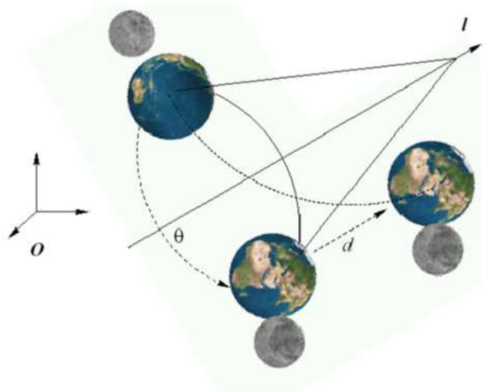
Geometric Algebra Computing
Transformationen in LA und CGA
03.07.2019

Dr. Dietmar Hildenbrand

Technische Universität Darmstadt
Fachbereich Mathematik



Überblick

- In linearer Algebra
  - Homogene Koordinaten
  - Transformationen in linearer Algebra
- Transformationen in konformer geometrischer Algebra

Homogene Koordinaten

- definiere Äquivalenzklasse:

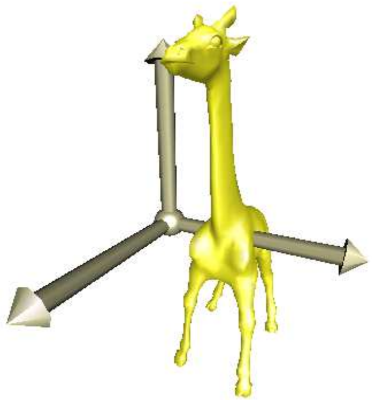
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \cong \begin{pmatrix} sx \\ sy \\ sz \\ s \end{pmatrix} \qquad \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}$$

- 3D (inhomogene) Koordinaten
- 4D homogene Koordinaten
- Skalierungsfaktor s bzw. Gewicht w ungleich 0

Translation

- Translation als Matrix-Multiplikation in homogenen Koordinaten
  - Bsp. Translation um den Vektor $(x_0, y_0, z_0)^t$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

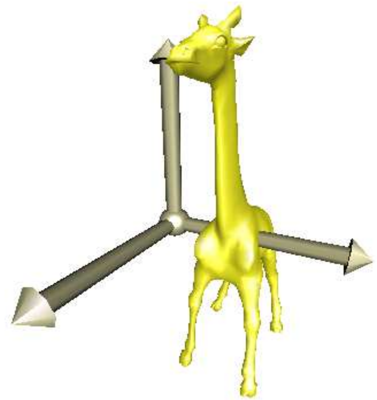


- Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation

- Translation als Matrix-Multiplikation in homogenen Koordinaten
  - Bsp. Translation um den Vektor $(x_0, y_0, z_0)^t$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

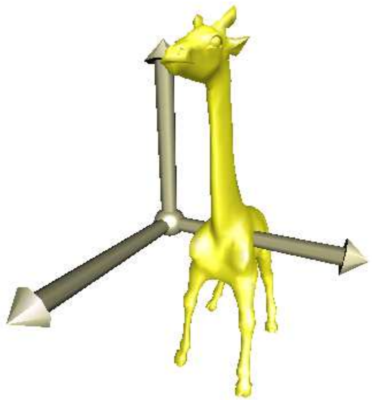


- Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation

- Translation als Matrix-Multiplikation in homogenen Koordinaten
  - Bsp. Translation um den Vektor $(x_0, y_0, z_0)^t$

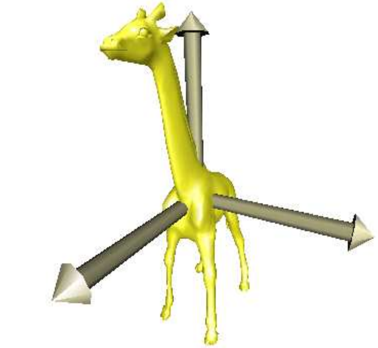
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$



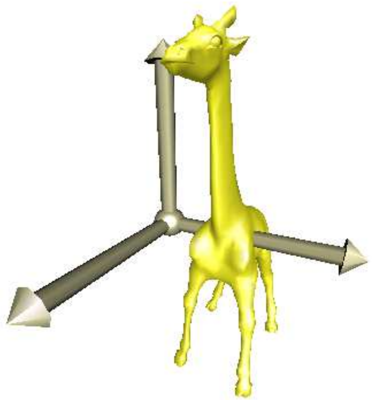
- Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation

- Translation als Matrix-Multiplikation in homogenen Koordinaten
  - Bsp. Translation um den Vektor $(x_0, y_0, z_0)^t$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

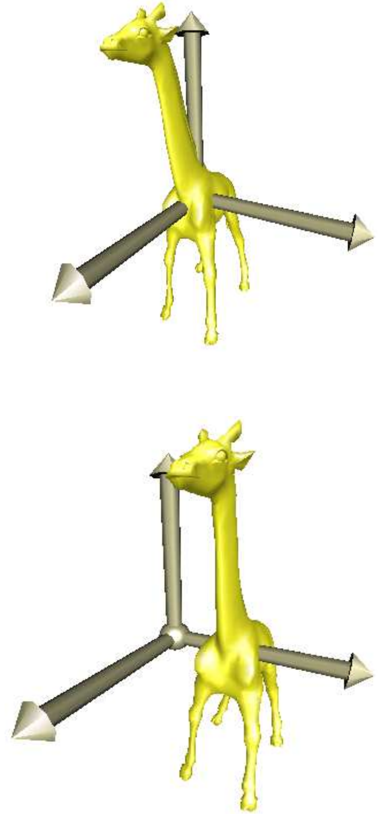


- Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation

- Translation als Matrix-Multiplikation in homogenen Koordinaten
  - Bsp. Translation um den Vektor $(x_0, y_0, z_0)^t$

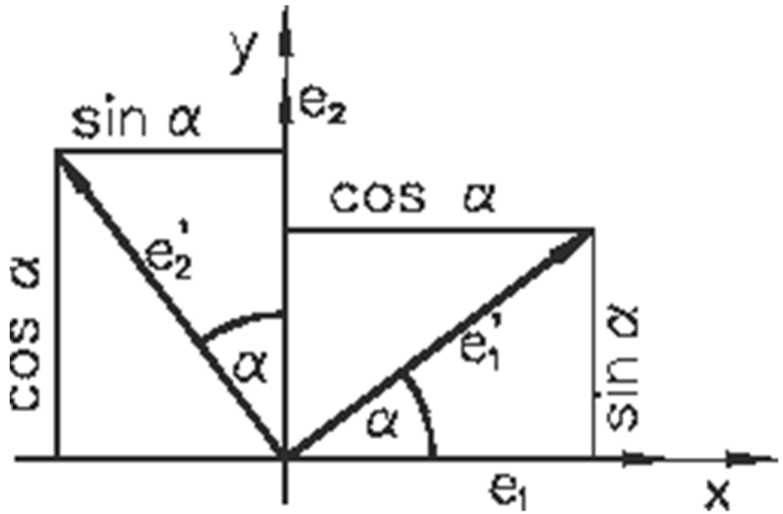
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$



- Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Rotation

- Eine Rotation R_α um den Winkel α um die z-Achse in mathematisch positive Richtung ergibt für die Basisvektoren folgende Beziehung:
  - $R_\alpha((1, 0, 0)^t) = (\cos\alpha, \sin\alpha, 0)$
  - $R_\alpha((0, 1, 0)^t) = (-\sin\alpha, \cos\alpha, 0)$
  - $R_\alpha((0, 0, 1)^t) = (0, 0, 1)$



Rotation

- Die zugehörige 3 x 3 Matrix ergibt sich daher zu

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- In homogenen Koordinaten folgt für die Rotation R_α um die z-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation

- Bei Rotation R_α um die x-bzw. y-Achse ergeben sich analog folgende homogene Darstellungen.
  - Drehung mit dem Winkel α um die x-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  - Drehung mit dem Winkel α um die y-Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

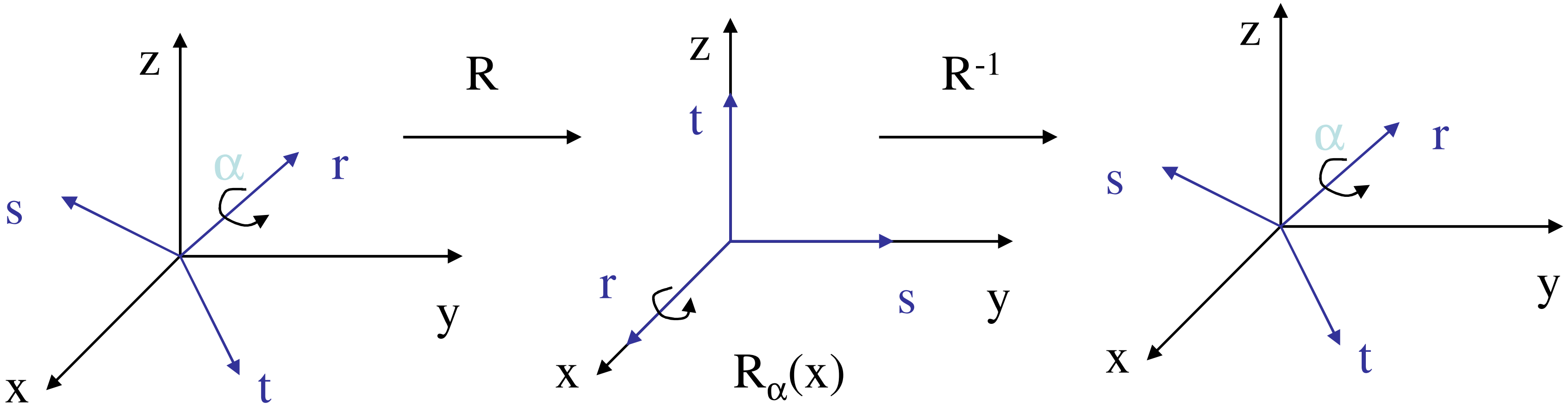
Rotation um beliebige Achse (durch Ursprung)

Berechnung von R

- Drehung $\mathrm{R}(x,y,z)$ um beliebige Achse in Richtung des normierten Vektors $\mathbf{r}=(x,y,z)^t$ um den Winkel α
- Orthonormale Basis $(\mathbf{r},\mathbf{s},\mathbf{t})$ bestimmen
  - erster Basisvektor ist $\mathbf{r}$
  - zweiter Basisvektor $\mathbf{s}$ soll senkrecht auf $\mathbf{r}$ stehen:

$$s = \frac{r \times e_x}{\left\| r \times e_x \right\|} \qquad \text{oder (falls } r \| e_x \text{)} \qquad s = \frac{r \times e_y}{\left\| r \times e_y \right\|}$$

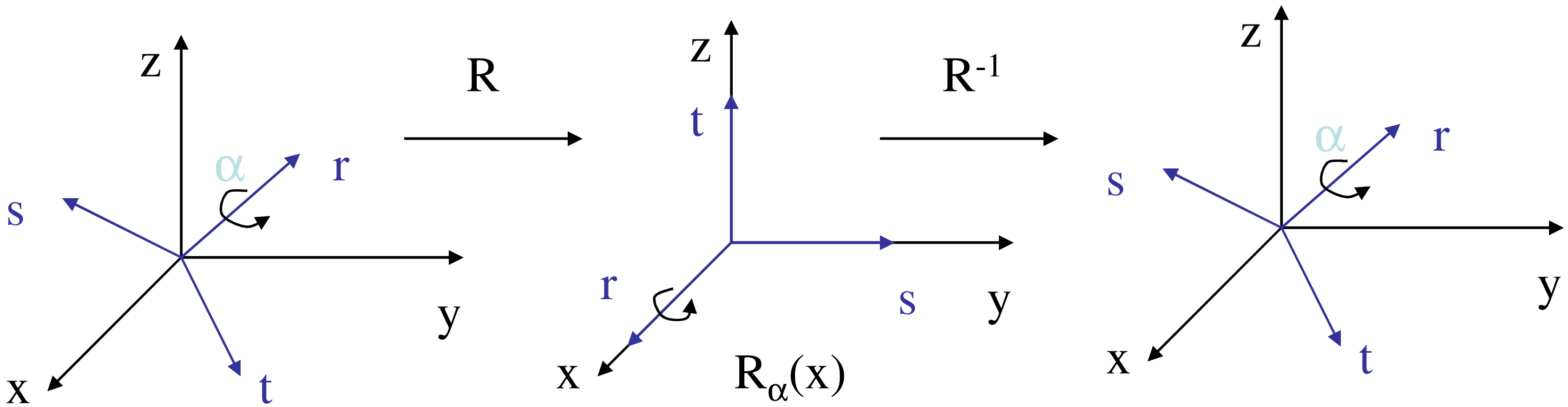
  - dritter Basisvektor $\mathbf{t} = \mathbf{r} \times \mathbf{s}$



Rotation um beliebige Achse (durch Ursprung)

Berechnung von R

- Vektoren $(\mathbf{r},\mathbf{s},\mathbf{t})$ werden in die Spalten der Transformationsmatrix geschrieben
- T-Matrix ist orthogonal und transformiert
  - $\mathbf{e}_x \rightarrow \mathbf{r}$, $\mathbf{e}_y \rightarrow \mathbf{s}$, $\mathbf{e}_z \rightarrow \mathbf{t}$. (das ist R^{-1})
  - Für orthonormierte Matrizen A gilt stets $\mathrm{A}^{-1}=\mathrm{A}^{t}$.
- Also: R ergibt sich, indem man die Vektoren $(\mathbf{r},\mathbf{s},\mathbf{t})$ in die Zeilen von A schreibt



Rotation um beliebige Achse (durch Ursprung)

- Dreht man im Uhrzeigersinn um den Vektor (x,y,z) und den Winkel α, so gilt mit den Abkürzungen $s=\sin(\alpha)$, $c=\cos(\alpha)$ und $t=1-\cos(\alpha)$

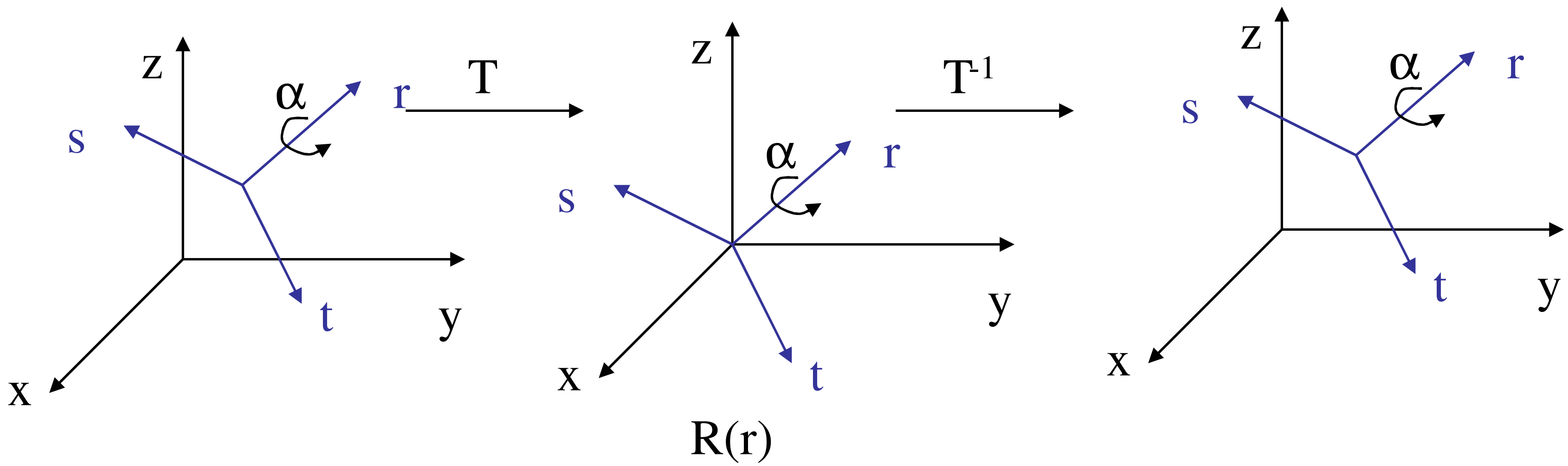
$$R_{(x,y,z)} = \begin{bmatrix} t\cdot x^2+c & t\cdot x\cdot y-s\cdot z & t\cdot x\cdot z+s\cdot y & 0 \\ t\cdot x\cdot y+s\cdot z & t\cdot y^2+c & t\cdot y\cdot z-s\cdot x & 0 \\ t\cdot x\cdot z-s\cdot y & t\cdot y\cdot z+s\cdot x & t\cdot z^2+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

- Für kleine Winkel α ($\alpha < 1°$) kann man sin α durch die Bogenlänge α und cos α durch 1 approximieren

$$R_{(x,y,z)} \approx \begin{bmatrix} 1 & -z\cdot\alpha & y\cdot\alpha & 0 \\ z\cdot\alpha & 1 & -x\cdot\alpha & 0 \\ -y\cdot\alpha & x\cdot\alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation um beliebige Raumachse

- Die bisher diskutierten Rotationen lassen den Ursprung fest
- Rotationsachse durch eine beliebige Achse im Raum
  - Verschiebung des Rotationszentrums in den Ursprung
  - anschließende Rotation und
  - Zurückverschiebung in das Rotationszentrum



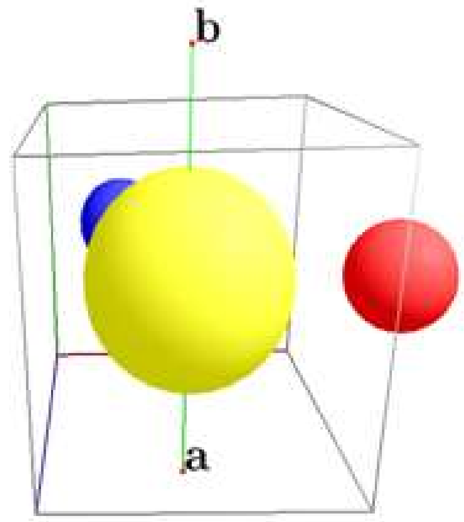
Rotation um beliebige Raumachse

- Beispiel
  - Rotation in positiver Richtung um eine Achse durch den Punkt (x_0, y_0, z_0) und um den Winkel α
  - Die Richtung der Rotationsachse sei die z-Richtung.

$$p' = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot p$$

Transformationen in geometrischer Algebra

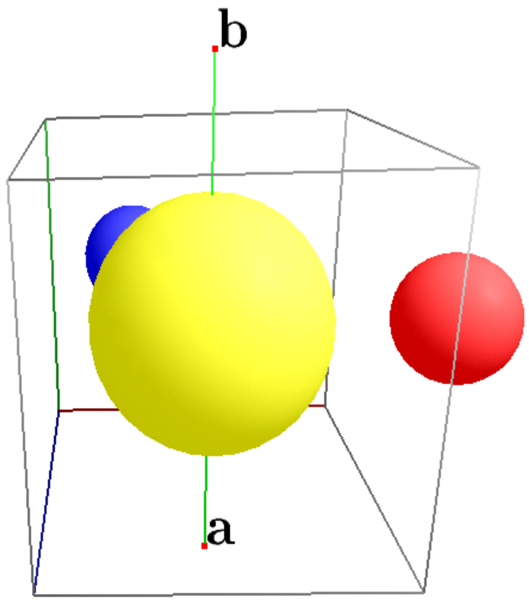
- Siehe CluCalc-Script ConformalTransformations.clu …



Transformations in geometric algebra

Transformations $\mathbf{a}_{\mathbf{transformed}} = \mathbf{V}\mathbf{a}\tilde{\mathbf{V}}$

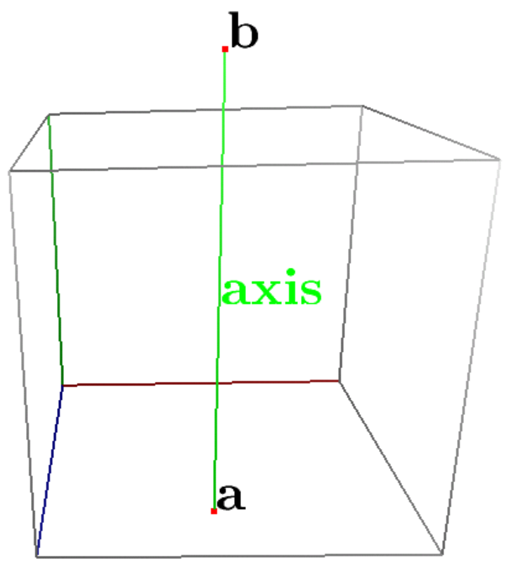
- V is an algebraic entity called versor
- $\tilde{V}$ its reverse
- versor describes kind of transformation
- Note : transformation operators are part of the algebra



RotationAxis

Rotation Axis

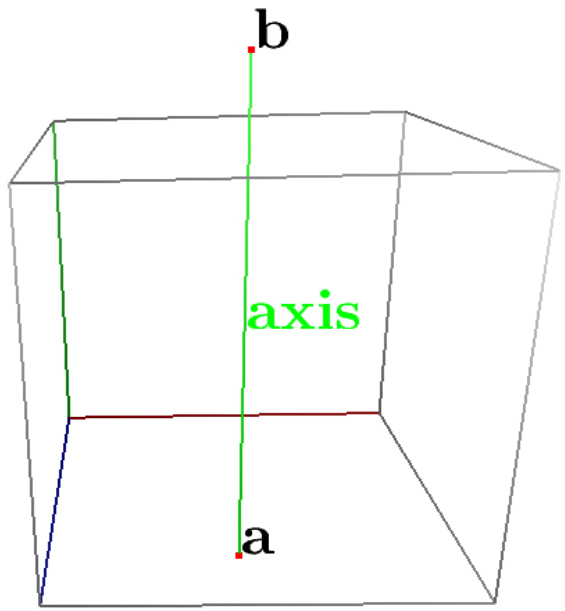
- $a = \mathbf{a} + \frac{1}{2}\mathbf{a}^2 e_\infty + e_0$
  (3D vector bold)
- $b = \mathbf{b} + \frac{1}{2}\mathbf{b}^2 e_\infty + e_0$
- $axis = (a \wedge b \wedge e_\infty)^*$



CLUScript example RotationAxis.clu

Rotation Axis

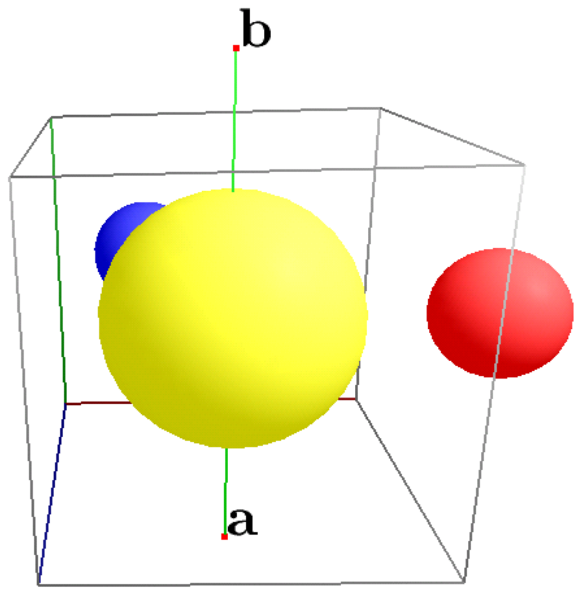
```
DefVarsN3();
:IPNS;

:Red;
:a = VecN3(0,-2,0);
:b = VecN3(0,2,0);

:Green;
axis = *(a^b^einf);
:axis;
?axis;
```



Rotor

$$\mathbf{Rotor\ R} = \mathbf{e}^{-\frac{\phi}{2}\mathbf{l}}$$

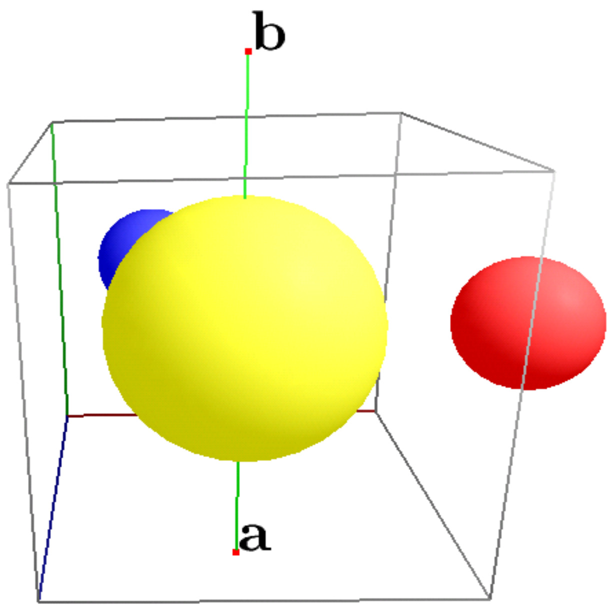
- ϕ is the rotation angle
- l is the rotation axis
- $R = cos(\frac{\phi}{2}) - l sin(\frac{\phi}{2})$



CLUScript example Rotor.clu

$$\text{Rotor } \mathbf{R} = e^{-\frac{\phi}{2}\mathbf{l}}$$

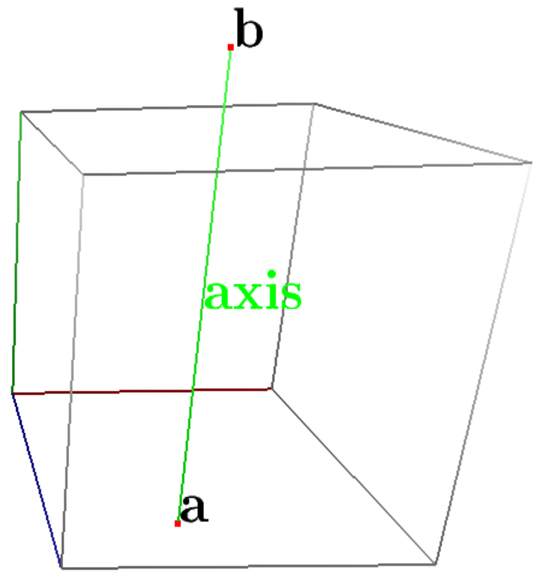
```
:Yellow;
:Sun = e0 -0.5*einf;

:Red;
:Earth =VecN3(2,0,0)-0.125*einf;

?R = exp(-angle/2*axis);

:Blue;
:R * Earth * ~R;
```



Translator

$$\textbf{Translator } \mathbf{T} = \mathbf{e}^{\frac{\mathbf{e}_\infty \mathbf{t}}{2}}$$

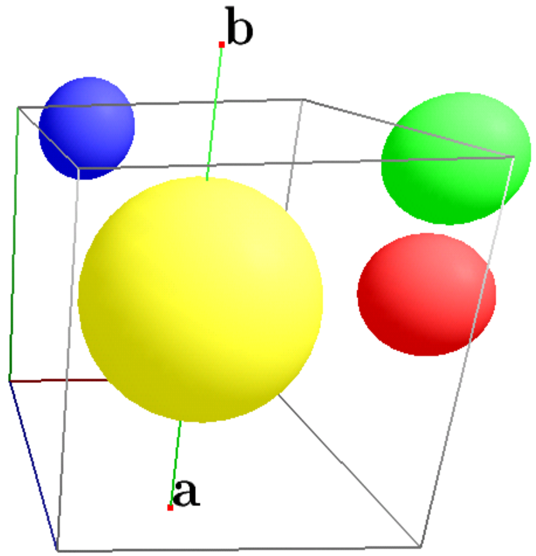
- $\mathbf{t}$ is the 3D translation vector
  $\mathbf{t} = t_1 e_1 + t_2 e_2 + t_3 e_3$
- $T = 1 + \frac{e_\infty \mathbf{t}}{2}$
- $T = e^{\frac{e_\infty \mathbf{t}}{2}} =$
  $1 + \frac{\frac{e_\infty \mathbf{t}}{2}}{1!} + \frac{(\frac{e_\infty \mathbf{t}}{2})^2}{2!} + \frac{(\frac{e_\infty \mathbf{t}}{2})^3}{3!} ...)$
- since $(e_\infty)^2 = 0$



Rigid Body Motion

$$\mathbf{Rigid\ Body\ Motion\ M = RT}$$

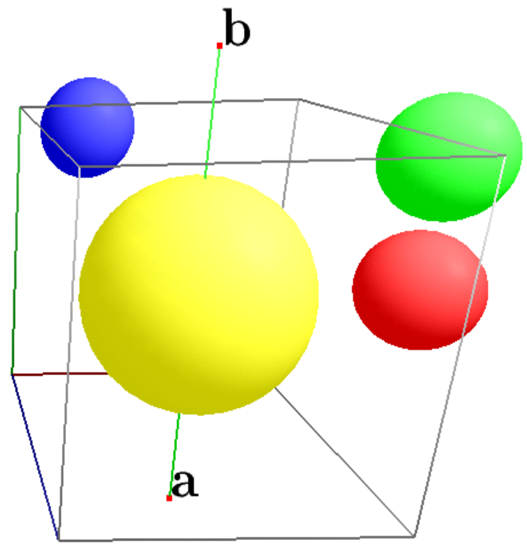
- R is a rotor
- T is a translator



CLUScript example RigidBody.clu

Rigid Body Motion $M = RT$

```
:Yellow;
:Sun = e0 -0.5*einf;
:Red;
:Earth=VecN3(2,0,0)-0.125*einf;

?R = exp(-angle/2*axis);

TVEC = angle/4*e2;
?T= exp(e*TVEC/2);

:Green;
:T* Earth * ~T;
?Motor=R*T;

:Blue;
:Motor* Earth * ~Motor;
```

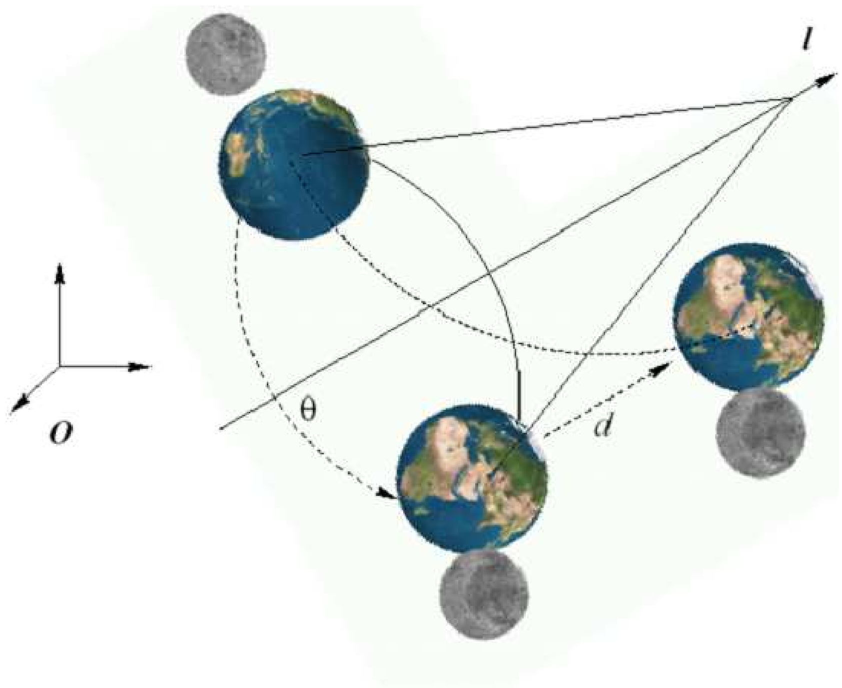


Screw Motion

$$M = e^{-\frac{\theta}{2}l + e_\infty \mathbf{d}}$$

- l is a bivector
  representing an arbitrary axis
- θ is the rotation angle
- $\mathbf{d}$ is a 3D vector
  parallel to the axis l

If l is zero $\rightarrow$ pure translation
If $\mathbf{d}$ is zero$\rightarrow$ pure rotation



Interpolation of motions

Interpolation of motions

two twists T_1 and T_2:

$T_1 = -\frac{\theta_1}{2} l_1 + e_\infty \mathbf{d_1}$

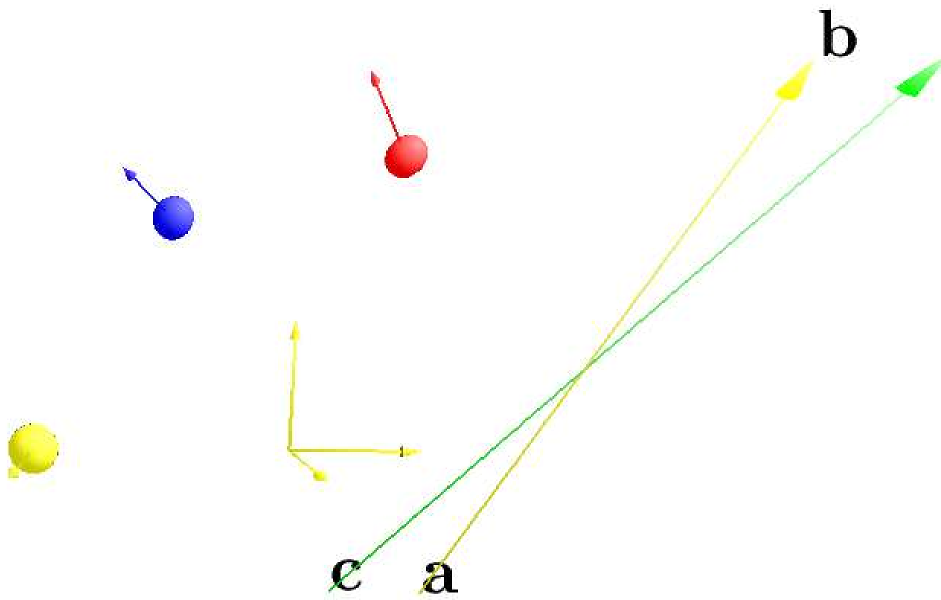
$T_2 = -\frac{\theta_2}{2} l_2 + e_\infty \mathbf{d_2}$

linear interpolation

$T(t) = (1-t) * T_1 + t * T_2$

results in the Motor

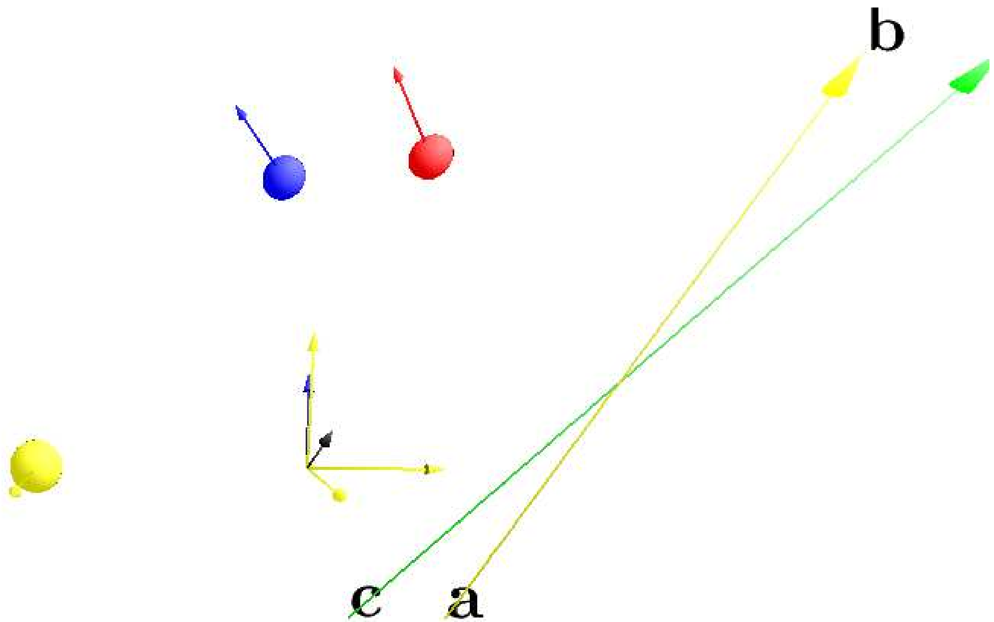
$M(t) = e^{T(t)}$



Velocities

Velocities

- Screw velocity $V = \dot{T}(t)$
  - translational velocity
    $\mathbf{v} = e_0 \cdot V$
  - rotational velocity
    $\omega = -\frac{V + e_\infty \mathbf{v}}{e_{123}}$
- e.g. interpolation via
  $T(t) = (1-t) * T_1 + t^m * T_2$
  - Screw velocity
    $V = mt^{m-1} * T_2 - T_1$

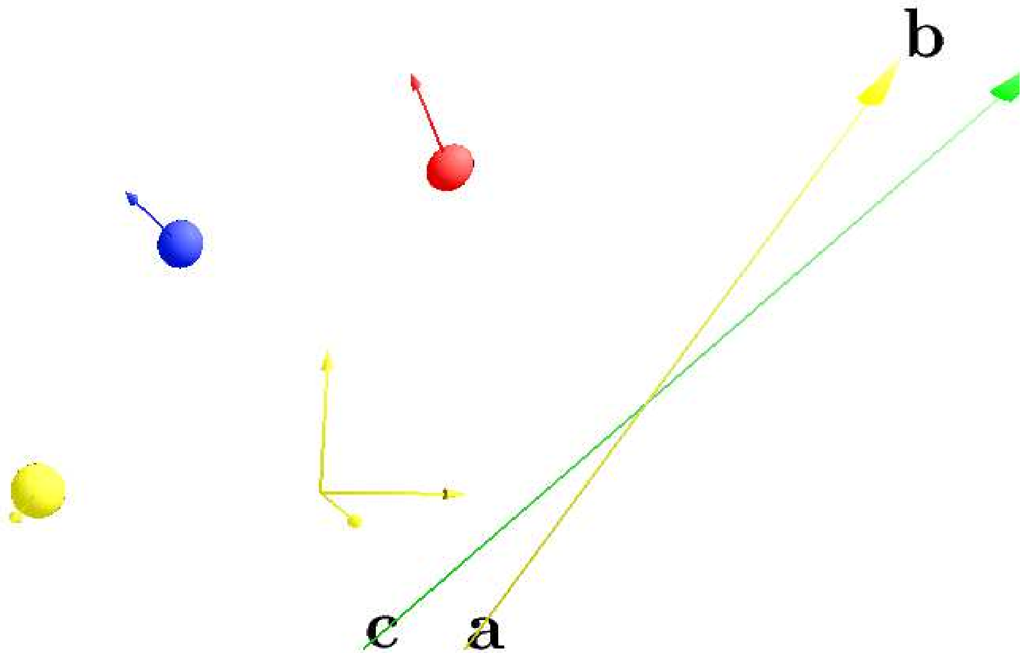


Dynamics

Dynamics

- *comomentum*
  $P = e_{123} I_{body} \omega + m\mathbf{v}e_0$
  I_{body} beeing the inertia tensor
  linearily mapping ω
  to another 3D vector

- kinetic energy
  $K = \frac{1}{2} V \cdot P$



Als nächstes …

numerische Stabilität der Berechnung von Rotationen

- a) wie berechnet man viele hintereinander ausgeführte Rotationen in linearer Algebra, wie in geometrischer Algebra?
- b) was passiert, wenn durch Transformationsmatrizen beschriebene Rotationen numerisch ungenau werden?
- c) was passiert, wenn durch Rotoren beschriebene Rotationen numerisch ungenau werden?
- d) in welchem Sinn, sind Rotoren damit numerisch stabiler als Transformationsmatrizen?

vielen Dank für die
Aufmerksamkeit