

Computergraphik II

Winter 2012/2013

9 Computer Animation

Versionsdatum: 30. November 2012



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 9-1-

Computergraphik II

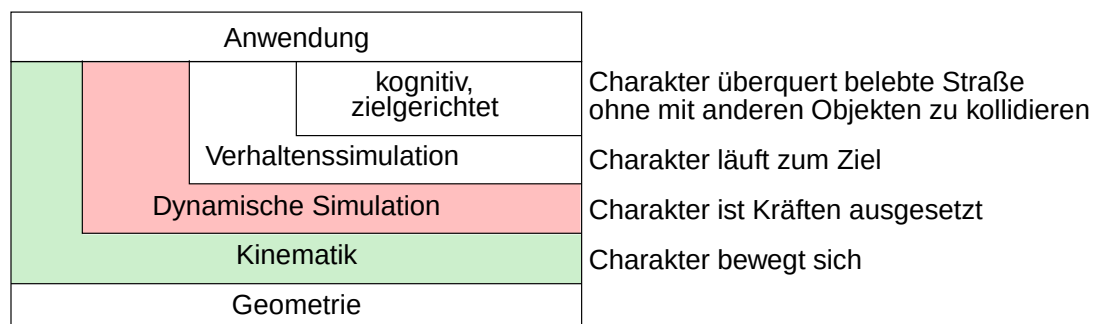
9 Computer Animation ...



Animation = „sich über die Zeit verändernde Objekte“, z.B.

1. globale Position und Orientierung
2. Bewegung/Veränderung des Objektes in sich bzw. in Relation zu anderen Objekten

Hierarchie der Ansätze zur Bewegungskontrolle:



Kinematik: ○ Bewegungslehre (Physik); Bewegung = direkte Geometrie-Modifikation

- Veränderbare Zeitschritte ⇒ Geometrie muss für jeden Zeitpunkt bestimmbar sein



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 9-2-

Computergraphik II

9.1 Keyframe Animation

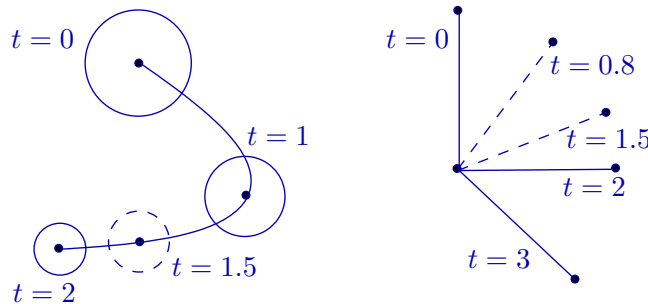


Bezug zur Trickfilm-Animation: Interpolation vorgegebener Schlüsselbilder:

	Zeichentrick-Animation	Computeranimation
Schlüsselbild	Künstler/Choreograph	Animator
In-Betweens	Zeichner	Algorithmus

- Grundsätzliches Vorgehen:**
1. Beschreibung der Szene $\mathcal{O}$ durch zeitabhängige Parameter (Animationsparameter) $\phi_1, \phi_2, \dots, \phi_N$
 2. Zwischenzeitpunkt t : Interpolation der Parameter $\mathcal{O}(\phi_1(t), \dots, \phi_N(t))$

Animationsparameter: Unterscheide *Position* und *Orientierung*



zeitabhängig: Zentrum $\mathbf{Z}$, Radius r

zeitabhängig: Richtung $\vec{r}$



9.1.1 Interpolation der Position



Aufgabe:

Interpolation von Positions-Parametern wird auf Basis geeigneter Kurven-Klassen durchgeführt

Konkrete Aufgabe:

- Gegeben: Zeit-Positions-Paare $(t_i, \mathbf{P}_i)$, $i = 1, \dots, N$
Gesucht: Kurve $\mathbf{C} : \mathbf{C}(t_i) = \mathbf{P}_i$, $i = 1, \dots, N$

Interpolations-Kurven

B-Splines (siehe Abschnitt „Interpolation mit B-Splines“)

- stückweise (kubische) Polynomkurven
- C^2 -stetige Kurven

Catmull-Rom Splines (siehe auch Hermite-Polynome)

- C^1 -stetige kubische Polynomkurven

heuristische Bestimmung der Tangente (ggf. nachträglich Anpassung)



9.1.2 Beschreibung der Orientierung



Orientierung: Entsteht durch Rotation bzgl. einer Ausgangslage

Interpolation von Orientierung = Interpolation von Rotation

Beschreibung von Rotation: Rotation um Winkel ϕ um x -Achse in homogenen Koordinaten:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi & 0 \\ 0 & \sin \phi & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Versuch: Matrix-Interpolation zur Interpolation zweier Orientierungen:

Gegeben: $R_x(-90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $R_x(90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Gesucht: $\frac{1}{2} (R_x(-90^\circ) + R_x(90^\circ)) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \neq R_x(0^\circ) = Id$



9.1.2 Beschreibung der Orientierung ...



Euler-Winkel

Ansatz: Sequentielle Drehung um x -, y - & z -Achse in **lokalen Koordinaten**
Pitch, Yaw, Roll Rotation $R_x(\phi_x), R_y(\phi_y), R_z(\phi_z)$ um x -, y - bzw. z -Achse

Reihenfolge: Beeinflusst Ergebnis $\Rightarrow$ feste Reihenfolge, z.B. x, y, z .

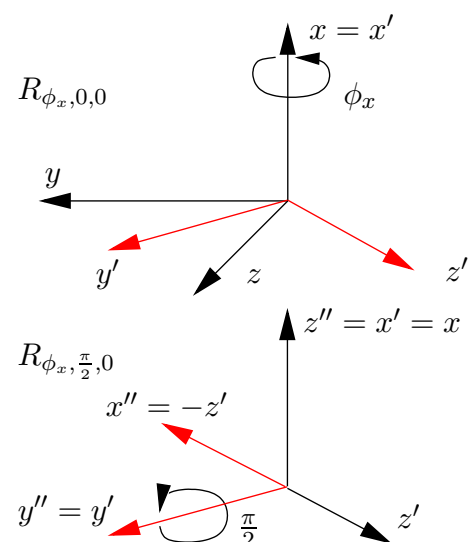
Gesamtrotation für Abarbeitungsreihenfolge x, y, z :

$$R_{\phi_x, \phi_y, \phi_z} := R_x(\phi_x) \circ R_y(\phi_y) \circ R_z(\phi_z)$$

Reihenfolge entspricht Top-Down Ansatz bei Graphik-Programmierung.

Problem: Mehrdeutigkeit (*Gimbal-Lock*)

$$R_{\phi_x, \frac{\pi}{2}, \phi_z} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \sin(\phi_x + \phi_z) & \cos(\phi_x + \phi_z) & 0 & 0 \\ -\cos(\phi_x + \phi_z) & \sin(\phi_x + \phi_z) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



9.1.2 Beschreibung der Orientierung ...



Mehrdeutigkeit der Euler-Winkel & Animation

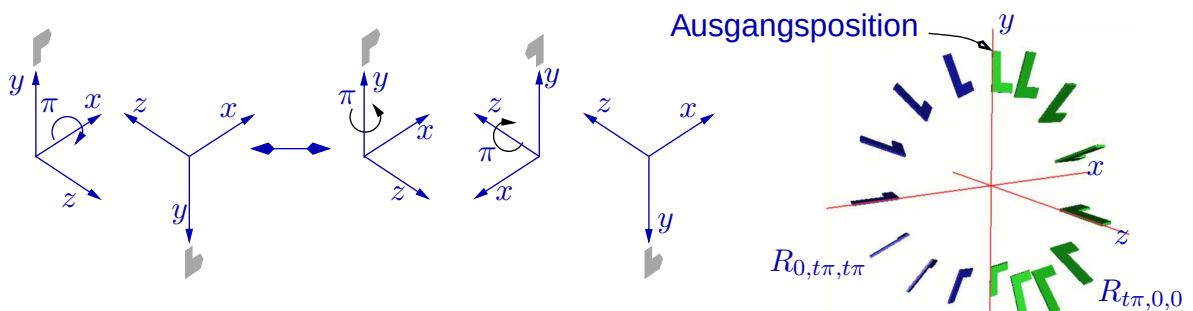
Beachte: Für Animation muss Repräsentation der Orientierung eindeutig sein

Denn: Sonst entstehen „beliebige“ Interpolationsergebnisse

Beispiel: ○ $R_{\pi,0,0}$ und $R_{0,\pi,\pi}$ beschreiben dieselbe Orientierung

- Interaktionspfade zwischen $R_{0,0,0}$ und $R_{\pi,0,0} = R_{0,\pi,\pi}$ sind unterschiedlich

$$R_{t\pi,0,0} \neq R_{0,t\pi,t\pi}, t \in]0, 1[$$



Erkenntnis: Euler-Winkel ungeeignete Repräsentation für Animation



9.1.3 Exkurs: Komplexe Zahlen



Komplexe Zahlen

Allgemein: Komplexe Zahlen erweitern die reellen Zahlen derart, dass Wurzeln negativer Zahlen definiert sind.

Imaginäre Einheit: $i := \sqrt{-1}$ als Lösung von $x^2 = -1$

Darstellung komplexer Zahlen: $c = x + i \cdot y$, mit $x, y \in \mathbb{R}$

Realteil von c : $\Re(c) = x$, Imaginärteil von c : $\Im(c) = y$

Mengensymbol: Die Menge der komplexen Zahlen wird mit $\mathbb{C}$ bezeichnet

Rechenregeln gelten wie gewohnt unter Beachtung von $i^2 = -1$

Addition: $c_1 + c_2 = x_1 + i \cdot y_1 + x_2 + i \cdot y_2 = (x_1 + x_2) + i \cdot (y_1 + y_2)$

$\Rightarrow \Re(c_1 + c_2) = x_1 + x_2$; $\Im(c_1 + c_2) = y_1 + y_2$

Multiplikation:

$$c_1 \cdot c_2 = (x_1 + i \cdot y_1) \cdot (x_2 + i \cdot y_2) = x_1 \cdot x_2 + i \cdot x_1 \cdot y_2 + i \cdot y_1 \cdot x_2 + i^2 \cdot y_1 \cdot y_2$$

$$\Rightarrow c_1 \cdot c_2 = (x_1 \cdot x_2 - y_1 \cdot y_2) + i \cdot (x_1 \cdot y_2 + x_2 \cdot y_1) \quad (\text{da } i^2 = -1)$$

$$\Rightarrow \Re(c_1 \cdot c_2) = x_1 \cdot x_2 - y_1 \cdot y_2$$
; $\Im(c_1 \cdot c_2) = x_1 \cdot y_2 + x_2 \cdot y_1$



Addition und Multiplikation komplexer Zahlen sind kommutativ!

9.1.3 Exkurs: Komplexe Zahlen ...



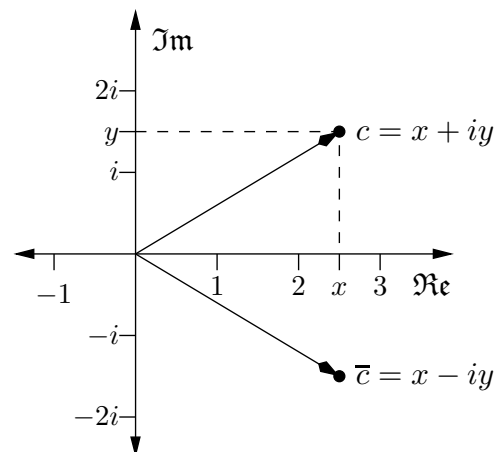
Eigenschaften komplexer Zahlen

Gauß'sche Zahlenebene: Man kann sich den komplexen Zahlenbereich als Ebene vorstellen:

- „Basisvektoren“ sind 1 und i

$$\mathbb{C} \ni x + i \cdot y \leftrightarrow \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$$

Länge: Analog zur Vektorlänge, $|c|^2 = x^2 + y^2$



Konjugation: „Spiegeln“ des Vektors an der Realteilachse

$$c = x + i \cdot y \Rightarrow \bar{c} := x - i \cdot y$$

$$\text{Damit ergibt sich: } c \cdot \bar{c} = (x + iy) \cdot (x - iy) = x^2 - (iy)^2 = x^2 + y^2 = |c|^2$$



9.1.3 Exkurs: Komplexe Zahlen ...



Polarkoordinaten

Jedes $c = x + iy \in \mathbb{C}$ lässt sich schreiben als:

$$c = r \cdot \cos \alpha + r \cdot i \sin \alpha, \text{ mit } r = |c|, \alpha \in [0, 2\pi[$$

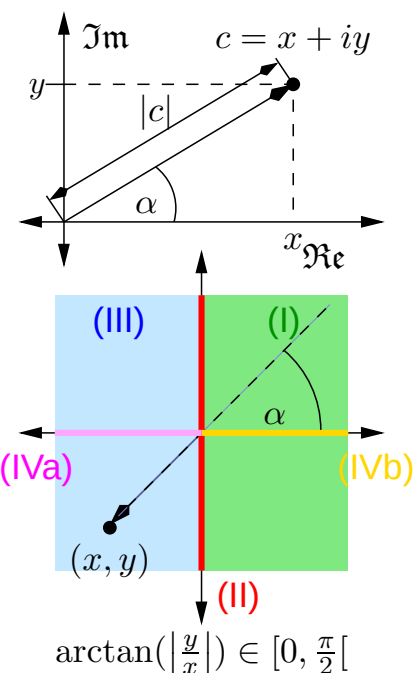
Aufgrund von $\cos^2 \alpha + \sin^2 \alpha = 1$ (Pythagoras) ist

$$\begin{aligned} |c|^2 &= r^2 \cos^2 \alpha + r^2 \sin^2 \alpha \\ &= r^2 (\cos^2 \alpha + \sin^2 \alpha) = r^2 \end{aligned}$$

Umrechnung: Für $c = x + iy$ ist $r = \sqrt{x^2 + y^2}$ und

$$\alpha = \arctan2(y, x)$$

$$:= \begin{cases} \operatorname{sgn}(y) \cdot \arctan\left(\left|\frac{y}{x}\right|\right) & x > 0, y \neq 0 \text{ (I)} \\ \operatorname{sgn}(y) \cdot \frac{\pi}{2} & x = 0, y \neq 0 \text{ (II)} \\ \operatorname{sgn}(y) \cdot (\pi - \arctan\left(\left|\frac{y}{x}\right|\right)) & x < 0, y \neq 0 \text{ (III)} \\ \pi & x < 0, y = 0 \text{ (IVa)} \\ 0 & x > 0, y = 0 \text{ (IVb)} \end{cases}$$



9.1.3 Exkurs: Komplexe Zahlen ...



Rotation mit komplexen Zahlen

Betrachte folgende Untermenge: $\mathcal{Z} = \{c : |c| = 1\}$

- Es gilt: $1 = |c| = x^2 + y^2 \Rightarrow \exists_1 \alpha \in [0, 2\pi[: \cos \alpha = x \wedge \sin \alpha = y$ (S.O.)
- Für $c_1, c_2 \in \mathcal{Z}$ mit $c_i = \cos \alpha_i + i \cdot \sin \alpha_i$ ist (Additionstheoreme)

$$\begin{aligned} c_1 \cdot c_2 &= \underbrace{(\cos \alpha_1 \cdot \cos \alpha_2 - \sin \alpha_1 \cdot \sin \alpha_2)}_{=\cos(\alpha_1+\alpha_2)} + i \underbrace{(\cos \alpha_1 \cdot \sin \alpha_2 + \sin \alpha_1 \cdot \cos \alpha_2)}_{=\sin(\alpha_1+\alpha_2)} \\ &= \cos(\alpha_1 + \alpha_2) + i \cdot \sin(\alpha_1 + \alpha_2) \in \mathcal{Z} \end{aligned}$$

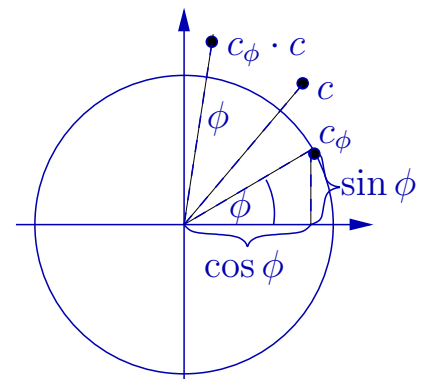
Rotation um ϕ mittels $c_\phi = \cos \phi + i \cdot \sin \phi \in \mathcal{Z}$:

$$R_\phi : \begin{array}{ccc} \mathbb{C} & \longrightarrow & \mathbb{C} \\ c & \mapsto & R_\phi(c) = c_\phi \cdot c \end{array}$$

denn $c_\phi \cdot c = |c| (\cos(\alpha + \phi) + i \cdot \sin(\alpha + \phi))$

Exponential-Schreibweise von Polar-Koord.:

$$c = x + i \cdot y = |c| (\cos \alpha + i \sin \alpha) = |c| e^{i\alpha}$$



9.1.3 Exkurs: Komplexe Zahlen ...



Rotation mit komplexen Zahlen (Forts.)

Gegeben: Punkt $P = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$ in der Ebene und Drehwinkel ϕ

Durchführung der Rotation

1. Interpretiere P als komplexe Zahl in $\mathbb{C}$: $z = x + iy$
2. Rotiere z in $\mathbb{C}$: $z' = z \cdot e^{i\phi}$
3. Interpretiere z' wieder als Punkt in $\mathbb{R}^2$ ab: $\begin{pmatrix} x' \\ y' \end{pmatrix} \in \mathbb{R}^2$

Klassisches Vorgehen mit Matrizen in 2D: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

Rotation mit komplexen Zahlen in 2D bietet hierzu keine Vorteile

...aber: Sir William Hamilton (1805-1865) hat folgendes herausgefunden:

- eine Erweiterung komplexer Zahlen auf 3D ist nicht möglich
- eine Verallgemeinerung für $2n$ -dimensionale Räume geht jedoch



Quaternionen= verallgemeinerte komplexe Zahlen für 4D-Rotationen

Bemerkung: Exponential-Schreibweise für Polarkoordinaten

Funktionen wie $\sin$, $\cos$, $\exp$ sind über **unendliche Reihen** definiert:

$$\cos x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}, \quad \sin x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}, \quad e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Damit können in diese Funktionen problemlos komplexe Parameter eingesetzt werden.

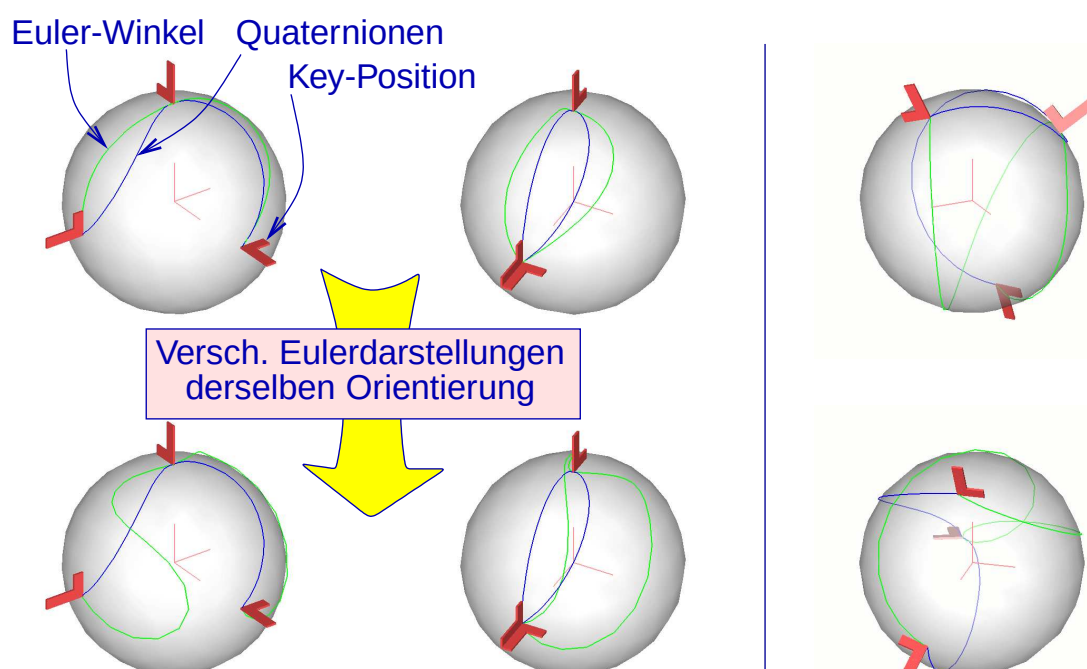
Einsetzen von $i\phi$ als Parameter in die Exponentialfunktion liefert:

$$\begin{aligned} e^{i\phi} &= \sum_{k=0}^{\infty} \frac{(i\phi)^n}{n!} = \sum_{k=0}^{\infty} \frac{(i\phi)^{2k}}{(2k)!} (\text{gerade Expon.}) + \sum_{k=0}^{\infty} \frac{(i\phi)^{2k+1}}{(2k+1)!} (\text{ungerade Expon.}) \\ &= \sum_{k=0}^{\infty} \frac{(i^2)^k \cdot \phi^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{i(i^2)^k \cdot \phi^{2k+1}}{(2k+1)!} = \sum_{k=0}^{\infty} \frac{(-1)^k \phi^{2k}}{(2k)!} + i \sum_{k=0}^{\infty} \frac{(-1)^k \phi^{2k+1}}{(2k+1)!} \\ &= \cos \phi + i \sin \phi \end{aligned}$$



9.1.4 Quaternionen

Euler-Winkel vs. Quaternionen (Demo-Programm)



9.1.4 Quaternionen ...



Definition: Quaternionen

Quaternionen hat eine reale und drei imaginäre Komponenten s bzw. x, y, z .

Schreibweise: Mit imaginäre Einheiten i, j, k :

$$\underline{q} := s + ix + jy + kz = (s, \vec{v}), \quad \vec{v} = (x, y, z) \quad \text{Zahlenmenge: } \mathbb{H}$$

mit $i^2 = j^2 = k^2 = -1$, $ij = k$, $jk = i$, $ki = j$ und $ji = -k$, $kj = -i$, $ik = -j$

Addition: $\underline{q}_1 + \underline{q}_2 = (s_1 + s_2, \vec{v}_1 + \vec{v}_2)$

Multiplikation: Durch Anwendung obiger Regeln folgt:

$$\begin{aligned} \underline{q}_1 \underline{q}_2 &= (s_1 + ix_1 + jy_1 + kz_1)(s_2 + ix_2 + jy_2 + kz_2) \\ &= s_1 s_2 - (x_1 x_2 + y_1 y_2 + z_1 z_2) + i(s_1 x_2 + s_2 x_1 + y_1 z_2 - y_2 z_1) \\ &\quad + j(s_1 y_2 + s_2 y_1 + z_1 x_2 - z_2 x_1) + k(s_1 z_2 + s_2 z_1 + x_1 y_2 - x_2 y_1) \\ &= (s_1 s_2 - (\vec{v}_1 \cdot \vec{v}_2), s_1 \vec{v}_2 + s_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2) \end{aligned}$$

Unsymmetrie $ij = -ji, jk = -kj, ki = -ik \Rightarrow$ Multiplik. ist **nicht kommutativ**.



9.1.4 Quaternionen ...



Eigenschaften von Quaternionen

Konjugierte:

$$\underline{\bar{q}} = (s, -\vec{v}), \quad \underline{\bar{q}}_1 \underline{\bar{q}}_2 = (s_1 s_2 - (\vec{v}_1 \cdot \vec{v}_2), -s_1 \vec{v}_2 - s_2 \vec{v}_1 + \underbrace{\vec{v}_1 \times \vec{v}_2}_{= -\vec{v}_2 \times \vec{v}_1}) = \overline{(\underline{q}_2 \underline{q}_1)}$$

Länge:

$$\underline{q} \underline{\bar{q}} = (s^2 + (\vec{v} \cdot \vec{v}), \underbrace{s\vec{v} - s\vec{v} + \vec{v} \times (-\vec{v})}_{=\vec{0}}) \in \mathbb{R}, \quad |\underline{q}| = \sqrt{\underline{q} \underline{\bar{q}}} = \sqrt{s^2 + (\vec{v} \cdot \vec{v})}$$

Einheitsquaternionen: Quaternionen mit der Länge 1, d.h. $|\underline{q}| = 1$

Winkel: Für $|\underline{q}_1| = |\underline{q}_2| = 1$ gilt:

$$\cos(\angle(\underline{q}_1, \underline{q}_2)) = \Re(\underline{q}_1 \underline{\bar{q}}_2) = s_1 s_2 + (\vec{v}_1 \cdot \vec{v}_2)$$

Das entspricht dem inneren Produkt der zugehörigen 4D-Vektoren

(s_1, x_1, y_1, z_1) und (s_2, x_2, y_2, z_2)



9.1.4 Quaternionen ...



Rotation mit Quaternionen

Problem: Einheitsquaternionen beschreiben Rotationen in 4D

Fragen bei der Nutzung von Quaternionen in 3D

- Wie wird 3D in 4D eingebettet?
- Wie kann man 4D-Rotationen in 3D übertragen?

Folgendes Quaternion beschreibt 3D-Rotation (Achse $\hat{\mathbf{v}}$, Winkel ϕ):

$$\underline{\mathbf{q}} = (\cos \frac{\phi}{2}, \sin \frac{\phi}{2} \hat{\mathbf{v}}), \quad |\underline{\mathbf{q}}| = 1$$

Durchführung der Rotation: Gegeben sei Punkt $\mathbf{P} \in \mathbb{R}^3$

1. Übertragung von $\mathbf{P}$ in 4D: $\underline{\mathbf{q}}_{\mathbf{P}} = (0, \vec{\mathbf{p}})$, $\vec{\mathbf{p}}$ ist der Ortsvektor von $\mathbf{P}$
2. Durchführung der Rotation: $\underline{\mathbf{q}}'_{\mathbf{P}} = \underline{\mathbf{q}} \underline{\mathbf{q}}_{\mathbf{P}} \overline{\underline{\mathbf{q}}}$
3. Interpretation in 3D: $\underline{\mathbf{q}}'_{\mathbf{P}} = (0, R_{\hat{\mathbf{v}}, \phi}(\mathbf{P}))$, d.h. $s' = 0$, $R_{\hat{\mathbf{v}}, \phi}(\mathbf{P}) = (x', y', z')$

Hintereinanderausführung von Rotationen entspr. Quaternionen-Multiplik.:

$$R_{\underline{\mathbf{q}}_2} \left(R_{\underline{\mathbf{q}}_1}(\mathbf{P}) \right) \hat{=} \underline{\mathbf{q}}_2 (\underline{\mathbf{q}}_1 \underline{\mathbf{q}}_{\mathbf{P}} \overline{\underline{\mathbf{q}}_1}) \overline{\underline{\mathbf{q}}_2} = (\underline{\mathbf{q}}_2 \underline{\mathbf{q}}_1) \underline{\mathbf{q}}_{\mathbf{P}} (\overline{\underline{\mathbf{q}}_1} \overline{\underline{\mathbf{q}}_2}) = (\underline{\mathbf{q}}_2 \underline{\mathbf{q}}_1) \underline{\mathbf{q}}_{\mathbf{P}} \overline{(\underline{\mathbf{q}}_2 \underline{\mathbf{q}}_1)} \hat{=} R_{\underline{\mathbf{q}}_2 \underline{\mathbf{q}}_1}(\mathbf{P})$$



9.1.4 Quaternionen ...



Beispiel:

Rotation von $\mathbf{P} = (1, 2, 0)$ um $\hat{\mathbf{v}} = (1, 0, 0)$ mit Winkel π (Ergebnis: $(1, -2, 0)$).

$$\underline{\mathbf{q}} = (\cos(\pi/2), \sin(\pi/2)\hat{\mathbf{v}}) = (0, (1, 0, 0)); \quad \underline{\mathbf{q}}_{\mathbf{P}} = (0, (1, 2, 0)); \quad \overline{\underline{\mathbf{q}}} = (0, -(1, 0, 0))$$

Berechnung von $R_{\underline{\mathbf{q}}}(\mathbf{P})$ in zwei Schritten:

1. $\underline{\mathbf{q}}_{\mathbf{P}} \overline{\underline{\mathbf{q}}} = \left(- \left(\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \right), \left(\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \times \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \right) \right) = (1, (0, 0, 2))$
2. $\underline{\mathbf{q}}(\underline{\mathbf{q}}_{\mathbf{P}} \overline{\underline{\mathbf{q}}}) = \left(- \left(\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \right), \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \right) = (0, (1, -2, 0))$

Das Quaternion $(0, (1, -2, 0))$ entspricht dem Punkt $(1, -2, 0)$





Mehrdeutigkeit von Quaternionen

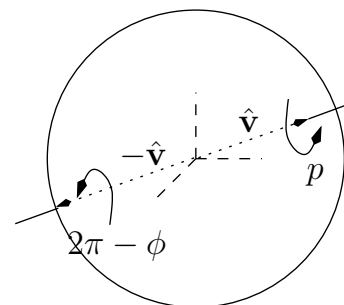
Beobachtung: $\underline{q}$ und $-\underline{q}$ liefert dasselbe Ergebnis:

1. Multiplikation bzgl. reeller Faktoren ist **kommutativ**, z.B.
 $a \cdot (s, \vec{v}) \cdot b = (ab s, ab \vec{v}) = ab(s, \vec{v})$
2. damit gilt: $R_{\underline{q}}(\mathbf{P}) \hat{=} \underline{q} \mathbf{q}_P \bar{\underline{q}} = (-1)^2 \underline{q} \mathbf{q}_P \bar{\underline{q}} = (-\underline{q}) \mathbf{q}_P (-\bar{\underline{q}}) \hat{=} R_{-\underline{q}}(\mathbf{P})$

Analyse: $-\underline{q} = (-\cos \frac{\phi}{2}, -\sin \frac{\phi}{2} \hat{v})$ entspricht Rotation um $-\hat{v}$ mit Winkel $2\pi - \phi$, denn

$$\cos(\pi - \frac{\phi}{2}) = -\cos(\frac{\phi}{2}) \text{ und } \sin(\pi - \frac{\phi}{2}) = \sin(\frac{\phi}{2})$$

$$\Rightarrow -\underline{q} = (\cos(\pi - \frac{\phi}{2}), \sin(\pi - \frac{\phi}{2})(-\hat{v}))$$



Mehrdeutigkeit: Beide Rotationen liefern dasselbe Ergebnis! Weitere Mehrdeutigkeiten gibt es nicht!



9.1.4 Quaternionen ...



Beispiel: Mehrdeutigkeit bei Euler-Winkeln

Erinnerung: Für Eulerwinkel ist $R_{\pi,0,0} = R_{0,\pi,\pi}$

Frage: Wie sieht das bei Quaternionen aus?

$$R_{\pi,0,0} \hat{=} R_{\underline{q}_x} \text{ mit } \underline{q}_x = (0, (1, 0, 0)), \text{ da } \sin \frac{\pi}{2} = 1, \cos \frac{\pi}{2} = 0$$

$$R_{0,\pi,0} \hat{=} R_{\underline{q}_y} \text{ mit } \underline{q}_y = (0, (0, 1, 0)), \quad R_{0,0,\pi} \hat{=} R_{\underline{q}_z} \text{ mit } \underline{q}_z = (0, (0, 0, 1))$$

$$R_{0,\pi,\pi} \hat{=} R_{\underline{q}_y} \circ R_{\underline{q}_z} = R_{\underline{q}_y \underline{q}_z} = R_{\underline{q}_x}$$

$$\text{da } (0, (0, 1, 0)) \cdot (0, (0, 0, 1)) = \left(0, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) = (0, (1, 0, 0))$$

Ergebnis: $R_{\pi,0,0}$ und $R_{0,\pi,\pi}$ entsprechen demselben Einheitsquaternion
 $\underline{q} = (0, (1, 0, 0))$





Umrechnung Quaternionen $\leftrightarrow$ Rotationsmatrizen

Umrechnung des Einheitsquaternion $\underline{q} = (\cos \frac{\phi}{2}, \sin \frac{\phi}{2} \hat{v}) = (s, (x, y, z))$ in Rotationsmatrix R :

$$R_{\underline{q}} = \begin{pmatrix} 1 - 2(y^2 + z^2) & 2xy - 2sz & 2xz + 2sy \\ 2xy + 2sz & 1 - 2(x^2 + z^2) & 2yz - 2sx \\ 2xz - 2sy & 2yz + 2sx & 1 - 2(x^2 + y^2) \end{pmatrix} \quad (1)$$

Umrechnung der Rotationsmatrix $R = (r_{ij})_{i,j=1,\dots,3}$ in Einheitsquaternion. Aus Gleichung (1)

$$\begin{aligned} r_{11} + r_{22} + r_{33} &= 3 - 4(x^2 + y^2 + z^2) = 3 - 4(1 - s^2) = -1 - 4s^2 \\ \Rightarrow s &= \pm \frac{1}{2} \sqrt{r_{11} + r_{22} + r_{33} + 1} \\ r_{32} - r_{23} &= 4sx \Rightarrow x = \frac{r_{32} - r_{23}}{4s}, \quad \text{analog: } y = \frac{r_{13} - r_{31}}{4s}, \quad z = \frac{r_{21} - r_{12}}{4s} \end{aligned}$$

Beachte: Positives bzw. negatives s liefern zwei Quaternionen $\underline{q}^+, \underline{q}^-$, aber $\underline{q}^- = -\underline{q}^+$ beschreiben dieselbe Orientierung!



9.1.5 Interpolation der Orientierung



Ziel: Interpolation von Orientierungen, in Euler-Notation:

Gegeben: Zeit-Orientierungs-Paare $(t_i, R_{\vec{\phi}^i})$, $\vec{\phi}^i = (\phi_x^i, \phi_y^i, \phi_z^i)$, $i = 1, \dots, N$
Gesucht: Funktion R mit $R(t_i) = R_{\vec{\phi}^i}$, $i = 1, \dots, N$

Interpolation mit Euler-Winkeln: ○ Direkte Interpolation z.B. mit B-Splines

○ Problem: Ergebnis von der Repräsentation der Orientierung abhängig

Interpolation mit Quaternionen:

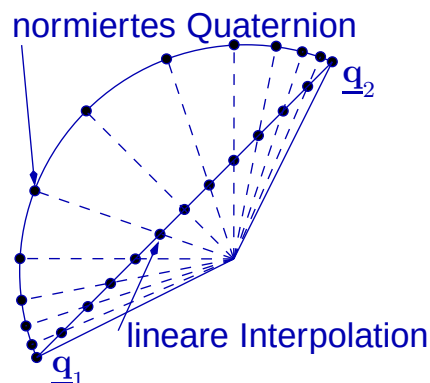
Erinnerung: Rotationen werden durch 4D-Einheitsquaternionen beschrieben

Problem: Interpolierte Quaternionen müssen Einheitsquaternionen sein.

Bézier, B-Spline leisten das nicht:

$$\mathbf{C}(i) \in \mathbb{R}^4, \|\mathbf{C}(i)\| = 1 \not\Rightarrow \|\mathbf{C}(u)\| = 1.$$

Normierung liefert verzerrte Schrittweiten



9.1.5 Interpolation der Orientierung ...



Sphärische lineare Interpolation (slerp)

Ziel: „Lineare“ Interpolation auf einer Kugeloberfläche bzw. Großkreisbogen

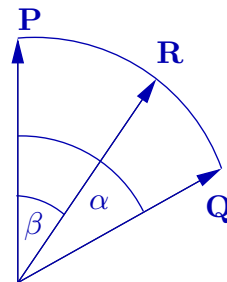
Gegeben: P, Q, R auf Einheitskreis (bzw. auf Großkreis), R „zwischen“ P und Q : $\alpha = \angle(P, Q), \beta = \angle(P, R)$.

Lösung: Zusammenhang zwischen Winkel und Punkt:

$$R = \frac{\sin(\alpha - \beta)}{\sin \alpha} P + \frac{\sin(\beta)}{\sin \alpha} Q$$

Damit gilt für $\beta = t\alpha, t \in [0, 1]$:

$$R(t) = \frac{\sin((1-t)\alpha)}{\sin \alpha} P + \frac{\sin(t\alpha)}{\sin \alpha} Q$$



Linearen Interpolation von Rotationen durch sphärische Interpolation von Einheitsquaternionen:

$$\underline{q}(t) = \frac{\sin((1-t)\alpha)}{\sin \alpha} \underline{q}_1 + \frac{\sin(t\alpha)}{\sin \alpha} \underline{q}_2, \quad \cos \alpha = s_1 s_2 + (\vec{v}_1 \cdot \vec{v}_2)$$



9.1.5 Interpolation der Orientierung ...



Bemerkung: Details zur Interpolation auf Kreisbögen

Gegeben: P, Q, R Punkte auf dem Einheitskreis, R „zwischen“ P und Q , $\alpha = \angle(P, Q)$

o.B.d.A.: Punkte liegen in der Ebene und $P = (1, 0)$

$\Rightarrow Q = (\cos \alpha, \sin \alpha)$ und $R = (\cos \beta, \sin \beta)$

Gesucht: Linearkombination $R = aP + bQ$

Prüfe, ob $a = \frac{\sin((1-t)\alpha)}{\sin \alpha}$ und $b = \frac{\sin(t\alpha)}{\sin \alpha}$ „funktionieren“

$$\begin{aligned} aP + bQ &= \frac{\sin(\alpha - \beta)}{\sin \alpha} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{\sin \beta}{\sin \alpha} \cdot \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix} \\ &= \begin{pmatrix} \frac{\sin \alpha \cos \beta - \cos \alpha \sin \beta}{\sin \alpha} + \frac{\sin \beta \cdot \cos \alpha}{\sin \alpha} \\ 0 + \frac{\sin \beta \cdot \sin \alpha}{\sin \alpha} \end{pmatrix} = \begin{pmatrix} \cos \beta \\ \sin \beta \end{pmatrix} = R \end{aligned}$$



9.1.5 Interpolation der Orientierung ...



Sphärische Spline Interpolation

Auswertung von Bézier- und B-Spline Kurven basiert auf sukzessiver linearer Interpolation aus Kontrollpunkten (de Casteljau, de Boor)

Frage: Gilt das auch für *Catmull-Rom-Splines* mit Interpolationspunkten P_i ?

Also: Entstehen Bézier-Kontrollpunkte durch lineare Interpolation aus P_i ?

Konkret: i -tes Bézier-Segment mit KPs $C_0^i, \dots, C_3^i$ und Interpolationspunkte $P_{i-1}, \dots, P_{i+2}$:

$$\begin{aligned} C_0^i &= P_i, & C_3^i &= P_{i+1} \text{ (Endpunkte)}, & \vec{t}_i &= \frac{1}{2} (P_{i+1} - P_{i-1}) \text{ (Tangenten)} \\ C_1^i &= P_i + \frac{1}{3} \vec{t}_i = P_i + \frac{1}{6} (P_{i+1} - P_{i-1}) = \frac{1}{6} [2 (\frac{1}{2} P_i + \frac{1}{2} P_{i+1}) - P_{i-1}] + \frac{5}{6} P_i \\ C_2^i &= P_{i+1} - \frac{1}{3} \vec{t}_{i+1} = P_{i+1} - \frac{1}{6} (P_{i+2} - P_i) \\ &= \frac{1}{6} [2 (\frac{1}{2} P_{i+1} + \frac{1}{2} P_i) - P_{i+2}] + \frac{5}{6} P_{i+1} \end{aligned}$$

Sphärische Spline Interpolation:

Verwende slerp-Operation anstelle linearer Interpolation!



9.1.5 Interpolation der Orientierung ...



Sphärische Spline Interpolation (Forts.)

Gegeben: Zu interpolierende Quaternionen-Sequenz $\underline{q}_1, \dots, \underline{q}_n$

Frage: Welcher Repräsentant von $R_{\underline{q}_i}$ ist zu wählen, $\underline{q}_i$ oder $-\underline{q}_i$?

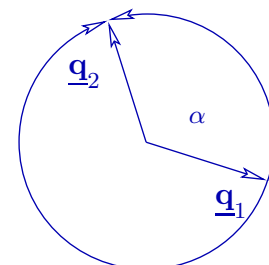
Analogie zum Einheitskreis: Der kleineren Winkel repräsentiert die „kurze“ Strecke.

Regel:

$$\angle(\underline{q}_i, \underline{q}_{i+1}) > \frac{\pi}{2} \Leftrightarrow \Re(\underline{q}_i \overline{\underline{q}_{i+1}}) < 0 \Rightarrow \underline{q}_{i+1} \leftarrow -\underline{q}_{i+1}$$

in sequentieller Reihenfolge $i = 1, i = 2, \dots, i = n - 1$

Beachte: $\underline{q}_i$ wird **zweimal** mit **halbem Winkel** angewendet, daher $> \frac{\pi}{2} (= 90^\circ)$



9.2 Spline-basierte Animation



Idee: Spline-basierte Animation

Keyframe-Animation: Vorgabe von Parameter zu Key-Zeitpunkten;
Berechnung interpolierender Spline-Kurven

Spline-basiert: Direkte Vorgabe der Spline-Kurven

Fragen: 1. Welche Animationsparameter sind für Spline-basiert geeignet?

- + : Bewegung eines Körpers durch den Raum (intuitive Bewegungsbahn)
- : Fingerspitze eines schlenkernden Arms eines gehenden Charakters

2. Wie kommt der Zeitbezug ins Spiel (keine Keyframes mehr!)?

Wichtig: Gleichförmige Abtastung des Parameters u erzeugt keine gleichförmige Bewegung entlang der Kurve!

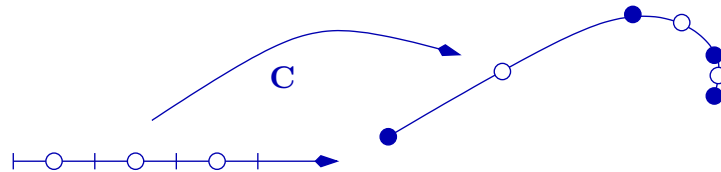
⇒ Ziel: Positionierung entlang der Kurve anhand der **Weglänge**



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 9-27-

Computergraphik II



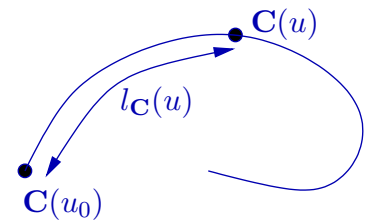
9.2.1 Pfad und Geschwindigkeit



Ziel: Bewegung eines Körpers entlang einer Kurve

Bewegungspfad des Objekt im Raum

$$\begin{aligned} C(u) : \mathbb{R} &\longrightarrow \mathbb{R}^3 \\ u &\mapsto (x(u), y(u), z(u)) \end{aligned}$$



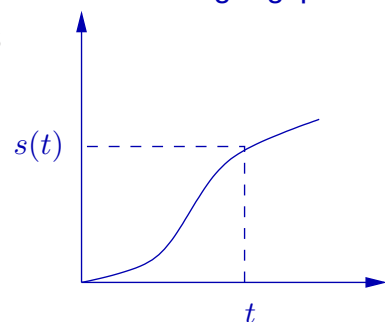
Wegfunktion: Zurückgelegter Weg auf dem Pfad

$$\begin{aligned} \text{zum Zeitpunkt } t : \mathbb{R} &\longrightarrow \mathbb{R} \\ s(t) : t &\mapsto s(t) \end{aligned}$$



Ziel: Bewegung des Objektes entlang C , so dass zum Zeitpunkt t der Weg $s(t)$ zurückgelegt ist.

Bogenlängenfunktion $l_C(u)$ misst Länge des Kurvenstückes für Pfad C vom Startpunkt $C(u_0)$ bis $C(u)$



Unterscheide: Modellierungsparameter u (Bewegungspfad), Zeit-Parameter t (Wegfunktion)

Wegfunktion



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 9-28-

Computergraphik II

9.2.1 Pfad und Geschwindigkeit ...



Ermittlung der Bogenlänge

Gegeben: Bewegungspfad $C(u)$ mit Start- und Zielparameter u_0 bzw. u

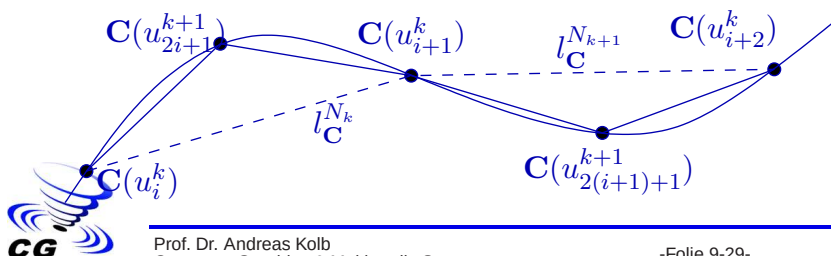
Abschätzung der Bogenlänge $l_C(u)$ durch Länge eines Kantenzuges:

$$l_C(u) \approx l_C^N(u) = \sum_{i=0}^{N-1} \|C(u_{i+1}) - C(u_i)\|, \text{ wobei } u_i = u_0 + i\Delta_N, \Delta_N = \frac{u - u_0}{N}$$

Exakte Lösung durch Grenzbetrachtung:

$$l_C^N(u) = \sum_{i=0}^{N-1} \|C(u_{i+1}) - C(u_i)\| = \sum_{i=0}^{N-1} \left\| \frac{C(u_{i+1}) - C(u_i)}{\Delta_N} \right\| \Delta_N \xrightarrow{N \rightarrow \infty} \int_{u_0}^u \|C'(\tau)\| d\tau$$

Implementation: I.a. Integral nicht lösbar $\Rightarrow$ Abschätzung mit $N_k = 2^k$



Abbruch, falls
wenig Genau-
igkeitszuwachs:
 $l_C^{N_{k+1}} - l_C^{N_k} < \varepsilon$



9.2.1 Pfad und Geschwindigkeit ...



Ermittlung des Parameters u (feste Pfad-Kurve)

Beobachtung: Ermittlung von l_C wird durch Auswertung von C dominiert

Ansatz für feste Pfad-Kurve: Speicherung von $l_C(u)$ in **Look-Up Tabelle**

1. Speicherung von $l_C(u_i)$ für äquidistante Parameter u_i
2. Speicherung von u_i^* für äquidistante Bogenlängen
 $\Rightarrow$ effizienterer Zugriff auf gesuchte u -Parameter

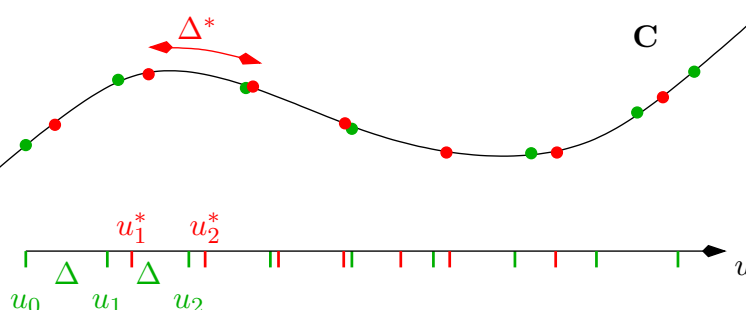
Hinweis: Berechnung von u_i^* durch Bisektion mittels der u_i, l_i Werte

Variante 1:

u_0	0
u_1	l_1
u_2	$l_1 + l_2$

$$u_i = u_0 + i \cdot \Delta$$

$$l_i = \|C(u_i) - C(u_{i-1})\|$$



Variante 2:

0	u_0^*
l_1^*	u_1^*
l_2^*	u_2^*

$$l_{i+1}^* - l_i^* = \Delta^*$$

$$l_i^* = l_C(u_i^*)$$



9.2.1 Pfad und Geschwindigkeit ...



Ermittlung des Parameters u (allgemeiner Fall)

Aufgabe: Zu gegebener Wegfunktion s und Zeitpunkt t : Für welchen Parameterwert u gilt $l_C(u) = s(t)$

Ansatz: Suche Nullstelle von $g(u) = l_C(u) - s(t)$

Beachte: $l_C(u)$ ist monoton wachsend $\Rightarrow$ eindeutige Nullstelle von $g(u)$.

Newton-Verfahren ermittelt Nullstelle von $g(u)$:

Initial: Startparameter u_0

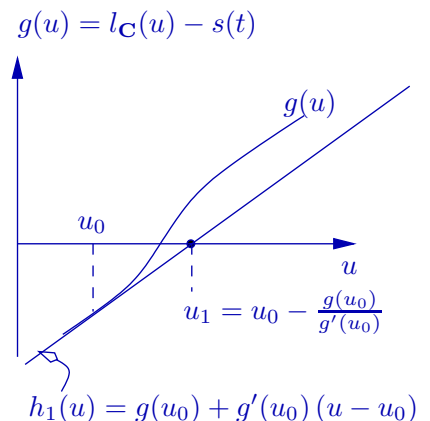
Iteration: Lege Tangente durch $g(u_i)$:

$$h_i(u) = g(u_i) + g'(u_i)(u - u_i)$$

u_{i+1} ist Nullstelle von $h_i(u)$:

$$u_{i+1} = u_i - \frac{g(u_i)}{g'(u_i)} = u_i - \frac{l_C(u_i) - s(t)}{\|C'(u_i)\|}$$

$$\text{da } g'(u) = \frac{d}{du} \left(\int_{u_0}^u \|C'(\tau)\| d\tau - s(t) \right) = \|C'(u)\|$$



9.2.2 Formkontrolle



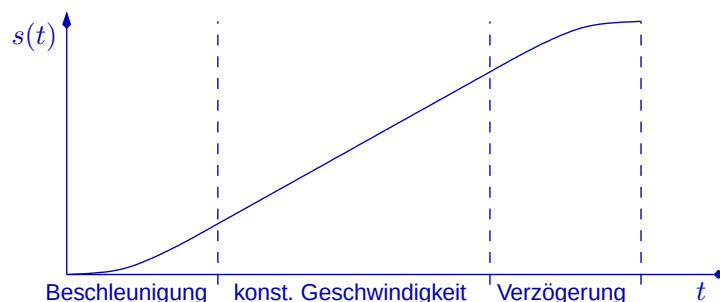
Bislang: Kontrolle des Kurvenverlaufs beschränkt auf

Bézier-Kurven/B-Spline: Wahl von Kontrollpunkten

Bézier-Splines/Catmull-Rom Splines: Angabe von Interpolationspunkten und Wahl/Abschätzung von Tangenten

Problem: Viele Anwendungen benötigen andere Kontrollmöglichkeiten, z.B. Festlegung einer *Wegfunktion*

Beispiel: Konstante Beschleunigung - konstante Geschwindigkeit - konstante Verzögerung



Ziel: Bessere Animations-spezifische Kontrollmöglichkeiten

9.2.2 Formkontrolle ...



Ziel: Ease-In/Ease-Out

Definition einer Funktion: $\text{ease}(t) = \begin{cases} \text{„weiche Beschleunigung“} & \text{für } t \in [0, \frac{1}{2}] \\ \text{„weiche Verzögerung“} & \text{für } t \in [\frac{1}{2}, 1] \end{cases}$

Ansatz: Sinus-Ease-Funktion

Idee: Sinus hat auf $[-\frac{\pi}{2}, \frac{\pi}{2}]$ das gewünschte Verhalten

Definition: Anpassung auf $t \in [0, 1]$: $\text{ease}_{\sin}(t) = \frac{1}{2}\sin(t\pi - \frac{\pi}{2}) + \frac{1}{2} \in [0, 1]$

Anpassung der Beschleunigung/Verzögerung durch Skalierung des Parameters t

Beispiel: Beschleunigung von 0 auf v_0 in der Zeit $t = 0$ bis $t = t_0$:

$$\text{ease}_{\sin}^{v_0, t_0}(t) = \frac{4v_0 t_0}{\pi} \text{ease}_{\sin}\left(\frac{t}{t_0}\right), \quad \text{d.h. } t \in [0, t_0] \Rightarrow \frac{t}{t_0} \in [0, \frac{1}{2}]$$

$$\begin{aligned} \text{Geschwindigkeit } v(t) &= (\text{ease}_{\sin}^{v_0, t_0})'(t) = \frac{4v_0 t_0}{2\pi} \cos\left(\frac{t\pi}{2t_0} - \frac{\pi}{2}\right) \frac{\pi}{2t_0} \\ &= v_0 \cos\left(\frac{\pi}{2t_0}(t - t_0)\right) \implies v(t_0) = v_0 \end{aligned}$$



9.2.2 Formkontrolle ...



Geschwindigkeit und Beschleunigung

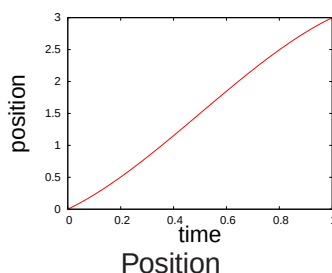
Bislang: Bewegungskontrolle über Wegfunktion

Interaktion: Meist Kontrolle über Geschwindigkeit bzw. Beschleunigung

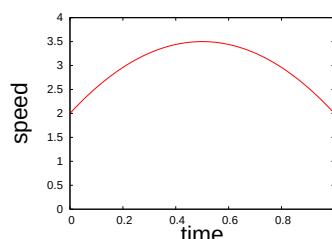
Newton: Zusammenhang zw. Weg $s(t)$, Geschwindigkeit $v(t)$ und Beschleunigung $a(t)$

$$\text{Konst. } v : s(t) = s(t_0) + (t - t_0) \cdot v \quad \text{Konst. } a : v(t) = v(t_0) + (t - t_0) \cdot a$$

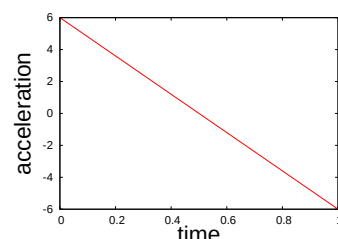
$$\begin{aligned} \text{Variables } v : s(t) &= s(t_0) + \int_{t_0}^t v(\tau) d\tau & \text{Variables } a : v(t) &= v(t_0) + \int_{t_0}^t a(\tau) d\tau \\ \implies v(t) &= s'(t) & \implies a(t) &= v'(t) \end{aligned}$$



Position



Geschwindigkeit



Beschleunigung





Ansatz: Parabolische Ease-Funktion

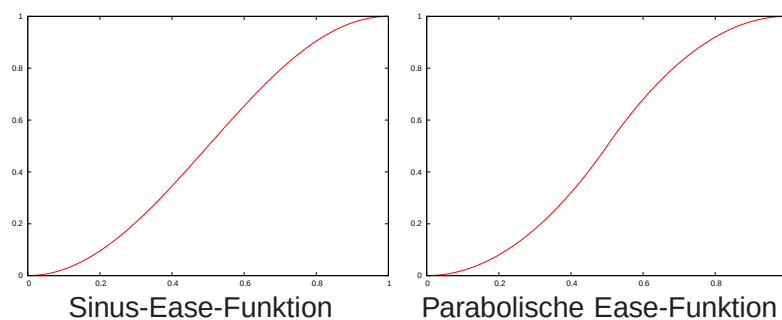
Idee: Verwende eine *konstante Beschleunigung*, damit

konst. Beschleunigung ($a = 1$) $\implies$ lineare Geschwindigkeit $\implies$ quadr. Wegfunktion

$$\text{ease}_{\text{parabol}}(t) = \begin{cases} 2t^2 & \text{falls } t \in [0, \frac{1}{2}] \\ 1 - 2(1 - t)^2 & \text{falls } t \in [\frac{1}{2}, 1] \end{cases}$$

Beachte: Anpassung der Beschleunigung/Verzögerung muss berücksichtigen, dass $(\text{ease}_{\text{sin}})'(0.5) = 1 \neq (\text{ease}_{\text{parabol}})'(0.5) = 2$

Vergleich:



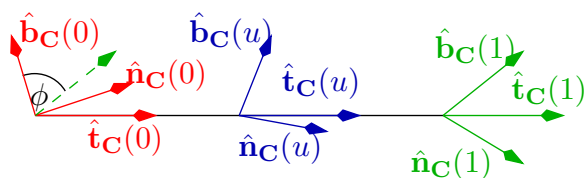
9.2.3 Kamera-Animation



Erinnerung: Frenet-Frame definiert lokale Koordinaten für Kurven (Kamera-Pfad)

Problem der fehlenden Krümmung für Geraden $\implies$ kein Frenet-Frame berechenbar

Interpolation der Frames für $u \in [0, 1]$, wenn Frame für $u \in \{0, 1\}$ bekannt



Tangente: $\hat{t}_C(u) = \hat{t}_C(0) = \hat{t}_C(1)$

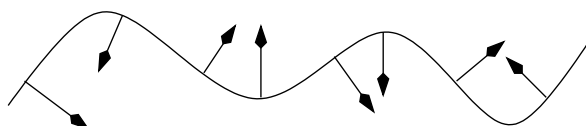
Normale: $\hat{n}_C(u) = R_{\hat{t}_C}(u \cdot \phi) \hat{n}_C(0)$

Binormale: $\hat{b}_C(u) = R_{\hat{t}_C}(u \cdot \phi) \hat{b}_C(0)$

wobei $\cos \phi = (\hat{n}_C(0) \cdot \hat{n}_C(1))$

Problem extremer Bewegungen z.B. meanderförmig

Beispiel einer Kurve mit Normalenvektoren:



Alternative Ansätze zur Kamera-Ausrichtung

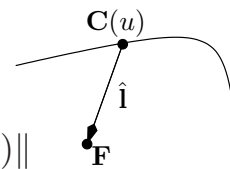
Lokale Koordinaten: Orthogonale Vektoren $\{\hat{\mathbf{l}}, \hat{\mathbf{u}}, \hat{\mathbf{w}} = \hat{\mathbf{l}} \times \hat{\mathbf{u}}\}$ mit Blickrichtung $\hat{\mathbf{l}}$ und Up-Vektor $\hat{\mathbf{u}}$ im Kurvenpunkt $\mathbf{C}(u)$

Bestimmung von $\hat{\mathbf{l}}$: ○ Fokuspunkt $\mathbf{F}$:

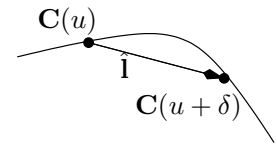
$$\hat{\mathbf{l}} = (\mathbf{F} - \mathbf{C}(u)) / \|\mathbf{F} - \mathbf{C}(u)\|$$

○ Vorausschauend (Sekante):

$$\hat{\mathbf{l}} = (\mathbf{C}(u+\delta) - \mathbf{C}(u)) / \|\mathbf{C}(u+\delta) - \mathbf{C}(u)\|$$



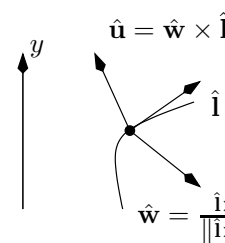
Fokuspunkt



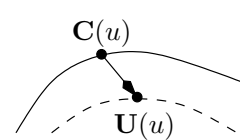
Vorausschauend

Bestimmung von $\hat{\mathbf{u}}$: ○ Globaler Up-Vektor z.B. y -Achse

$$\vec{\mathbf{w}} = \hat{\mathbf{l}} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \hat{\mathbf{w}} = \frac{\vec{\mathbf{w}}}{\|\vec{\mathbf{w}}\|}, \quad \hat{\mathbf{u}} = \hat{\mathbf{w}} \times \hat{\mathbf{l}}$$



Globaler Up-Vektor



Up-Vektor-Pfad

○ Up-Vektor durch 2. Punkt $\mathbf{U}(u)$

$$\vec{\mathbf{w}} = \hat{\mathbf{l}} \times (\mathbf{U}(u) - \mathbf{C}(u)), \quad \hat{\mathbf{w}} = \frac{\vec{\mathbf{w}}}{\|\vec{\mathbf{w}}\|}, \quad \hat{\mathbf{u}} = \hat{\mathbf{w}} \times \hat{\mathbf{l}}$$



9.3 Deformationen, Morphing und Warping

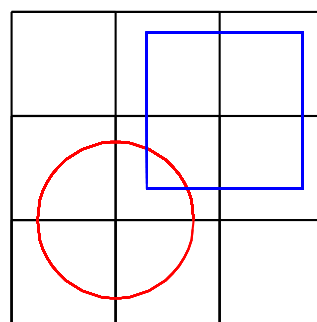


Bezeichnung:

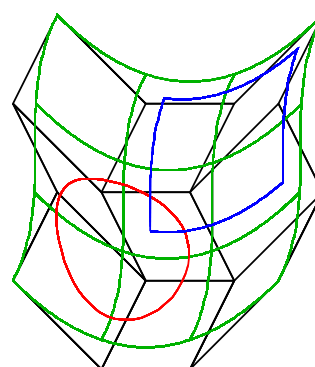
Deformation: Veränderung komplexer Objekte (z.B. Verbiegen oder Verdrehen) durch **punktwise Transformation** des umgebenden Raums

Warping: Variation der Kontrollparameter *eines Objektes* über die Zeit

Morphing: Kombinationen mehrerer Objekte zur Erzeugung „gemischter“ Geometrien



Original Objekte



Deformierte Objekte
(schwarz: Deformationsgitter)





Bezeichnung:

Deformationen (auch Space-Warp): Abbildung des Raumes auf sich selbst:

$$D : \begin{matrix} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ \mathbf{P} & \mapsto & D(\mathbf{P}) \end{matrix} \quad \text{bzw.} \quad D : \begin{matrix} \mathbb{R}^3 & \longrightarrow & \mathbb{R}^3 \\ \mathbf{P} & \mapsto & D(\mathbf{P}) \end{matrix}$$

Bezugsquader Q : Deformation D wird relativ zu einem Bezugsquader definiert:

$$Q = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$$

Normierte Koordinaten: Berechnung von $(u, v, w) \in [-1, 1]^3$ aus $(x, y, z) \in Q$:

Hinrechnung:
$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} (2x - (x_{\min} + x_{\max})) / (x_{\max} - x_{\min}) \\ (2y - (y_{\min} + y_{\max})) / (y_{\max} - y_{\min}) \\ (2z - (z_{\min} + z_{\max})) / (z_{\max} - z_{\min}) \end{pmatrix}$$

Rückrechnung:
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{(1-u)}{2} x_{\min} + \frac{(1+u)}{2} x_{\max} \\ \frac{(1-v)}{2} y_{\min} + \frac{(1+v)}{2} y_{\max} \\ \frac{(1-w)}{2} z_{\min} + \frac{(1+w)}{2} z_{\max} \end{pmatrix}$$



9.3.1 Deformationen ...



Ansatz: Tapering(Verjüngen), 2D oder 3D

Idee: Skalierung in Abhängigkeit einer zweiten Koordinate mittels *Taperingfunktion*:

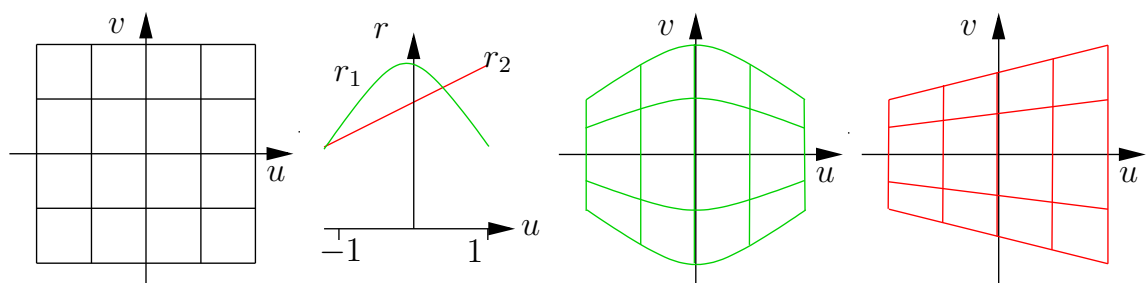
Vorgehen (2D):

Konstante Koordinate: beispielsweise u

Taperingfunktion: $r(u) = 2 - u^2$, d.h. $r(-1) = r(1) = 1$, $r(0) = 2$

Skalierung: $D(u, v, w) = (u, r(u)v, r(u)w)$

Beispiele:



9.3.1 Deformationen ...



Ansatz: Twisting(Verdrehen), 3D

Idee: Verdrehung abhängig von einer konstanten Koordinate

Vorgehen (3D):

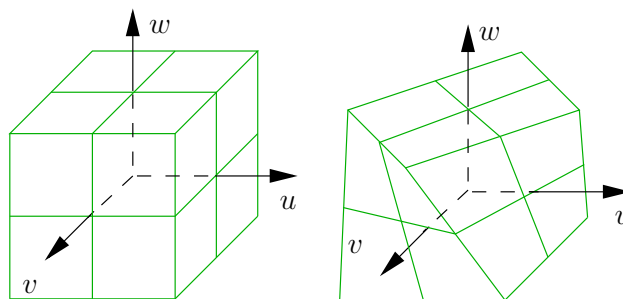
Konst. Koordinate: beispielsweise w

Max. Drehwinkel: α

Verdrehen: Mit $\phi(w) = \frac{\alpha(w+1)}{2}$, d.h. $\phi(-1) = 0$, $\phi(1) = \alpha$

$$D(u, v, w) = \begin{pmatrix} \cos(\phi(w)) & -\sin(\phi(w)) & 0 \\ \sin(\phi(w)) & \cos(\phi(w)) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Beispiel: Verdrehung um die w -Achse



9.3.1 Deformationen ...



Ansatz: Bending(Verbiegen), 2D oder 3D

Idee: Biegung in einer Ebene, Parameter $u \hat{=}$ Biegewinkel, $v \hat{=}$ Biegeradius

Vorgehen (2D): Verbiegen mit Zentrum (u_0, v_0) & Winkelsextrema ϕ_{min}, ϕ_{max}

Biegewinkel (u -Parameter):

$$\phi(u) = \frac{1-u}{2}\phi_{min} + \frac{1+u}{2}\phi_{max}$$

Biegeradius (v -Parameter):

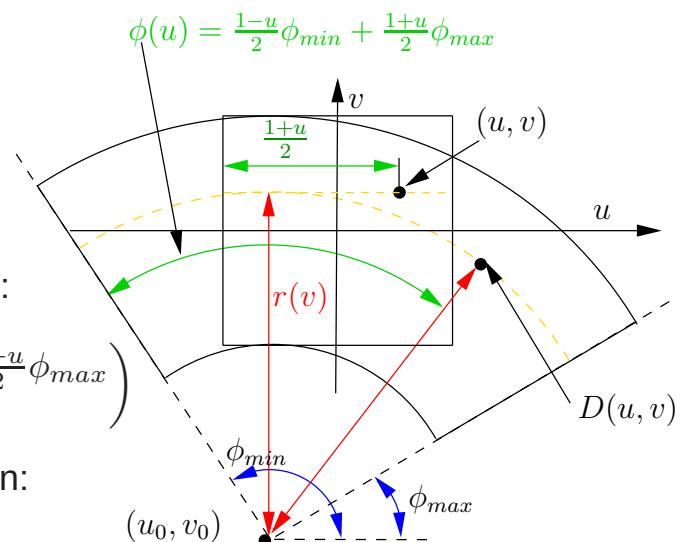
$$r(v) = v - v_0$$

Verbiegen (Polar-Koord. bzgl. u_0, v_0):

$$D(u, v) = \begin{pmatrix} \phi(u) \\ r(v) \end{pmatrix} = \begin{pmatrix} \frac{1-u}{2}\phi_{min} + \frac{1+u}{2}\phi_{max} \\ v - v_0 \end{pmatrix}$$

Verbiegen in kartesische Koordinaten:

$$D(u, v) = \begin{pmatrix} r(v) \cos(\phi(u)) \\ r(v) \sin(\phi(u)) \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$$



9.3.1 Deformationen ...

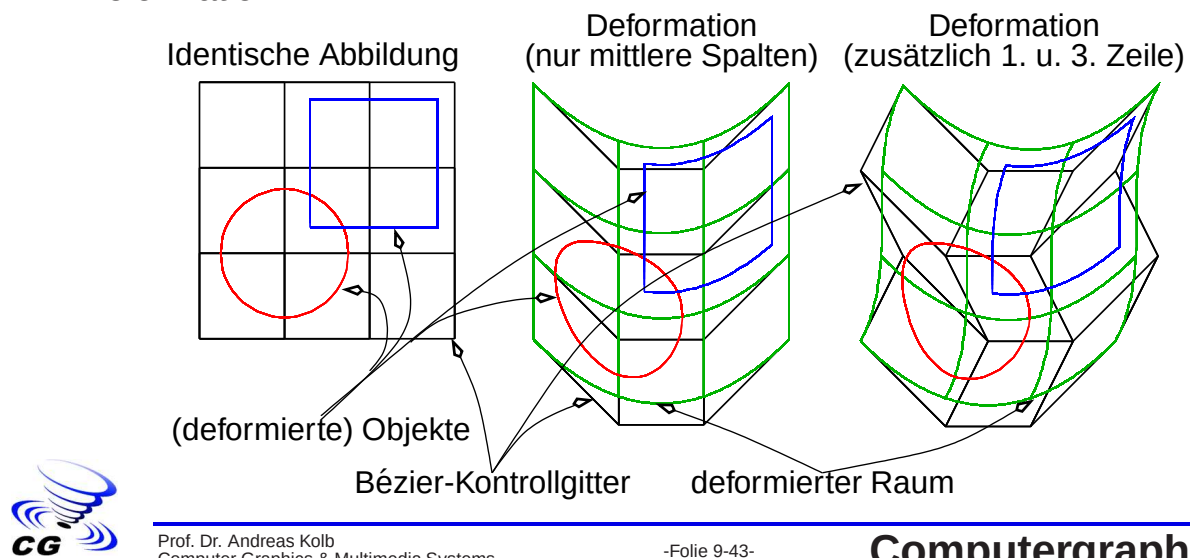


Freiform Deformation, 2D

Ziel: Freie Verzerrung des Raumes

Initial: Bilde Bezugsquader Q mit Hilfe der Bézier-Fläche $C(u, v)$ identisch auf sich selbst ab.

Deformation: Veränderung der Kontrollpunkte in der Ebene definiert Deformation



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 9-43-

Computergraphik II

9.3.1 Deformationen ...



Freiform-Deformation (Forts.)

Identische Abbildung $Q \rightarrow [0, 1]^2$:

1. $(x, y) \in Q$ wird in das Parametergebiet $[0, 1]^2$ abgebildet:

$$u(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad v(y) = \frac{y - y_{\min}}{y_{\max} - y_{\min}}$$

2. Initiale Kontrollpunkte C_{ij} , $i, j = 0, \dots, 3$ für kubische Bézier-Fläche:

$$C_{ij} = \left(\begin{pmatrix} (1 - \frac{i}{3})x_{\min} + \frac{i}{3}x_{\max} \\ (1 - \frac{j}{3})y_{\min} + \frac{j}{3}y_{\max} \end{pmatrix} \right) \Rightarrow D(x, y) = \sum_{i,j=0}^3 C_{ij} B_i^3(u(x)) B_j^3(v(y)) = \begin{pmatrix} x \\ y \end{pmatrix}$$

3D-Fall: Analog zum 2D-Fall mit *tri-variaten* Bézier-Volumen:

$$C_{ijk} = \left(\begin{pmatrix} (1 - \frac{i}{3})x_{\min} + \frac{i}{3}x_{\max} \\ (1 - \frac{j}{3})y_{\min} + \frac{j}{3}y_{\max} \\ (1 - \frac{k}{3})z_{\min} + \frac{k}{3}z_{\max} \end{pmatrix} \right)$$

$$\Rightarrow D(x, y, z) = C(u(x), v(y), w(z)) = \sum_{i,j,k=0}^3 C_{ijk} B_i^3(u(x)) B_j^3(v(y)) B_k^3(w(z)) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



Prof. Dr. Andreas Kolb
Computer Graphics & Multimedia Systems

-Folie 9-44-

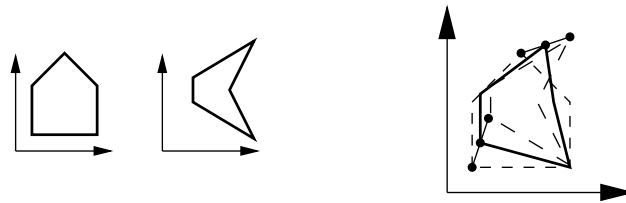
Computergraphik II

Idee: Direkte Anwendung der Interpolationstechnik auf Geometrien
 $\Rightarrow$ Fließender Übergang zwischen den Formen

Grundsätzlicher Ansatz

Ausgangssituation: Kontrollpunkte P_i $i = 1, \dots, k$ eines Objektes die zu Zeitpunkten $t_1, \dots, t_n$ bekannt sind: $P_i(t_1), \dots, P_i(t_n)$, $i = 1, \dots, k$

Vorgehen: Interpolation liefert $P_i(t)$ für beliebiges t , womit die Geometrie zu jedem Zeitpunkt bestimmbar ist.



9.3.2 Morphing ...

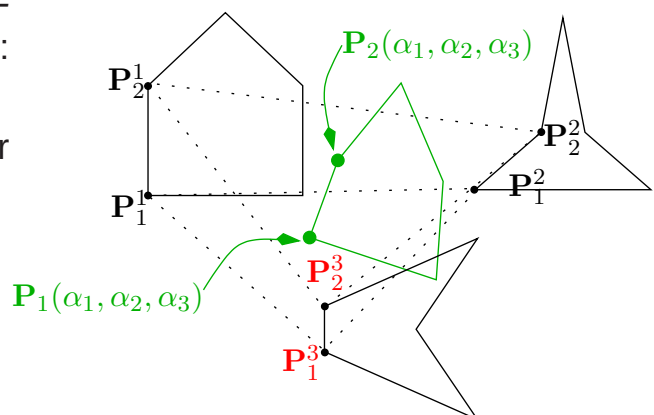
Morphing mit Affin-Kombinationen

Verwende *Freiform-Prototypen (Blendshapes)* und mische diese Positionen mit Hilfe von Affin-Kombinationen.

Ausgangssituation: Kontrollpunkte P_i eines Objektes die für *Key-Positionen* $1, \dots, k$ bekannt sind: $P_i^1, \dots, P_i^k$.

Vorgehen: Affin-Kombination: Für Gewichte α_j mit $\sum_{j=0}^n \alpha_j = 1$:

$$P_i = P_i(\alpha_1, \dots, \alpha_k) = \sum_{j=0}^k \alpha_k P_j$$

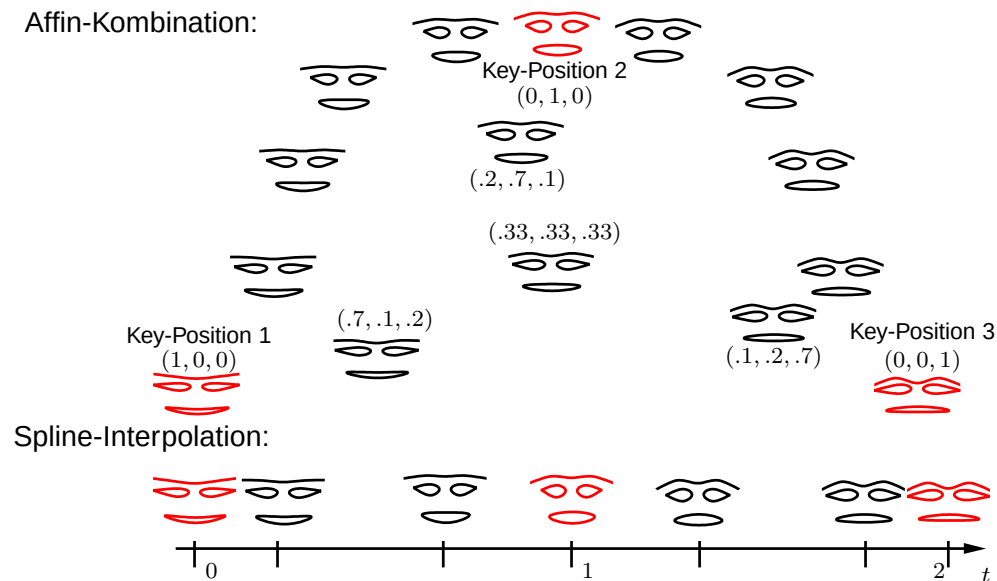


Animation: Variiere die Gewichtsparameter α_k über die Zeit: $\alpha_i(t)$

Morphing-Beispiel

Ausgangslage: 3 Prototypen mit Semantik („fröhlich“, „erschrocken“, „ängstlich“)

Animation: Veränderung der Gewichte über die Zeit



Weitere Morphingaspekte

Farbe: Neben der Form, kann auch die Farbe etc. verändert werden

Beispiel: Morphing in einem Bild

1. Warping (FFD) stellt Pixel in in Beziehung: $(x, y) = D^0(x, y) \rightarrow D^1(x, y)$
 $\Rightarrow$ Bézier-Kontrollpunkte für Start- und Endbild C_{ij}^{init} (initial), C_{ij}^{end}
2. Zwischenbild (Zeit t) durch Interpolation von Kontrollpunkten & Farbe:

$$C_{ij}^t = (1 - t) \cdot C_{ij}^{\text{init}} + t \cdot C_{ij}^{\text{end}} \Rightarrow D^t$$

$$I(D^t(x, y)) = (1 - t) \cdot I(D^0(x, y)) + t \cdot I(D^1(x, y))$$

