Interaktive Computergrafik

Vorlesung im Sommersemester 2013
Kapitel 7: Animation und Keyframing

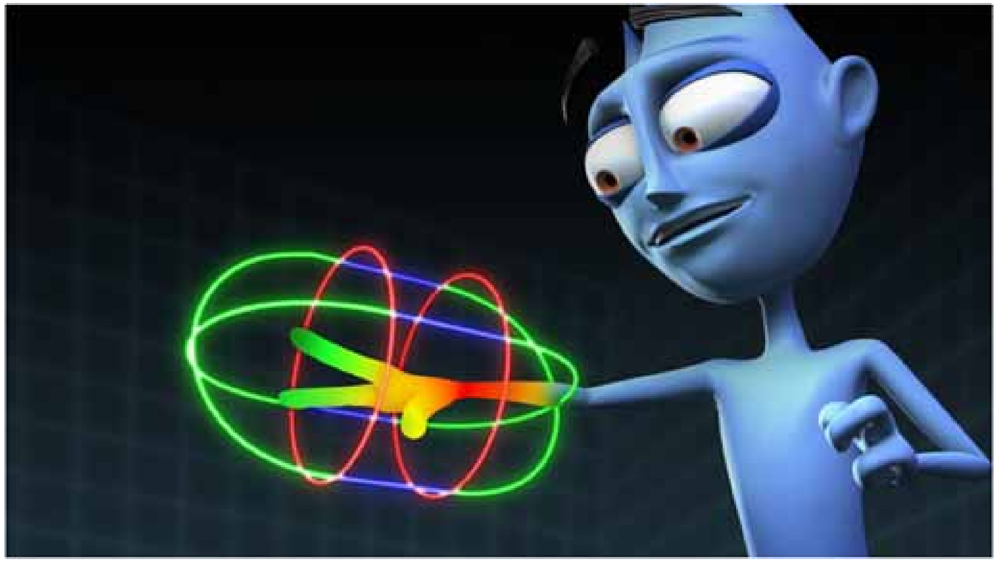
Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

KIT
Karlsruhe Institute of Technology

Animation und Keyframing

Inhalt

- Keyframing und kinematische Animation
- Interpolation von Ort und Orientierung
- Inverse Kinematik
- Skinning



Animation

Begrifflichkeiten

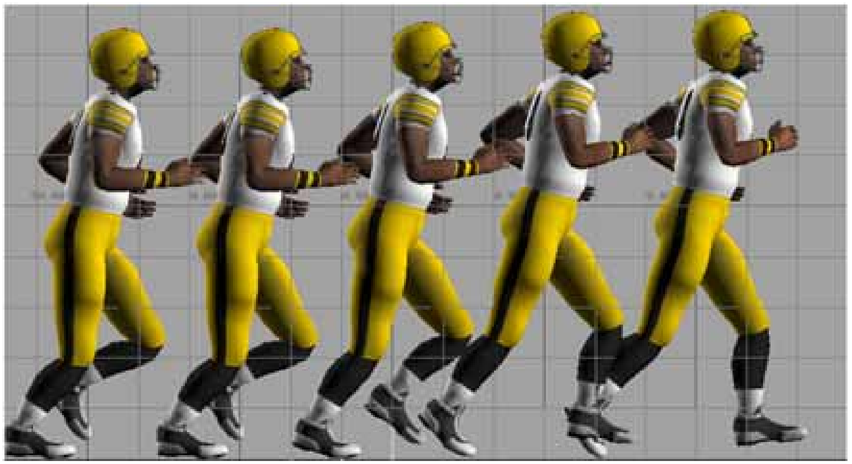
- kinematische Animation
  - direkte Kinematik: Bestimmung einer Bewegung aus vorgegebenen, zeitlich veränderlichen Parametern
    - Ort, Geschwindigkeit, Beschleunigung (aber nicht Kraft)
    - Pfad-Animation, z.B. Bewegung entlang eines Pfads
  - inverse/indirekte Kinematik: Bestimmung aus Randbedingungen

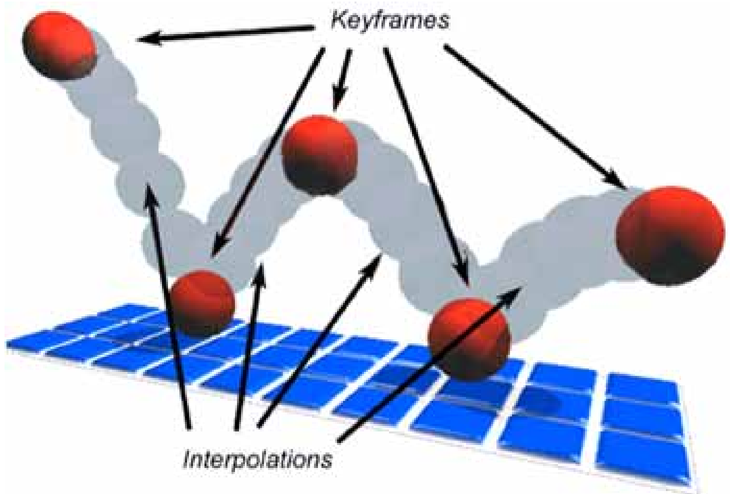
- dynamisches Modell (Simulation):
  - basierend auf physikalischen Gesetzen (insb. Kraftgesetze)
  - physikalische Eigenschaften der Objekte
  - Anfangsbedingungen
  - Starrkörper, deformierbare Körper, Flüssigkeiten, …

Keyframe- und kinematische Animation

Keyframe-Animation

- Vorgabe von „Schlüsselbildern“ (keyframes) durch den Animator, anschließende Interpolation (in betweening)
  - verwendet i.d.R. (Kontroll-)Punkte zur Festlegung der Animation
  - klassische Methode der traditionellen Animation
  - direkte Kontrolle durch den Animator
- Fragestellungen:
  - Interpolation von Position, Orientierung, …
  - Übertragung von Deformationen auf Oberflächen (Skelett→Mesh)

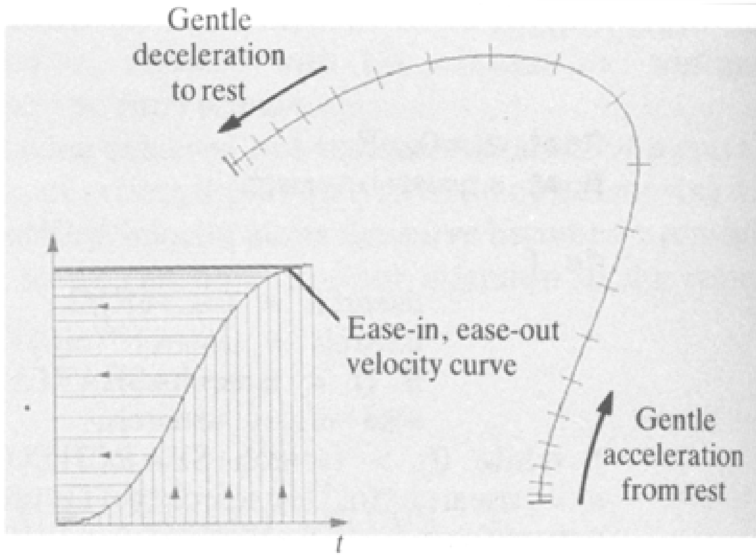




Keyframe- und kinematische Animation

Interpolation

- (stückweise-)linear
- Bézier-Kurven, Splines
- Hermite-Splines
- slow-in, slow-out
  (auch: ease-in, ease-out)

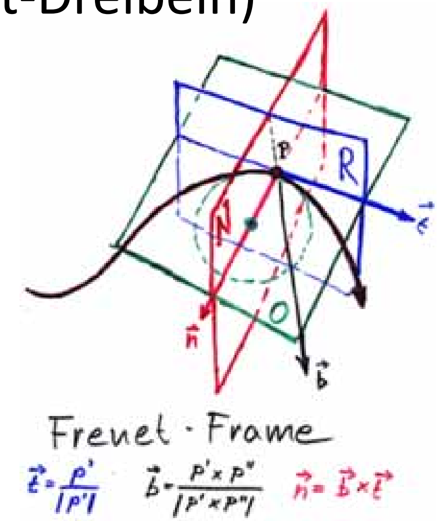


- können wir direkt auf Positionen anwenden, z.B. bei der Pfadanimation
  - Objekt folgt einem vorgegeben Pfad aus Kontrollpunkten
  - Pfadverfolgung zur Generierung der Sicht der ersten Person
  - Bestimmung der Orientierung?
  - Kontrolle über Geschwindigkeit?

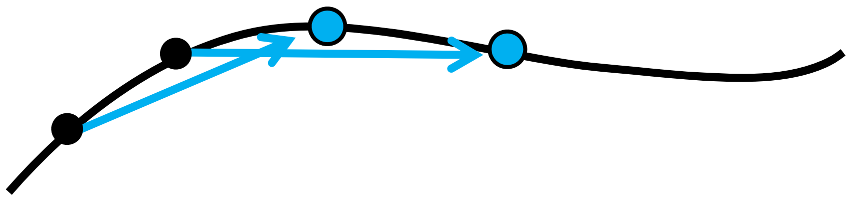
Keyframe- und kinematische Animation

Pfadanimation

- Pfad ergibt sich durch Interpolation zwischen Orts-Keyframes
  - 1) Orientierung direkt ebenfalls durch Keyframes definieren
  - 2) Orientierung durch Ableitung(en) der Kurve (Frenet-Dreibein)
    - lokales Bezugsystem/Tangenten weisen oft eine schnell variierende Orientierung auf



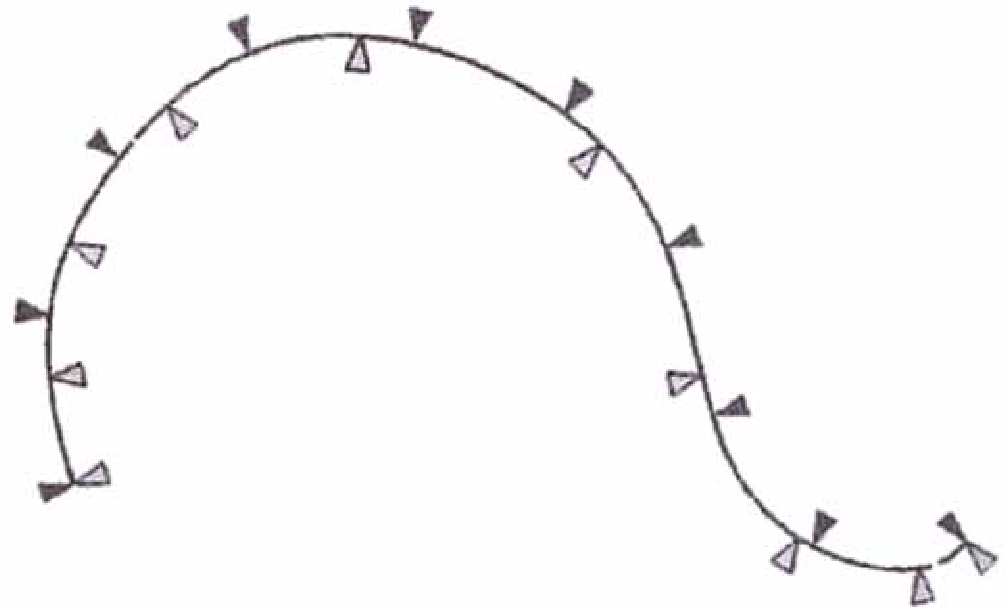
  - 3) Center-of-Interest (COI) Animation
    - Blick gerichtet auf COI, z.B. ein anderes (bewegtes) Objekt oder ein animierter Punkt entlang einer Kurve



Keyframe- und kinematische Animation

Ablaufgeschwindigkeit

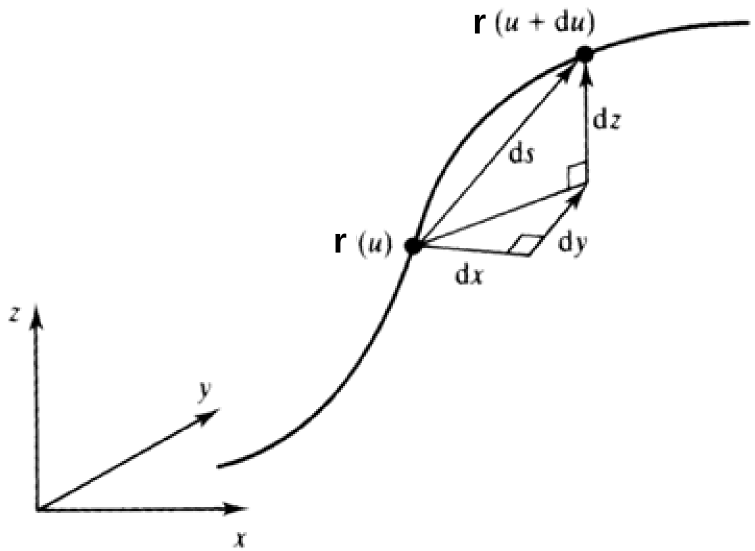
- Interpolation zwischen Ort (und Orientierung) durch Splines
- mit welcher Geschwindigkeit werden diese Kurven durchlaufen?
- gegeben: interpolierte Positionen $\boldsymbol{r}(u)$
- gesucht: Steuerung der Geschwindigkeit des Durchlaufens der Kurve



Ablaufgeschwindigkeit von Keyframe-Animationen

Erster Ansatz: Konstanter Geschwindigkeitsbetrag

- Bogenlängenparametrisierung $\boldsymbol{r}_{arc}(s)$
- Umparametrisierung $u \rightarrow s$
- Bogenlänge



$$s(u) = \int_{u_0}^{u} \sqrt{\frac{dr(u')}{du'} \cdot \frac{dr(u')}{du'}}\, du'$$

- in der Praxis: numerische Integration,
  z.B. Simpson-Regel/Keplersche Fassregel
  (Näherung einer Kurve durch eine einfach integrierbare Parabel)

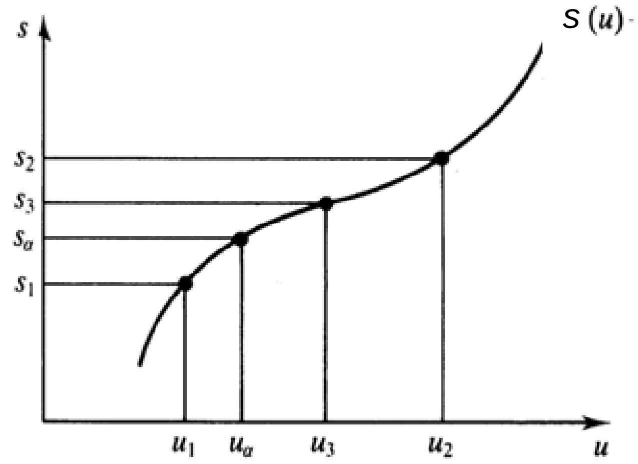
$$\int_{u_1}^{u_2} f(u')du' = \frac{u_2 - u_1}{6}\left(f(u_1) + 4f\left(\frac{1}{2}(u_1 + u_2)\right) + f(u_2)\right)$$

- damit können wir $s(u)$ berechnen und tabellieren

Ablaufgeschwindigkeit von Keyframe-Animationen

Gesucht: Umkehrfunktion $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{s})$

- Schritt 1: Suche durch Bisektion (Intervallhalbierungsverfahren)
  - effizient, da $s(u)$ monoton steigend
  - liefert: $s \in [s_i; s_{i+1}) \Rightarrow u \in [u_i; u_{i+1})$



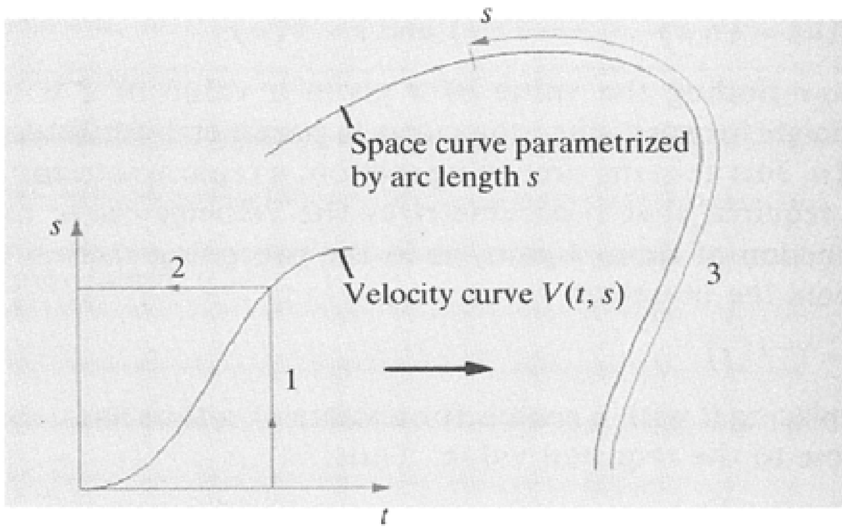
- Schritt 2: suche exaktes u
  - es gilt: $s = s(u) \Leftrightarrow s - s(u) = 0$
  - suche Nullstelle,
    z.B. mit Newton-Raphson-Methode

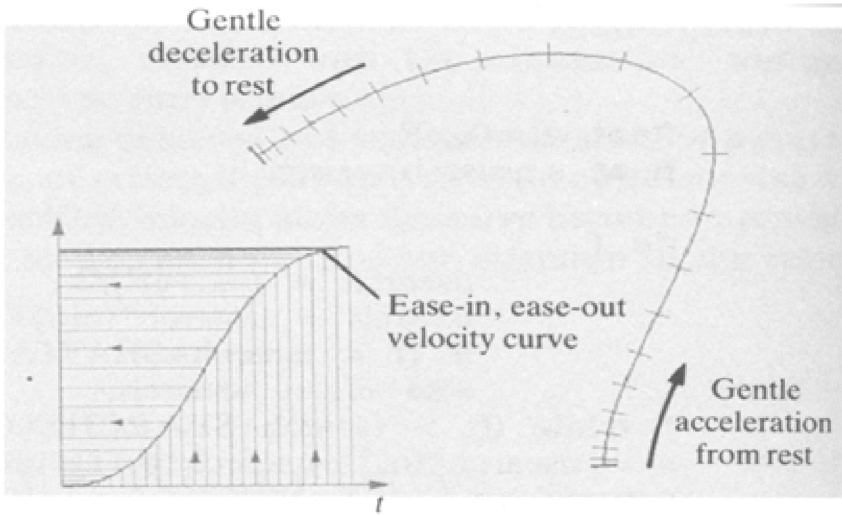
- $\Rightarrow$ Berechnung von $\boldsymbol{r}_{arc}(s) = \boldsymbol{r}\big(u(s)\big)$ mit tabelliertem $s(u)$
  - erlaubt konstante Bahngeschwindigkeit

Ablaufgeschwindigkeit von Keyframe-Animationen

Steuerung der Geschwindigkeit

- mit einer Geschwindigkeitskurve (ordnet Zeit einer Strecke zu)
- Bahngeschwindigkeit: $\frac{ds}{dt}$
- Position entlang der Kurve $\boldsymbol{r}_{time}(t)$:
  - $s = V(t)$
  - $\boldsymbol{r}_{time}(t) = \boldsymbol{r}\left(u\big(V(t)\big)\right)$





Keyframe- und kinematische Animation

Interpolation von Rotationen/Orientierung

- Orientierung (eines 3D-Objekts) ist durch eine Rotationsmatrix definiert

(lineare) Interpolation von	Problem
von Richtungsvektoren?	Länge
von Polarwinkeln?	Singularität an den Polen
von Rotationsmatrizen?	keine Orthogonalität
Eulerwinkeln?	Kardanische Blockade

- wie also interpoliert man Rotationen?

Euler Rotationen

- jede Rotation kann durch 3 Rotationen um die Hauptachsen (x, y, z) ausgedrückt werden (Leonhard Euler 1707 – 1783)
  - man kann auch andere Achsen und Reihenfolgen festlegen, z.B. Luftfahrtnorm (DIN 9300) (Yaw-Pitch-Roll z, y', x'')

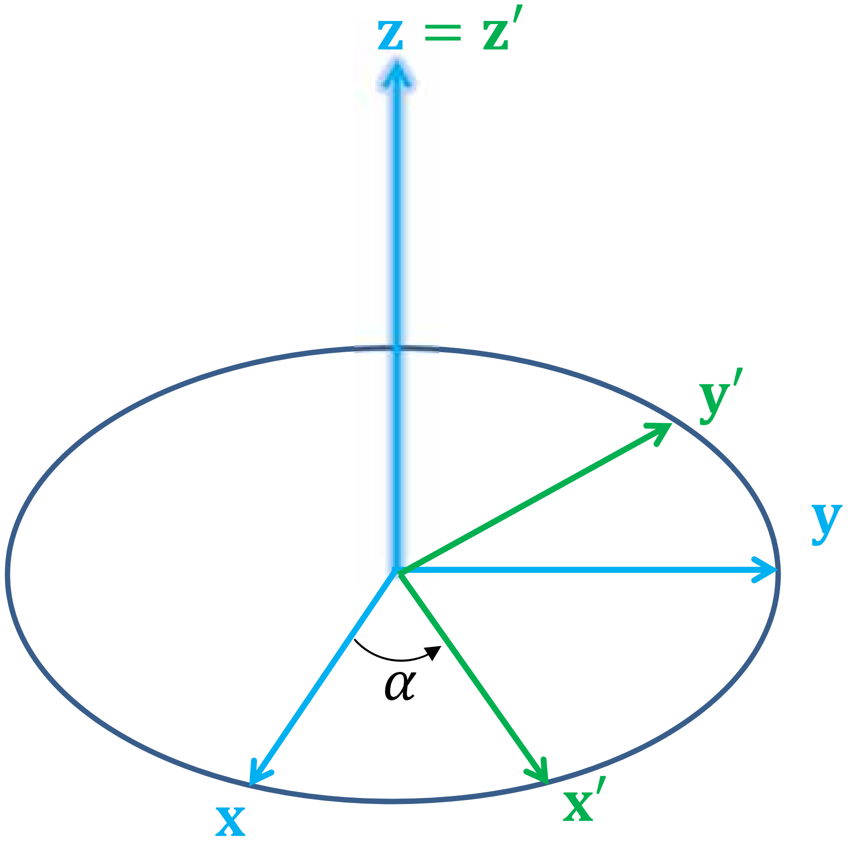
- sind die Rotationen um die x-, y- bzw. z-Achse ψ, θ und ϕ dann ist die Rotationsmatrix:

$$\mathbf{R} = \mathbf{R_z}(\phi)\mathbf{R_y}(\theta)\mathbf{R_x}(\psi) =$$
$$\begin{pmatrix} \cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi \\ -\sin\theta & \sin\psi\cos\theta & \cos\psi\cos\theta \end{pmatrix}$$

- die Winkel ψ, θ und ϕ heißen Eulerwinkel
  - … und beschreiben die Orientierung eines Objektes
  - … zusammen mit der Festlegung der Achsen und der Reihenfolge

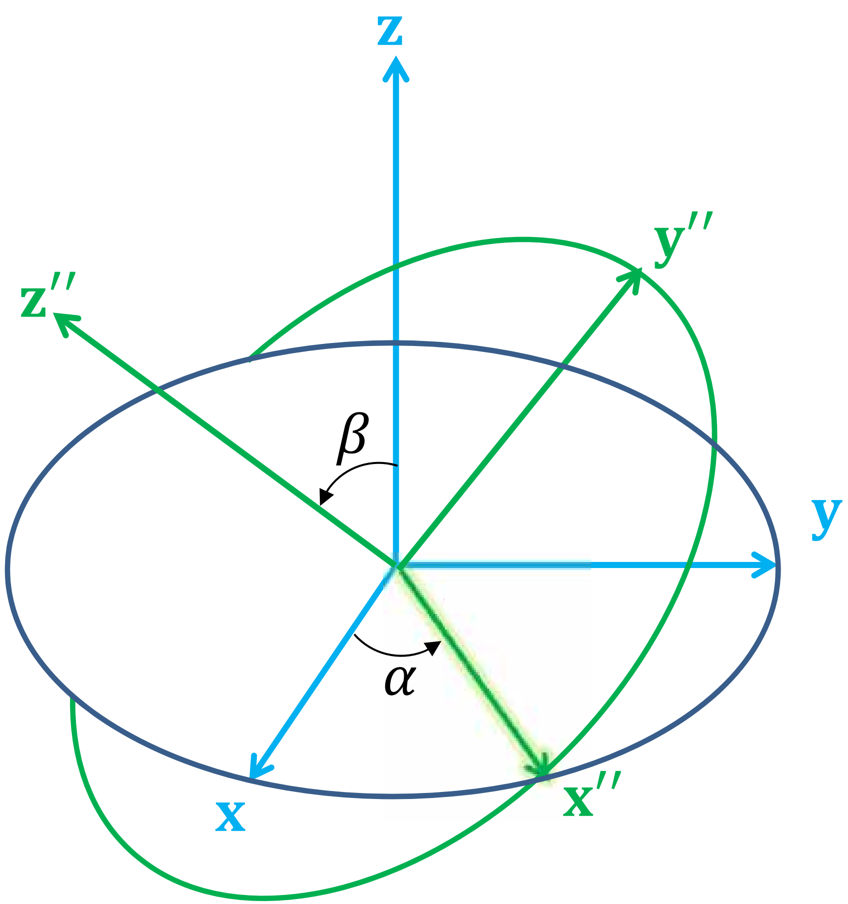
Euler Rotationen

- Rotation kann ebenfalls durch Eulerrotation $\mathbf{R} = \mathbf{R_z}(\gamma)\mathbf{R_x}(\beta)\mathbf{R_z}(\alpha)$ ausgedrückt werden, also Eulerwinkel α, β, γ für die Rotation um $\mathbf{z}$-$\mathbf{x}$-$\mathbf{z}$



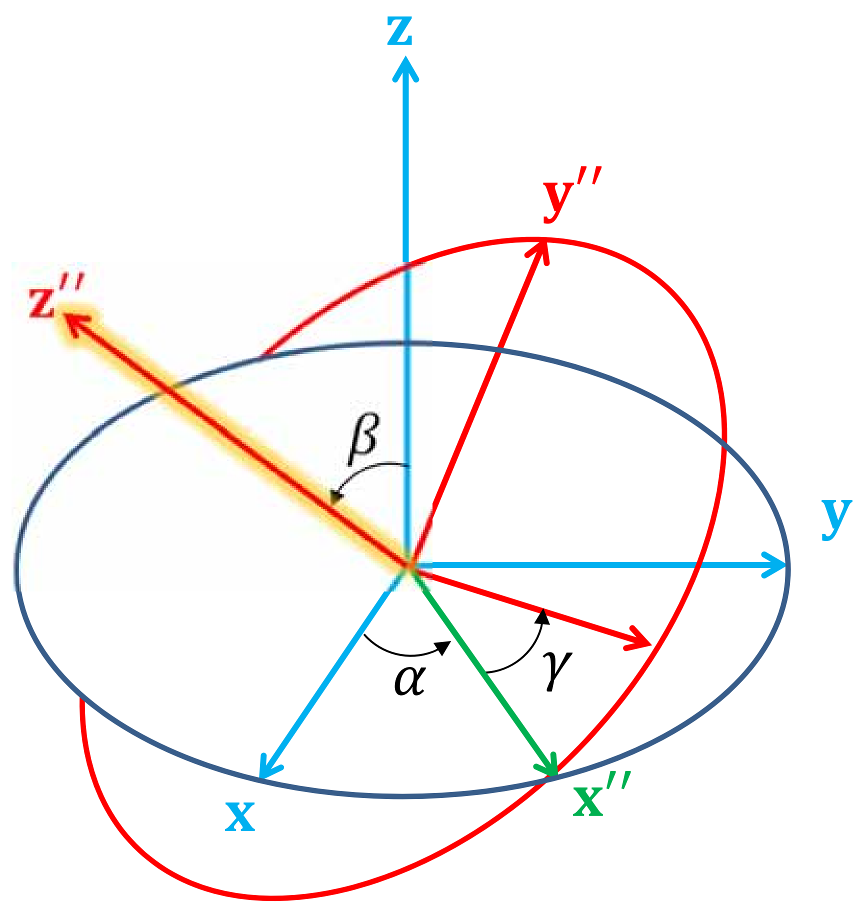
Euler Rotationen

- Eulerwinkel α, β, γ für die Rotation um $\mathbf{z}$-$\mathbf{x}$-$\mathbf{z}$



Euler Rotationen

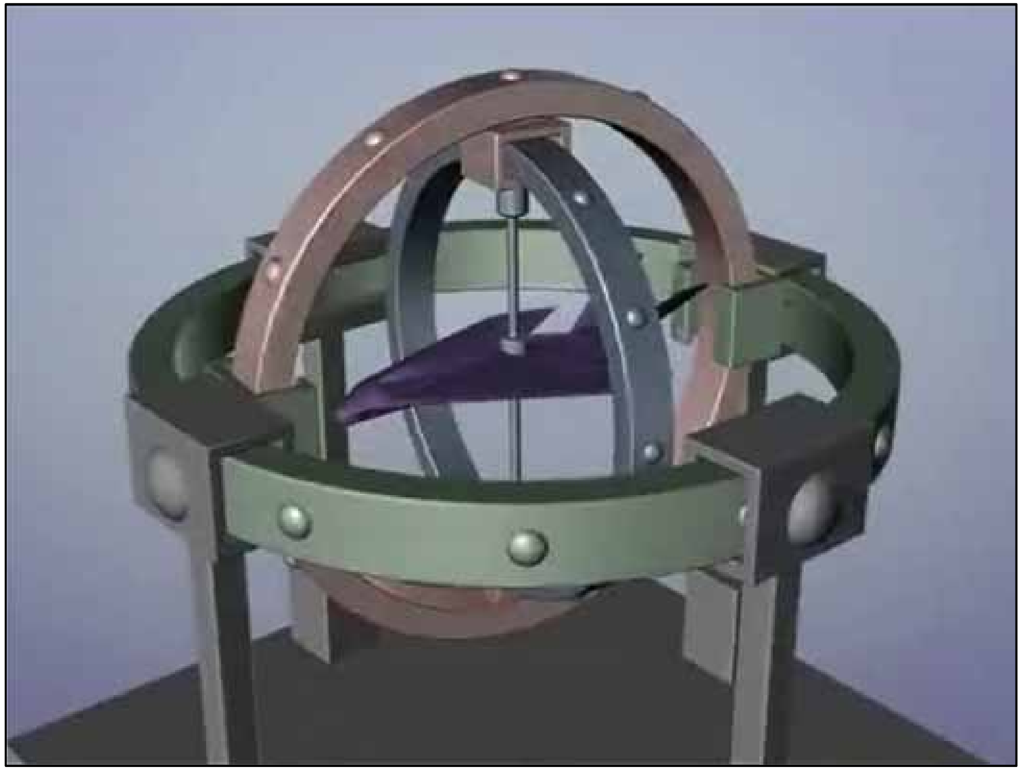
- Eulerwinkel α, β, γ für die Rotation um $\mathbf{z}$-$\mathbf{x}$-$\mathbf{z}$



Euler Rotationen

Kardanische Blockade (engl. Gimbal Lock) und weitere Probleme

- die 1. und 3. Rotationsachse können zusammenfallen,
  d.h. nur Summe aus 1. und 3. Winkel ist relevant (siehe Video)
- Abhängigkeit von der Reihenfolge, da Rotationen nicht kommutativ sind
- Bahngeschwindigkeit nicht konstant

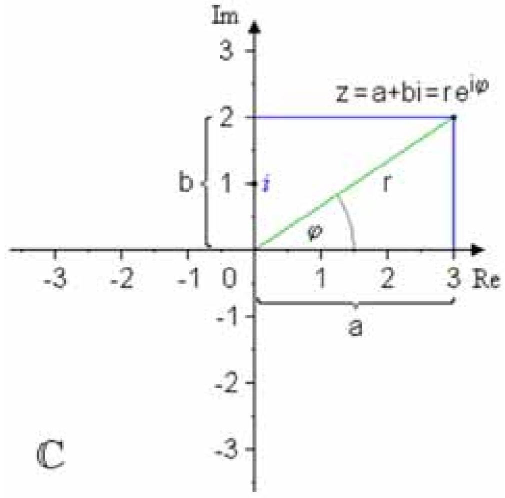


http://www.youtube.com/watch?v=zc8b2Jo7mno

Darstellung von Rotationen

Komplexe Zahlen und Quaternionen

- Multiplikation zweier komplexer Zahlen bewirkt
  - Addition der Winkel, Multiplikation der Länge
  - eine Zahl $\mathrm{z} = a + b\mathbf{i} = e^{i\varphi} \in \mathbb{C}$ mit $|z| = 1$ beschreibt eine Rotation



- Idee der Quaternionen (Hamilton, „Elements of Quaternions“, 1866)
  - Verallgemeinerung der komplexen Zahlen (u.a. für Rotationen in 3D)
  - Definition von $\mathbb{H}$
    - $q(s, x, y, z) = s\mathbf{1} + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, mit
    - $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -\mathbf{1}$
    - $\mathbf{ij} = \mathbf{k}$, $\mathbf{ji} = -\mathbf{k}$ (analog weitere zyklische Permutationen)
  - Kurzschreibweise: $q(s, \boldsymbol{v}) = s\mathbf{1} + v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$

Quaternionen

- Addition:

$$q_1 + q_2 = (s_1, \boldsymbol{v}_1) + (s_2, \boldsymbol{v}_2) = (s_1 + s_2, \boldsymbol{v}_1 + \boldsymbol{v}_2)$$

- Multiplikation:

$$q_1 q_2 = (s_1, \boldsymbol{v}_1)(s_2, \boldsymbol{v}_2) = (s_1 s_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2, s_1 \boldsymbol{v}_2 + s_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2)$$

- Konjugiert:

$$\overline{q} = \overline{(s, \boldsymbol{v})} = (s, -\boldsymbol{v})$$

- Skalarprodukt:

$$q_1 \cdot q_2 = s_1 s_2 + \boldsymbol{v}_1 \cdot \boldsymbol{v}_2$$

Quaternionen

- Norm:

$$|q| = \sqrt{q \cdot q} = \sqrt{q\overline{q}} = \sqrt{s^2 + x^2 + y^2 + z^2}$$

- Multiplikation ist normtreu:

$$|q_1 q_2| = |q_1||q_2|$$

- Einheitsquaternion:

$$|q| = 1 \quad \Leftrightarrow \quad q\overline{q} = 1 \quad \Leftrightarrow \quad q^{-1} = \overline{q}$$

- Alternative Form von Einheitsquaternionen:

$$q = (s, \boldsymbol{v}) = (\cos\varphi, \sin\varphi\, \boldsymbol{n})\,, \quad \text{mit } |\boldsymbol{n}| = 1$$

Quaternionen

- wir können einen Punkt $\boldsymbol{r} \in \mathbb{R}^3$ mit rein imaginärem Quaternion identifizieren (Einbetten des $\mathbb{R}^3$ in $\mathbb{H}$):

$$p = (0, \boldsymbol{r})$$

  - sogenannte reine Quaternionen (pure quaternions)

- weiter ist der Operator R_q über das Quaternionenprodukt mit einem Einheitsquaternion q definiert als

$$R_q(p) = qpq^{-1}$$

- R_q beschreibt die Rotation des Punktes $\boldsymbol{r}$
  - welche Rotation beschreibt q?

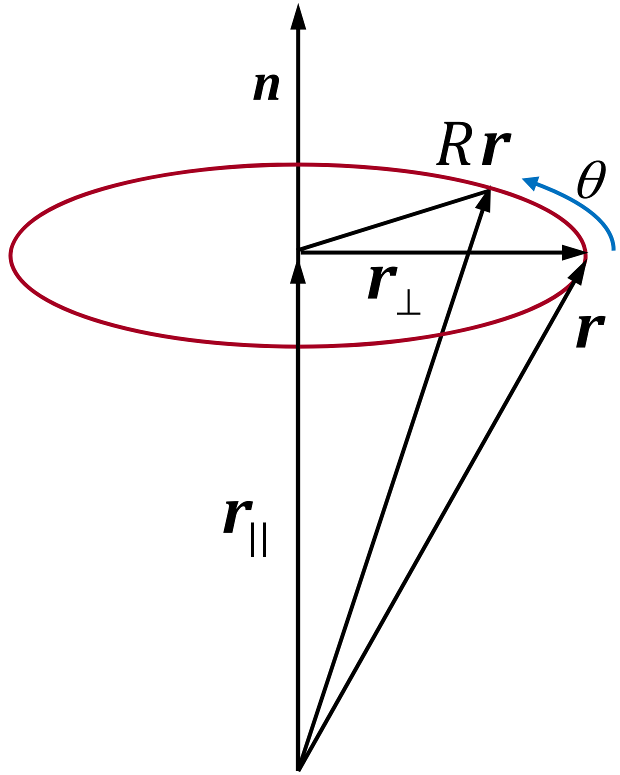
Quaternionen

Vorbereitung zur Herleitung der Rotationseigenschaft von $\boldsymbol{R_q}$

- Rotation von $\boldsymbol{r}$ um eine normierte Achse $\boldsymbol{n}$ um den Winkel θ:

$$R\boldsymbol{r} = \boldsymbol{r}_{||} + \cos\theta\,\boldsymbol{r}_{\perp} + \sin\theta\,\boldsymbol{n}\times\boldsymbol{r}_{\perp}$$

mit $\boldsymbol{r}_{\perp} = \boldsymbol{r} - \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r})$ und $\boldsymbol{r}_{||} = \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r})$



- Einsetzen und Umformen ergibt:

$$R\boldsymbol{r} = \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r}) + \cos\theta(\boldsymbol{r} - \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r})) + \sin\theta\,\boldsymbol{n}\times\boldsymbol{r}$$

$$= \cos\theta\,\boldsymbol{r} + (1-\cos\theta)\boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r}) + \sin\theta\,\boldsymbol{n}\times\boldsymbol{r}$$

Quaternionen

Herleitung der Rotationseigenschaft von $\boldsymbol{R_q}$

- eben definierter Operator

$$R_q(p) = qpq^{-1} = qp\overline{q} = (s,\boldsymbol{v})(0,\boldsymbol{r})(s,-\boldsymbol{v})$$

$$= (0,(s^2 - \boldsymbol{v}\cdot\boldsymbol{v})\boldsymbol{r} + 2\boldsymbol{v}(\boldsymbol{v}\cdot\boldsymbol{r}) + 2s\boldsymbol{v}\times\boldsymbol{r})$$

mit $q = (s,\boldsymbol{v}) = (\cos\varphi, \sin\varphi\,\boldsymbol{n})$

$$R_q(p) = (0,(\cos^2\varphi - \sin^2\varphi)\boldsymbol{r} +$$

$$2\sin^2\varphi\,\boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r}) + 2\cos\varphi\sin\varphi\,\boldsymbol{n}\times\boldsymbol{r})$$

$$= (0,\cos 2\varphi\,\boldsymbol{r} + (1-\cos 2\varphi)\boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{r}) + \sin 2\varphi\,\boldsymbol{n}\times\boldsymbol{r})$$

$\Rightarrow$ Rotation um Achse $\boldsymbol{n}$ um Winkel $\theta = 2\varphi$

Quaternionen

- Alternativ: Rotation eines Punktes $\boldsymbol{r}$ um Winkel θ um die Achse $\boldsymbol{n}$ durch das Einheitsquaternion

$$q = (\cos(\theta/2), \sin(\theta/2)\,\boldsymbol{n}), \quad \text{mit } |\boldsymbol{n}| = 1$$

- und dem imaginären Quaternion

$$p = (0, \boldsymbol{r})$$

- Operation

$$R_q(p) = qp\overline{q}$$

Quaternionen

Nacheinanderausführung von Rotationen

- Euler-Theorem: zwei nacheinander folgende Rotationen sind identisch zu einer neuen Rotation
- Produkt von zwei Einheitsquaternionen ist wieder Einheitsquaternion (Gruppeneigenschaft)

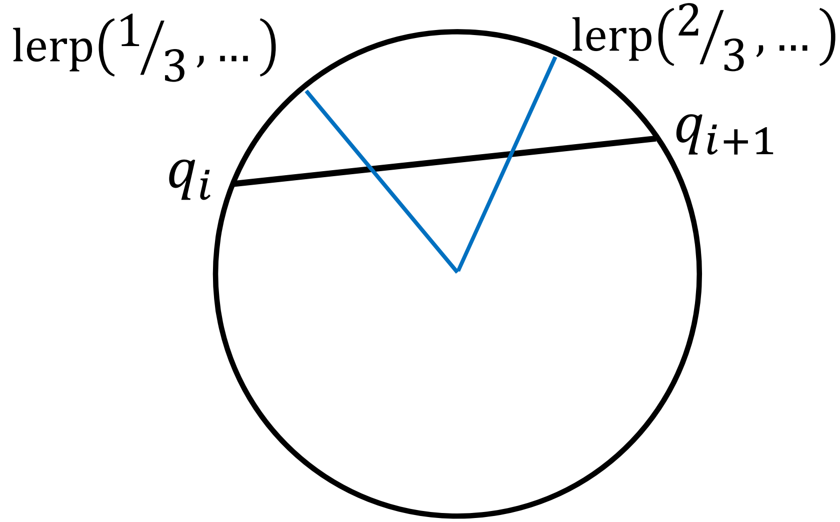
$$R_{q''} = R_q R_{q'} \quad \text{mit} \quad q'' = qq'$$

- Vorteile:
  - Eindeutigkeit bei Rotation zwischen Anfangs- und Endorientierung
  - kein Gimbal Lock

Interpolation von Rotationen

- Ziel: Orientierungen als Keyframes $q_0, q_1, \ldots, q_{n-1}$
  - wie sieht interpolierte Rotationsbewegung aus?
  - Glattheit? Segmentgrenzen?

- einfachste Lösung:
  - $q(t) = \mathrm{lerp}(t, q_i, q_{i+1}) = (1-t)q_i + tq_{i+1}$
  - wichtig: $q(t)$ hinterher immer normieren!
  - Vorteil: kein Gimbal Lock
  - Problem: keine konstante Winkelgeschwindigkeit (an den Enden langsamer als in der Mitte)



Interpolation von Rotationen

Sphärische lineare Interpolation von Quaternionen

- slerp (= **s**pherical **l**inear int**erp**olation) auf der 4D-Kugel

$$\mathrm{slerp}(t, q_i, q_{i+1}) = \frac{\sin\big(\Omega(1-t)\big)}{\sin\Omega} q_i + \frac{\sin(\Omega t)}{\sin\Omega} q_{i+1}$$

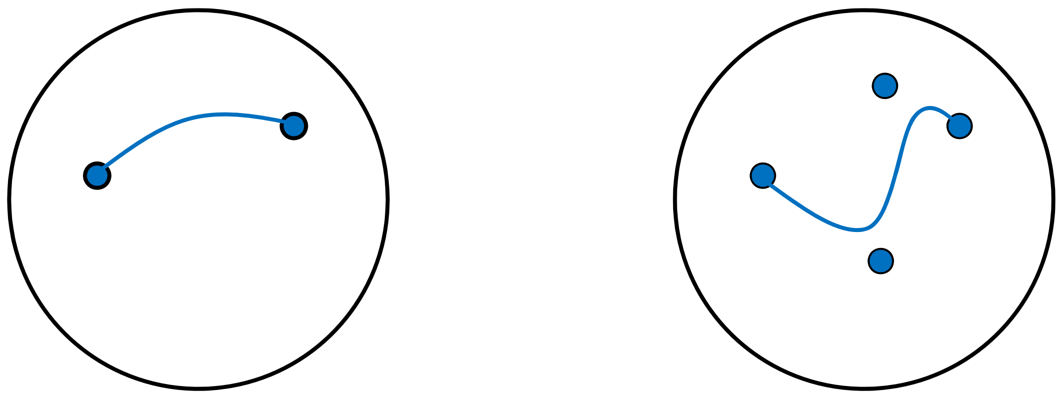
  - mit $q_1 \cdot q_2 = \cos\Omega$
  - für $t \in [0; 1]$ und Einheitsquaternionen q_i, q_{i+1}

- Problem: welche von beiden Richtungen auf der 4D Kugel?
  - q und $-q$ beschreiben dieselbe Rotation, da $qpq^{-1} = (-q)p(-q^{-1})$
  - Lösung: interpoliere zwischen

    q_i und q_{i+1} wenn $|q_i - q_{i+1}| < |q_i - (-q_{i+1})|$
    q_i und $-q_{i+1}$ sonst

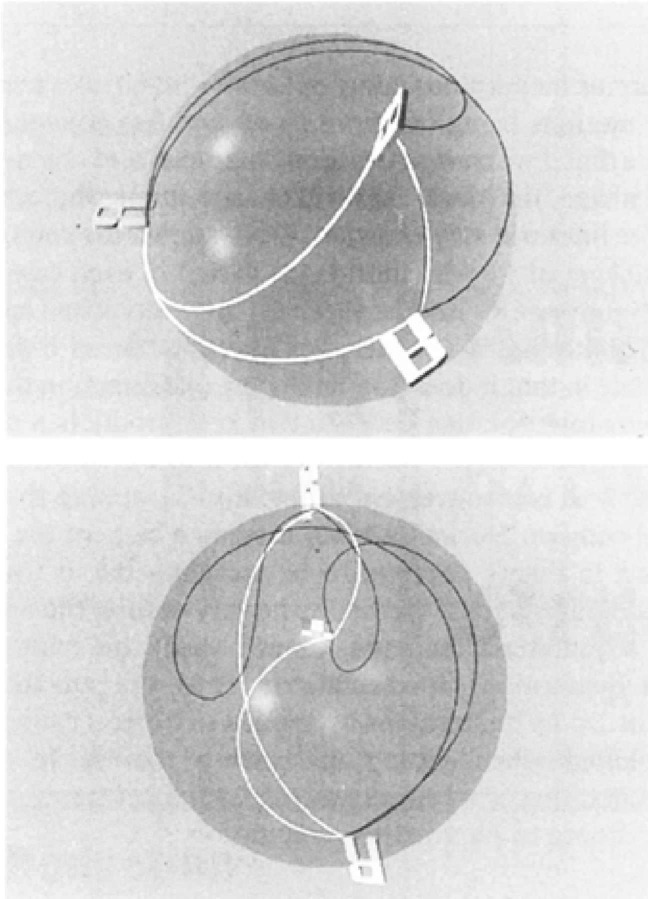
Interpolation von Rotationen

- Problem bei mehr als 2 Keyframes: glatte Übergänge zw. Segmenten
  - Lösung: sphärische Splines (analog zum Kurvenfall)
  - z.B. sphärische kubische Bézier-Kurve über 2 „Eck“-Quaternionen und 2 Ableitungen (siehe [Watt, Watt 92])

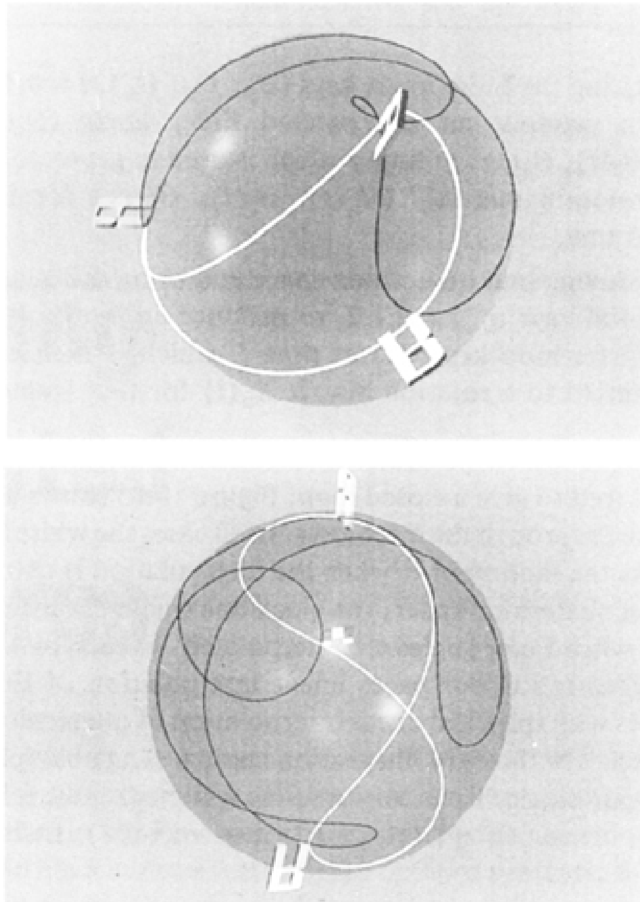


SLERP   Sphärische Bézier-Kurve

Interpolation von Rotationen

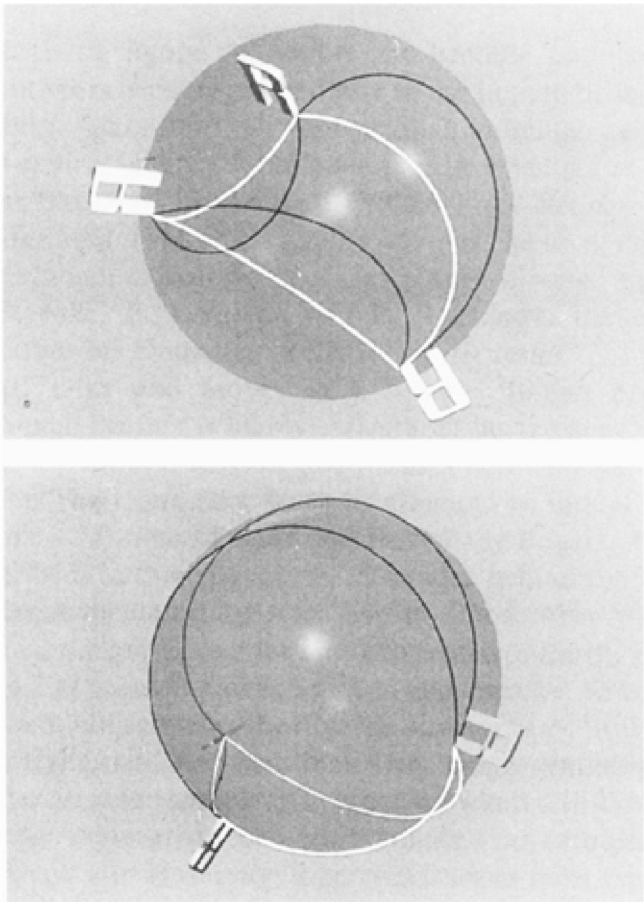


lineare Interpolation
(weiß: Quaternionen,
schwarz: Eulerwinkel)

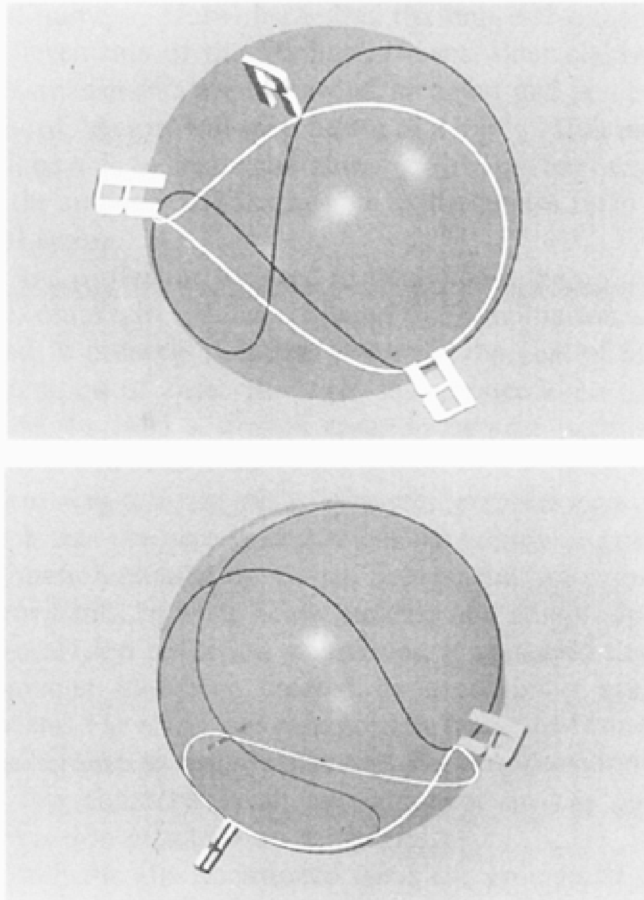


kubische Spline-Interpolation
(weiß: Quaternionen,
schwarz: Eulerwinkel)

Interpolation von Rotationen



lineare Interpolation
(weiß: Quaternionen,
schwarz: Eulerwinkel)

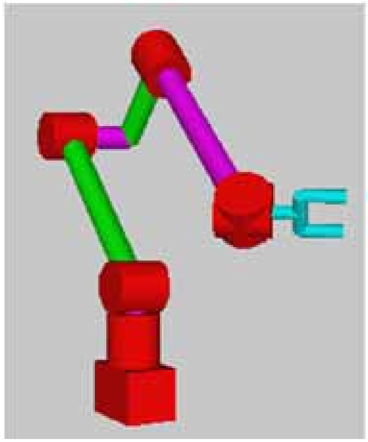


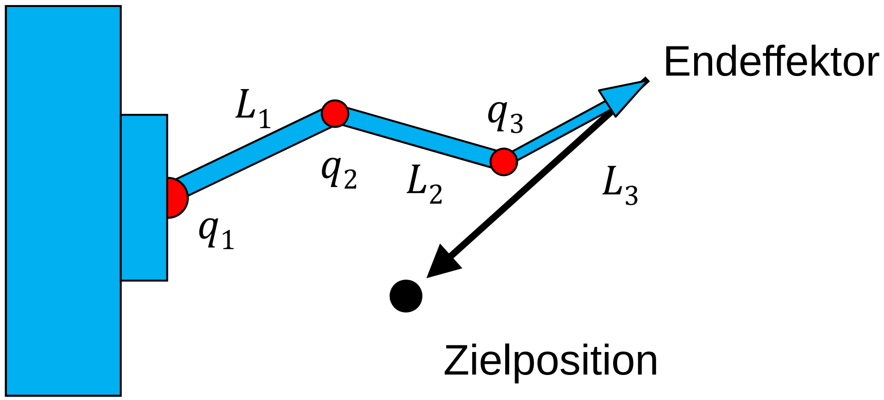
kubische Spline-Interpolation
(weiß: Quaternionen,
schwarz: Eulerwinkel)

Inverse Kinematik

Grundidee: Bestimmung einer Bewegung aus Randbedingungen

- insbesondere von Gelenkwinkeln eines mehrgliedrigen Modells aus einem Zielpunkt
- Eingabe: Zielkonfiguration eines Endeffektors (Endpunkt eines Arms etc.)
- Ziel der Berechnung: Parameter (Winkel) an den Gelenken
- meist numerische Lösung (oft auch keine analytische Lösung möglich)

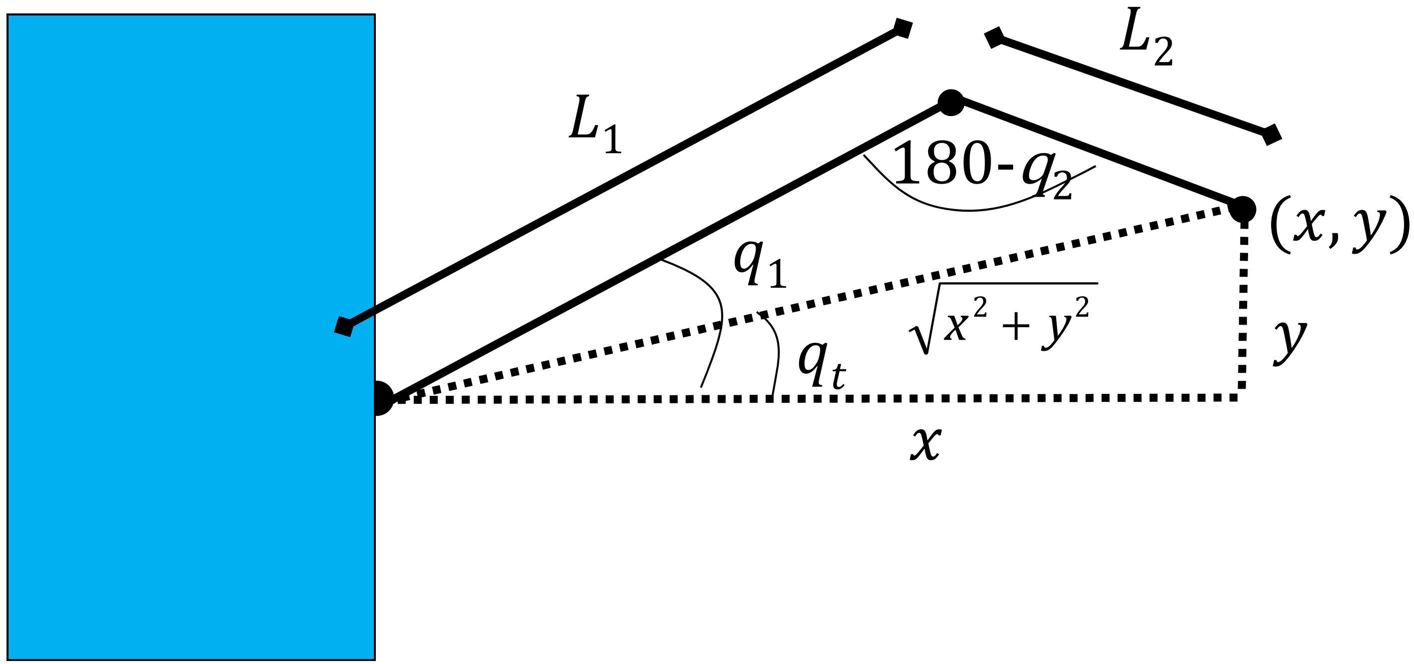




Inverse Kinematik

Einfaches Beispiel (Analytische Lösung hier noch möglich)

- Animation durch Interpolation des Posenvektors (aller Winkelparameter) vom Start- zum Zielwert



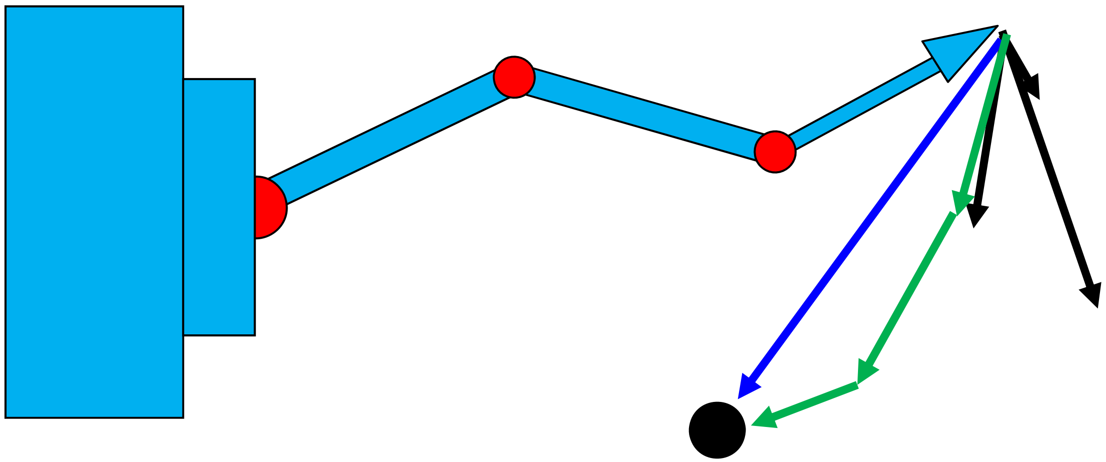
Inverse Kinematik

Skizze der numerischen Lösung

- Lage des Endeffektors $\binom{P}{\alpha}$ ist eine Funktion der Parameter Θ:

$$\binom{P}{\alpha} = F(\Theta) \quad \text{und} \quad \binom{dP}{d\alpha} = \frac{\partial F}{\partial \Theta} d\Theta$$

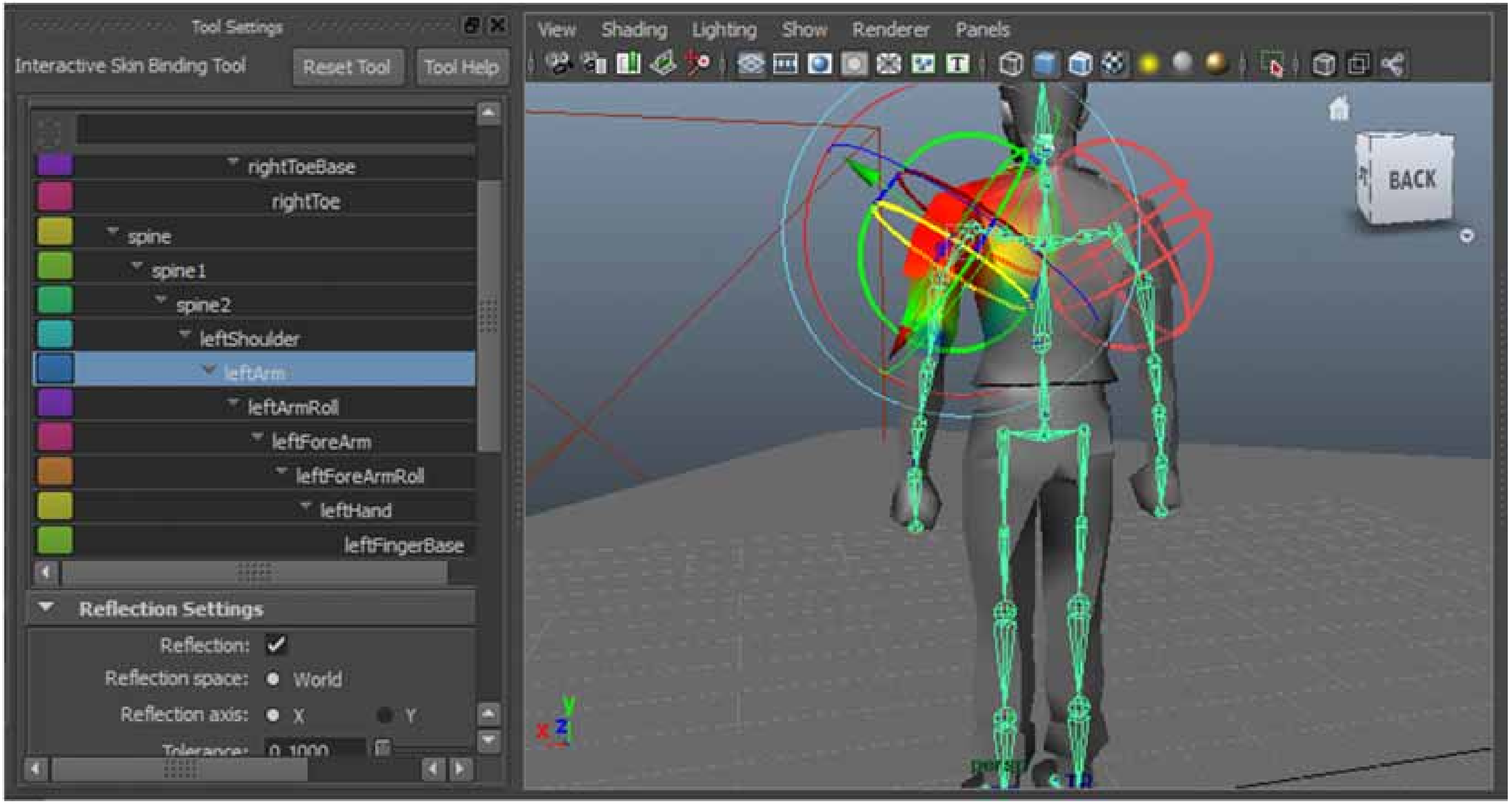
- Linearisierung: wäre F eine lineare Funktion könnte man die Zielkonfiguration nähern (i.d.R. über-/unterbestimmtes System)
  - daher: kleine Schritte, lineare Approximation der eigentlich gekrümmten Bewegung



(Character-)Skinning

Grundidee

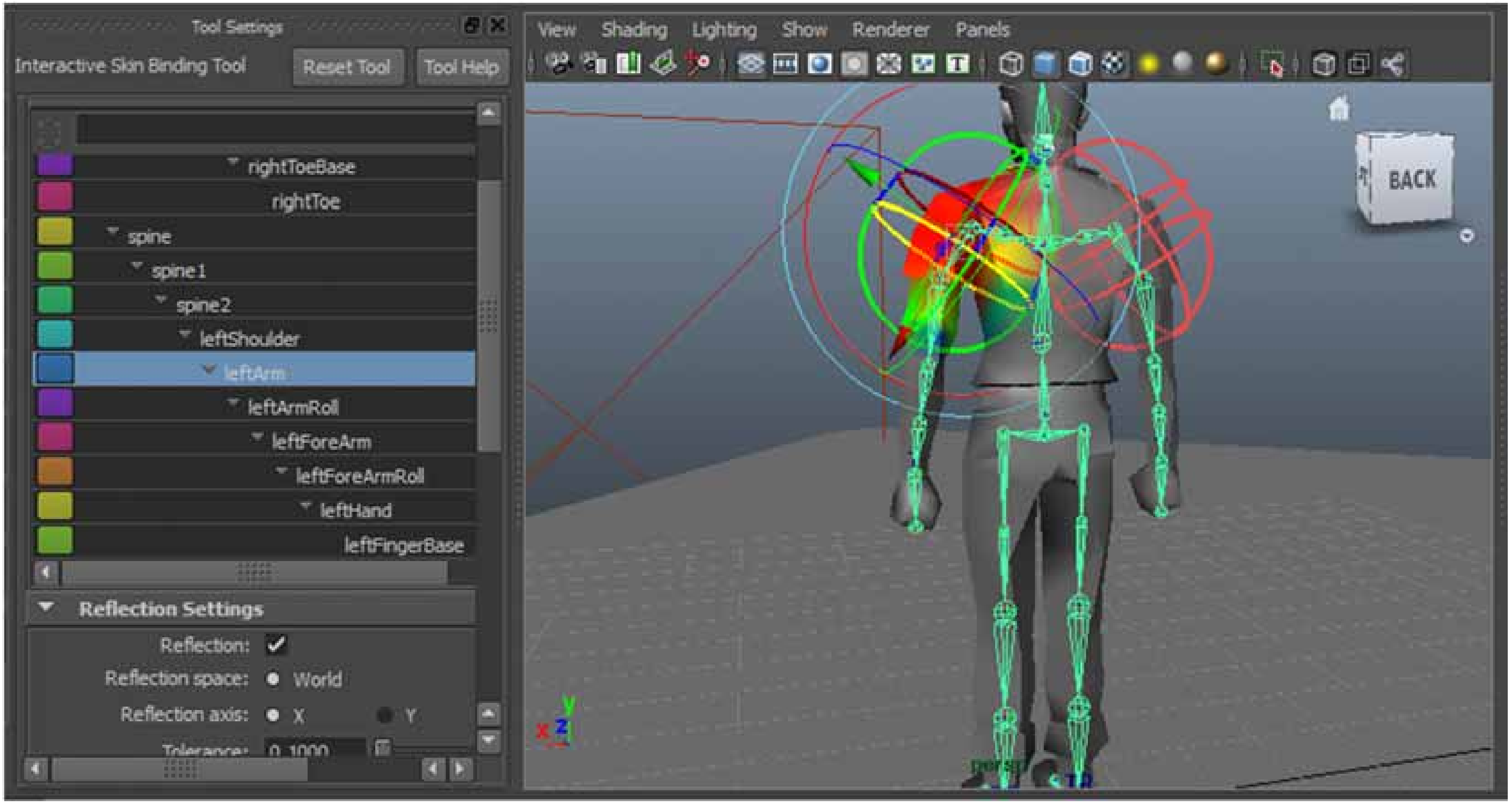
- Skelette repräsentieren einen hierarchischen Bewegungsapparat
- Animation: direkte Kinematik und Interpolation oder inverser Kinematik
- aber: wie überträgt man die Animation auf ein Dreiecksnetz?



(Character-)Skinning

Grundidee

- Vertexpositionen werden durch „Bones“ beeinflusst
- für jedes Segment können wir eine Transformationsmatrix (aus Position und Orientierung bestimmen)



(Character-)Skinning

Skin-Weights und Berechnung von Vertexpositionen

- Einfluss eines Segments auf einen Vertex wird durch ein Gewicht festgelegt (automatisch mit manueller Nachbearbeitung)
- im Vertex-Shader
  - berücksichtige mehrere Segmente/Transformationen (meist bis zu 4)
  - berechne transformierte Vertexpositionen $\boldsymbol{x}_i' = \mathrm{M}_i\boldsymbol{x}$
  - endgültige Vertexposition: Affinkombination $\boldsymbol{x}' = \sum w_i \cdot \boldsymbol{x}_i'$