

Interaktive Computergrafik

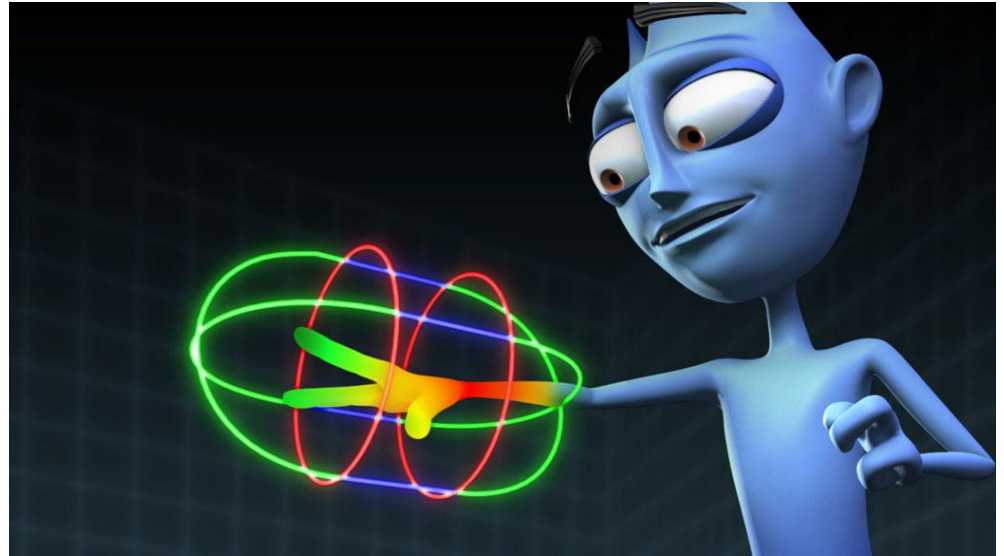
Vorlesung im Sommersemester 2017
Kapitel 7: Animation und Keyframing
(kein Prüfungsstoff)

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



Inhalt

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



Begrifflichkeiten

▶ kinematische Animation

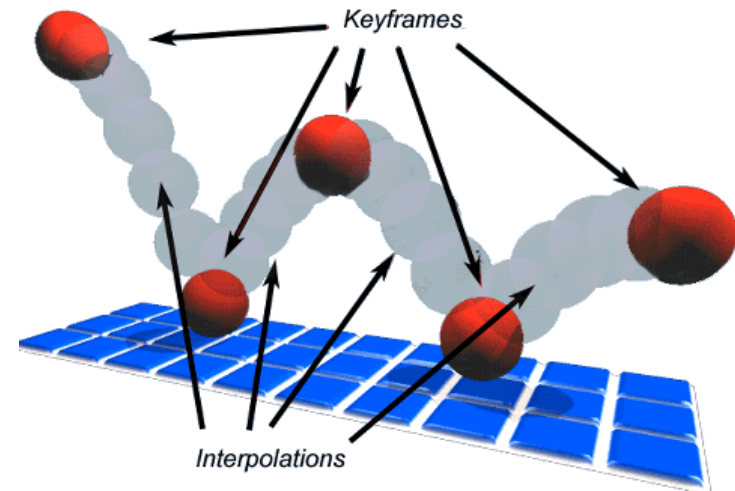
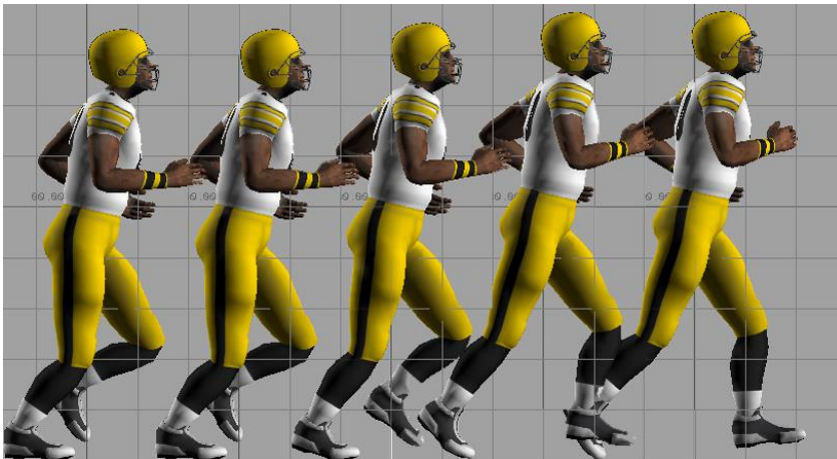
- ▶ **direkte Kinematik**: Bestimmung einer Bewegung aus vorgegebenen, zeitlich veränderlichen Parametern
 - ▶ Ort, Geschwindigkeit, Beschleunigung (aber nicht Kraft)
 - ▶ Pfad-Animation, z.B. Bewegung entlang eines Pfads
- ▶ **inverse/indirekte Kinematik**: Bestimmung aus Randbedingungen

▶ **dynamisches Modell (Simulation)**:

- ▶ basierend auf physikalischen Gesetzen (insb. Kraftgesetze)
- ▶ physikalische Eigenschaften der Objekte
- ▶ Anfangsbedingungen
- ▶ Starrkörper, deformierbare Körper, Flüssigkeiten, ...

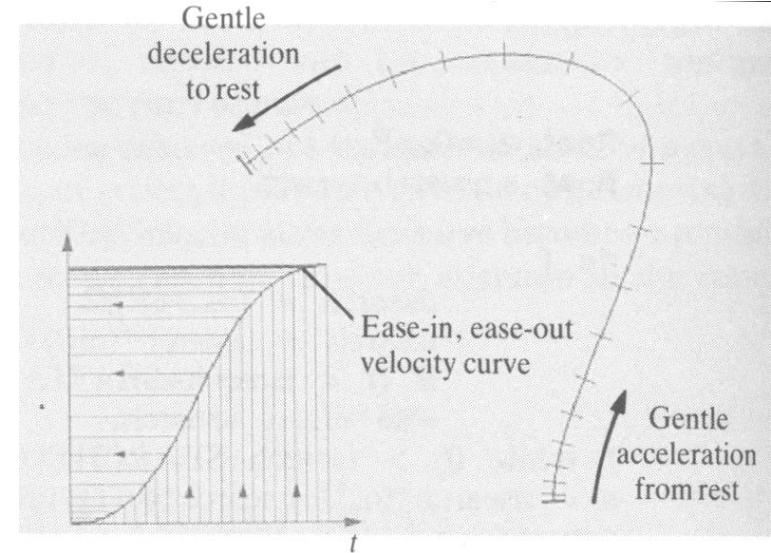
Keyframe-Animation

- ▶ 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)



Interpolation

- ▶ (stückweise-)linear
- ▶ Bézier-Kurven, B-Splines
- ▶ Hermitesche Splines
- ▶ slow-in, slow-out
(auch: ease-in, ease-out)

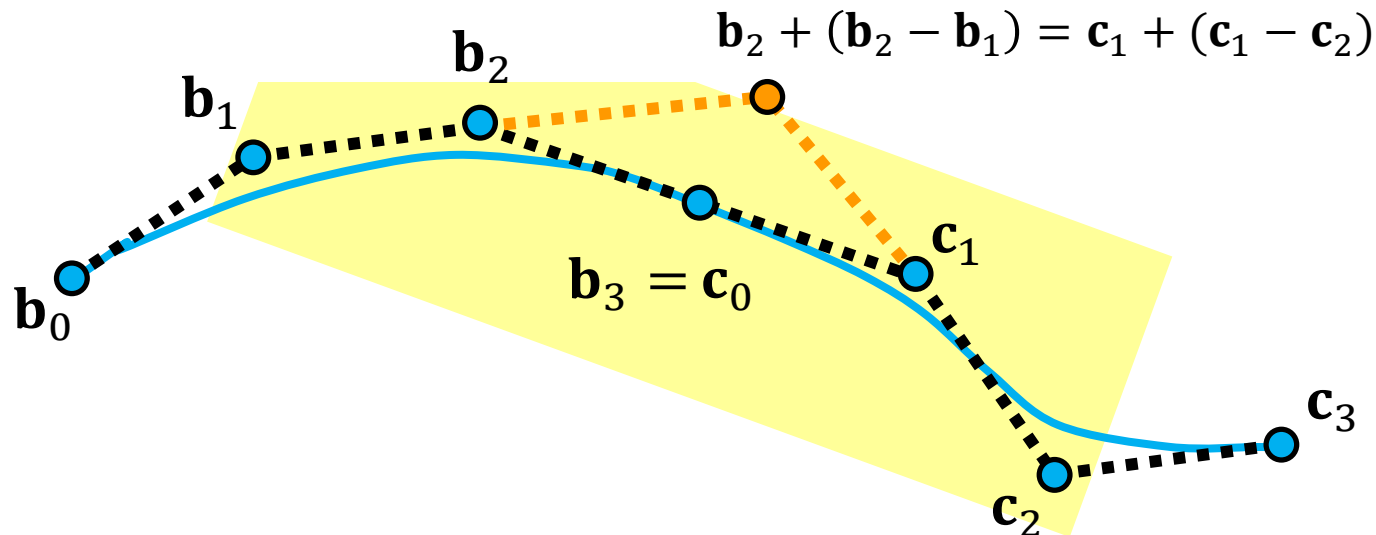


- ▶ können wir direkt auf Positionen anwenden, z.B. bei der Pfadanimation
 - ▶ Objekt folgt einem vorgegeben Pfad aus Kontrollpunkten
 - ▶ Pfadverfolgung zur Generierung der Sicht der ersten Person
 - ▶ Bestimmung der Orientierung?
 - ▶ Kontrolle über Geschwindigkeit?

C^k -Übergang zweier Bézierkurven (gleicher Grad)

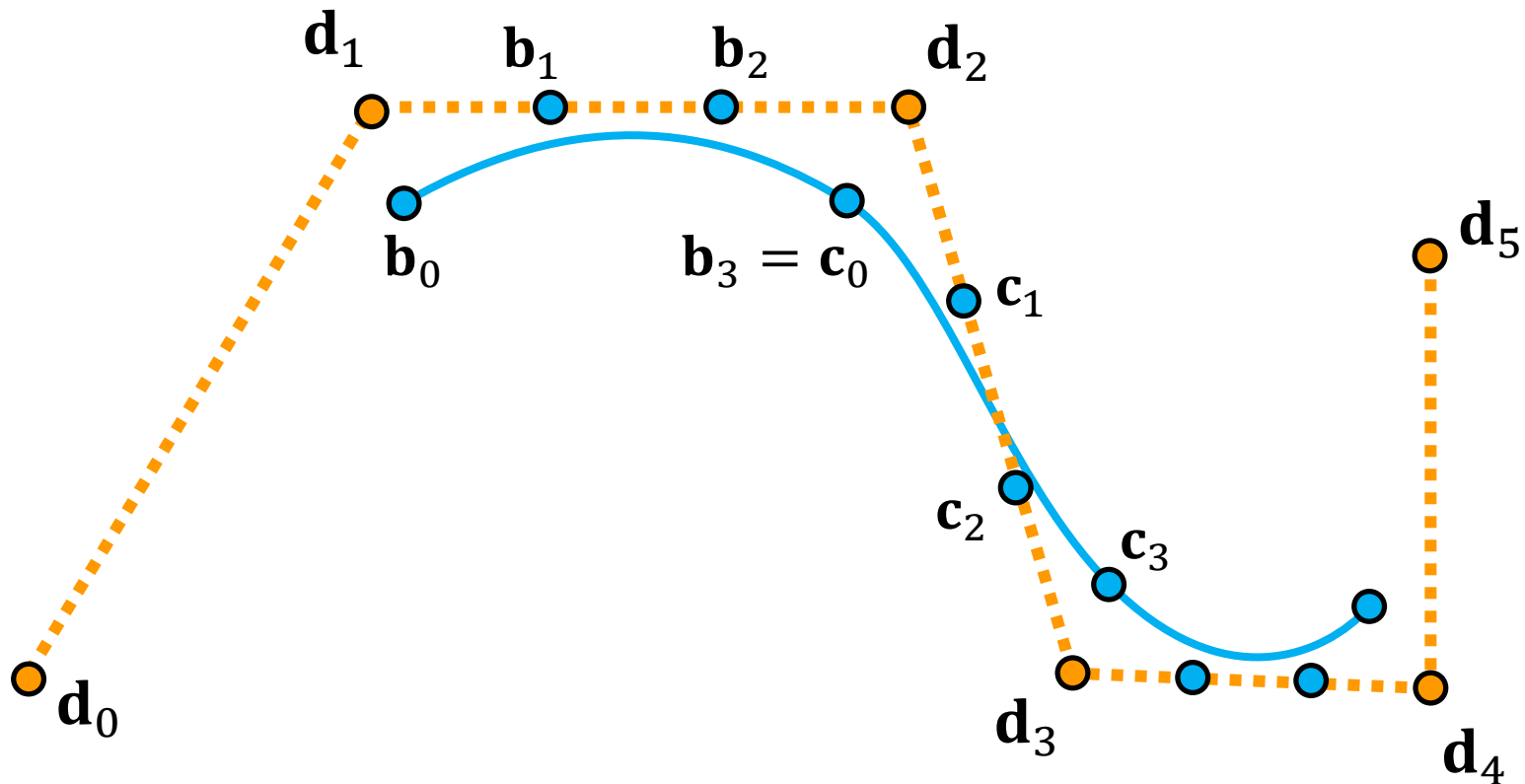
- ▶ $F(u) = \sum_{i=0}^n B_i^n(u) \mathbf{b}_i$ und $G(u) = \sum_{i=0}^n B_i^n(u) \mathbf{c}_i$
 - ▶ C^0 -stetig $\Leftrightarrow F(1) = G(0) \Leftrightarrow \mathbf{b}_n = \mathbf{c}_0$
 - ▶ C^1 -stetig $\Leftrightarrow F'(1) = G'(0) \Leftrightarrow \mathbf{b}_n - \mathbf{b}_{n-1} = \mathbf{c}_1 - \mathbf{c}_0$ (und $\mathbf{b}_n = \mathbf{c}_0$)
 - ▶ C^2 -stetig $\Leftrightarrow C^1$ -stetig und $F''(1) = G''(0)$
 $\Leftrightarrow \mathbf{b}_{n-1} + (\mathbf{b}_{n-1} - \mathbf{b}_{n-2}) = \mathbf{c}_1 + (\mathbf{c}_1 - \mathbf{c}_2)$

- ▶ Beispiel: kubische Bézierkurven und „A-Frame“-Eigenschaft



B-Splines definiert durch A-Frames

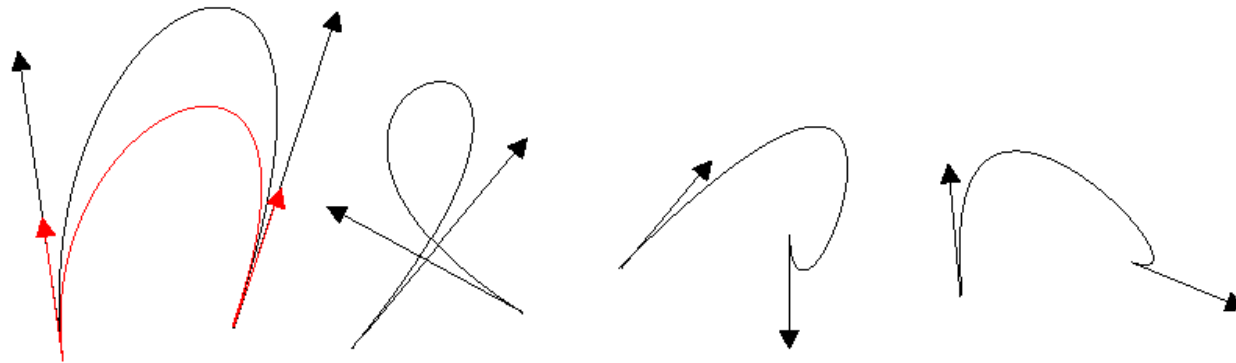
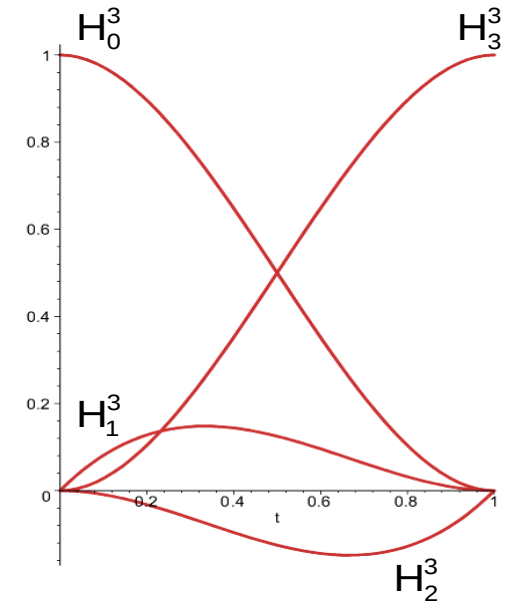
- ▶ statt Bézier-Punkte anzugeben kann man die Ecken der A-Frames festlegen, um einen C^2 -stetigen sog. **B-Spline** zu konstruieren
- ▶ diese neuen Kontrollpunkte $\mathbf{d}_i$ nennt man de Boor-Punkte
- ▶ je 4 de Boor-Punkte definieren eine kubische Bézier-Teilkurve



Univariate Interpolation

Hermite'sche Splines

- ▶ Koeffizienten beschreiben die Endpunkte und Ableitungen/Tangenten
- ▶ $f(x) = H_0^3(t) \cdot f(x_i) + H_1^3(t) \cdot f'(x_i) + H_2^3(t) \cdot f'(x_{i+1}) + H_3^3(t) \cdot f(x_{i+1})$
- ▶ mit $t = \frac{x-x_i}{x_{i+1}-x_i} \in [0; 1]$ für $x_i \leq x \leq x_{i+1}$



$$H_0^3(t) = (1-t)^2(1+2t)$$

$$H_1^3(t) = t(1-t)^2$$

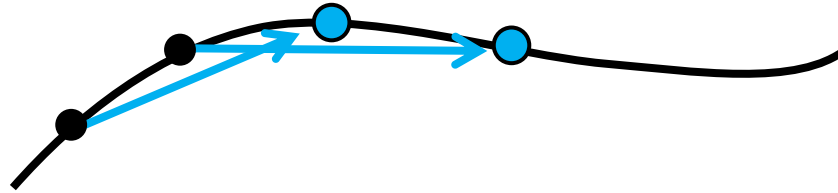
$$H_2^3(t) = -t^2(1-t)$$

$$H_3^3(t) = (3-2t)t^2$$

Pfadanimation

► Pfad ergibt sich durch Interpolation zwischen Orts-Keyframes

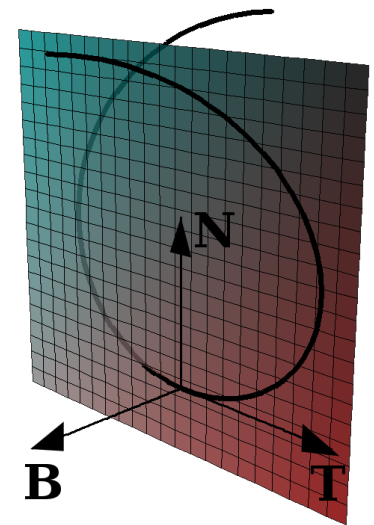
- 1) Orientierung direkt ebenfalls durch Keyframes definieren
- 2) Center-of-Interest (COI) Animation Blick gerichtet auf COI, z.B. ein anderes (bewegtes) Objekt oder ein animierter Punkt entlang einer Kurve



- 3) Orientierung durch Ableitung(en) der Kurve (Frenet-Dreibein)

► lokales Bezugssystem/Tangenten weisen oft eine schnell variierende Orientierung auf

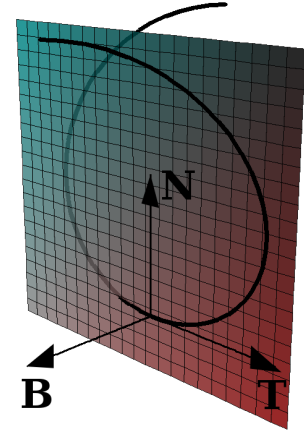
► Kurve $\mathbf{r}(s)$: $\mathbf{T} = \frac{d\mathbf{r}}{ds}$, $\mathbf{N} = \frac{d\mathbf{T}}{ds}$, $\mathbf{B} = \mathbf{T} \times \mathbf{N}$



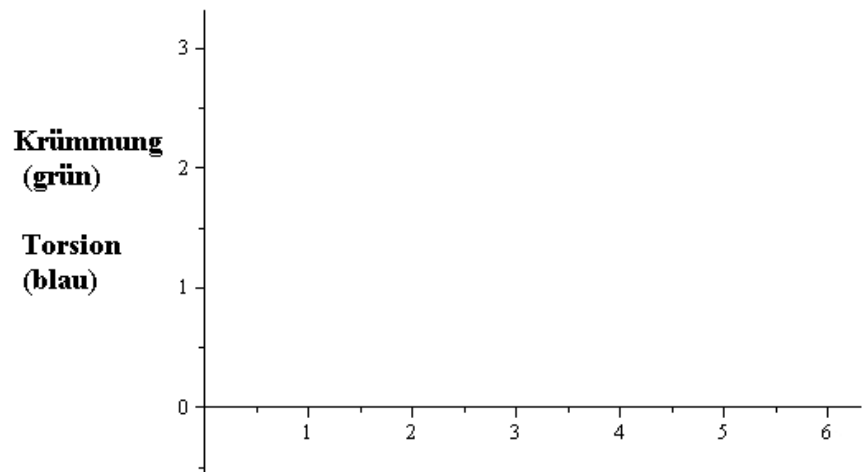
Keyframe- und kinematische Animation

Frenet-Dreibein

- ▶ Ableitung(en) der Kurve (Frenet-Dreibein)
 - ▶ lokales Bezugssystem/Tangenten weisen oft eine schnell variierende Orientierung auf
 - ▶ Kurve $\mathbf{r}(s)$: $\mathbf{T} = \frac{d\mathbf{r}}{ds}$, $\mathbf{N} = \frac{d\mathbf{T}}{ds}$, $\mathbf{B} = \mathbf{T} \times \mathbf{N}$

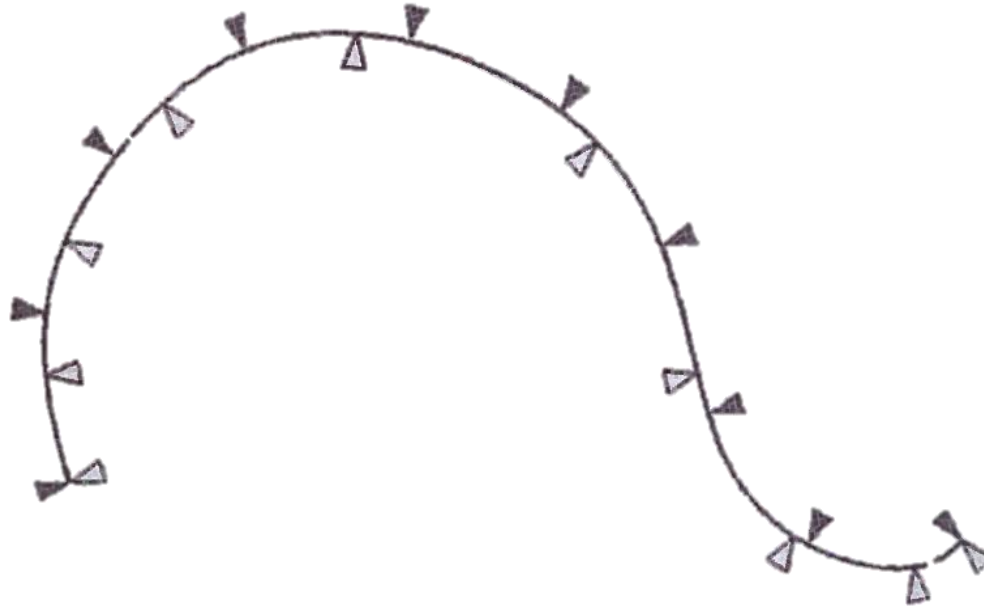


Torus-Knoten mit Tangentialvektor (braun), Normalenvektor (grün) und Binormalenvektor (blau)



Ablaufgeschwindigkeit

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



Erster Ansatz: Konstanter Geschwindigkeitsbetrag

► Bogenlängenparametrisierung $\mathbf{r}_{arc}(s)$

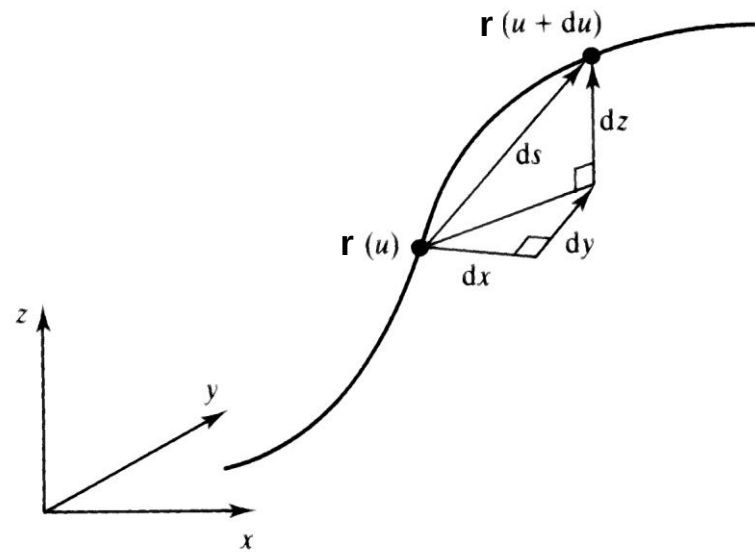
► Umparametrisierung $u \rightarrow s$

► Bogenlänge

$$s(u) = \int_{u_0}^u \left\| \frac{d\mathbf{r}(u')}{du'} \right\| du'$$

► in der Praxis: numerische Integration,
z.B. Simpson-Regel/Keplersche Fassregel

(Näherung einer Kurve durch eine einfach integrierbare Parabel)



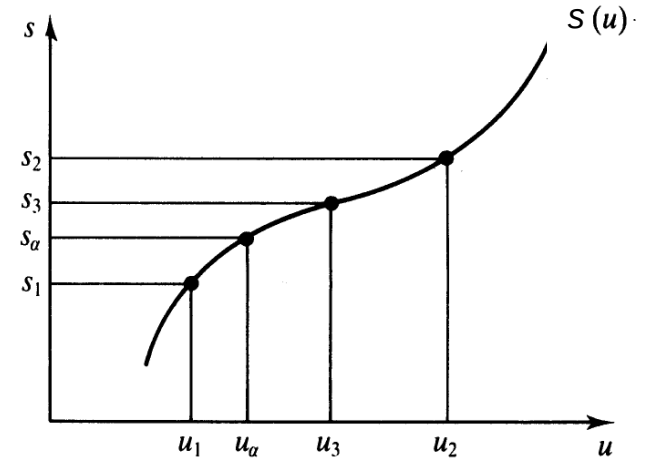
$$\int_{u_1}^{u_2} f(u') du' = \frac{u_2 - u_1}{6} \left(f(u_1) + 4f\left(\frac{1}{2}(u_1 + u_2)\right) + f(u_2) \right)$$

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

Ablaufgeschwindigkeit von Keyframe-Animationen

Gesucht: Umkehrfunktion $u = u(s)$

- ▶ Schritt 1: Suche durch Bisektion (Intervallhalbierungsverfahren)
 - ▶ effizient, da $s(u)$ monoton steigend
 - ▶ suche Intervall $[u_i; u_{i+1})$ so, dass $s(u_i) \leq s \leq s(u_{i+1})$

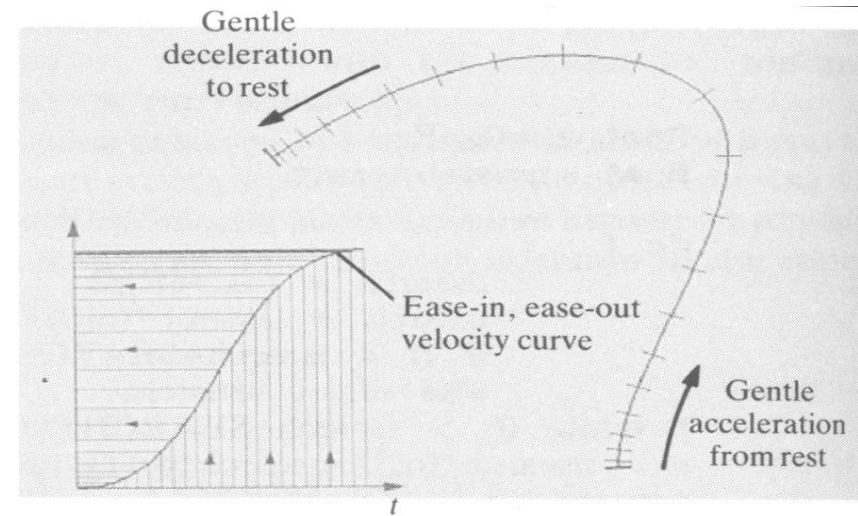
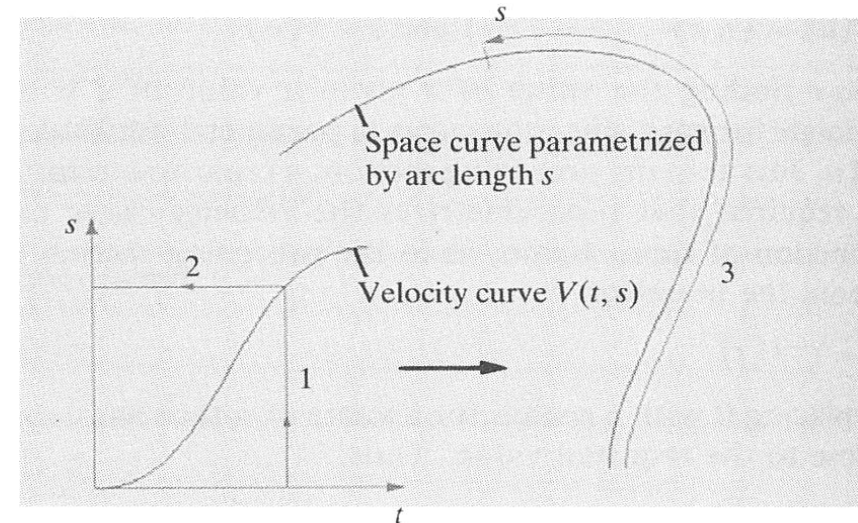


- ▶ Schritt 2: suche exaktes u
 - ▶ es gilt: $s = s(u) \Leftrightarrow s - s(u) = 0$
 - ▶ suche Nullstelle, z.B. mit Newton-Raphson-Methode
- ▶ $\Rightarrow$ Berechnung von $\mathbf{r}_{arc}(s) = \mathbf{r}(u(s))$ mit tabelliertem $s(u)$
 - ▶ erlaubt konstante Bahngeschwindigkeit

Ablaufgeschwindigkeit von Keyframe-Animationen

Steuerung der Geschwindigkeit

- ▶ mit einer Geschwindigkeitskurve (ordnet Zeit einer Strecke zu)
- ▶ Bahngeschwindigkeit: $\frac{ds}{dt}$
- ▶ Position entlang der Kurve $\mathbf{r}_{time}(t)$:
 - ▶ $V(t)$ = Zurückgelegte Strecke = Integral über Geschwindigkeit
 - ▶ $s = V(t)$
 - ▶ $\mathbf{r}_{time}(t) = \mathbf{r}(u(V(t)))$



Interpolation von Rotationen/Orientierung

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

► (lineare) Interpolation von

► von Richtungsvektoren?

► von Polarwinkeln?

► von Rotationsmatrizen?

► Eulerwinkeln?

Problem

Länge

Singularität an den Polen

keine Orthogonalität

Kardanische Blockade

► so nicht ☺: Classic Game Postmortem,
GDC 2013, David Braben
<http://www.gdcvault.com/play/1014628/>



- ▶ 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
- ▶ Probleme bei der Interpolation/Fortsetzung von Rotationen
 - ▶ wenn z.B. Kamera-/Betrachterorientierung in einer interaktiven Anwendung repräsentiert werden soll
- ▶ Beispiel: Rotationen um die y -, x - bzw. z -Achse (yaw, pitch, roll) mit Eulerwinkeln ψ , θ und ϕ :

$$\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_x(\theta)\mathbf{R}_y(\psi)$$

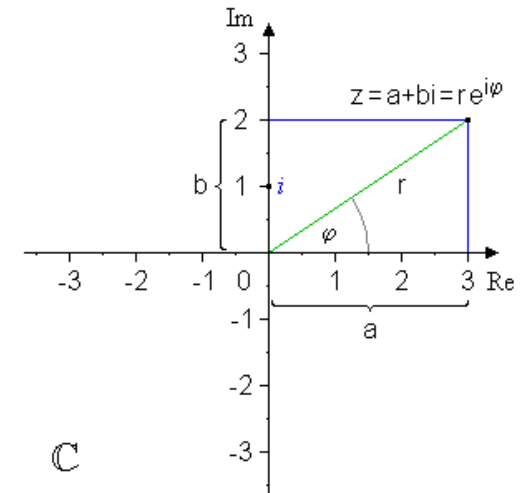
- ▶ wenn $\psi = 0^\circ$ (keine Rotation um die y -Achse) und $\theta = 90^\circ$ (um die x -Achse)
 - ▶ dann ist keine Rotation um die z -Achse mehr möglich, da diese der y -Achse entspricht

Darstellung von Rotationen



Komplexe Zahlen und Quaternionen

- ▶ Multiplikation zweier komplexer Zahlen bewirkt
 - ▶ Addition der Winkel, Multiplikation der Länge
 - ▶ eine Zahl $z = a + b\mathbf{i} = e^{i\varphi} \in \mathbb{C}$ 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 $\mathbb{H}$ (Menge der Quaternionen)
 - ▶ $q(s, x, y, z) = s\mathbf{1} + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, mit
 - ▶ $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -\mathbf{1}$
 - ▶ $\mathbf{ij} = \mathbf{k}, \mathbf{ji} = -\mathbf{k}$ (analog weitere zyklische Permutationen)
 - ▶ Kurzschreibweise: $q(s, \mathbf{v}) = s\mathbf{1} + v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$

► Addition: $q_1 + q_2 = (s_1, \mathbf{v}_1) + (s_2, \mathbf{v}_2) = (s_1 + s_2, \mathbf{v}_1 + \mathbf{v}_2)$

► Multiplikation (nicht kommutativ!):

$$\begin{aligned} q_1 q_2 &= (s_1, \mathbf{v}_1)(s_2, \mathbf{v}_2) = \\ &= (s_1 + v_{1,x}\mathbf{i} + v_{1,y}\mathbf{j} + v_{1,z}\mathbf{k})(s_2 + v_{2,x}\mathbf{i} + v_{2,y}\mathbf{j} + v_{2,z}\mathbf{k}) = \dots = \\ &= (s_1 s_2 - \mathbf{v}_1 \mathbf{v}_2, s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \end{aligned}$$

► Multiplikation mit einem Skalar ist kommutativ:

$$rq = (r, \mathbf{0})(s, \mathbf{v}) = (rs, r\mathbf{v})$$

► Konjugat:

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

► Skalarprodukt $\cdot : \mathbb{H} \times \mathbb{H} \rightarrow \mathbb{R}$

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

Quaternionen



▶ Norm $|\cdot|: \mathbb{H} \rightarrow \mathbb{R}$

$$|q| = \sqrt{q \cdot \bar{q}} = \sqrt{s^2 + v_x^2 + v_y^2 + v_z^2}$$

▶ Multiplikation ist normtreu:

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

▶ Einheitsquaternion:

$$|q| = 1 \Leftrightarrow q \bar{q} = 1 \Leftrightarrow q^{-1} = \bar{q}$$

▶ jedes Einheitsquaternion lässt sich darstellen als:

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

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

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

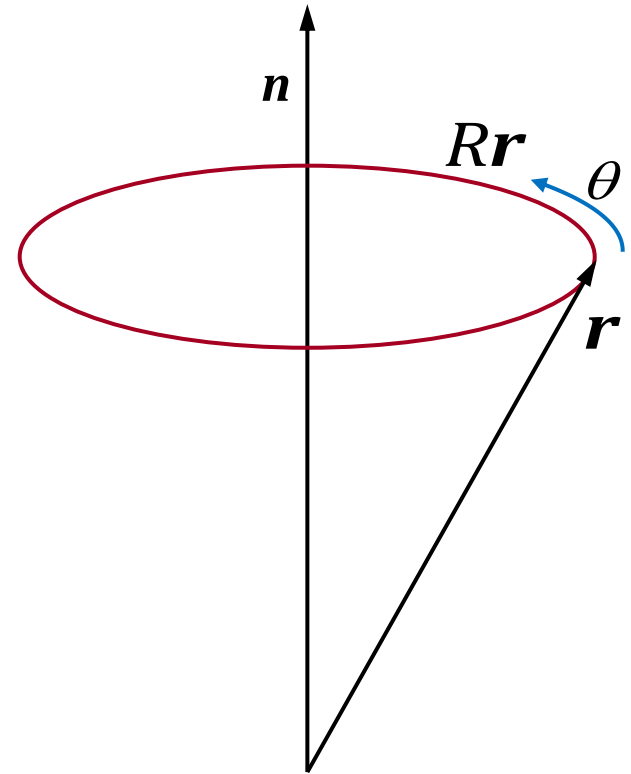
- ▶ sogenannte reine Quaternionen (pure quaternions)
- ▶ weiter ist der Operator R_q über das Quaternionenprodukt mit einem Einheitsquaternion q definiert als
$$R_q(p) = qpq^{-1}$$
- ▶ R_q beschreibt eine Rotation von $\mathbf{r}$ um...? Welche Rotation beschreibt q ?
- ▶ $a \cdot q$ mit $a \in \mathbb{R} \setminus 0$ beschreibt dieselbe Rotation: $(aq)p(aq)^{-1} = aa^{-1} \dots$

Vorbereitung zur Herleitung der Rotationseigenschaft von R_q

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

$$R\mathbf{r} = \mathbf{r}_{\parallel} + \cos \theta \mathbf{r}_{\perp} + \sin \theta (\mathbf{n} \times \mathbf{r}_{\perp})$$

$$\text{mit } \mathbf{r}_{\perp} = \mathbf{r} - \mathbf{n}(\mathbf{n} \cdot \mathbf{r}) \text{ und } \mathbf{r}_{\parallel} = \mathbf{n}(\mathbf{n} \cdot \mathbf{r})$$

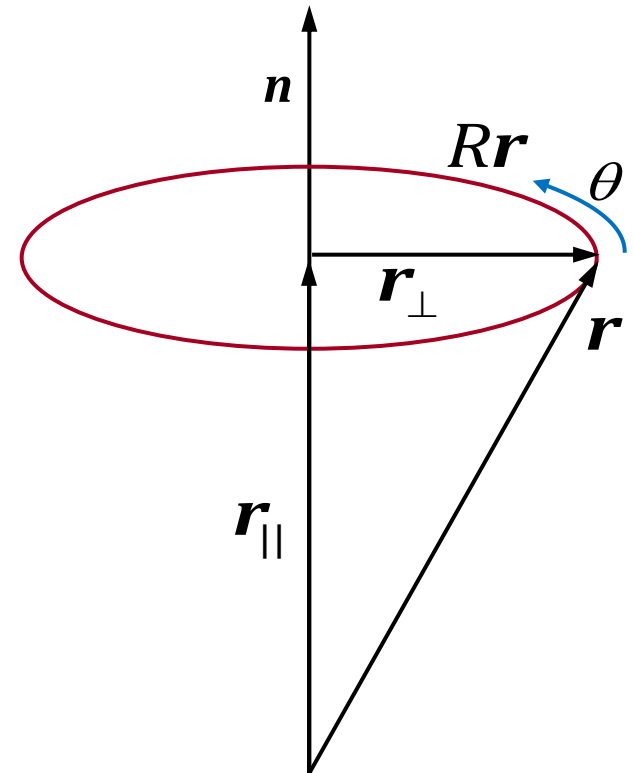


Vorbereitung zur Herleitung der Rotationseigenschaft von R_q

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

$$R\mathbf{r} = \mathbf{r}_{\parallel} + \cos \theta \mathbf{r}_{\perp} + \sin \theta (\mathbf{n} \times \mathbf{r}_{\perp})$$

$$\text{mit } \mathbf{r}_{\perp} = \mathbf{r} - \mathbf{n}(\mathbf{n} \cdot \mathbf{r}) \text{ und } \mathbf{r}_{\parallel} = \mathbf{n}(\mathbf{n} \cdot \mathbf{r})$$



Vorbereitung zur Herleitung der Rotationseigenschaft von R_q

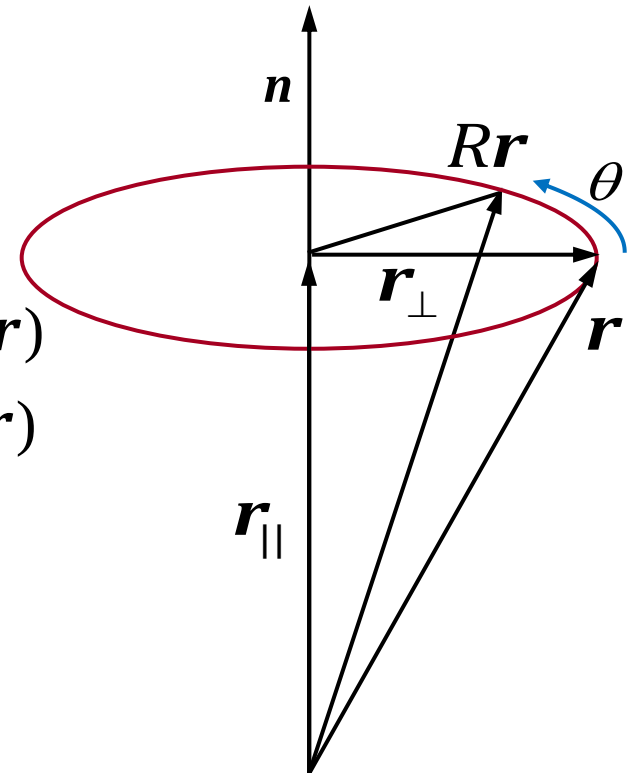
- Rotation von $\mathbf{r}$ um eine normierte Achse $\mathbf{n}$ um den Winkel θ :

$$R\mathbf{r} = \mathbf{r}_{\parallel} + \cos \theta \mathbf{r}_{\perp} + \sin \theta (\mathbf{n} \times \mathbf{r}_{\perp})$$

$$\text{mit } \mathbf{r}_{\perp} = \mathbf{r} - \mathbf{n}(\mathbf{n} \cdot \mathbf{r}) \text{ und } \mathbf{r}_{\parallel} = \mathbf{n}(\mathbf{n} \cdot \mathbf{r})$$

- Einsetzen und Umformen ergibt:

$$\begin{aligned} R\mathbf{r} &= \mathbf{n}(\mathbf{n} \cdot \mathbf{r}) + \cos \theta (\mathbf{r} - \mathbf{n}(\mathbf{n} \cdot \mathbf{r})) + \sin \theta (\mathbf{n} \times \mathbf{r}) \\ &= \cos \theta \mathbf{r} + (1 - \cos \theta) \mathbf{n}(\mathbf{n} \cdot \mathbf{r}) + \sin \theta (\mathbf{n} \times \mathbf{r}) \end{aligned}$$



Herleitung der Rotationseigenschaft von R_q

► eben definierter Operator

$$\begin{aligned} R_q(p) &= qpq^{-1} = qp\bar{q} = (s, \mathbf{v})(0, \mathbf{r})(s, -\mathbf{v}) \\ &= (0, (s^2 - \mathbf{v} \cdot \mathbf{v})\mathbf{r} + 2\mathbf{v}(\mathbf{v} \cdot \mathbf{r}) + 2s\mathbf{v} \times \mathbf{r}) \end{aligned}$$

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

$$\begin{aligned} R_q(p) &= (0, (\cos^2 \varphi - \sin^2 \varphi)\mathbf{r} + \\ &\quad 2\sin^2 \varphi \mathbf{n}(\mathbf{n} \cdot \mathbf{r}) + 2\cos \varphi \sin \varphi \mathbf{n} \times \mathbf{r}) \\ &= (0, \cos 2\varphi \mathbf{r} + (1 - \cos 2\varphi)\mathbf{n}(\mathbf{n} \cdot \mathbf{r}) + \sin 2\varphi \mathbf{n} \times \mathbf{r}) \end{aligned}$$

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

Beschreibung einer Rotation durch ein Quaternion

- ▶ Rotation eines Punktes $\mathbf{r}$ um Winkel θ um die Achse $\mathbf{n}$ durch das Einheitsquaternion

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

- ▶ und dem imaginären Quaternion

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

- ▶ Operation

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

- ▶ eine Rotation beschrieben durch Quaternionen kann natürlich einfach in Matrixdarstellung gebracht werden

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
 - ▶ keine Kardanische Blockade

Interpolation von Rotationen



▶ Ziel: Orientierungen als Keyframes $q_0, q_1, \dots, q_{n-1}$

▶ wie sieht interpolierte Rotationsbewegung aus?

▶ Glattheit? Segmentgrenzen?

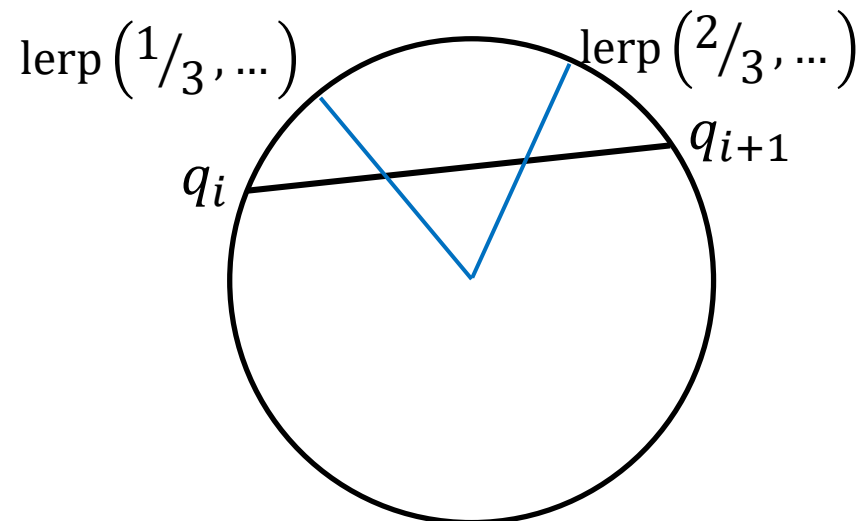
▶ einfachste Lösung:

▶ $q(t) = \text{lerp}(t, q_i, q_{i+1}) = (1 - t)q_i + tq_{i+1}$

▶ wichtig: $q(t)$ hinterher immer normieren!

▶ Vorteil: kein Gimbal Lock

▶ Problem: keine konstante Winkelgeschwindigkeit (an den Enden langsamer als in der Mitte)



Sphärische lineare Interpolation von Quaternionen

- ▶ lerp beschreibt eine Sekante
- ▶ slerp (= **s**pherical **l**inear **i**nter**p**olation) beschreibt Bogen auf der 4D-Kugel

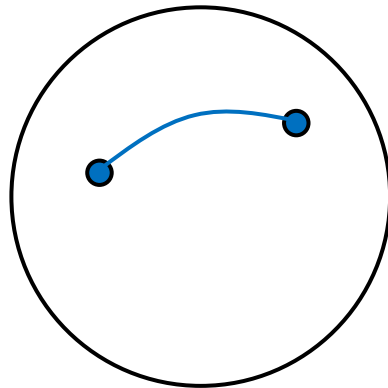
$$\text{slerp}(t, q_i, q_{i+1}) = q_i(\bar{q}_i q_{i+1})^t = \frac{\sin(\Omega(1-t))}{\sin \Omega} q_i + \frac{\sin(\Omega t)}{\sin \Omega} q_{i+1} = \dots$$

- ▶ 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

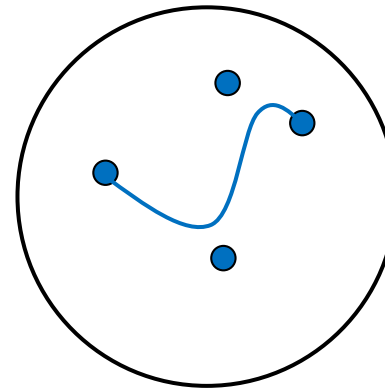
Interpolation von Rotationen



- ▶ 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])

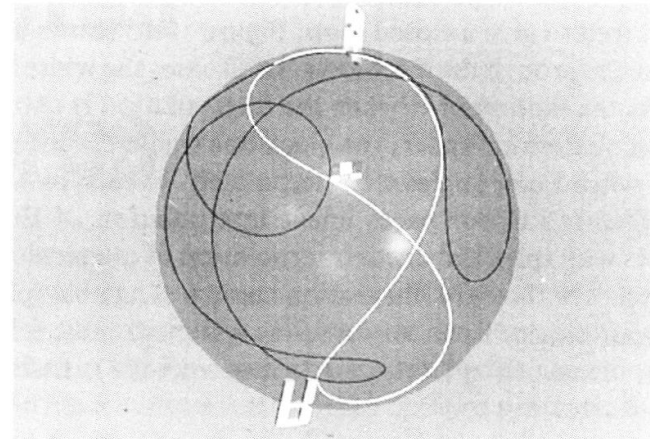
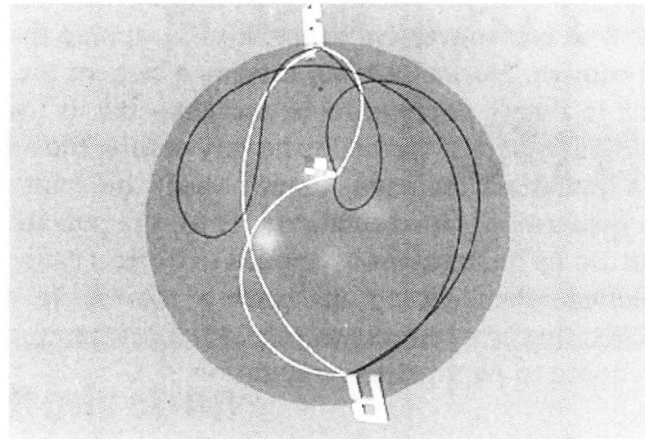
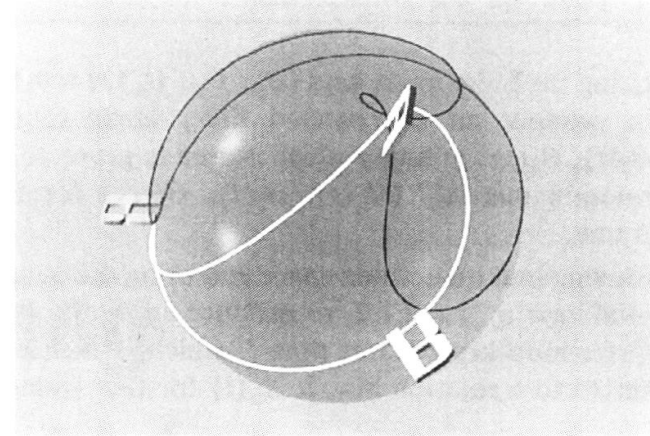
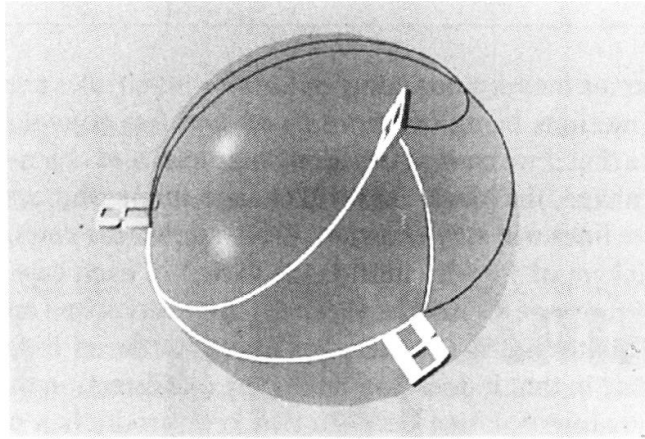


SLERP



Sphärische Bézier-Kurve

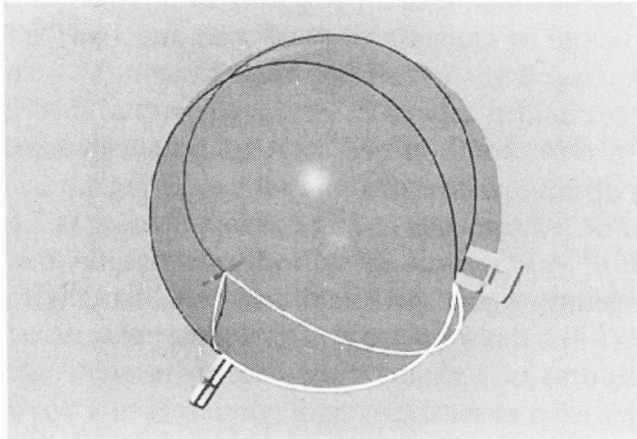
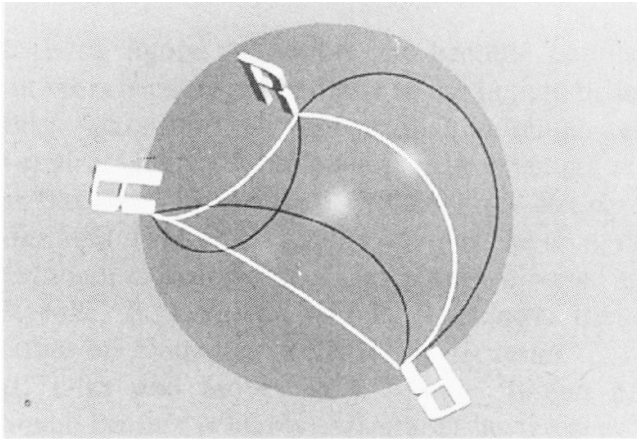
Interpolation von Rotationen



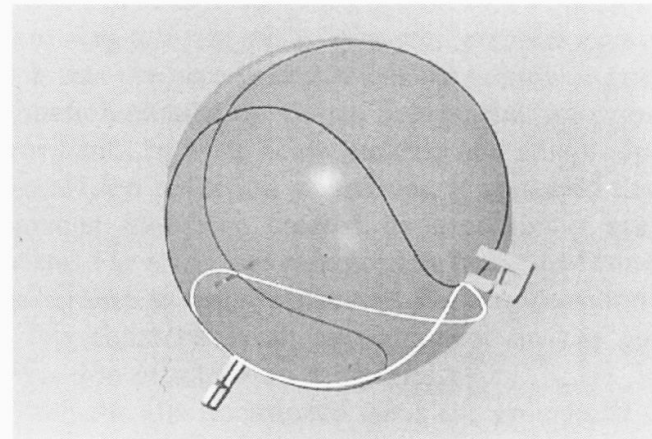
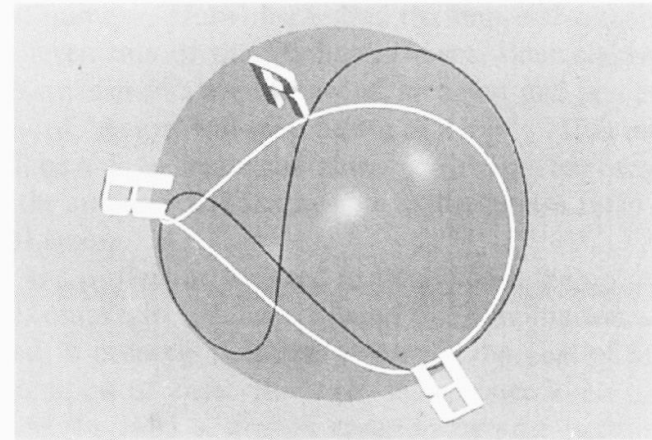
lineare Interpolation
(weiß: Quaternionen,
schwarz: Eulerwinkel)

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

Interpolation von Rotationen



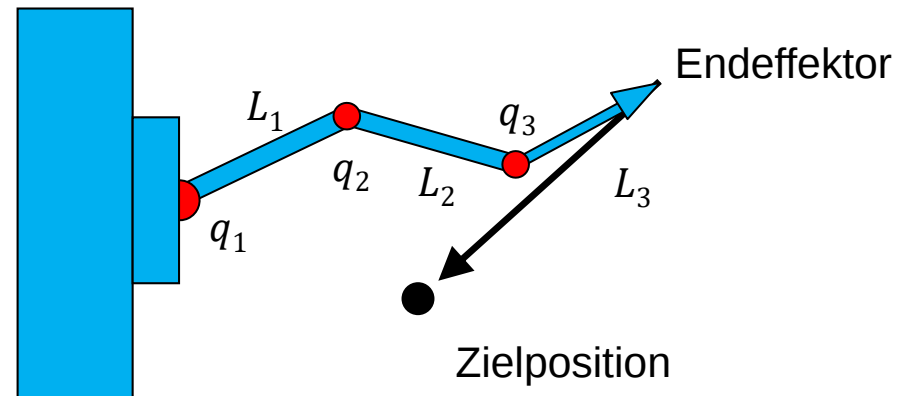
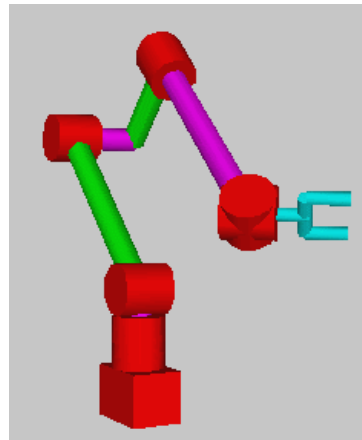
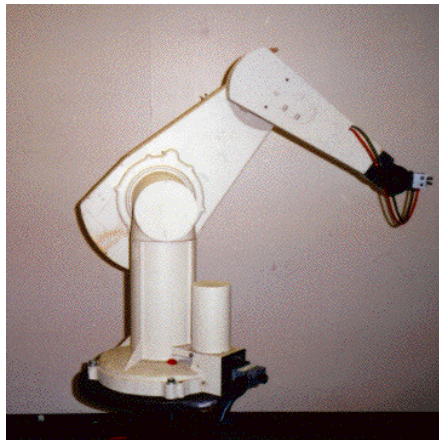
lineare Interpolation
(weiß: Quaternionen,
schwarz: Eulerwinkeln)



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

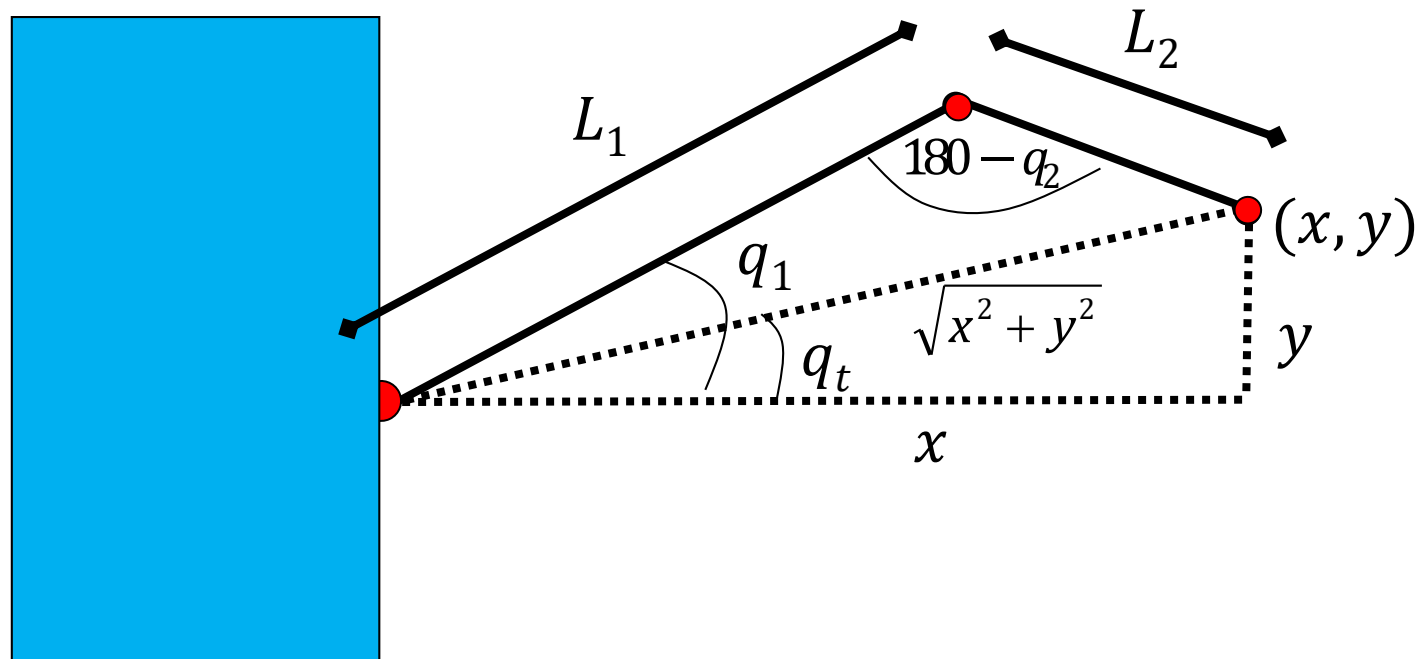
Grundidee: Bestimmung einer Bewegung aus Randbedingungen

- ▶ 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)



Einfaches Beispiel (analytische Lösung hier noch möglich)

- ▶ Animation durch Interpolation des Posenvektors (alle Längen-/Winkelparameter) vom Start- zum Zielwert

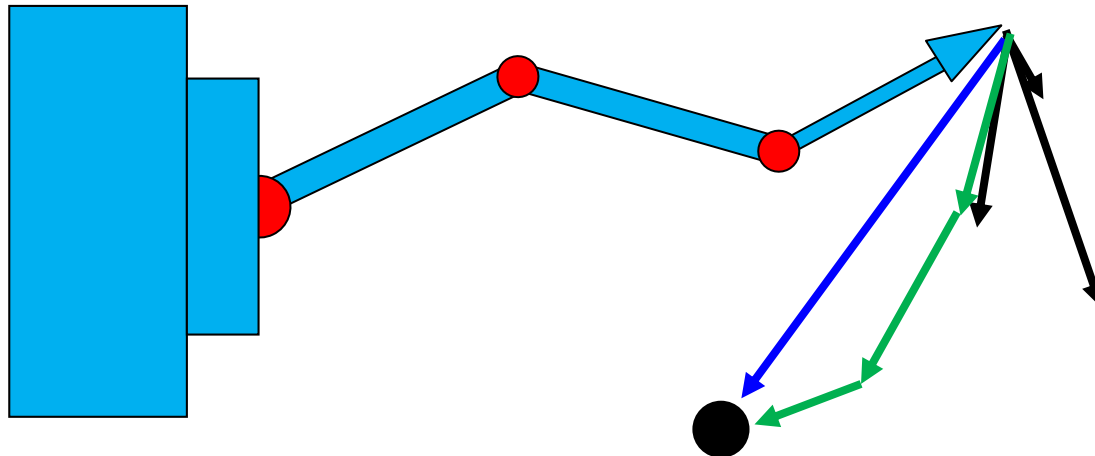


Skizze der numerischen Lösung

- ▶ Lage des Endeffektors $\begin{pmatrix} P \\ \alpha \end{pmatrix}$ ist eine Funktion der Parameter Θ :

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

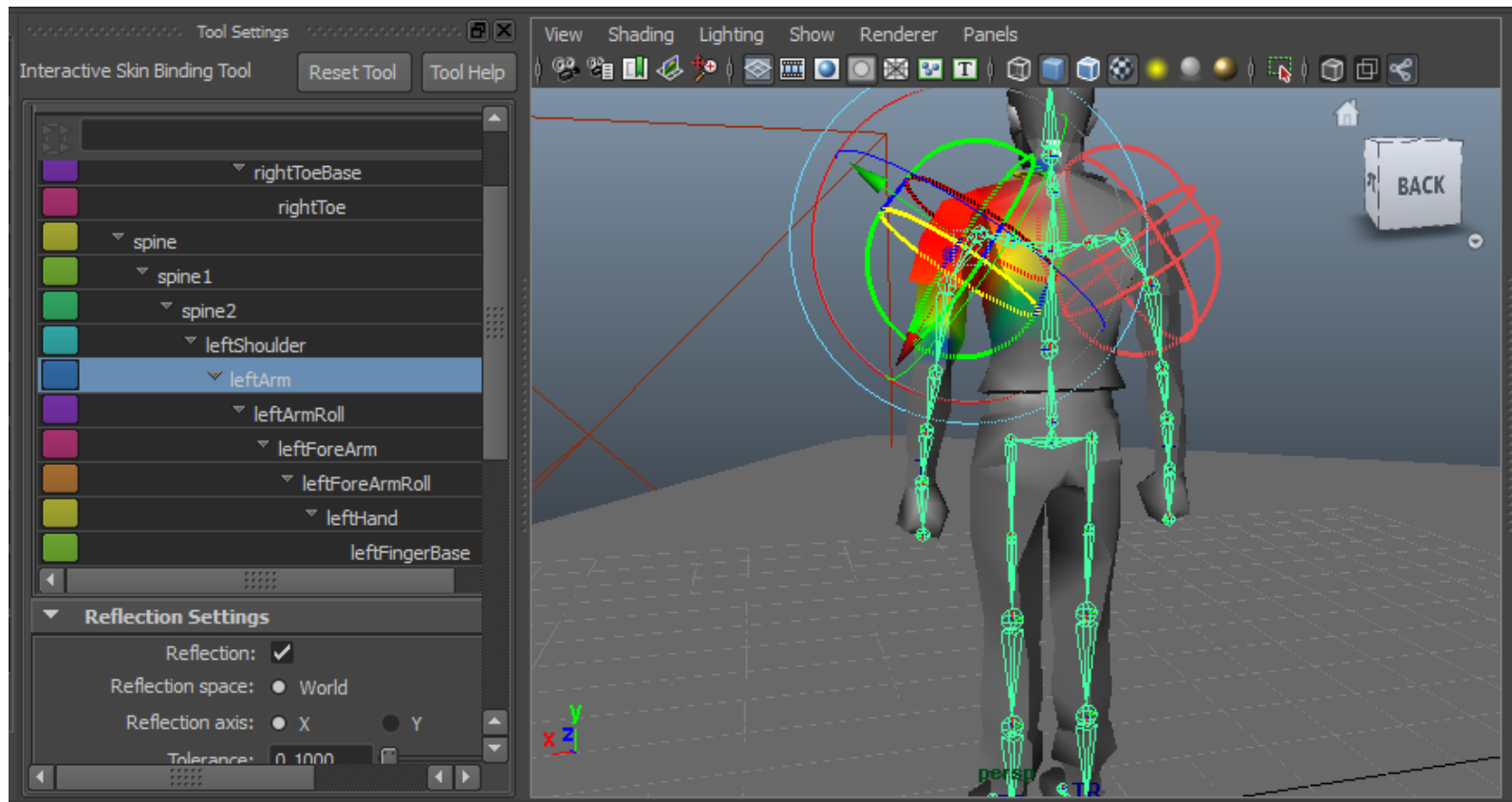
- ▶ Linearisierung: wäre F eine lineare Funktion könnte man die Zielkonfiguration berechnen/nähern, i.d.R. über-/unterbestimmtes System
- ▶ daher: kleine Schritte, **lineare Approximation der eigentlich gekrümmten Bewegung**



(Character-)Skinning

Grundidee

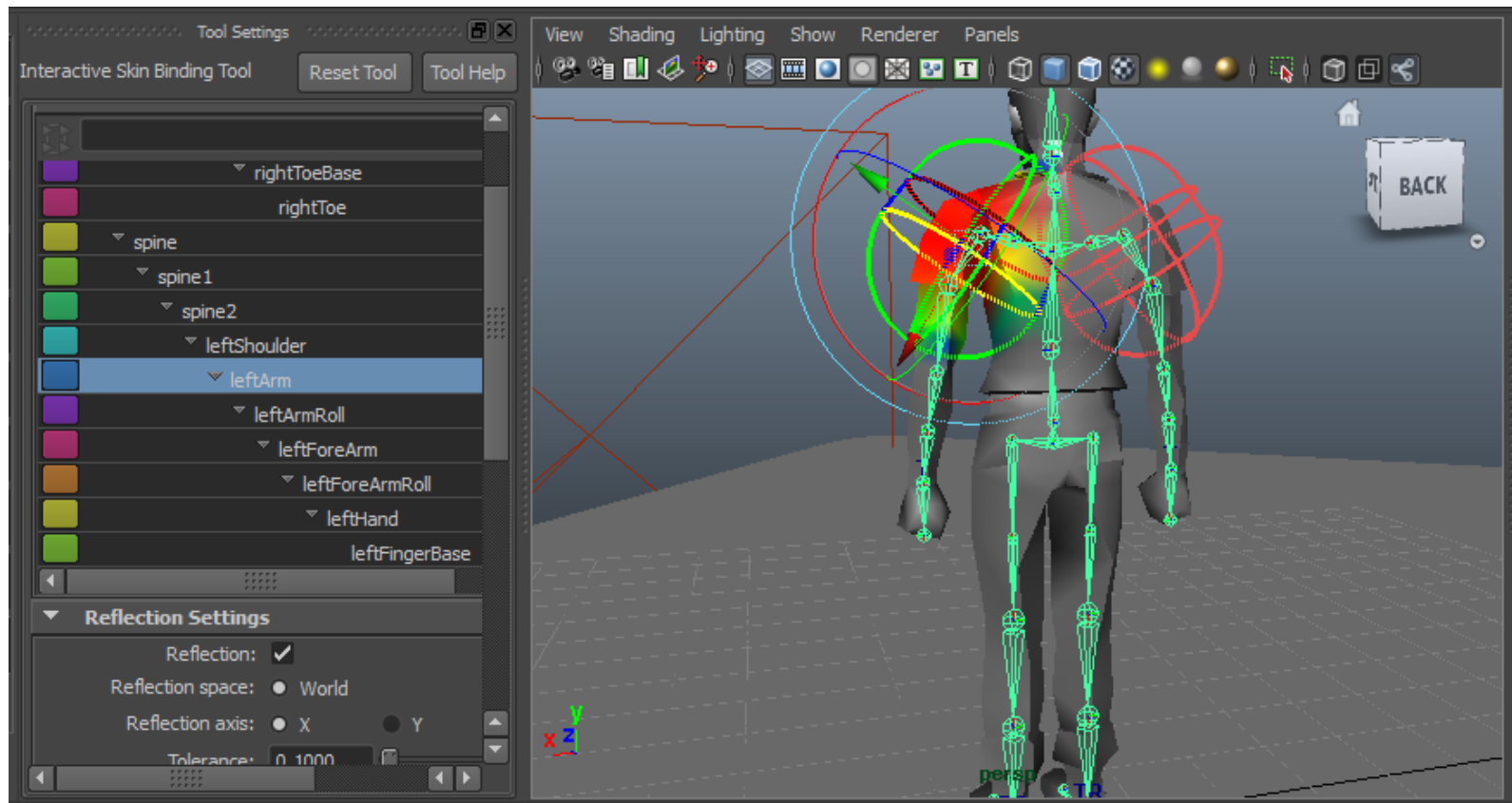
- ▶ Skelette repräsentieren einen hierarchischen Bewegungsapparat
- ▶ Animation: direkte Kinematik und Interpolation oder inverse Kinematik
- ▶ aber: wie überträgt man die Animation auf ein Dreiecksnetz?



(Character-)Skinning

Grundidee

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

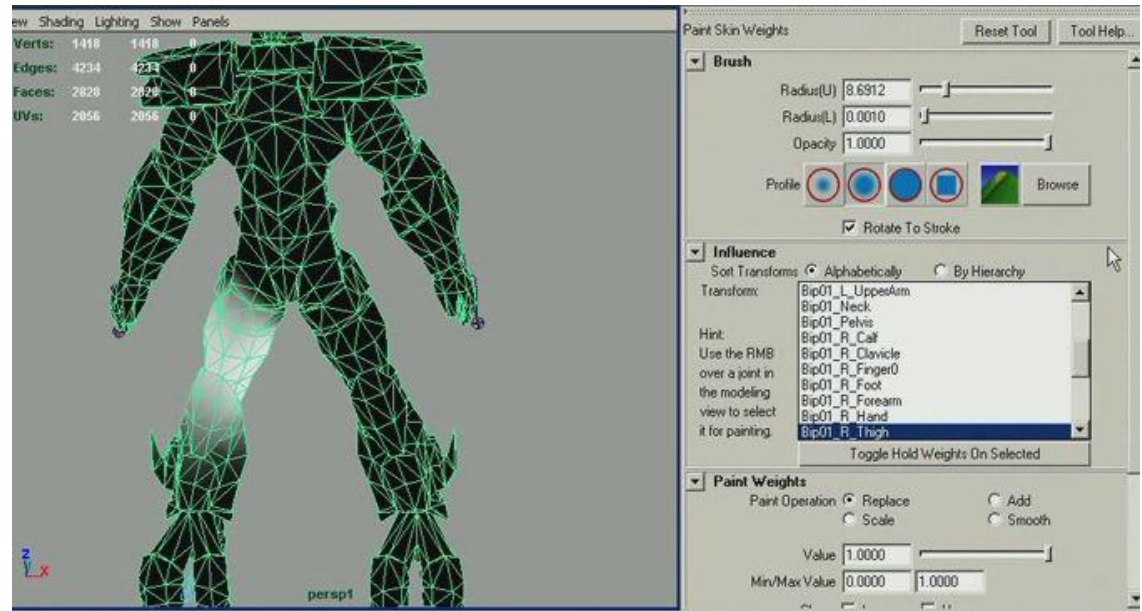
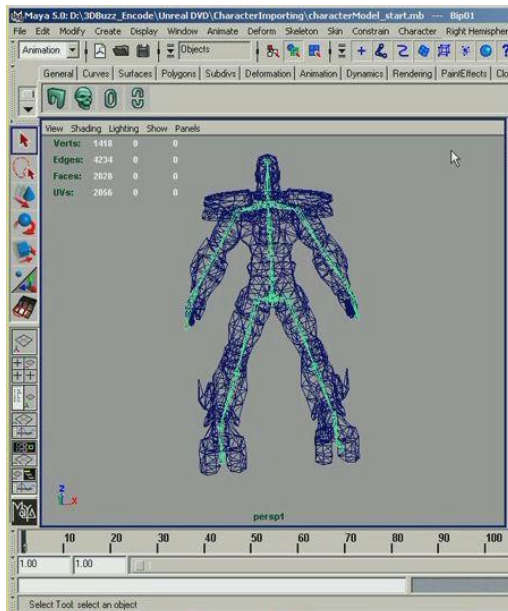


(Character-)Skinning



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 $x'_i = M_i x$
 - ▶ endgültige Vertexposition: Affinkombination $x' = \sum w_i \cdot x'_i$



(Character-)Skinning



Ausblicke / Aktuelle Forschungsthemen

▶ <http://www.youtube.com/watch?v=j6dwCtcy8DA>