# Rotation einer Schachtel(Quader)

Ziel dieses Artikels ist die Darstellung einer rotierenden Schachtel mit Hilfe von Geogebra-2D. Dazu braucht es im Wesentlichen 2 Schritte:

1. Die Darstellung der Schachtel im Kamera-System (Augen-System)
2. Die Projektion dieser transformierten Schachtel auf ein Zeichenblatt - wobei sich die Frage ergibt: welche Flächen sind sichtbar?

Die mathematischen Grundlagen, die wir benötigen, ist die Darstellung von Translationen, Spiegelungen und Drehungen von Punkten bzw. Koordinatensystemen (KS). Bei diesen Abbildungen handelt es sich um lineare Abbildungen und sie sind daher mit Matrizen darstellbar.

Zuerst zu den **Drehungen**. Ein guter Überblick befindet sich in der Wikipedia.

Wichtig sind folgende Fakten:

- Passive Drehungen (also Drehung des KS) sind invers zu aktiven.
- $(R_\alpha)^{-1} = R_{-\alpha}$
- Drehmatrizen lassen das skalare Produkt (Winkel und Norm) invariant (Kongruenzabbildungen) und sind daher orthogonal!

  Es gilt nämlich:
  $$(A\,\vec{x}) \cdot \vec{y} = (a_{ij}x_j)y_i = x_j(a_{ij}y_i) = \vec{x} \cdot (A^T\vec{y})$$
  oder in Klammernnomenklatur des skalaren Produkts
  $$\langle A\,x, y\rangle = \langle x, A^T\,y\rangle$$
  Mit der Eigenschaft der Invarianz des skalaren Produkts und obiger Eigenschaft ergibt sich:
  $$\langle R\,x, R\,y\rangle = \langle x, y\rangle \Rightarrow \langle x, R^T\,R\,y\rangle = \langle x, y\rangle \Rightarrow R^T\,R = I$$
  also $(R_\alpha)^{-1} = (R_\alpha)^T$

Wir beschränken uns im Folgenden auf den $\mathbb{R}^3$.

Führt man homogene Koordinaten ein, lassen sich auch Translationen als Matrixmultiplikation darstellen. Die "Umrechnung" ist trivial - man führt eine 4.-te Koordinate ein und setzt sie 1.

$$\vec{x}_h = \begin{pmatrix} \vec{x} \\ 1 \end{pmatrix}$$

Alle Transformationsmatrizen leiten sich jetzt von $(4 \times 4)$ Einheitsmatrizen ab. Die für die Translation lässt sich jetzt schreiben:

$$\vec{c} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow T\,\vec{c} = \begin{pmatrix} x+1 \\ y+2 \\ z+3 \\ 1 \end{pmatrix}$$

Auch bei den Drehmatrizen führt das zu analogen Ergebnissen (hier wird $\vec{e}_1$ um die z-Achse um $\alpha$ gedreht):

$$R_{z,\alpha} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \Rightarrow R_{z,\alpha}\ \vec{e}_1 = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \\ 0 \\ 1 \end{pmatrix}$$

(Hinweis: Wer die Drehmatrizen herleiten möchte, braucht sie nur unbekannt ansetzen und auf die Basisvektoren $\vec{e}_i$ ansetzen mit bekanntem Ergebnis! Dadurch ergeben sich die 9 Variablen;
Übrigens mit $R_{z,\alpha}\, R_{z,\beta} = R_{z,\alpha+\beta}$ ergeben sich die Summensätze der cos- und sin-Funktion! )

Was wir jetzt noch brauchen ist die Spiegelung um die y-z-Ebene $M_{yz}$ - sie dreht das Vorzeichen der x-Koordinaten.

Hier noch die Zusammenfassung:

Transformationsmatrizen für homogene Koordinaten

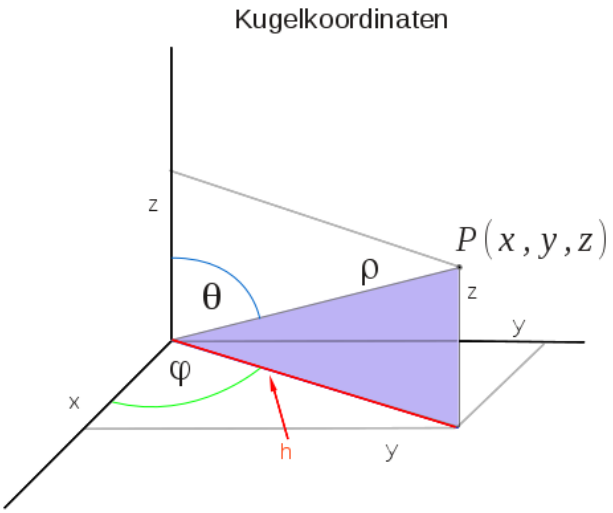
$$T = \begin{pmatrix} 1 & 0 & 0 & x_T \\ 0 & 1 & 0 & y_T \\ 0 & 0 & 1 & z_T \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad M_{yz} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_z = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

$R_x$ bezeichnet Drehung um x-Achse; beachte den anderen Aufbau von $R_y$
$M_{yz}$ ist Spiegelung um die y-z-Ebene(x-Koord. Wechseln Vorzeichen)

Was wir jetzt noch brauchen, um mit dem Rechnen zu beginnen, ist der Punkt, wo die Kamera (Augen) steht: dazu verwenden wir Kugelkoordinaten ($\rho$, $\theta$, $\varphi$).

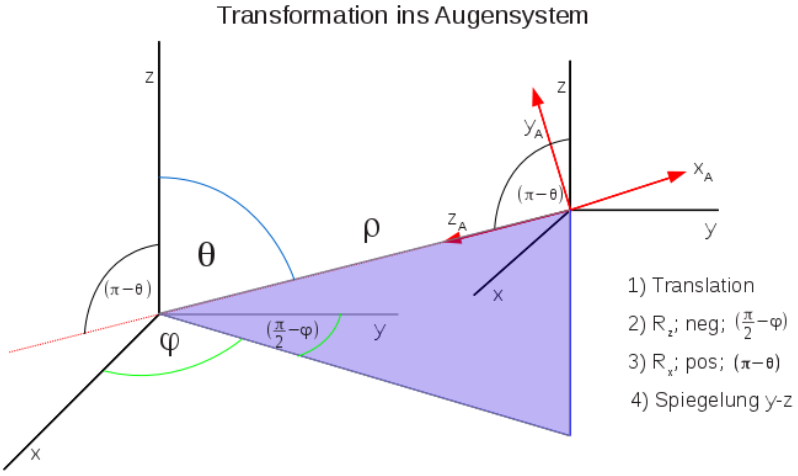
Die Umrechnung in kartesische Koordinaten sollte durch wiederholte Anwendung des "Pythagoras" keine Schwierigkeit darstellen:




Hier die Lösung

$z = ?$
$h = ?$
$y = ?$
$x = ?$

Das Augensystem wählen wir so, dass die $x_A$-Achse waagrecht nach rechts geht, die $y_A$-Achse nach oben (wie gewohnt) und die $z_A$-Achse von den Augen weg zum Ursprung des anderen Koordinatensystems.




Beachte, dass Transformationen von KS invers sind zu Transformationen zu Punkten

Hier ist einer der "springenden Punkte" in unserer Rechnung, darum schauen Sie sich die Zeichnung genau an und versuchen Sie die einzelnen Schritte nachzuvollziehen (auf den "alten" Koordinaten-Ursprung konzentrieren):

1. Translation - eh klar!
2. Drehung um die $z-$Achse im Uhrzeigersinn(math. negativ) bis $y-$Achse im blauen Dreieck liegt.
3. Jetzt Drehung um die $x-$Achse gegen den Uhrzeigersinn bis die $z-$Achse auf der Verbindungslinie der KS-Ursprünge "einrastet"
4. Zum Schluss das Umklappen der $x-$Achse (Spiegelung) - wir erhalten ein Linkssystem!

Die "Gesamtmatrix" - die das alles macht - lassen wir uns mit einem CAS berechnen. Ich habe *wxMaxima* verwendet: zum einen ist es kostenlos, zum anderen für meine Zwecke vollkommen ausreichend und für jede Plattform (ich arbeite seit 20 Jahren fast ausschließlich auf LINUX) erhältlich.

Rotationsmatrix um z-Achse mit $\alpha$

```
(%i1) R_z:matrix([cos(%alpha),-sin(%alpha),0,0],
                 [sin(%alpha),cos(%alpha),0,0],
                 [0,0,1,0],
                 [0,0,0,1]);
```

$$(\%o1)\quad \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotationsmatrix um x-Achse mit $\alpha$

```
(%i2) R_x:matrix([1,0,0,0],
                 [0,cos(%alpha),-sin(%alpha),0],
                 [0,sin(%alpha),cos(%alpha),0],
                 [0,0,0,1]);
```

$$(\%o2)\quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotationsmatrix um y-Achse mit $\alpha$

```
(%i3) R_y:matrix([cos(%alpha),0,sin(%alpha),0],
                 [0,1,0,0],
                 [-sin(%alpha),0,cos(%alpha),0],
                 [0,0,0,1]);
```

$$(\%o3)\quad \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translationsmatrix um Vektor (x_T, y_T, z_T)

(%i4)
```
T:matrix([1,0,0,x_T],
         [0,1,0,y_T],
         [0,0,1,z_T],
         [0,0,0,1]);
```

(%o4)
$$\begin{pmatrix} 1 & 0 & 0 & x\_T \\ 0 & 1 & 0 & y\_T \\ 0 & 0 & 1 & z\_T \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Spiegelungsmatrix um y-z-Ebene

(%i5)
```
M_yz:matrix([-1,0,0,0],
            [0,1,0,0],
            [0,0,1,0],
            [0,0,0,1]);
```

(%o5)
$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Wir setzen die entsprechenden Werte für x_T, y_T und z_T ein

(%i6)
```
x_T:%rho * sin(%theta) * cos(%phi)$
```

(%i7)
```
y_T:%rho * sin(%theta) * sin(%phi)$
```

(%i8)
```
z_T:%rho * cos(%theta)$
```

Evaluieren T mit diesen Werten neu und invertieren wegen KS-Transformation

(%i9)
```
A: invert(ev(T));
```

(%o9)
$$\begin{pmatrix} 1 & 0 & 0 & -\cos(\phi)\ \rho\sin(\theta) \\ 0 & 1 & 0 & -\sin(\phi)\ \rho\sin(\theta) \\ 0 & 0 & 1 & -\rho\cos(\theta) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Setzen für $\alpha$ in $R_z$ ein - Invertierung entfällt - hebt sich mit math. negativ auf!

(%i10)
```
%alpha:%pi/2 - %phi;
```

(%o10)
$$\frac{\pi}{2} - \phi$$

(%i11)
```
B: ev(R_z);
```

(%o11)
$$\begin{pmatrix} \sin(\phi) & -\cos(\phi) & 0 & 0 \\ \cos(\phi) & \sin(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

beide Transformationsmatrizen werden multipliziert und vereinfacht

```
(%i12) BA:B . A;
```

$$(\%o12)\quad \begin{pmatrix} \sin(\phi) & -\cos(\phi) & 0 & 0 \\ \cos(\phi) & \sin(\phi) & 0 & -\sin(\phi)^2 \rho \sin(\theta) - \cos(\phi)^2 \rho \sin(\theta) \\ 0 & 0 & 1 & -\rho\cos(\theta) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
(%i13) BA:trigsimp(BA);
```

$$(\%o13)\quad \begin{pmatrix} \sin(\phi) & -\cos(\phi) & 0 & 0 \\ \cos(\phi) & \sin(\phi) & 0 & -\rho\sin(\theta) \\ 0 & 0 & 1 & -\rho\cos(\theta) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

dieselbe Prozedur mit $R_x$ - wir nehmen $\alpha = \theta - \pi$ und ersparen uns so die Invertierung!

```
(%i14) %alpha:%theta - %pi;
```

$$(\%o14)\quad \theta - \pi$$

```
(%i15) C:ev(R_x);
```

$$(\%o15)\quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & -\cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
(%i16) CBA: C . BA;
```

$$(\%o16)\quad \begin{pmatrix} \sin(\phi) & -\cos(\phi) & 0 & 0 \\ -\cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\theta) & \sin(\theta) & 0 \\ -\cos(\phi)\sin(\theta) & -\sin(\phi)\sin(\theta) & -\cos(\theta) & \rho\sin(\theta)^2 + \rho\cos(\theta)^2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
(%i17) CBA:trigsimp(CBA);
```

$$(\%o17)\quad \begin{pmatrix} \sin(\phi) & -\cos(\phi) & 0 & 0 \\ -\cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\theta) & \sin(\theta) & 0 \\ -\cos(\phi)\sin(\theta) & -\sin(\phi)\sin(\theta) & -\cos(\theta) & \rho \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Zum Schluss noch die Spiegelungsmatrix - beachte Spiegelungsmatrizen sind ihre eigenen Inverse!

```
(%i18) Tr:M_yz . CBA;
```

$$(\%o18)\quad \begin{pmatrix} -\sin(\phi) & \cos(\phi) & 0 & 0 \\ -\cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\theta) & \sin(\theta) & 0 \\ -\cos(\phi)\sin(\theta) & -\sin(\phi)\sin(\theta) & -\cos(\theta) & \rho \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

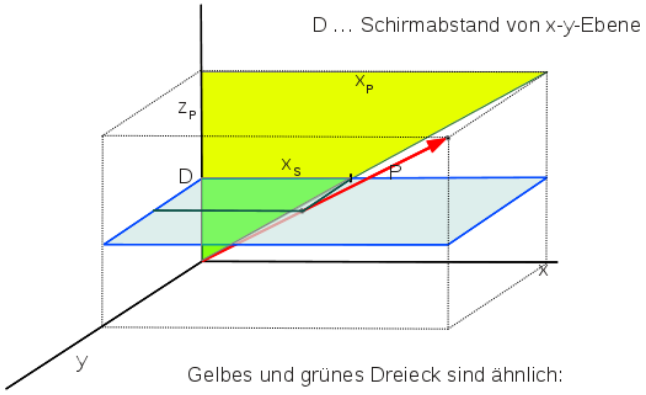
$$Tr = \begin{pmatrix} -\sin(\phi) & \cos(\phi) & 0 & 0 \\ -\cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\theta) & \sin(\theta) & 0 \\ -\cos(\phi)\sin(\theta) & -\sin(\phi)\sin(\theta) & -\cos(\theta) & \rho \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hier endlich das Endergebnis - 1 Matrix für Alles!

Damit wäre der erste Teil - Darstellung der Schachtel im Augensystem - erledigt.

Bleibt die perspektivische Projektion der Augenkoordinaten auf das Zeichenblatt - das ist allerdings nur mehr eine Angelegenheit von "ähnlichen Dreiecken". Schauen wir uns dazu die folgende Zeichnung an:

Berechnung der Bildschirmkoordinaten




Gelbes und grünes Dreieck sind ähnlich:

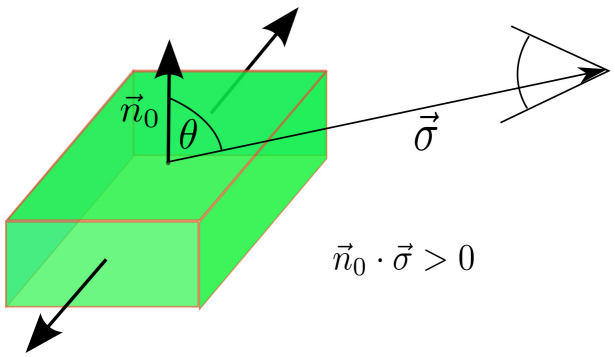
$$\frac{x_P}{z_P} = \frac{x_S}{D} \Rightarrow x_S = D\frac{x_P}{z_P}$$

Das schwarze KS sei unser Augensystem, der rote Pfeil ist ein Sehstrahl zu einem Punkt $P(x_P, y_P, z_P)$ der Schachtel. Die blaue Fläche ($D$-Einheiten Abstand parallel zur $x$-$y$-Ebene) sei das Zeichenblatt(Schirm). Was sind die Koordinaten $(x_S, y_S)$ dieses Punktes am Schirm? Aus den ähnlichen Dreiecken ergibt sich:

$$\boxed{\xi_S = \frac{D}{z_P}\,\xi_P \qquad \xi \in \{x,\, y\}}$$

Für $0 < D < z_P$ ergibt sich eine Verkleinerung.

So noch ein paar Worte zur **Sichtbarkeit** einer Fläche:




$$\vec{n}_0 \cdot \vec{\sigma} > 0$$

Wie man auf der Zeichnung sehen kann, ist eine Fläche nur dann sichtbar, wenn der Winkel zwischen dem Vektor $\vec{\sigma}$ und dem Normalvektor der Fläche (nach außen) spitz ist! $\vec{\sigma}$ beginnt bei einem beliebigen Punkt der Fläche (günstig ist ein Eckpunkt - der gilt gleich für mehrere Flächen) und der Augenposition. Bilden die Schachtelkanten ein Basis-Dreibein gilt $\vec{n}_0 = \pm\vec{e}_i$ und damit:

$$\pm\vec{e}_i \cdot \left[\rho \begin{pmatrix} \cos\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\theta \end{pmatrix} - \vec{p}\right] > 0$$

Da bei den $\vec{e}_i$ nur jeweils 1 Koordinate nicht verschwindet, entsteht eine skalare Sichtbarkeitsbedingung, die man in Geogebra bei den Eigenschaften des Polygons eintragen kann!
Hier nocheinmal die Zusammenfassung der Implementation in Geogebra
Hier die "fertige" Datei zum Ausprobieren - (Geogebra muss auf ihrem Computer installiert sein!)