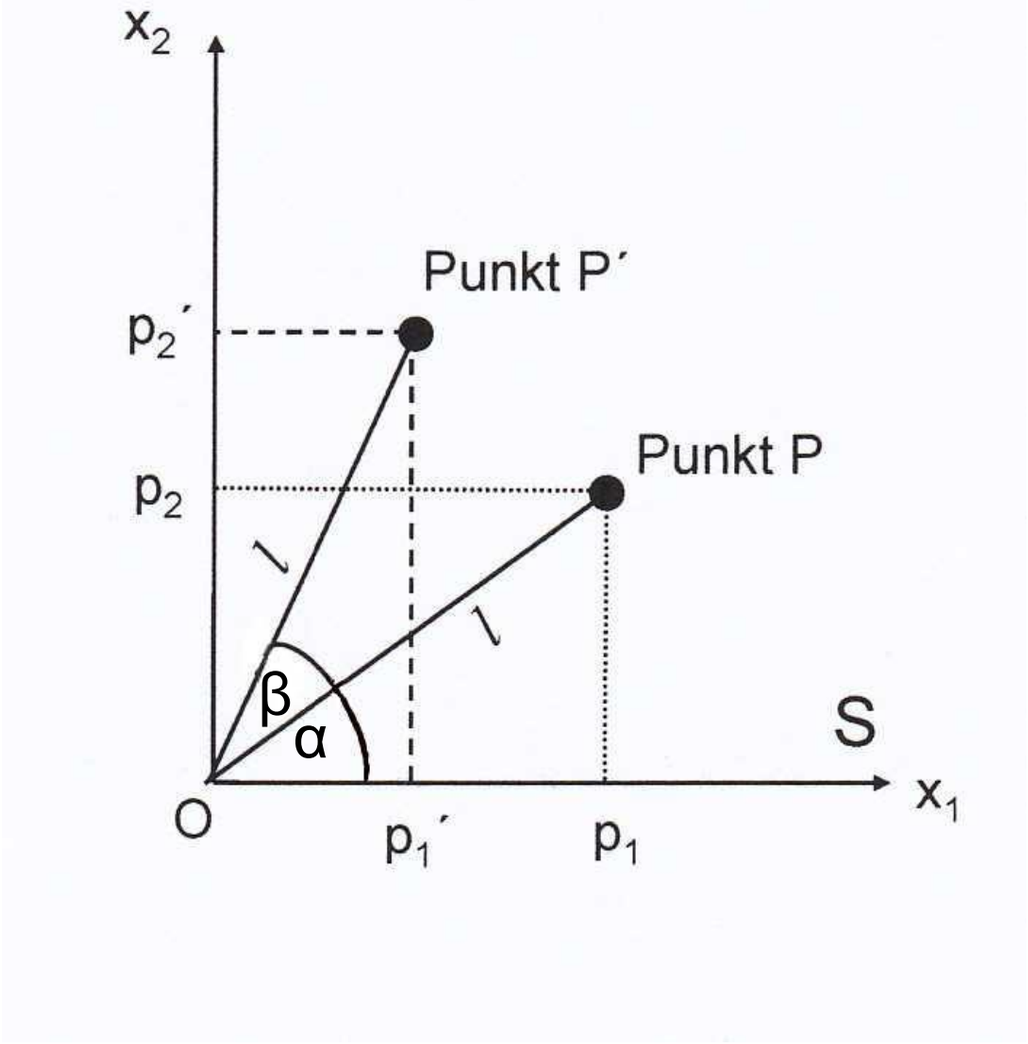
$$\mathbf{p}'_1 = l \cdot \cos(\alpha+\beta) = l \cdot (\cos\alpha \cdot \cos\beta - \sin\alpha \cdot \sin\beta)$$
$$= \mathbf{p}_1 \cdot \cos\beta - \mathbf{p}_2 \cdot \sin\beta$$

$$\mathbf{p}'_2 = l \cdot \sin(\alpha+\beta) = l \cdot (\sin\alpha \cdot \cos\beta + \cos\alpha \cdot \sin\beta)$$
$$= \mathbf{p}_2 \cdot \cos\beta + \mathbf{p}_1 \cdot \sin\beta$$

Rotationen um den Ursprung

- Mathematische Herleitung der Rotationsformel

$$\begin{bmatrix} p_1' \\ p_2' \end{bmatrix} = \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$$



Rotationen um den Ursprung

- Zusammenfassend: 2D Rotation um den Ursprung

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$$

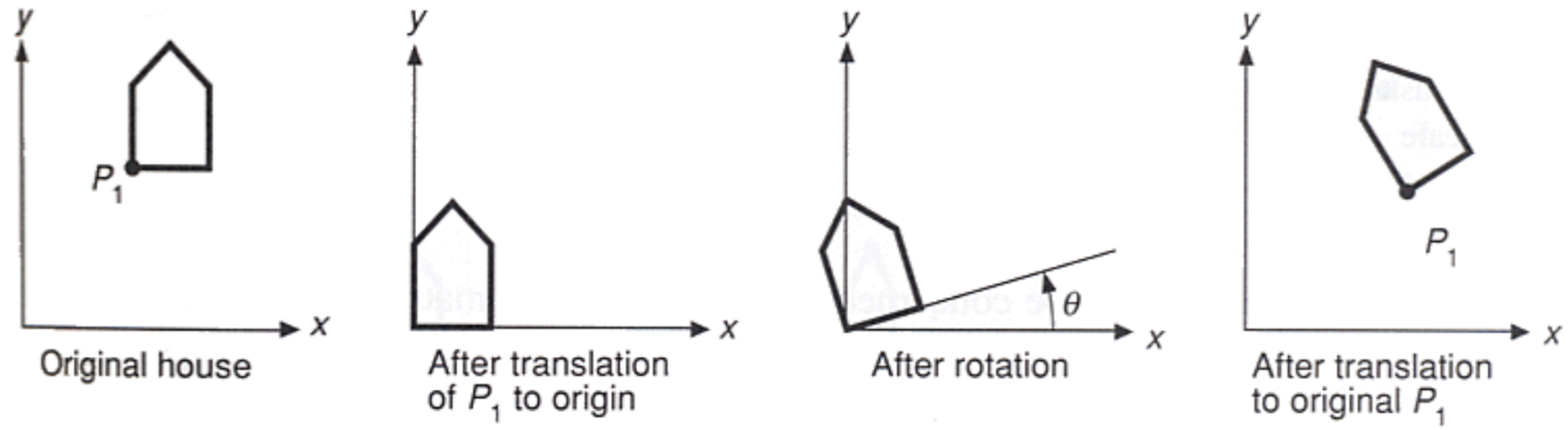
- In Vektor-Matrix Schreibweise: $P' = R(\varphi) \cdot P$ mit der Rotationsmatrix

$$R(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

- Die Matrix $R(\varphi)$ dreht Punkte um den Winkel φ in mathematisch positiver Richtung (gegen den Uhrzeigersinn).

Rotationen um einen beliebigen Punkt

- Rotation um einen beliebigen Punkt P1:
  - (1) Translation von P1 in den Ursprung
  - (2) Rotation um den Ursprung
  - (3) Translation von P1 in die Ursprüngliche Position

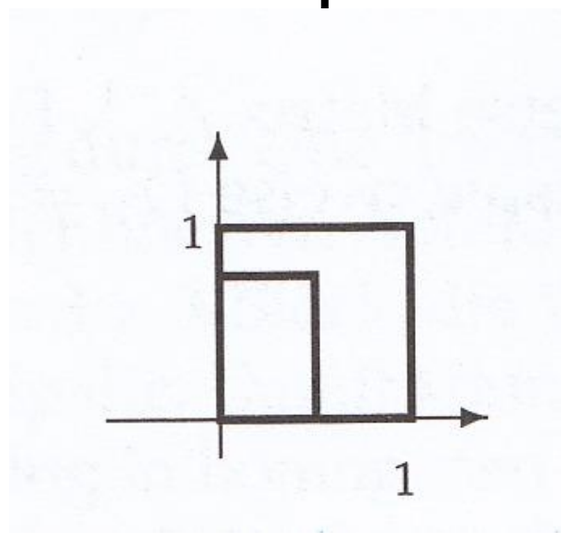


- Bemerkung: Die Matrizenmultiplikation ist nicht kommutativ, d.h. die Reihenfolge der Matrizen muss der Reihenfolge der Operationen entsprechen.

Skalierung

- Skalierung des Punktes $\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ mit den Faktoren α und β entspricht in Matrix-Vektor Schreibweise

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \alpha \cdot \mathbf{x} \\ \beta \cdot \mathbf{y} \end{bmatrix}$$



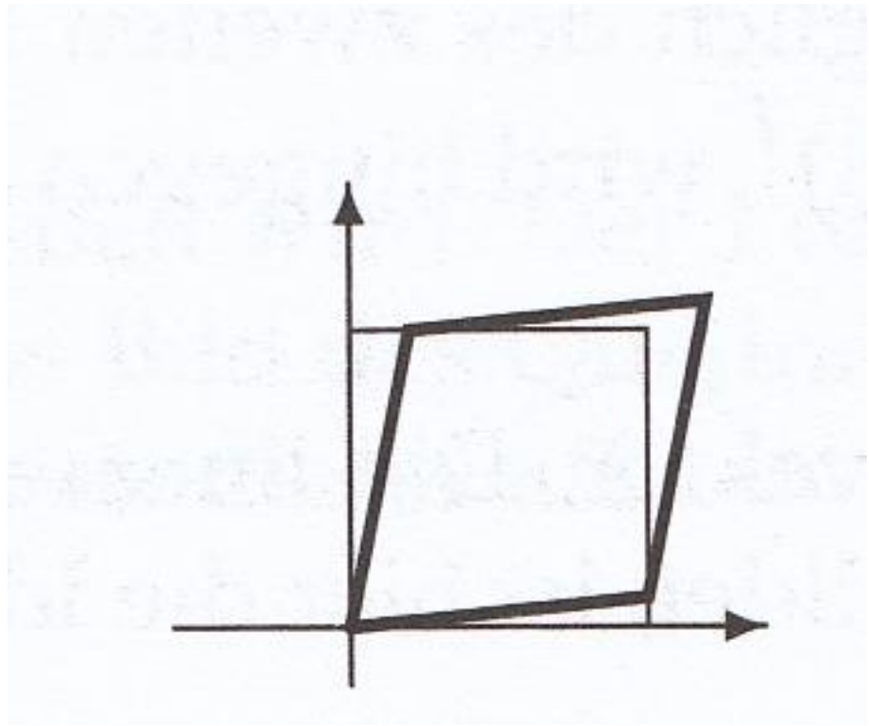
Skalierung

- Das bedeutet aber auch, dass ein Objekt welches nicht den Nullpunkt als Eckpunkt oder Mittelpunkt hat, „wandert“.
- „Wandern“ vermeiden: den Referenz-Eckpunkt nicht skalieren, oder Ursprung des lokalen Koordinatensystems entsprechend wählen.
  - ein weiterer Grund zur Verwendung lokaler Koordinatensysteme.
- Vorsicht: Skalierungen können Orthogonalität, Orientierung, Längen und Normalen zerstören, insbesondere wenn mit 0 oder negativen Werten skaliert wird.

Berechnung einer Scherung

- Eine Scherung ergibt sich, wenn Abhängigkeiten folgender Form bestehen:

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} + \alpha \cdot \mathbf{y} \\ \mathbf{y}' &= \beta \cdot \mathbf{x} + \mathbf{y} \end{aligned}$$



- In Matrix-Vektor Schreibweise:

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$$

Affine Transformationen

- Affine Transformationen lassen sich als Kombination einer linearen Abbildung A (kann immer durch eine Matrix dargestellt werden) und einer Translation T schreiben:

$$P' = A \cdot P + T$$

- Die bisher genannten Transformationen (Translation, Rotation, Skalierung, Scherung) sind affine Transformationen.
- Affine Invarianz von Teilungsverhältnissen:

  Für eine affine Transformation F und die Punkte P und Q gilt immer:

$$F(\lambda \cdot P + (1 - \lambda) \cdot Q) = \lambda \cdot F(P) + (1 - \lambda) \cdot F(Q) \text{ für } 0 \leq \lambda \leq 1$$

Eigenschaften affiner Abbildungen

- Das Bild einer Strecke von Q nach P unter einer affinen Abbildung F ist wieder eine Strecke.
- Eine affine Abbildung F ändert Teilverhältnisse $\lambda : (1 - \lambda)$ nicht.
- Es genügt Endpunkte Q und P einer Strecke abzubilden. Zwischenpunkte erhält man durch Interpolation von $F(Q)$ und $F(P)$.
- Unter affinen Abbildungen bleiben parallele Linien parallel.
- Affine Abbildungen sind nicht winkeltreu, und nicht längentreu.

Weitere Affine Transformationen

- Reflexion an der Gerade y = x : $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- Reflexion an der Gerade y = -x : $A = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$
- Reflexion an der x-Achse : $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
- Reflexion an der y-Achse : $A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
- Reflexion am Ursprung : $A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

Komposition von Transformationen

- Die Hintereinanderschaltung von Rotation, Translation und Skalierung führt auf die Abbildungsgleichung

$$P' = S \cdot (T + R \cdot P)$$

- Müssen jedoch mehrere solcher Transformationen hintereinander ausgeführt werden, so stört die Addition in der obigen Gleichung.
- Da heutige Graphikhardware insbesondere auch Matrizenmultiplikationen unterstützt, ist es günstig Transformationen ausschliesslich mittels Matrizenmultiplikation auszuführen.

$$P' = M_n \cdot \ \ldots \ \cdot M_3 \cdot M_2 \cdot M_1 \cdot P$$

Komposition von Transformationen

- falls dieselbe Komposition von Transformationen auf viele Punkte angewendet werden soll, ergibt sich ein enormer Geschwindigkeitsvorteil, da die komplette Transformation vorberechnet werden kann (Matrizenmultiplikation ist assoziativ).
- Die Matrizen können sukzessive auf einem Stack gesammelt werden und dann die Berechnung der Gesamt-Transformations-Matrix durchgeführt werden.

Punkte vs. Vektoren

- In der Computergraphik wird zwischen Punkten und Vektoren unterschieden
  - Punkte bezeichnen Positionen im Raum, Vektoren dagegen Verschiebungen (von Punkten) bzw. Richtungen (zwischen Punkten).
  - Vektoren sind invariant unter Verschiebungen des Koordinaten-ursprungs.
  - Mit dieser Unterscheidung können Aussagen unabhängig von einem konkreten Koordinatensystem getroffen werden.
  - Unterscheidung durch Zusatzkoordinate.

Homogene Koordinaten

- Das Tripel $(x, y, 1)^T$ stellt die sogenannten homogenen Koordinaten des Punktes $(x, y)^T \in \mathbb{R}^2$ dar.
- Analog besitzt ein Punkt $(x, y, z)^T \in \mathbb{R}^3$ $(x, y, z, 1)^T$ als homogene Koordinate.
- Das Tripel $(x, y, 0)^T$ stellt die homogenen Koordinaten eines Vektors $(x, y)^T \in \mathbb{R}^2$ dar.
- Für einen Vektor $(x, y, z)^T \in \mathbb{R}^3$ sind folglich $(x, y, z, 0)^T$ die homogenen Koordinaten.
- Bemerkung: Eigentlich gibt es unendlich viele Darstellungen $(x, y, w)^T (w \neq 0)$ desselben Punkes $(x, y)^T \in \mathbb{R}^2$ (analog für $(x, y, z)^T \in \mathbb{R}^3$). Man verwendet die sog. Standard-Darstellung mit $w = 1$.

Berechnungen in homogenen Koordinaten

- Bemerkungen:
  - In homogenen Koordinaten ist die Differenz zweier Punkte ein Vektor.

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \\ 1 \end{bmatrix} - \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{y}_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ 0 \end{bmatrix}$$

  - Ebenso ist die Differenz oder Summe zweier Vektoren ein Vektor.
  - Addition Punkt + Vektor ergibt Punkt.

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \\ 1 \end{bmatrix} + \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{y}_2 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ 1 \end{bmatrix}$$

  - Addition zweier Punkte macht keinen Sinn, da die Homogenisierende (= Zusatzkoordinate) im Ergebnis weder 0 noch 1 wäre.

Translation / Rotation in hom. Koordinaten

- Translation eines Punktes $\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ um den Vektor $\begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix}$:

$$\begin{bmatrix} 1 & 0 & \mathbf{t}_1 \\ 0 & 1 & \mathbf{t}_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}+\mathbf{t}_1 \\ \mathbf{y}+\mathbf{t}_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ 1 \end{bmatrix}$$

  - Bemerkung: Ein Vektor ist invariant bezüglich Translation.

- Rotation eines Punktes $\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ um den Winkel φ :

$$\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}\cdot\cos\varphi - \mathbf{y}\cdot\sin\varphi \\ \mathbf{x}\cdot\sin\varphi + \mathbf{y}\cdot\cos\varphi \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ 1 \end{bmatrix}$$

Skalierung / Rotation um beliebigen Punkt

- Skalierung eines Punktes $\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ mit den Faktoren λ_1 und λ_2:

$$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_1 \cdot \mathbf{x} \\ \lambda_2 \cdot \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ 1 \end{bmatrix}$$

- Rotation eines Punktes $\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ um einen Punkt $\begin{bmatrix} P_x \\ P_y \end{bmatrix}$ um den Winkel φ :

$$\begin{bmatrix} 1 & 0 & P_x \\ 0 & 1 & P_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -P_x \\ 0 & 1 & -P_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ 1 \end{bmatrix}$$

Transformationen im Raum: Translation

- Die Verschiebung eines Punktes $(\mathbf{x}, \mathbf{y}, \mathbf{z})^T$ um den Translationsvektor $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)^T$ ergibt den Punkt $(\mathbf{x}', \mathbf{y}', \mathbf{z}')^T$ mit:

$$\begin{bmatrix} 1 & 0 & 0 & \mathbf{t}_1 \\ 0 & 1 & 0 & \mathbf{t}_2 \\ 0 & 0 & 1 & \mathbf{t}_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x} + \mathbf{t}_1 \\ \mathbf{y} + \mathbf{t}_2 \\ \mathbf{z} + \mathbf{t}_3 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ \mathbf{z}' \\ 1 \end{bmatrix}$$

Skalierung in 3D

- Eine Skalierung mit den Faktoren s_1, s_2 und s_3 in den drei Achsenrichtungen hat die Gestalt:

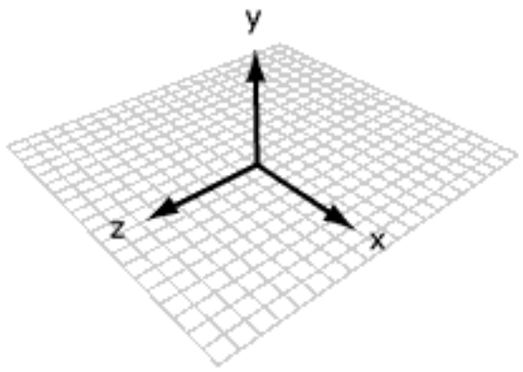
$$\begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 \cdot \mathbf{x} \\ s_2 \cdot \mathbf{y} \\ s_3 \cdot \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ \mathbf{z}' \\ 1 \end{bmatrix}$$

Rotationen in 3D

- Alle Rotationen erfolgen im mathematisch positiven Sinn (d.h. gegen den Uhrzeiger-Sinn)
- Der Betrachter „sitzt“ dabei auf der Rotationsachse und schaut in Richtung Ursprung des Koordinatensystems.
- Wir betrachten zunächst die Rotationen um die einzelnen Koordinatenachsen um den Winkel φ .

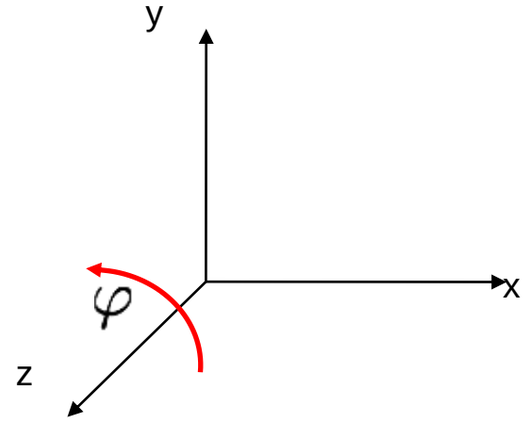
→ Transformationsmatrizen:

$$R_x(\varphi), R_y(\varphi), R_z(\varphi)$$



Rotation um die z-Achse

- Die Rotation eines Punktes $(\mathbf{x}, \mathbf{y}, \mathbf{z})^T$ um die z-Achse um den Winkel φ ergibt den Punkt $(\mathbf{x}', \mathbf{y}', \mathbf{z}')^T$ mit:

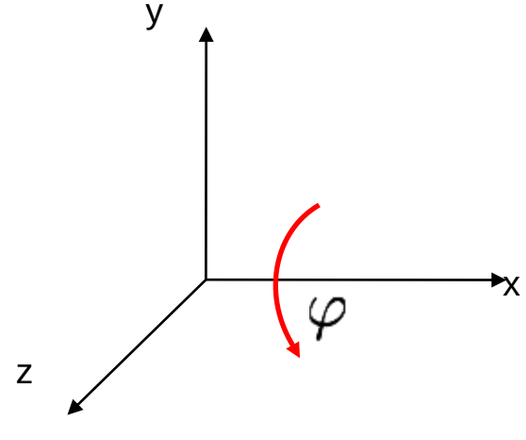


$$\underbrace{\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_z(\varphi)} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x} \cdot \cos\varphi - \mathbf{y} \cdot \sin\varphi \\ \mathbf{x} \cdot \sin\varphi + \mathbf{y} \cdot \cos\varphi \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ \mathbf{z}' \\ 1 \end{bmatrix}$$

- Bemerkung: Drehung um den Winkel φ um die z-Achse entspricht dem 2D-Fall, wobei die z-Koordinate konstant bleibt.

Rotation um die x-Achse

- Die Rotation eines Punktes $(\mathbf{x}, \mathbf{y}, \mathbf{z})^T$ um die x-Achse um den Winkel φ ergibt den Punkt $(\mathbf{x}', \mathbf{y}', \mathbf{z}')^T$ mit:

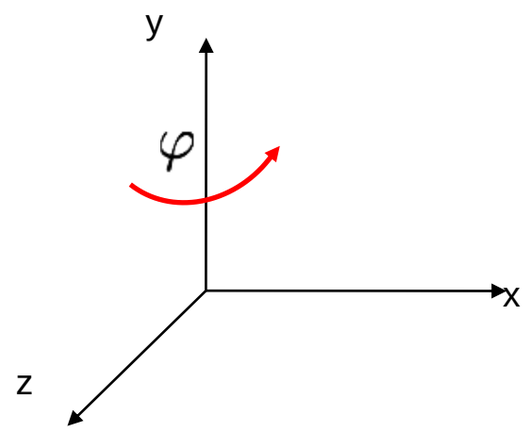


$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_x(\varphi)} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ \mathbf{y} \cdot \cos\varphi - \mathbf{z} \cdot \sin\varphi \\ \mathbf{y} \cdot \sin\varphi + \mathbf{z} \cdot \cos\varphi \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ \mathbf{z}' \\ 1 \end{bmatrix}$$

- Bemerkung: Drehung um den Winkel φ um die x-Achse entspricht dem 2D-Fall, wobei die x-Koordinate konstant bleibt.

Rotation um die y-Achse

- Die Rotation eines Punktes $(\mathbf{x}, \mathbf{y}, \mathbf{z})^T$ um die y-Achse um den Winkel φ ergibt den Punkt $(\mathbf{x}', \mathbf{y}', \mathbf{z}')^T$ mit:



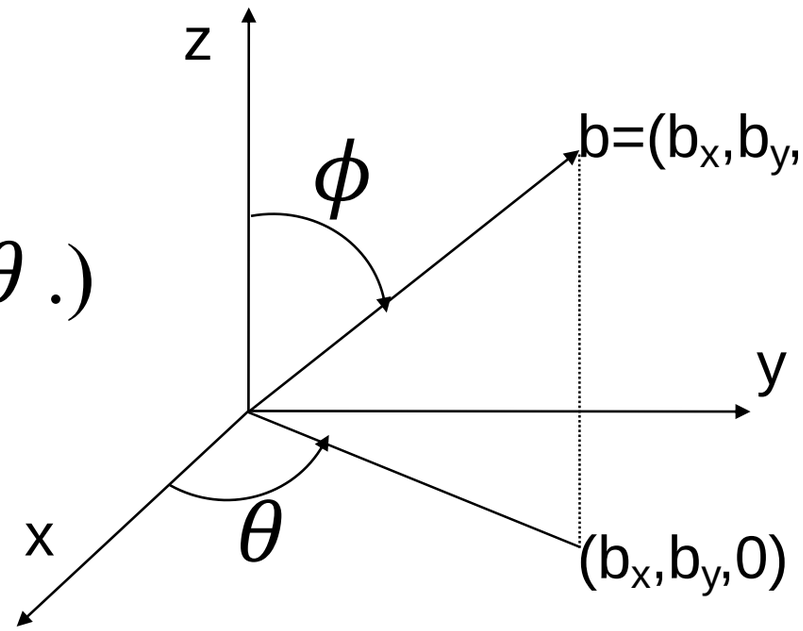
$$\underbrace{\begin{bmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_y(\varphi)} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x} \cdot \cos\varphi + \mathbf{z} \cdot \sin\varphi \\ y \\ -\mathbf{x} \cdot \sin\varphi + \mathbf{z} \cdot \cos\varphi \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \\ \mathbf{z}' \\ 1 \end{bmatrix}$$

- Bemerkung: Drehung um den Winkel φ um die y-Achse entspricht dem 2D-Fall, wobei die y-Koordinate konstant bleibt.

Rotation um eine beliebige Achse

- Jede Rotation um eine beliebige Achse kann aus Rotationen um die einzelnen Koordinatenachsen zusammengesetzt werden. (→ Euler: Darstellung von b_x,b_y,b_z durch ϕ und θ .)

Bem.: immer noch rechtshändiges System, hier nur gedreht



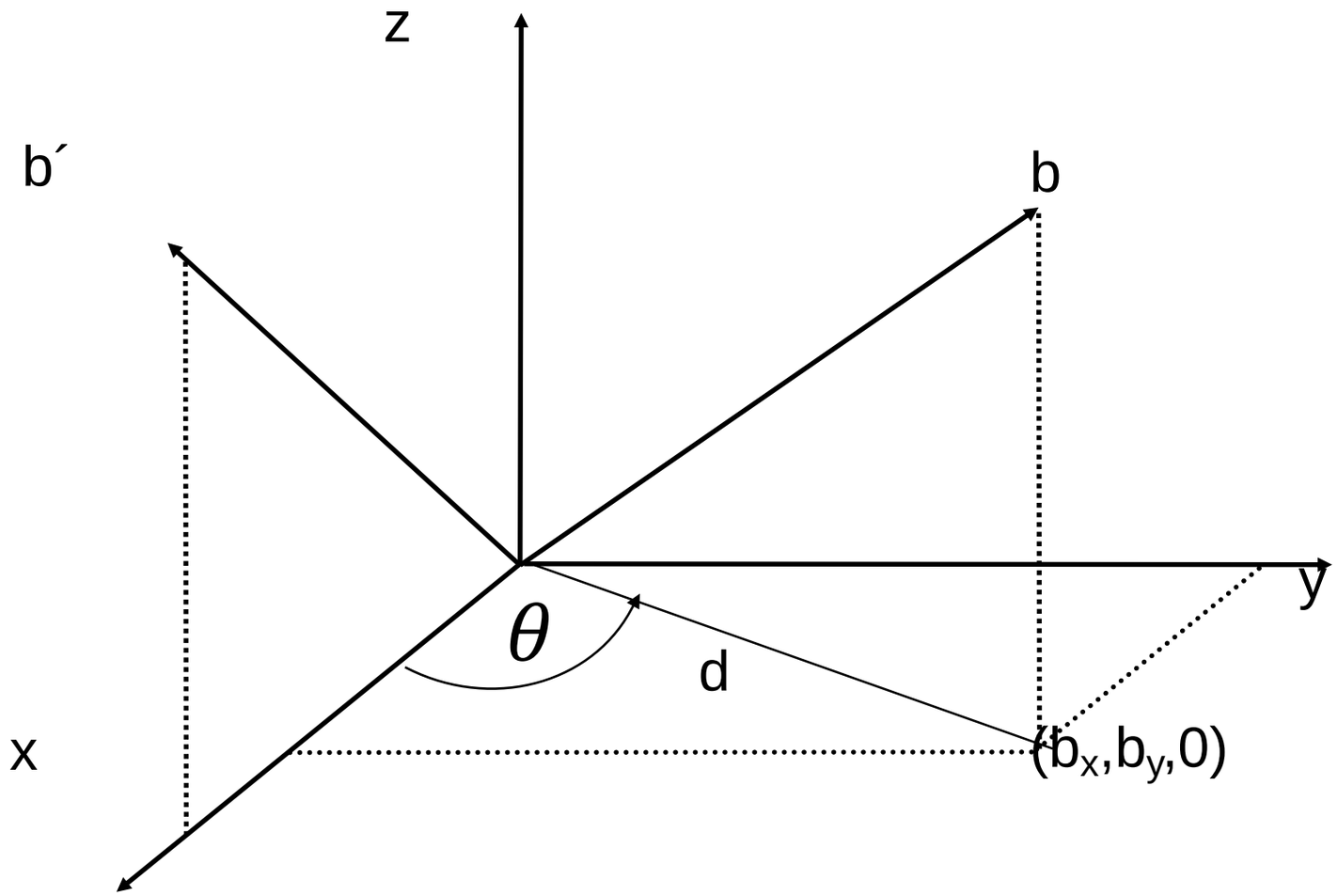
- Ziel: Rotation $R_G(\alpha)$ eines Punktes um eine beliebig orientierte Achse G im Raum um einen Winkel α .
- Sonderfall: Drehachse G geht durch den Ursprung und wird von einem Vektor $\mathbf{b} = (\mathbf{b}_x, \mathbf{b}_y, \mathbf{b}_z)$ mit $\|\mathbf{b}\| = 1$ generiert.

Rotation um eine beliebige Achse

- Gesucht: Koordinaten eines Punktes P nach einer Drehung um die Achse G um den Winkel α .
- Vorgehensweise:
  - Der Punkt P wird so transformiert, dass die Drehachse mit der z-Achse zusammenfällt.
  - Anschließend wird für die Drehung um α die Rotationsmatrix $R_z(\alpha)$ verwendet.
  - Hinterher werden die „Hilfstransformationen“ wieder Rückgängig gemacht.
  - Ist G mit der z-Achse identisch, entfallen die Hilfstransformationen.

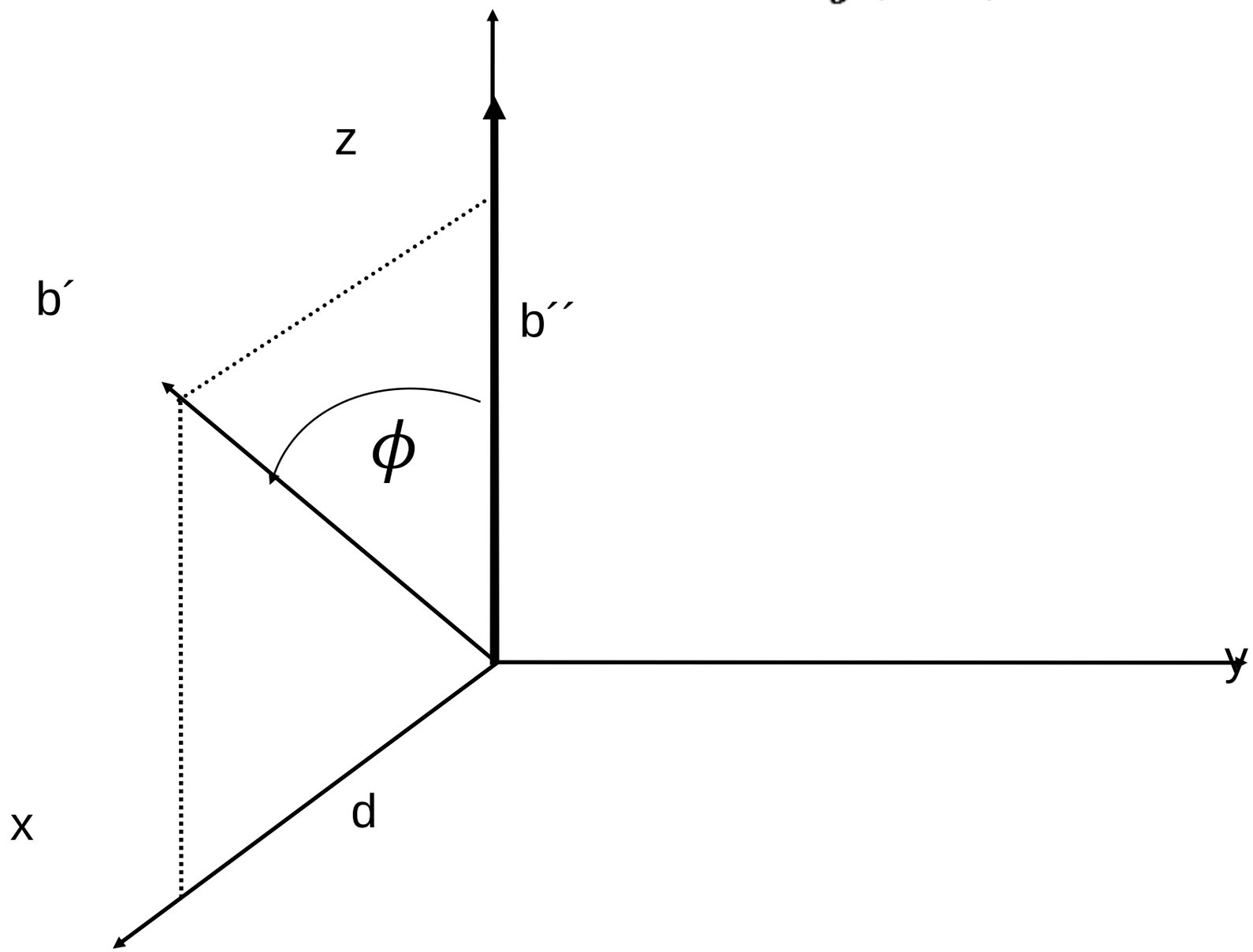
Rotation um eine beliebige Achse

$$R_z(-\theta)$$



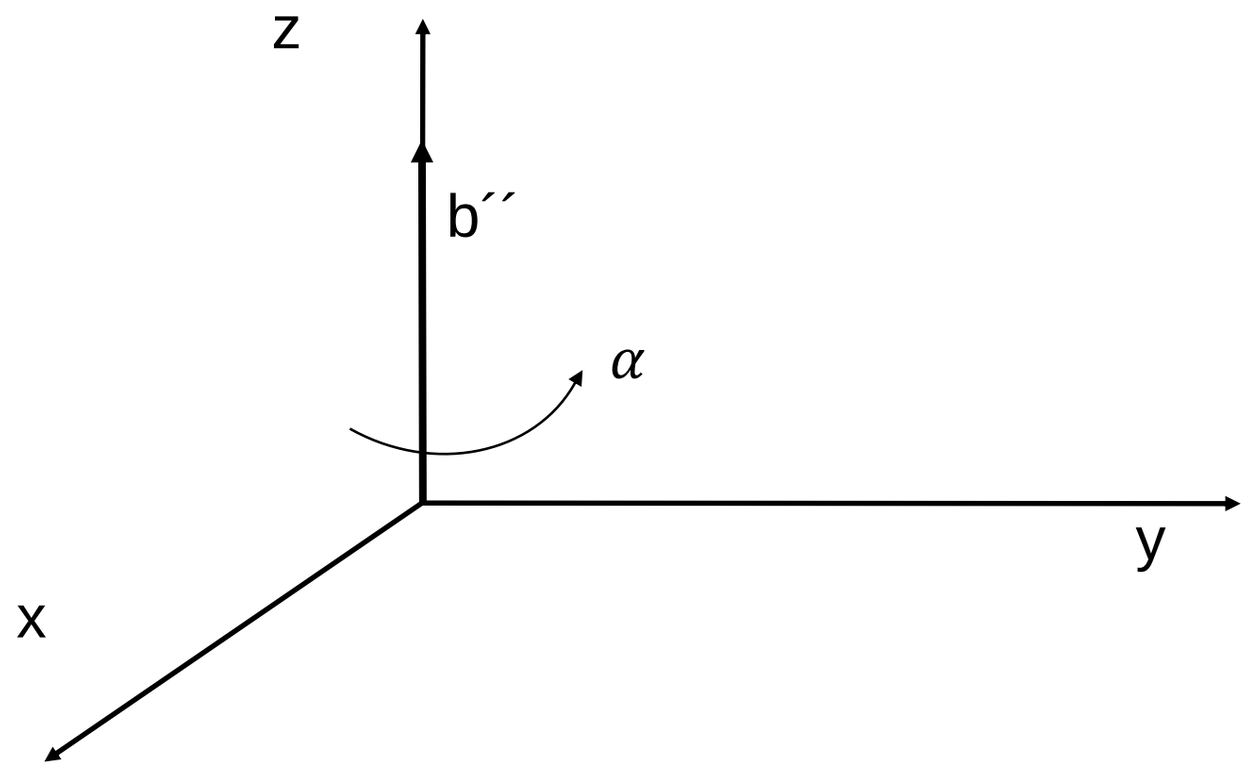
Rotation um eine beliebige Achse

- Schritt 2: Drehe b‘ auf die z-Achse $R_y(-\phi)$



Rotation um eine beliebige Achse

- Schritt 3: Drehung um z-Achse $R_z(\alpha)$.



- Die Rotationen aus den Schritten 1 und 2 werden in umgekehrter Reihenfolge wieder rückgängig gemacht.

Ergebnis: Rotation um beliebige Achse

- Ergebnis:
  Die Gesamttransformation lässt sich in einem Schritt durch die Verknüpfung aller Transformationen realisieren:

$$R_b(\alpha) = R_z(\theta)R_y(\phi)R_z(\alpha)R_y(-\phi)R_z(-\theta)$$

- Allgemeiner Fall:
  Ist die Drehachse eine allgemeine Gerade

$$G : a + \lambda \mathbf{b} \quad \lambda \in \mathbb{R}, \|\mathbf{b}\| = 1, a = (a_x, a_y, a_z)$$

  so ist vor Schritt 1 und nach Schritt 5 eine entsprechende Translation einzuschieben, also

$$R_G(\alpha) = T(a_x, a_y, a_z)R_z(\theta)R_y(\phi)R_z(\alpha)R_y(-\phi)R_z(-\theta)T(-a_x, -a_y, -a_z)$$