# Einführung in die Spieleprogrammierung

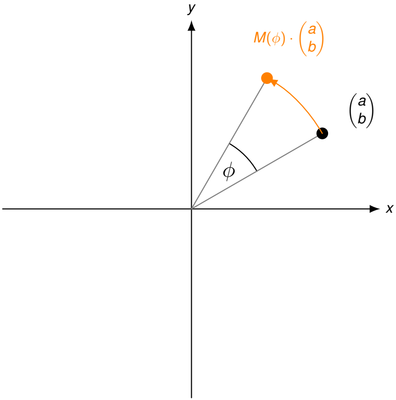
## Rotationen

Tobias Huber

huber@hcm-lab.de

May 13, 2019

- Rotationen in 2D
- Eulerwinkel
- Quaternionen




Im $\mathbb{R}^2$ kann jede Drehung (gegen den Uhrzeigersinn) um einen Winkel $\phi$ durch ein Matrix

$$M(\phi) = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix}$$

dargestellt werden.

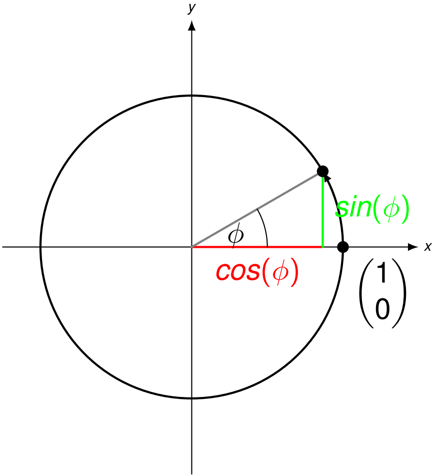
# Rotationen in 2D

Da Drehung linear ist (kann man beweisen müssen wir nicht) reicht es diese Matrix auf den Basis Vektoren $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ und $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ zu überprüfen, da:

$$\begin{pmatrix} a \\ b \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1}$$

$$\implies M(\phi) \cdot \begin{pmatrix} a \\ b \end{pmatrix} = a\big(M(\phi) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}\big) + b\big(M(\phi) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}\big) \tag{2}$$

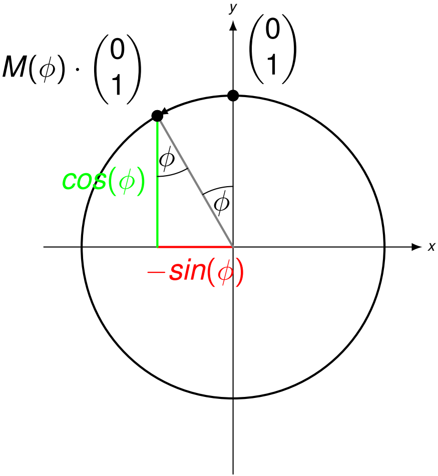
Universität Augsburg University




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

$$= \begin{pmatrix} cos(\phi) \\ sin(\phi) \end{pmatrix}$$

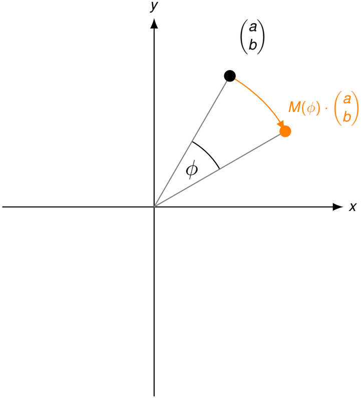
Das passt, da wir hier ein Dreieck mit rechtem Winkel und Hypothenuse der Länge 1 haben.




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

$$= \begin{pmatrix} -sin(\phi) \\ cos(\phi) \end{pmatrix}$$

Bisher haben wir mathematisch positiv gedreht ( von $(1, 0)$ nach $(0, 1)$ ) Was wenn wir links herum drehen wollen:



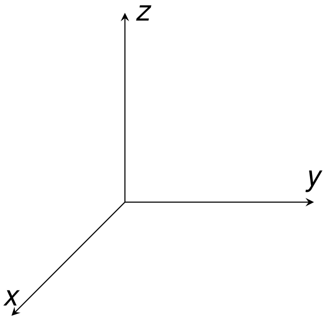

In diesem Fall erhalten wir die Drehmatrix durch einsetzen des negativen Winkels.

$$M(-\phi) = \begin{bmatrix} cos(-\phi) & -sin(-\phi) \\ sin(-\phi) & cos(-\phi) \end{bmatrix}$$

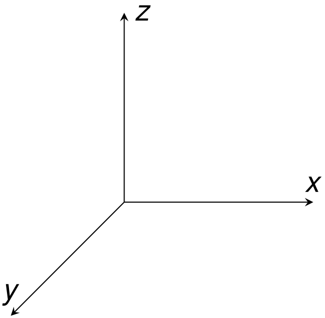
$$= \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix}$$

- Rotationen in 2D
- Eulerwinkel
- Quaternionen

Rechthand Koordinatensystem. (Blender)




Linkshand Koordinatensystem. (Unity)



# Eulerwinkel

Im $\mathbb{R}^3$ können wir alle mögichen Drehungen durch die sogenannten Eulerwinkel, die Drehungen um die Koordinatenachsen, darstellen.

Rotation um x-Achse: $$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}$$

Rotation um y-Achse: $$\begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix}$$

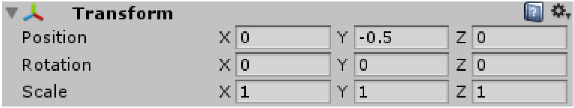
Hierbei dreht man um die y-Achse andersherum, da wir nach der Linkehandregel von $z$ nach $x$ drehen wollen.

Rotation um z-Achse: $$\begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Eulerwinkel

Universität Augsburg University

Mit diesen Rotationsmatrizen kann jede Drehung durch drei xyz-Eulerwinkel (eigentlich Kardan oder Tait-Bryan-Winkel) $\alpha, \beta, \gamma$(yaw, pitch and roll angle) dargestellt werden.




$$R(\alpha, \beta, \gamma) =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Unity verwendet yxz-Eulerwinkel ( zuert um z dann x dann y).

Problem "Gimbal Lock":
Für $\beta = 90^\circ$ gilt $sin(\beta) = 1$ und $\cos(\beta) = 0$. Somit erhalten wir

$$R(\alpha, \beta, \gamma) = \tag{3}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

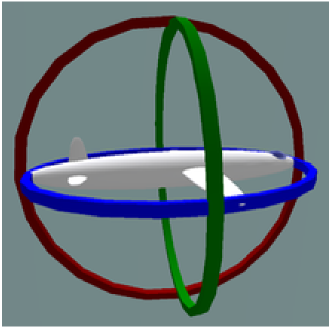
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ sin(\gamma) & cos(\gamma) & 0 \\ -cos(\gamma) & sin(\gamma) & 0 \end{bmatrix} \tag{5}$$

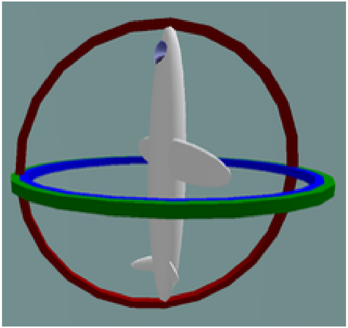
$$= \begin{bmatrix} 0 & 0 & 1 \\ cos(\alpha)sin(\gamma) + sin(\alpha)cos(\gamma) & cos(\alpha)cos(\gamma) - sin(\alpha)sin(\gamma) & 0 \\ sin(\alpha)sin(\gamma) - cos(\alpha)cos(\gamma) & sin(\alpha)cos(\gamma) + cos(\alpha)sin(\gamma) & 0 \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ sin(\alpha + \gamma) & cos(\alpha + \gamma) & 0 \\ -cos(\alpha + \gamma) & sin(\alpha + \gamma) & 0 \end{bmatrix} \tag{7}$$

Universität Augsburg University

Bei der endgültigen Matrix (7) sieht man, dass eine Änderung von $\alpha$ den gleichen Effekt hat wie dieselbe Änderung von $\gamma$.
$\Longrightarrow$ Verlust eines Freiheitsgrades.
Grafisch:





'How about sending me a fourth gimbal for Christmas'- Michael Collins auf der Apollo 11 Mission.

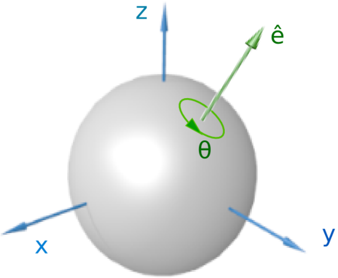
# Eulerwinkel

- Vorteile von Eulerwinkeln:
  - Intuitiv für Menschen verständlich
- Nachteile:
  - Anwendung der Rotation in mehreren Schritten
    - Reihenfolge ist wichtig
    - Gimbal-Lock
  - Mehrdeutigkeit (z. B. 180°Rotation um z-Achse = 180°Rotation um x-Achse + 180° um die y-Achse)
- Quaternionen

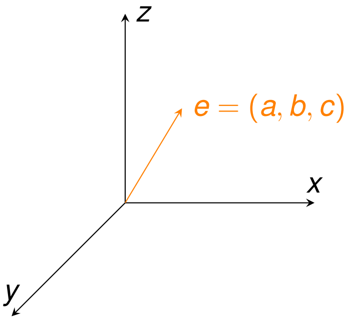
Universität Augsburg University

- Rotationen in 2D
- Eulerwinkel
- Quaternionen

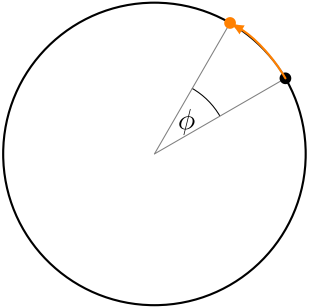
Jede Sequenz von Drehungen um den Ursprung entpsricht einer einzigen Drehung um eine Drehachse *e*. (Eulers Rotationstheorem/Der Satz vom Fußball)



Die Drehachse $e$ lässt sich durch einen Punkt auf der Einheitskugel (also einem Einheitsvektor) darstellen:




Den Drehwinkel $\theta$ kann man durch ein skalar $w \in \mathbb{R}$ darstellen.




- Drehung als Quadruple $(w, a, b, c)$

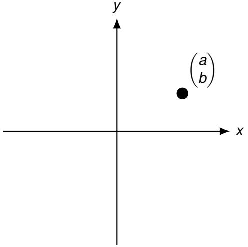
# Quaternionen

- Warum nicht einfach als Vektor in $\mathbb{R}^4$ ?
- Es gibt keine 'vernünftige' (nullteilerfreie) Multiplikation auf $\mathbb{R}^4$.
  - Wollen: $a * b = a * c \implies b = c$
  - Aber: $\begin{pmatrix}1\\0\end{pmatrix} * \begin{pmatrix}0\\2\end{pmatrix} = 0 = \begin{pmatrix}1\\0\end{pmatrix} * \begin{pmatrix}0\\4\end{pmatrix} \implies \begin{pmatrix}0\\2\end{pmatrix} = \begin{pmatrix}0\\4\end{pmatrix}$ ???
- Quaternionen besitzen Multiplikation, analog zu $\mathbb{C}$ als Erweiterung von $\mathbb{R}^2$.

Universität Augsburg University

# Quaternionen

- Erinnerung an $\mathbb{C}$:




Dieser Punkt entspricht $a + ib \in \mathbb{C}$, Multiplikation ergibt sich durch $i^2 = -1$ und das Distributiv Gesetz. (Fun Fact: Multiplication mit Punkten auf dem Einheitskreis entspricht einer Drehung im 2D-Raum, wie auf den ersten Folien)

- Quaternionen $\mathbb{H}$:
  Der Punkt $(w, a, b, c)$ entspricht $w + ai + bj + ck$ und Multiplikation ergibt sich durch: $i^2 = j^2 = k^2 = ijk = -1$.

# Rotation mit Quternion

- Quaternion $q$ für eine Drehung um Drehachse $v = (v_1, v_2, v_3)$ um Drehwinkel $\theta$ ist gegeben durch:

$$q = (cos(\theta/2), sin(\theta/2)v_1, sin(\theta/2)v_2, sin(\theta/2)v_3)$$

- Als Quaternion:

$$q = cos(\theta/2) + sin(\theta/2)(v_1 i + v_2 j + v_3 k)$$

- Einen beliebigen Vektor $x$ dreht man dann, indem man ihn als Quaternion mit Realteil 0 schreibt ( $x = 0 + x_1 i + x_2 j + x_3 k$ ) und dann

$$qxq^{-1}$$

berechnet.

Universität
Augsburg
University

Nützliche Quellen:
Rotationen in Unity:
https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity
https://docs.unity3d.com/ScriptReference/Transform.Rotate.html
Quaternions Stuff:
https://www.youtube.com/watch?v=d4EgbgTm0Bg
https://www.youtube.com/watch?v=zjMuIxRvygQ