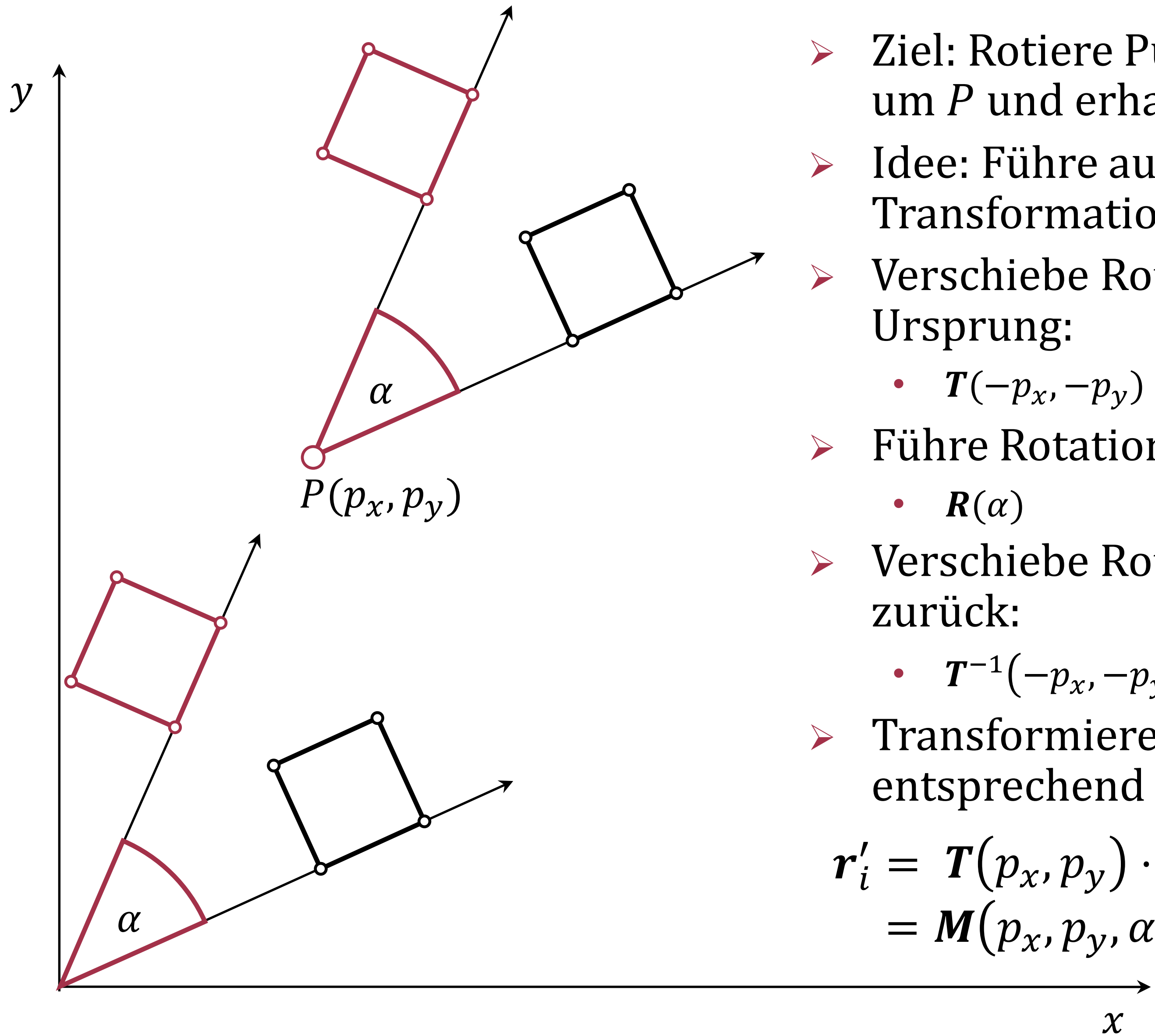
# Computergrafik

Universität Osnabrück, Henning Wenke, 2012-05-07

# Noch Kapitel III: Transformationen

# 2D Rotation um freies Rotationszentrum

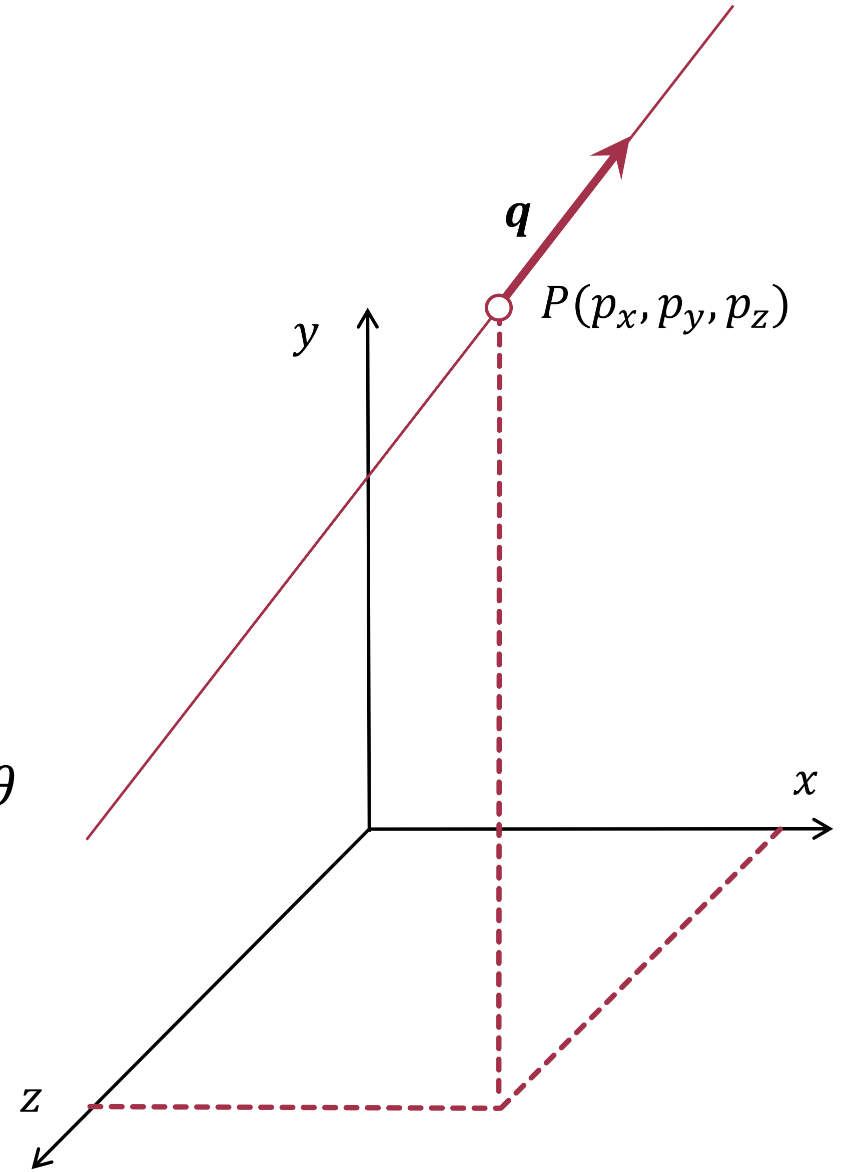



- Ziel: Rotiere Punkte $\boldsymbol{r}_i$ um Winkel $\alpha$ um $P$ und erhalte $\boldsymbol{r}'_i$
- Idee: Führe auf elementare Transformationen zurück
- Verschiebe Rotationszentrum in Ursprung:
  - $\boldsymbol{T}(-p_x, -p_y)$
- Führe Rotation um Ursprung aus:
  - $\boldsymbol{R}(\alpha)$
- Verschiebe Rotationszentrum zurück:
  - $\boldsymbol{T}^{-1}(-p_x, -p_y) = \boldsymbol{T}(p_x, p_y)$
- Transformiere Koordinaten $\boldsymbol{r}_i$ entsprechend gemäß:

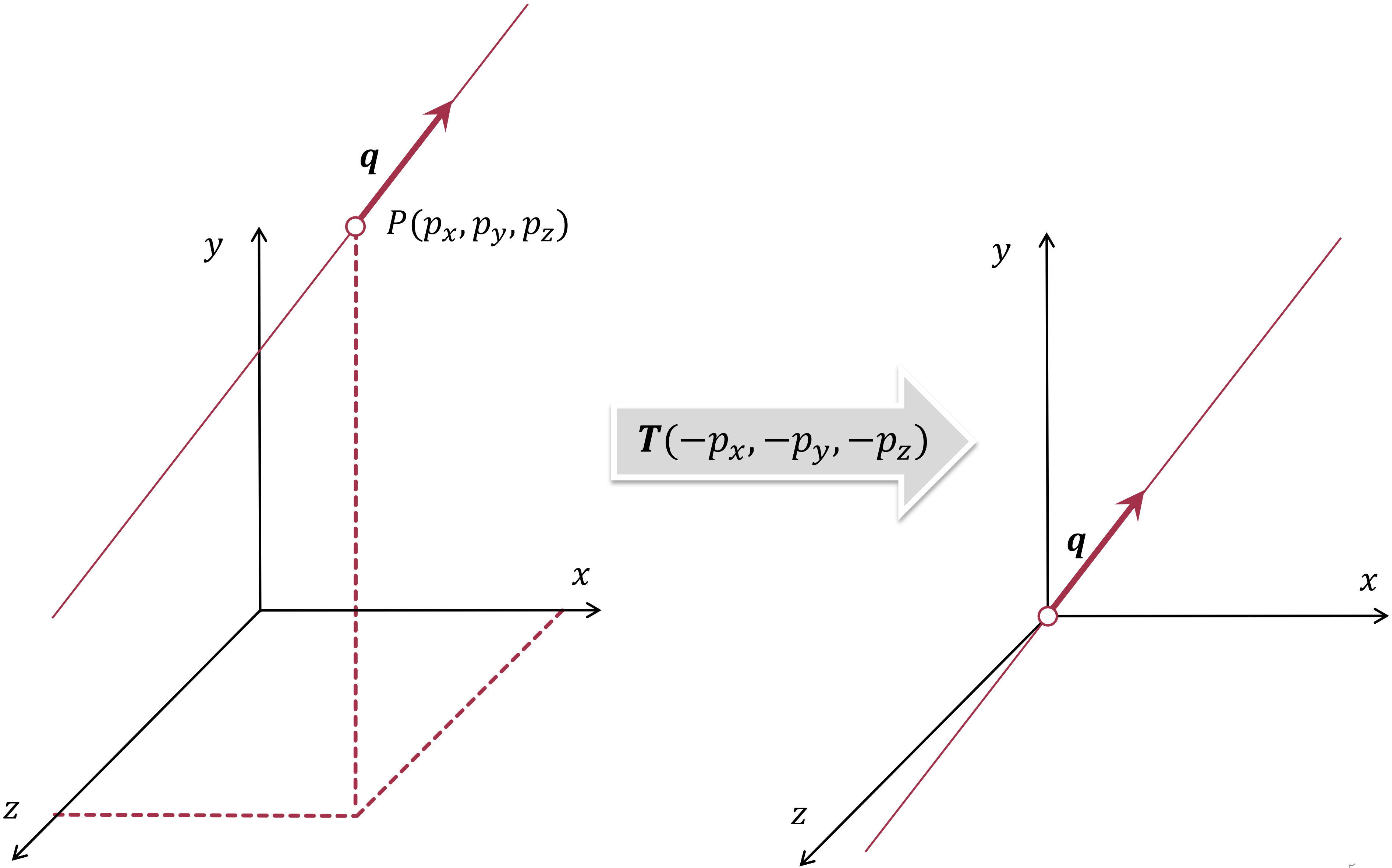
$$\begin{aligned} \boldsymbol{r}'_i &= \boldsymbol{T}(p_x, p_y) \cdot \boldsymbol{R}(\alpha) \cdot \boldsymbol{T}(-p_x, -p_y) \cdot \boldsymbol{r}_i \\ &= \boldsymbol{M}(p_x, p_y, \alpha) \cdot \boldsymbol{r}_i \end{aligned}$$

# Rotation um beliebige Achse

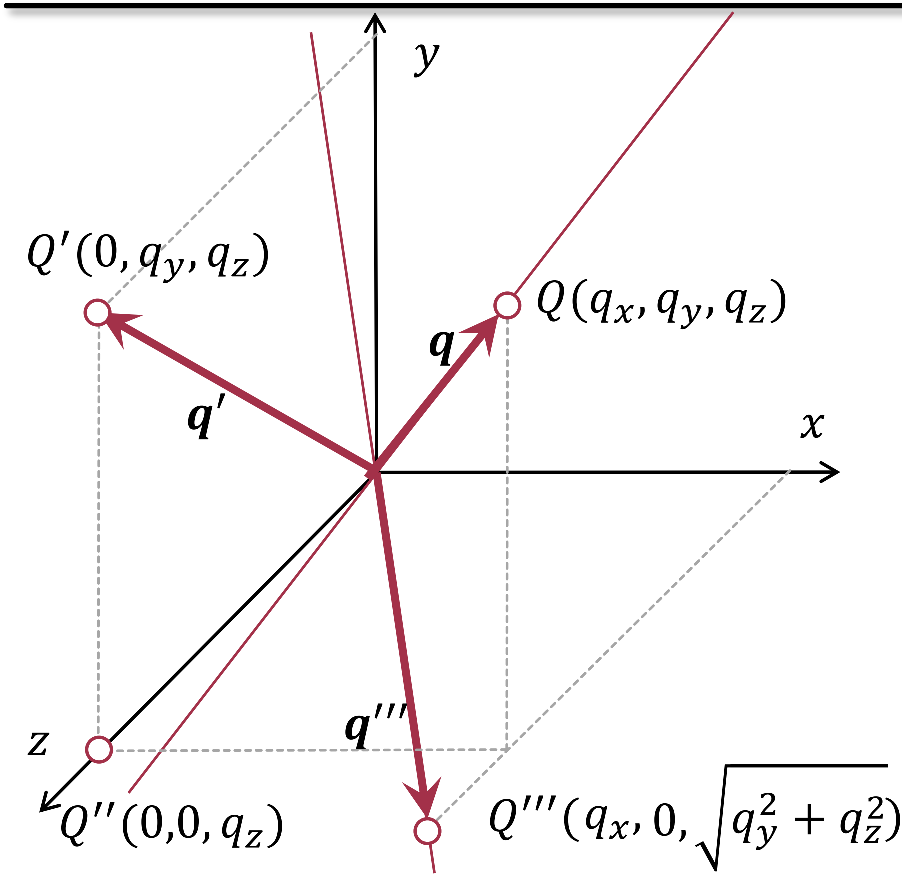
- Ziel: Rotiere um beliebige Achse um Winkel $\theta$
- Gegeben: Rotationsachse, definiert durch:
  - Punkt $P$
  - Normierter Richtungsvektor $\boldsymbol{q}$
- Idee:
  - Transformiere Rotationsachse z.B. in z-Achse
  - Rotiere um z-Achse um Winkel $\theta$
  - Rücktransformiere Rotationsachse
  - Wende resultierende Matrix auf alle Koordinaten an



# I. Translation, sodass Ursprungsgerade



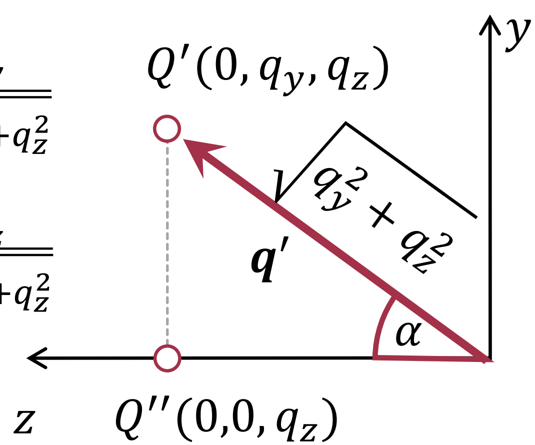
# II. Rotation um x-Achse in xz-Ebene




- Gegeben: Ursprungsgerade mit norm. Richtungsvektor $\boldsymbol{q}$
- Gesucht: Matrix $\boldsymbol{R_x}(\alpha)$ zur Rotation in Gerade in xz-Ebene mit Richtungsvektor $\boldsymbol{q}'''$
- Bestimme Projektion $\boldsymbol{q}'$ von $\boldsymbol{q}$ in yz-Ebene
- Berechne Winkel $\alpha$ (bzw. sin/cos davon) zwischen $\boldsymbol{q}'$ und z-Achse
- Dann gilt: $\boldsymbol{q}''' = \boldsymbol{R_x}(\alpha) \cdot \boldsymbol{q}$

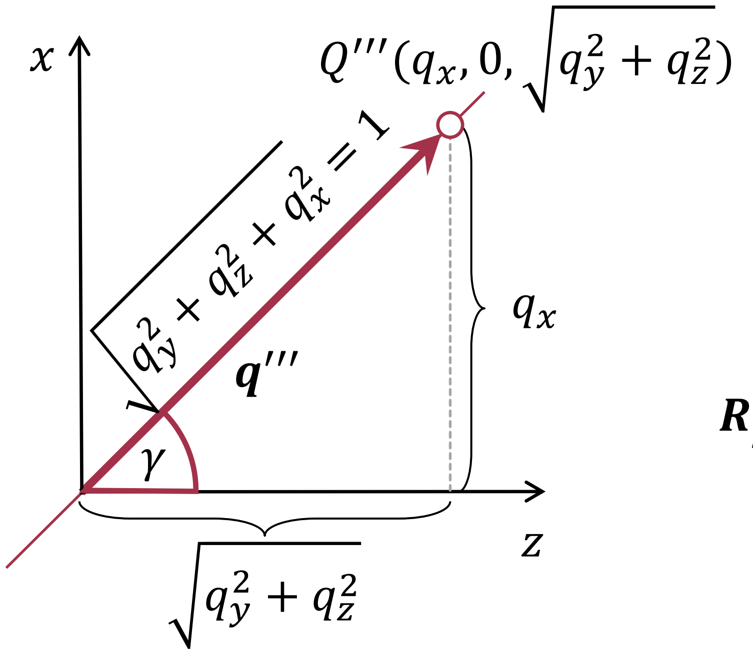
$$\sin(\alpha) = \frac{q_y}{\sqrt{q_y^2+q_z^2}}$$

$$\cos(\alpha) = \frac{q_z}{\sqrt{q_y^2+q_z^2}}$$




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

# III. Rotation um y-Achse in z-Achse




$\sin(\gamma) = q_x$

$\cos(\gamma) = \sqrt{q_y^2+q_z^2}$

- Gegeben: Ursprungsgerade in xz-Ebene mit norm. Richtungsvektor $\boldsymbol{q}'''$
- Gesucht: Matrix $\boldsymbol{R_y}(\beta)$ zur Rotation in Gerade in z-Achse

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

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

$$= \begin{pmatrix} \sqrt{q_y^2+q_z^2} & 0 & -q_x & 0 \\ 0 & 1 & 0 & 0 \\ q_x & 0 & \sqrt{q_y^2+q_z^2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Probe:

$$\boldsymbol{R_y}(\beta) \cdot \boldsymbol{q}''' = \begin{pmatrix} 0 \\ 0 \\ \boldsymbol{q}_x^2+\boldsymbol{q}_y^2+\boldsymbol{q}_z^2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

# IV. Rotation um z-Achse um Winkel $\theta$

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

# V. Rücktransformation zu Ausgangsachse

- Inverse von III:
  - $\boldsymbol{R}_{\boldsymbol{y}}^{-1}(\beta) = \boldsymbol{R}_{\boldsymbol{y}}(-\beta)$
- Inverse von II:
  - $\boldsymbol{R}_{\boldsymbol{x}}^{-1}(\alpha) = \boldsymbol{R}_{\boldsymbol{x}}(-\alpha)$
- Inverse von I:
  - $\boldsymbol{T}^{-1}\left(-p_x, -p_y, -p_z\right) = \boldsymbol{T}(p_x, p_y, p_z)$

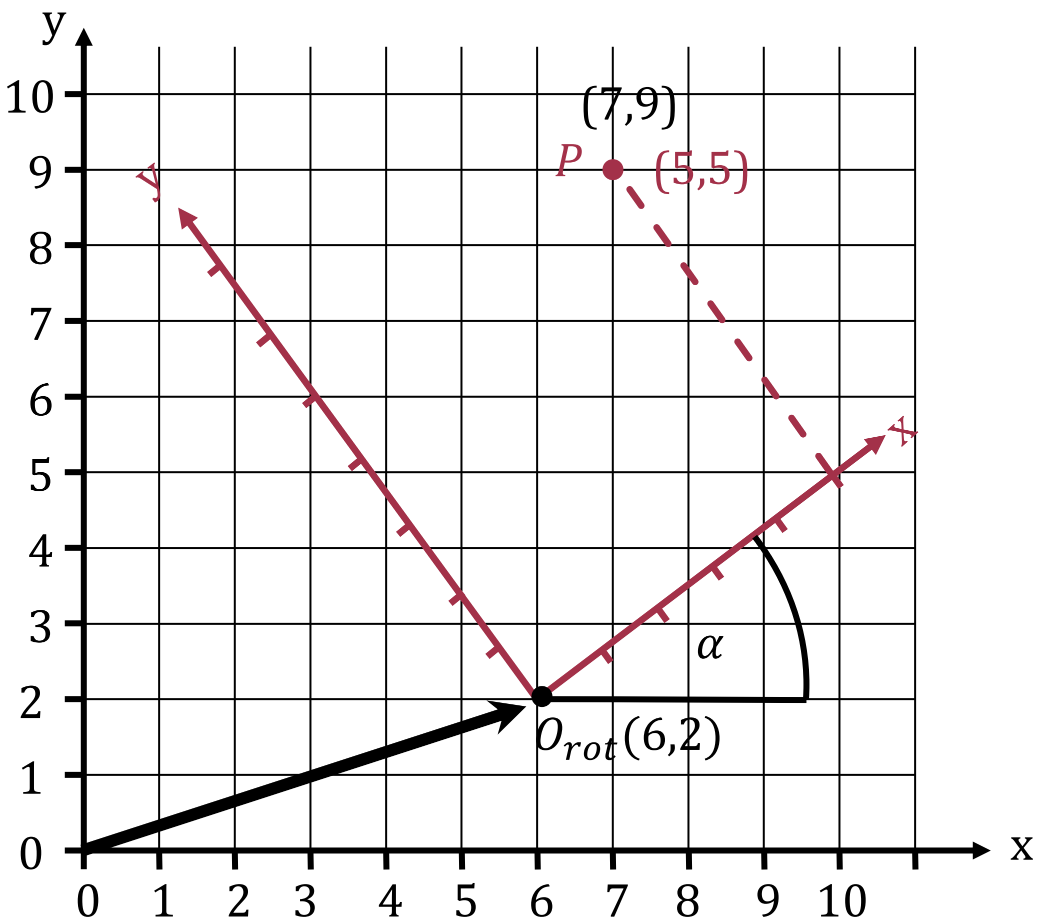
# Gesamttransformation

$$\begin{aligned}
\boldsymbol{R}(\boldsymbol{p},\boldsymbol{q},\theta) = \ & \boldsymbol{T}^{-\mathbf{1}}(-p_x,-p_y,-p_z) \\
& \cdot \boldsymbol{R}_{\boldsymbol{x}}^{-\mathbf{1}}\,(\alpha) \\
& \cdot \boldsymbol{R}_{\boldsymbol{y}}^{-1}\,(\beta) \\
& \cdot \boldsymbol{R}_{\boldsymbol{z}}\,(\theta) \\
& \cdot \boldsymbol{R}_{\boldsymbol{y}}\,(\beta) \\
& \cdot \boldsymbol{R}_{\boldsymbol{x}}\,(\alpha) \\
& \cdot \boldsymbol{T}(-p_x,-p_y,-p_z)
\end{aligned}$$

# 3.4

# Wechsel zwischen kartesischen Koordinatensystemen

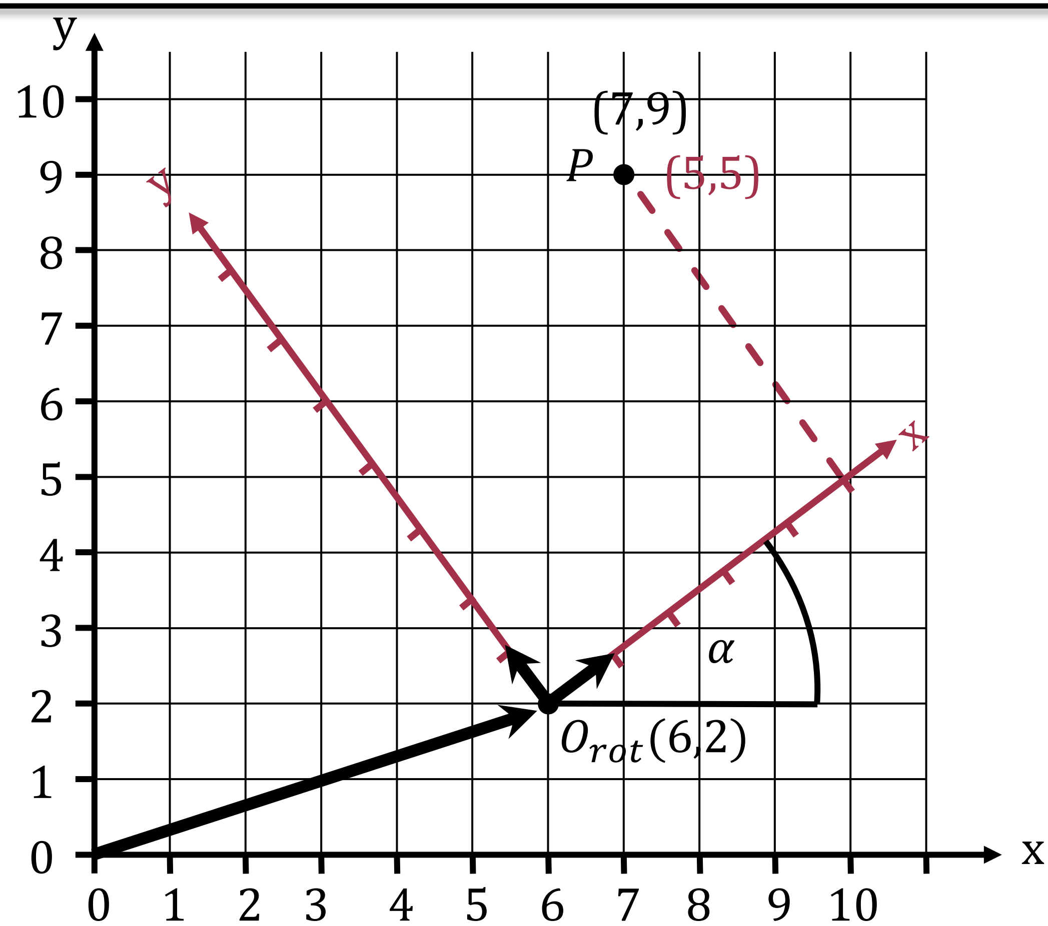
# Beschreibe P aus Sicht des schwarzen KS




- Transformiere schwarzes in rotes Koordinatensystem
- Drehe um $\alpha$ CCW
  - $cos(\alpha) = 4/5 = 0.8$
  - $sin(\alpha) = 3/5 = 0.6$
  - $\boldsymbol{R}(\alpha) = \begin{pmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- Verschiebe um $O_{rot}$
  - $\boldsymbol{T}(O_{rot}) = \begin{pmatrix} 1 & 0 & 6 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$
- Matrix für KS-Wechsel:
  - $\boldsymbol{M}_{rot \to schwarz} = \boldsymbol{T}(O_{rot}) \cdot \boldsymbol{R}(\alpha)$
  - $= \begin{pmatrix} 0.8 & -0.6 & 6 \\ 0.6 & 0.8 & 2 \\ 0 & 0 & 1 \end{pmatrix}$

$$\boldsymbol{p}_{schwarz} = \boldsymbol{M}_{rot \to schwarz} \cdot \boldsymbol{p}_{rot} = \begin{pmatrix} 0.8 & -0.6 & 6 \\ 0.6 & 0.8 & 2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 5 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \\ 9 \\ 1 \end{pmatrix}$$

# Beobachtung



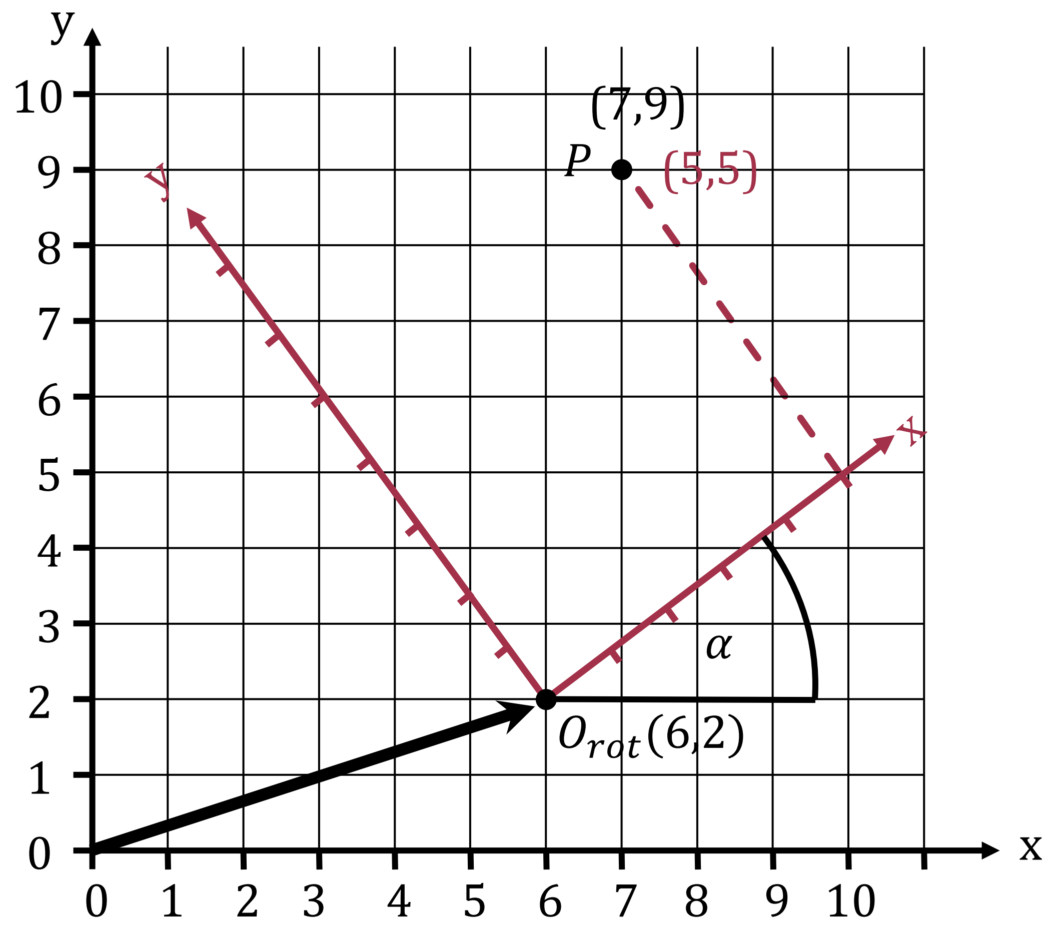

$$\boldsymbol{M_{rot \to schwarz}} = \begin{pmatrix} \boxed{\begin{matrix} 0.8 \\ 0.6 \\ 0 \end{matrix}} & \boxed{\begin{matrix} -0.6 \\ 0.8 \\ 0 \end{matrix}} & \boxed{\begin{matrix} 6 \\ 2 \\ 1 \end{matrix}} \end{pmatrix}$$

→ Ursprung von rot, beschrieben in schwarz

→ $\boldsymbol{e_y}$ von rot, beschrieben in schwarz

→ $\boldsymbol{e_x}$ von rot, beschrieben in schwarz

# Beschreibe P aus Sicht des roten KS




$$\boldsymbol{R}(\alpha) = \begin{pmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \boldsymbol{T}(O_{rot}) = \begin{pmatrix} 1 & 0 & 6 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

- Matrix für KS-Wechsel rot nach schwarz:
  - $\boldsymbol{M}_{\boldsymbol{r}\to\boldsymbol{s}} = \boldsymbol{T}(O_{rot}) \cdot \boldsymbol{R}(\alpha)$
  - $= \begin{pmatrix} 0.8 & -0.6 & 6 \\ 0.6 & 0.8 & 2 \\ 0 & 0 & 1 \end{pmatrix}$
- Matrix für KS-Wechsel schwarz nach rot:
  - $\boldsymbol{M}_{\boldsymbol{s}\to\boldsymbol{r}} = \boldsymbol{M}_{\boldsymbol{r}\to\boldsymbol{s}}^{-1}$
  - $\boldsymbol{M}_{\boldsymbol{s}\to\boldsymbol{r}} = \boldsymbol{R}(\alpha)^{-1} \cdot \boldsymbol{T}(O_{rot})^{-1}$
  - $\boldsymbol{M}_{\boldsymbol{s}\to\boldsymbol{r}} = \boldsymbol{R}(-\alpha) \cdot \boldsymbol{T}(-O_{rot})$
  - $\boldsymbol{M}_{\boldsymbol{s}\to\boldsymbol{r}} = \begin{pmatrix} 0.8 & 0.6 & -6 \\ -0.6 & 0.8 & -2 \\ 0 & 0 & 1 \end{pmatrix}$
- Test
  - $\boldsymbol{M}_{\boldsymbol{s}\to\boldsymbol{r}} \cdot P_s = P_r$
  - $\boldsymbol{M}_{\boldsymbol{s}\to\boldsymbol{r}} \cdot (7{,}9{,}1) = (5{,}5{,}1)$

# Koordinatensystemwechsel

$$\boldsymbol{M}_{\boldsymbol{B}\to\boldsymbol{A}} \cdot \boldsymbol{p}_{\boldsymbol{B}} = \boldsymbol{p}_{\boldsymbol{A}} = \begin{pmatrix} e^A_{(x,x)} & e^A_{(y,x)} & e^A_{(z,x)} & O^A_x \\ e^A_{(x,y)} & e^A_{(y,y)} & e^A_{(z,y)} & O^A_y \\ e^A_{(x,z)} & e^A_{(y,z)} & e^A_{(z,z)} & O^A_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p^B_x \\ p^B_y \\ p^B_z \\ 1 \end{pmatrix}$$

Punkt beschrieben in A

$\boldsymbol{e}_x$ von B

$\boldsymbol{e}_y$ von B

$\boldsymbol{e}_z$ von B

Ursprung von B

Punkt beschrieben in B

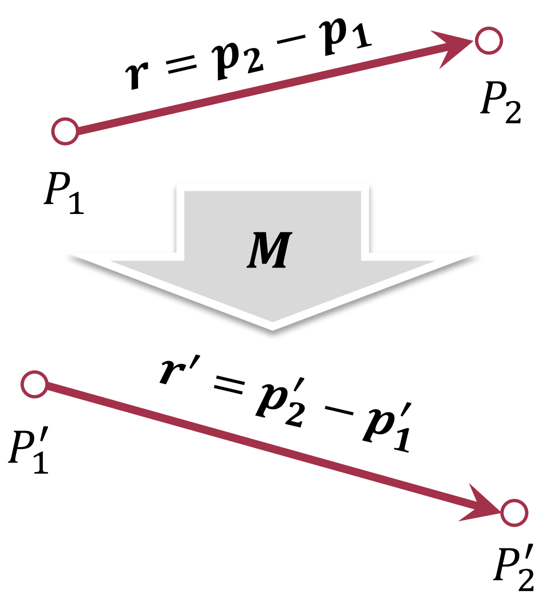
Beschrieben in A

$$(\boldsymbol{M}_{\boldsymbol{B}\to\boldsymbol{A}})^{-\boldsymbol{1}} \cdot \boldsymbol{p}_{\boldsymbol{A}} = \boldsymbol{M}_{\boldsymbol{A}\to\boldsymbol{B}} \cdot \boldsymbol{p}_{\boldsymbol{A}} = \boldsymbol{p}_{\boldsymbol{B}}$$

# 3.5

# Transformation von Richtungsvektoren

# Tangentenvektoren




- Bisher: Transformation von Ortsvektoren in homogenen Koordinaten
- Tangentenvektoren werden in gleicher Weise wie Ortsvektoren transformiert
- Allerdings kann auch obere linke 3x3 Matrix der Transformation in HK verwendet werden

$$\boldsymbol{p'_1} = \boldsymbol{M} \cdot \boldsymbol{p_1} \quad \boldsymbol{p'_2} = \boldsymbol{M} \cdot \boldsymbol{p_2}$$

$$\begin{aligned}\boldsymbol{r'} &= \boldsymbol{p'_2} - \boldsymbol{p'_1} \\ &= \boldsymbol{M} \cdot \boldsymbol{p_2} - \boldsymbol{M} \cdot \boldsymbol{p_1} \\ &= \boldsymbol{M} \cdot (\boldsymbol{p_2} - \boldsymbol{p_1}) \\ &= \boldsymbol{M} \cdot \boldsymbol{r}\end{aligned}$$

# Normalentransformation

- Hinweis: 3D Spalten- bzw. Zeilenvektor entspricht 1x3 bzw. 3x1 Matrix. Dann ist das Skalarprodukt: $\boldsymbol{a}^T \cdot \boldsymbol{b}$
- Gegeben:
  - Tangentenvektor: $\boldsymbol{r}$
  - Normale dazu: $\boldsymbol{n}$
  - Transformationsmatrix $\boldsymbol{M}$
  - $\boldsymbol{r}' = \boldsymbol{M} \cdot \boldsymbol{r}$
- Gesucht:
  - Zugehörige Transformationsmatrix $\boldsymbol{G}$ für Normalenvektoren
- Es gilt:
  - $\boldsymbol{r}^T \cdot \boldsymbol{n} = 0$
- Es soll gelten:
  - $(\boldsymbol{n}')^T \cdot \boldsymbol{r}' = 0$, mit: $\boldsymbol{n}' = \boldsymbol{G} \cdot \boldsymbol{n}$

$$(\boldsymbol{n}')^T \cdot \boldsymbol{r}' = 0$$

$$(\boldsymbol{G} \cdot \boldsymbol{n})^T \cdot (\boldsymbol{M} \cdot \boldsymbol{r}) = 0$$

$$\boldsymbol{n}^T \cdot \boldsymbol{G}^T \cdot \boldsymbol{M} \cdot \boldsymbol{r} = 0$$

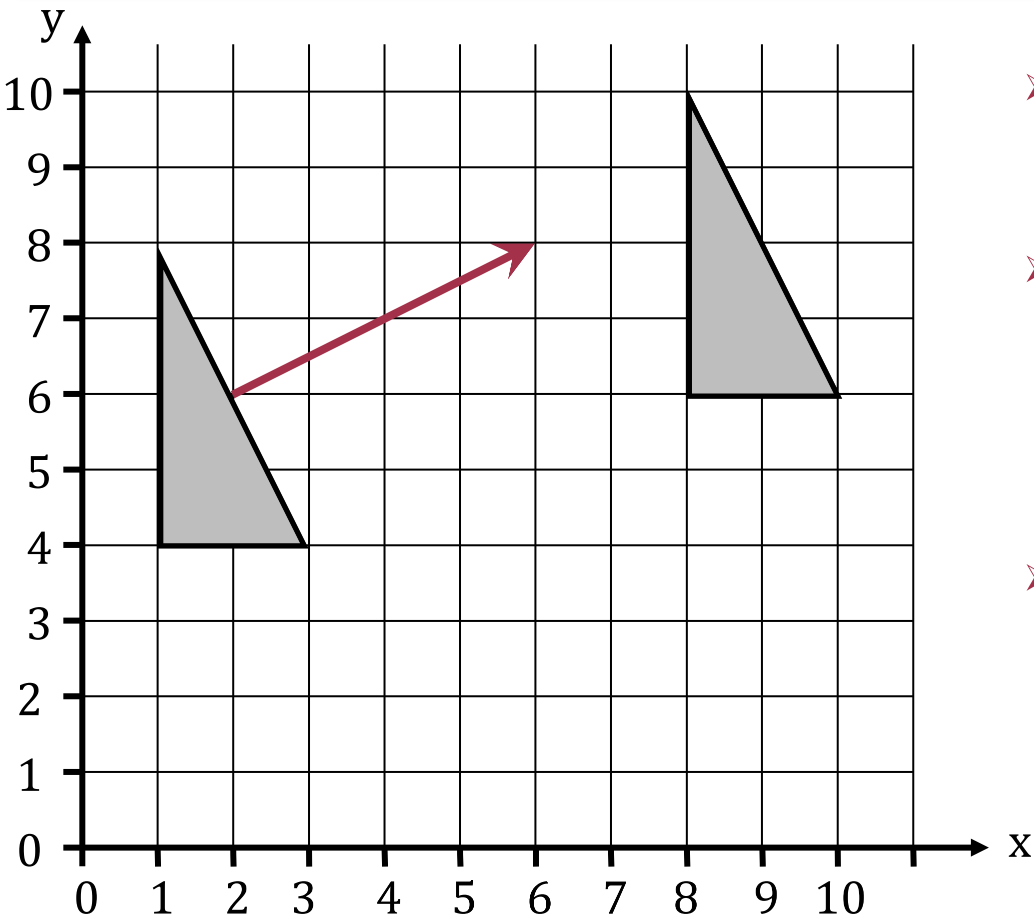
Ist erfüllt, wenn $\boldsymbol{G}^T \cdot \boldsymbol{M} = \boldsymbol{E}$, da $\boldsymbol{n}^T \cdot \boldsymbol{r} = 0$

$$\boldsymbol{G}^T = \boldsymbol{M}^{-1}$$

$$\boxed{\boldsymbol{G} = (\boldsymbol{M}^{-1})^T}$$

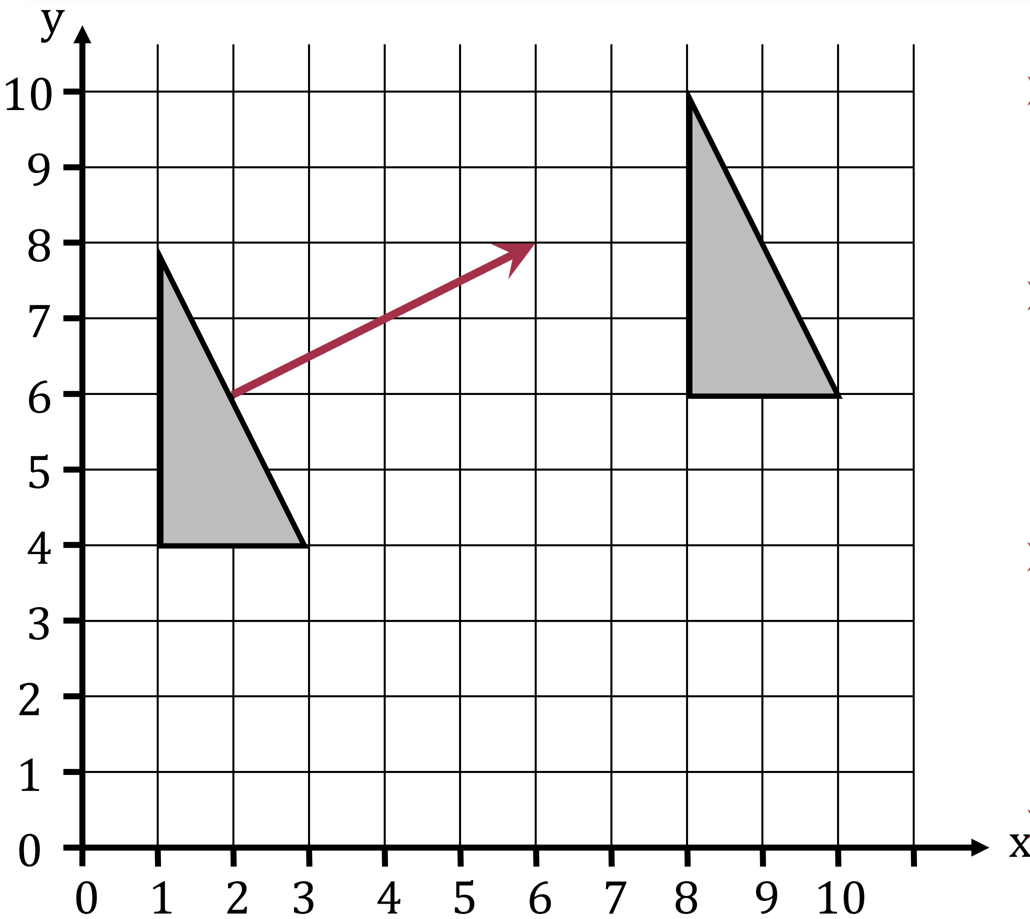
$$(A \cdot B)^T = B^T \cdot A^T$$

# Beispiel: Translation in homogenen Krd.



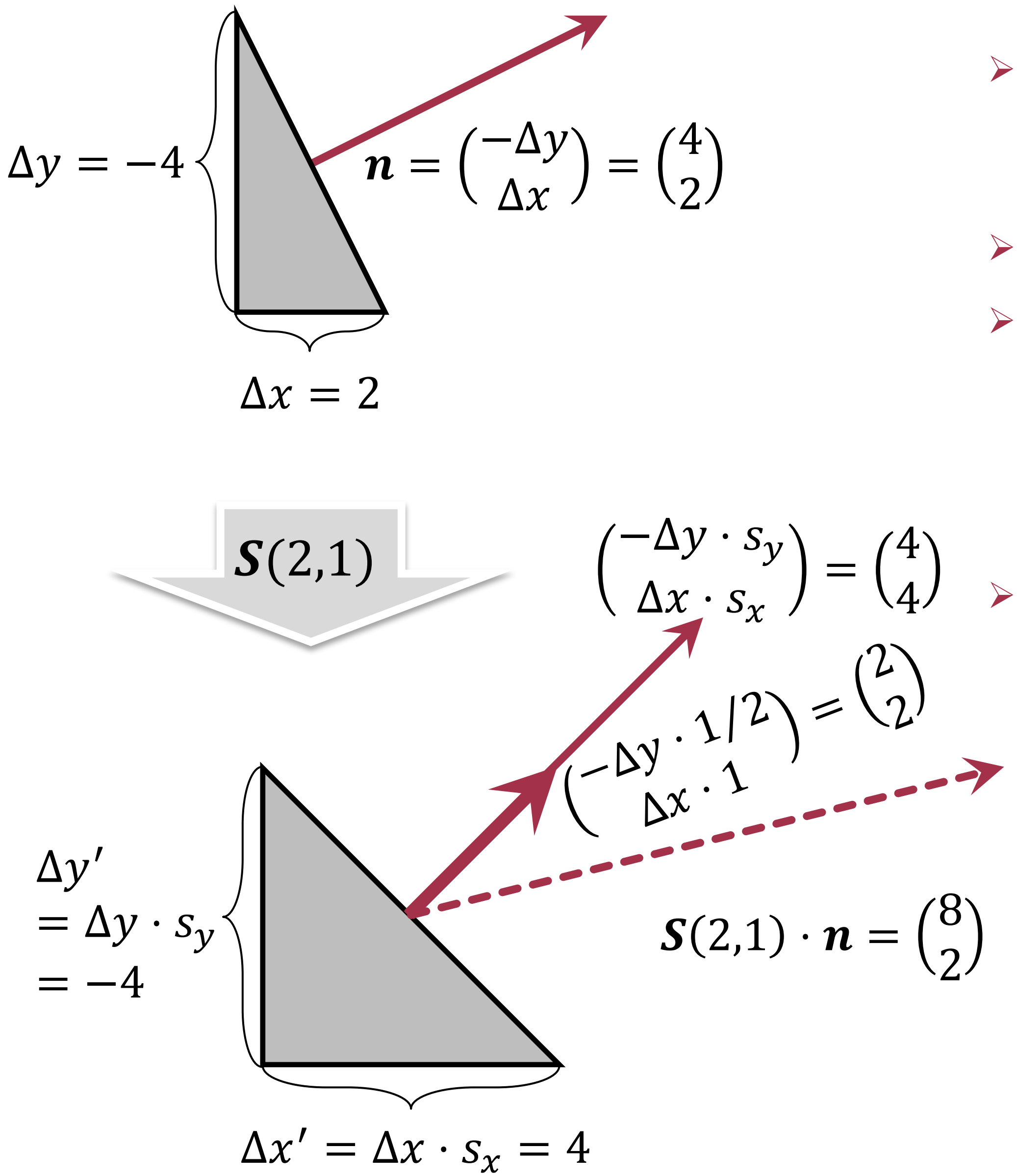

- Wir erwarten:
  - Keinerlei Auswirkung der Translation auf Richtungsvektoren
- Für $\boldsymbol{T}(t_x, t_y, t_z)$ erfüllt:
  - $\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}$
- Für $(\boldsymbol{T}^{-1})^T$ nicht:
  - $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -t_x & -t_y & -t_z & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ -t_x x - t_y y - t_z z \end{pmatrix}$
  - Kein Richtungsvektor mehr

# Beispiel: Translation



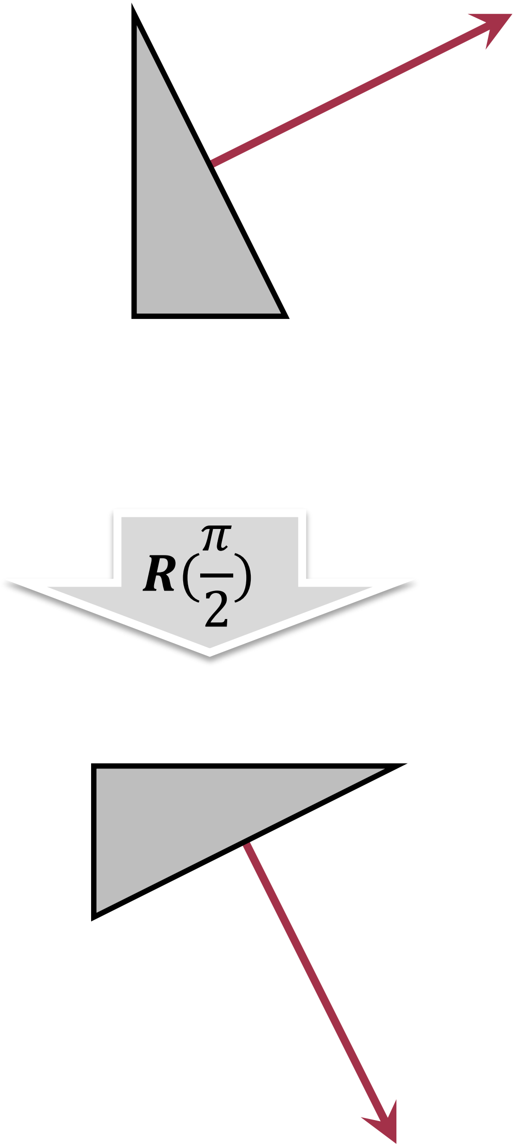

- Normalentransformation wird nicht in homogenen Koordinaten ausgeführt!
- Wir erwarten:
  - Keinerlei Auswirkung der Translation auf Richtungsvektoren
- Für $\boldsymbol{T}_{LO\ 3\times3}(t_x, t_y, t_z)$ erfüllt:
  - $\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$
- Für $(\boldsymbol{T}_{LO\ 3\times3}^{-1})^T$ auch

# Beispiel: Skalierung




- Wir erwarten:
  - Normale bleibt orthogonal zur Oberfläche
- Für $\boldsymbol{S}(s_x, s_y, s_z)$ nicht erfüllt
- Für $(\boldsymbol{S}^{-1})^T(s_x, s_y, s_z) = \boldsymbol{S}^T\left(\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z}\right) = \boldsymbol{S}\left(\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z}\right)$ erfüllt
- Achtung: Länge der Normalen ändert sich
  - Erneutes Normieren nötig

# Beispiel: Rotation




- Wir erwarten:
  - Normalenvektoren werden wie Ortsvektoren rotiert
- Für $\boldsymbol{R}_x(\phi), \boldsymbol{R}_y(\phi), \boldsymbol{R}_z(\phi)$ erfüllt
- Da Rotationsmatrizen orthogonal ebenfalls erfüllt für:
  - $(\boldsymbol{R}_x^{-1})^T(\phi) = \boldsymbol{R}_x(\phi)$
  - $\left(\boldsymbol{R}_y^{-1}\right)^T(\phi) = \boldsymbol{R}_y(\phi)$
  - $(\boldsymbol{R}_z^{-1})^T(\phi) = \boldsymbol{R}_z(\phi)$

# Zusammenfassung

- Tangentenvektoren können wie Ortsvektoren Transformiert werden
- Es genügt allerdings die obere linke 3x3 Matrix, da Translation wirkungslos
- Normalenvektoren werden mit der transponierten inversen der oberen linken 3x3 Matrix transformiert
- Normalentransformation oft einfacher:
  - Translation: unnötig
  - Rotation: Auch mit Originalmatrix möglich

# Kapitel IV: OpenGL

# Über OpenGL

- API für Rastergrafik
  - Prozedural
  - Hardwarenah
  - Plattform-, Betriebssystem- und Sprachunabhängig
- Spezifikationen variieren im Funktionsumfang
  - Diese Veranstaltung: Version 3.1 bzw. 3.3, Core Profile
- Implementation divergieren in:
  - Hardwarenutzung / Performance
  - Genauigkeit (etwas)
- Verwandte APIs:

# Entwicklungsgeschichte

- 1992 initiiert durch SGI…
- …und betreut durch das OpenGL ARB:

- Seit 2006 trägt die Khronos Group die Verantwortung

Over 100 members – any company worldwide is welcome to join

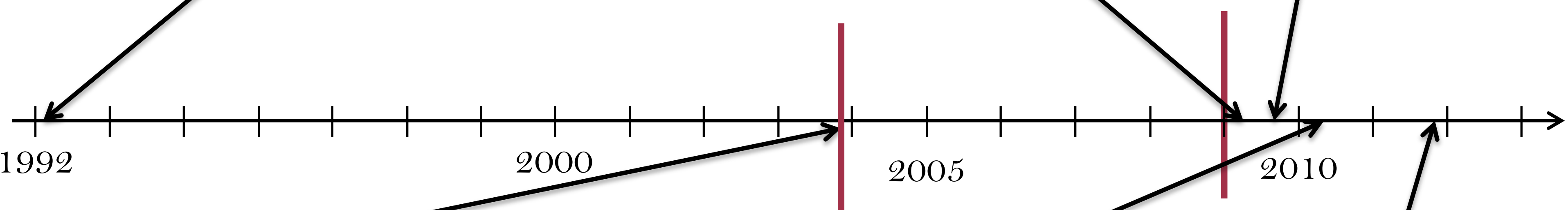
Board of Promoters

# Entwicklung der API

**OpenGL 1.x (ab 1992)**
- Feste, konfigurierbare Pipeline
- Spätere Versionen bringen weitere „Optionen“

**OpenGL 3.1(2009-03)**
- Legacy Funktionen aus Kern entfernt

**OpenGL 3.2 (2009-09)**
- Geometry Shader




**OpenGL 2.x (ab 2004)**
- Einführung programmierbarer Stages (VS / FS)
- Feste Pipeline bleibt parallel bestehen

**OpenGL 3.3 & 4.0 (2010-03)**
- Tesselation (nur 4.0)
- Bessere Zusammenarbeit mit OpenCL

**OpenGL 4.2 (aktuell, 2011-08)**

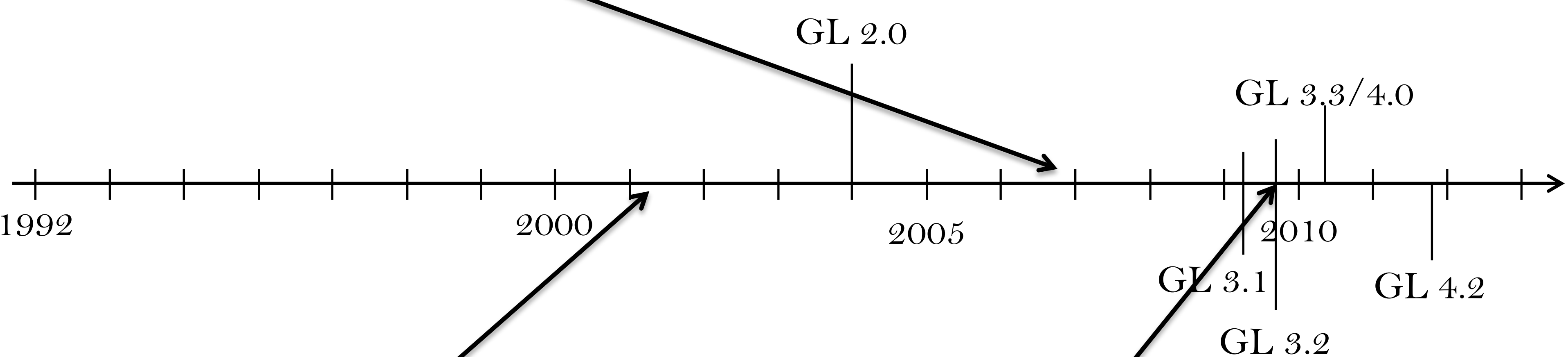
# Programmierstile

- OpenGL 1.x
  - Ausschließlich vorgegebene Algorithmen
  - Können aktiviert/deaktiviert und etwas konfiguriert werden
  - Nahezu kein Spielraum eigene Algorithmen zu formulieren
- OpenGL 2.x, 3.0, Compatibility Profile
  - Programmierbare Abschnitte
  - Feste Pipeline (mittlerweile in Software implementiert) sowie immediate Mode existieren nebenher
  - Können „vermischt“ werden
  - Resultiert in fragwürdigem Programmierstil
  - Inkompatibel mit schlankeren APIs (OpenGL ES 2, WebGL)
- OpenGL 3.1 und danach (Core Profile)
  - Legacy Funktionen entfernt
  - Aufwärtskompatibel

# Vergleich mit Hardwareentwicklung

## Nvidia GeForce 8800 (2006-11)

- OpenGL 3.3
- Cuda / OpenCL



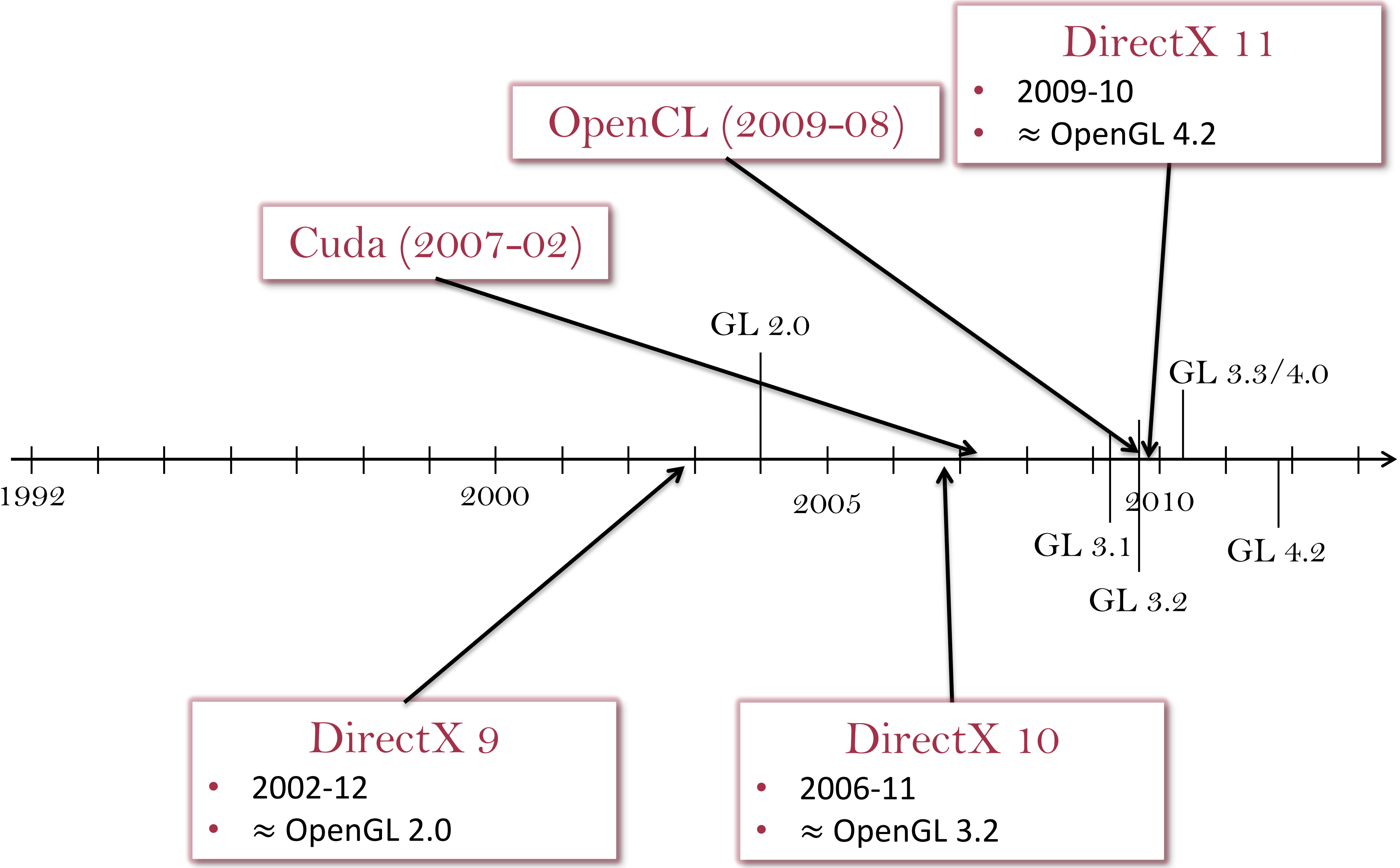

## Nvidia GeForce 3 (2001-03)

- Vertex & Fragment Shader (eingeschränkt)

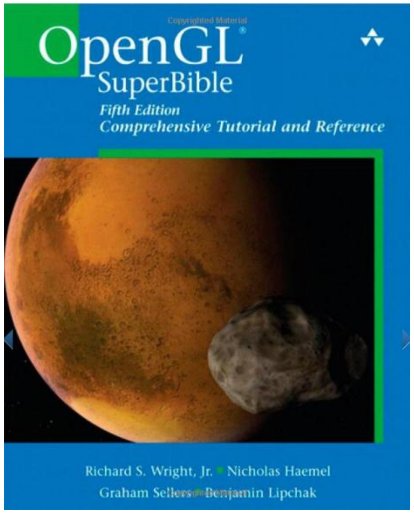
## ATI Radeon 5870 (2009-11)

- OpenGL 4.2
- OpenCL 1.1

# Vergleich mit proprietären APIs

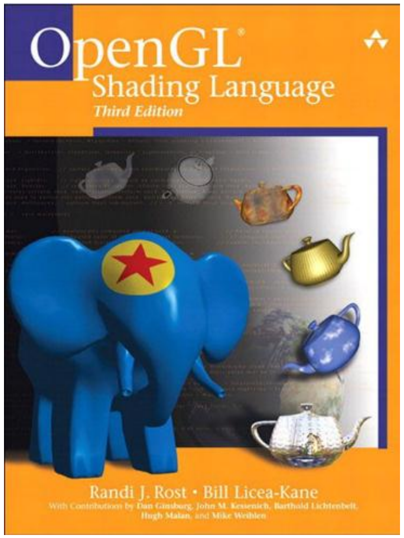


# Literatur OpenGL Programmierung




Wright, Haemel, Sellers, Lipchak
**OpenGL SuperBible**
Addison-Wesley 2010
(verfügbar unter Safari)

[Safari Link]




Rost, ...
**OpenGL Shading Language**
Addison-Wesley 2009
(verfügbar unter Safari)

# Links OpenGL Programmierung

- Tutorials, z.B. auf lwjgl.org, meist veraltet
- OpenGL 4.2 Specification
  - http://www.opengl.org/registry/doc/glspec42.core.20110808.pdf
- OpenGL Shading Language (GLSL) 4.20 Specification
  - http://www.opengl.org/registry/doc/GLSLangSpec.4.20.6.clean.pdf
- OpenGL 3.3 Reference Pages
  - http://www.opengl.org/sdk/docs/man3/
- GLSL 4.2 Reference Pages
  - http://www.opengl.org/sdk/docs/manglsl/
- OpenGL 4.2 Reference Pages
  - http://www.opengl.org/sdk/docs/man4/
- Quick Reference Card (blaue Befehle ignorieren!)
  - http://www.khronos.org/files/opengl42-quick-reference-card.pdf