Computergraphik II

Winter 2011/2012

9 Computer Animation

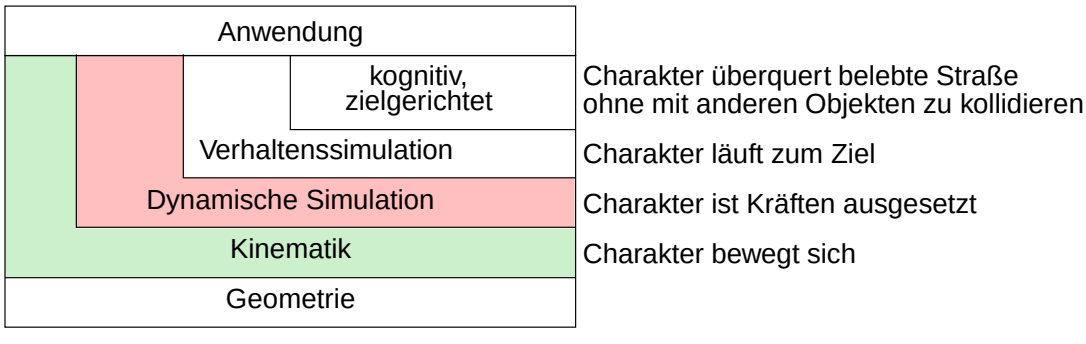
Versionsdatum: 18. November 2011

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

9.1 Keyframe Animation

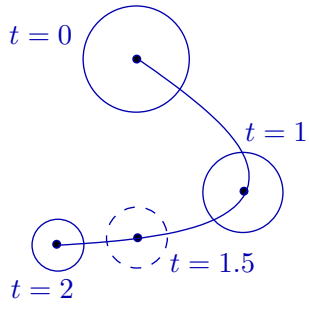
Bezug zur Trickfilm-Animation: Interpolation vorgegebener Schlüsselbilder:

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

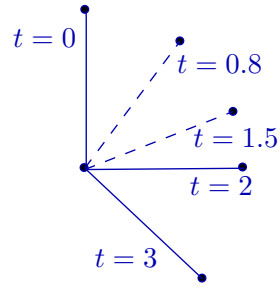
Grundsätzliches Vorgehen: 1. Beschreibung der Szene $\mathcal{O}$ durch *zeitabhängige Parameter (Animationsparameter)* $\phi_1, \phi_2, \ldots, \phi_N$

2. Zwischenzeitpunkt t: Interpolation der Parameter $\mathcal{O}(\phi_1(t), \ldots, \phi_N(t))$

Animationsparameter: Unterscheide *Position* und *Orientierung*



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



zeitabhängig: Richtung $\vec{\mathbf{r}}$

9.1.1 Interpolation der Position

Aufgabe:

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

Konkrete Aufgabe:

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

Gesucht: Kurve $\mathbf{C} : \mathbf{C}(t_i) = \mathbf{P}_i,\ i = 1, \ldots, N$

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

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

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.

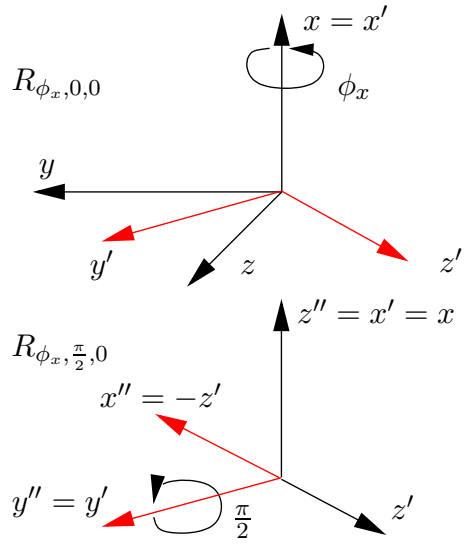
Gesamtrotation für Abarbeitungsreihenfolge x, y, z:

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

Reihenfolge entspricht Top-Down Ansatz bei Graphik-Programmierung.

Problem: Mehrdeutigkeit (*Gimbal-Lock*)

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



Mehrdeutigkeit der Euler-Winkel & Animation

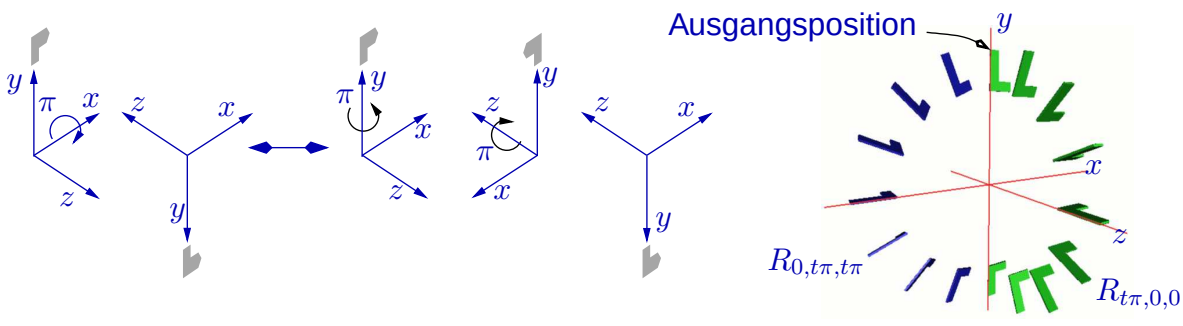
Beachte: Für Animation muss Repräsentation der Orientierung eindeutig sein

Denn: Sonst entstehen „beliebige“ Interpolationsergebnisse

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

❍ Interpolationspfade 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: $\mathfrak{Re}(c) = x$, Imaginärteil von c: $\mathfrak{Im}(c) = y$

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

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

Addition: $c_1 + c_2 = x_1 + i \cdot y_1 + x_2 + i \cdot y_2 = (x_1 + x_2) + i \cdot (y_1 + y_2)$
$\Rightarrow \mathfrak{Re}(c_1 + c_2) = x_1 + x_2$; $\mathfrak{Im}(c_1 + c_2) = y_1 + y_2$

Multiplikation:
$c_1 \cdot c_2 = (x_1 + i \cdot y_1) \cdot (x_2 + i \cdot y_2) = x_1 \cdot x_2 + i \cdot x_1 \cdot y_2 + i \cdot y_1 \cdot x_2 + i^2 \cdot y_1 \cdot y_2$
$\Rightarrow c_1 \cdot c_2 = (x_1 \cdot x_2 - y_1 \cdot y_2) + i \cdot (x_1 \cdot y_2 + x_2 \cdot y_1)$ (da $i^2 = -1$)
$\Rightarrow \mathfrak{Re}(c_1 \cdot c_2) = x_1 \cdot x_2 - y_1 \cdot y_2$; $\mathfrak{Im}(c_1 \cdot c_2) = x_1 \cdot y_2 + x_2 \cdot y_1$

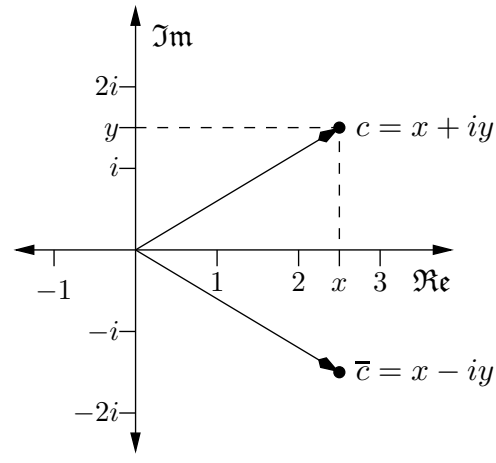
Addition und Multiplikation komplexer Zahlen sind kommutativ!

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

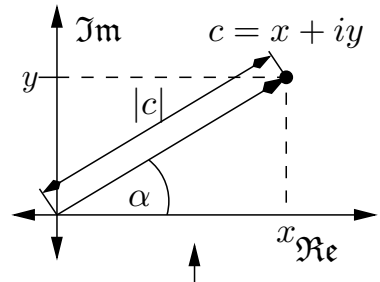
$$c = x + i \cdot y \Rightarrow \overline{c} := x - i \cdot y$$

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

Polarkoordinaten

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

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



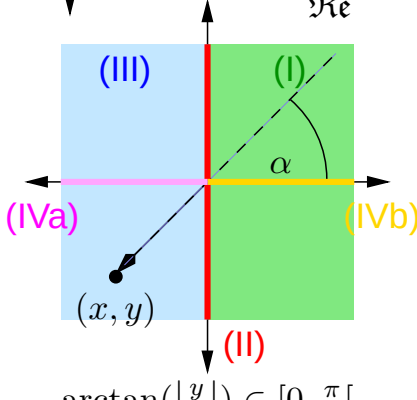
Aufgrund von $\cos^2\alpha + \sin^2\alpha = 1$ (Pythagoras) ist

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

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

$\alpha =$ arctan2(y, x)

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



$\arctan(\left|\frac{y}{x}\right|) \in [0, \frac{\pi}{2}[$

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)

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

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

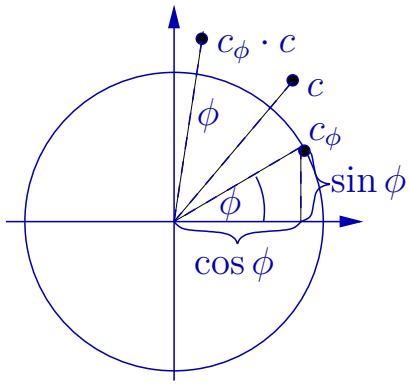
$$R_\phi : \begin{array}{ccc} \mathbb{C} & \longrightarrow & \mathbb{C} \\ c & \mapsto & R_\phi(c) = c_\phi \cdot c \end{array}$$

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

Exponential-Schreibweise von Polar-Koord.:

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

$$\Rightarrow \boxed{c_\phi \cdot c = |c| \, e^{i\alpha} \cdot e^{i\phi} = |c| \, e^{i(\alpha+\phi)}}$$



9.1.3 Exkurs: Komplexe Zahlen ...

Rotation mit komplexen Zahlen (Forts.)

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

Durchführung der Rotation

1. Interpretiere $\mathbf{P}$ als komplexe Zahl in $\mathbb{C}$: $z = x + iy$
2. Rotatiere 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:

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

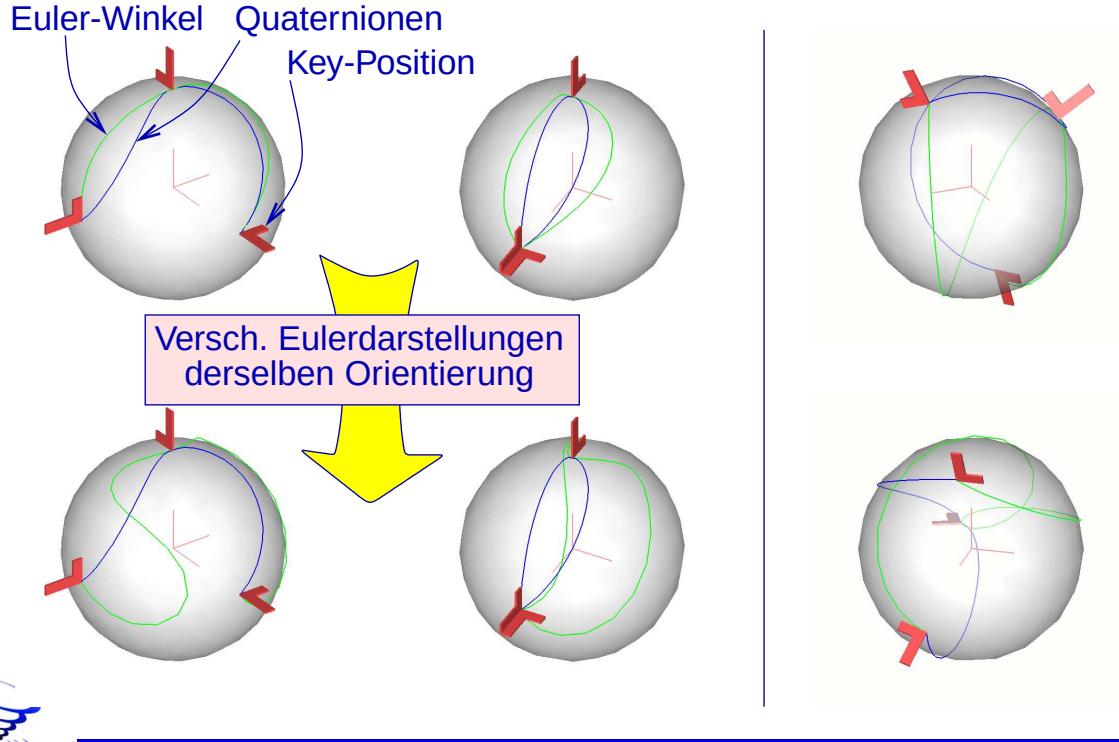
Damit können in diese Funktionen problemlos komplexe Parameter eingesetzt werden.

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:

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

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

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

Multiplikation: Durch Anwendung obiger Regeln folgt:

$$\begin{aligned}
\underline{\mathbf{q}_1\mathbf{q}_2} &= (s_1 + ix_1 + jy_1 + kz_1)(s_2 + ix_2 + jy_2 + kz_2) \\
&= s_1s_2 - (x_1x_2 + y_1y_2 + z_1z_2) + i(s_1x_2 + s_2x_1 + y_1z_2 - y_2z_1) \\
&\quad + j(s_1y_2 + s_2y_1 + z_1x_2 - z_2x_1) + k(s_1z_2 + s_2z_1 + x_1y_2 - x_2y_1) \\
&= (s_1s_2 - (\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_2), s_1\vec{\mathbf{v}}_2 + s_2\vec{\mathbf{v}}_1 + \vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2)
\end{aligned}$$

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

9.1.4 Quaternionen ...

Eigenschaften von Quaternionen

Konjugierte:

$$\overline{\underline{\mathbf{q}}} = (s, -\vec{\mathbf{v}}),\ \overline{\underline{\mathbf{q}}}_1\overline{\underline{\mathbf{q}}}_2 = (s_1s_2 - (\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_2), -s_1\vec{\mathbf{v}}_2 - s_2\vec{\mathbf{v}}_1 + \underbrace{\vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2}_{=-\vec{\mathbf{v}}_2 \times \vec{\mathbf{v}}_1}) = \overline{(\underline{\mathbf{q}}_2\underline{\mathbf{q}}_1)}$$

Länge:

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

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

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

$$\cos(\angle(\underline{\mathbf{q}}_1, \underline{\mathbf{q}}_2)) = \Re\mathfrak{e}(\underline{\mathbf{q}}_1\overline{\underline{\mathbf{q}}}_2) = s_1s_2 + (\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_2)$$

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)

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

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.:

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

Beispiel:

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

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

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 $(1, -2, 0)$

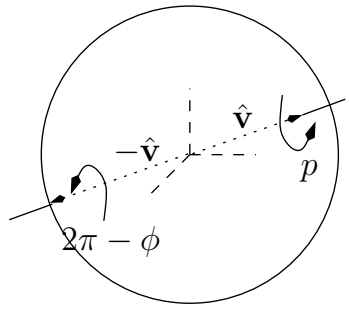
Mehrdeutigkeit von Quaternionen

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

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

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

$$\cos(\pi - \tfrac{\phi}{2}) = -\cos(\tfrac{\phi}{2}) \text{ und } \sin(\pi - \tfrac{\phi}{2}) = \sin(\tfrac{\phi}{2})$$
$$\Rightarrow -\underline{\mathbf{q}} = (\cos(\pi - \tfrac{\phi}{2}), \sin(\pi - \tfrac{\phi}{2})(-\hat{\mathbf{v}}))$$



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?

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

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

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

Umrechnung Quaternionen $\leftrightarrow$ Rotationsmatrizen

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

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

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

$$r_{11} + r_{22} + r_{33} = 3 - 4(x^2+y^2+z^2) = 3 - 4(1-s^2) = -1 + 4s^2$$

$$\Rightarrow s = \pm\tfrac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1}$$

$$r_{32} - r_{23} = 4sx \Rightarrow x = \frac{r_{32}-r_{23}}{4s}, \quad \text{analog: } y = \frac{r_{13}-r_{31}}{4s},\ z = \frac{r_{21}-r_{12}}{4s}$$

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

9.1.5 Interpolation der Orientierung

Ziel: Interpolation von Orientierungen, in Euler-Notation:

Gegeben: Zeit-Orientierungs-Paare $(t_i, R_{\vec{\phi}^i})$, $\vec{\phi}^i = \left(\phi_x^i, \phi_y^i, \phi_z^i\right)$, $i = 1, \ldots, N$

Gesucht: Funktion R mit $R(t_i) = R_{\vec{\phi}^i}$, $i = 1, \ldots, N$

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

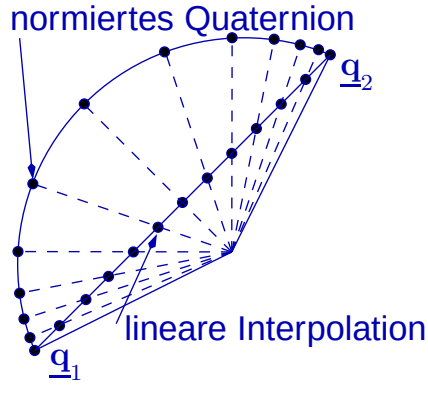
❍ Problem: Ergebnis von der Repräsentation der Orientierung abhängig

Interpolation mit Quaternionen:

Erinnerung: Rotationen werden durch $4D$-**Einheitsquaternionen** beschrieben

Problem: Interpolierte Quaternionen müssen Einheitsquaternionen sein.
Bézier, B-Spline leisten das nicht:
$\mathbf{C}(i) \in \mathbb{R}^4, \|\mathbf{C}(i)\| = 1 \not\Longrightarrow \|\mathbf{C}(u)\| = 1.$

Normierung liefert verzerrte Schrittweiten



Sphärische lineare Interpolation (slerp)

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

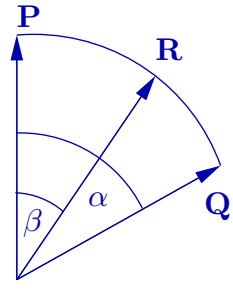
Gegeben: $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ auf Einheitskreis (bzw. auf Großkreis), $\mathbf{R}$ „zwischen“ $\mathbf{P}$ und $\mathbf{Q}$: $\alpha = \angle(\mathbf{P}, \mathbf{Q}), \beta = \angle(\mathbf{P}, \mathbf{R})$.

Lösung: Zusammenhang zwischen Winkel und Punkt:

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

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

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



Linearen Interpolation von Rotationen durch sphärische Interpolation von Einheitsquaternionen:

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

Bemerkung: Details zur Interpolation auf Kreisbögen

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

o.B.d.A.: Punkte liegen in der Ebene und $\mathbf{P} = (1, 0)$
$\Rightarrow \mathbf{Q} = (\cos\alpha, \sin\alpha)$ und $\mathbf{R} = (\cos\beta, \sin\beta)$

Gesucht: Linearkombination $\mathbf{R} = a\mathbf{P} + b\mathbf{Q}$

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

$$\begin{aligned} a\mathbf{P} + b\mathbf{Q} &= \frac{\sin(\alpha-\beta)}{\sin\alpha} \cdot \begin{pmatrix}1\\0\end{pmatrix} + \frac{\sin\beta}{\sin\alpha} \cdot \begin{pmatrix}\cos\alpha\\\sin\alpha\end{pmatrix} \\ &= \begin{pmatrix} \frac{\sin\alpha\cos\beta - \cos\alpha\sin\beta}{\sin\alpha} \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{\sin\beta\cdot\cos\alpha}{\sin\alpha} \\ \sin\beta \end{pmatrix} = \begin{pmatrix}\cos\beta\\\sin\beta\end{pmatrix} = \mathbf{R} \end{aligned}$$

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 $\mathbf{P}_i$?
Also: Entstehen Bézier-Kontrollpunkte durch lineare Interpolation aus $\mathbf{P}_i$?

Konkret: i-tes Bézier-Segment mit KPs $\mathbf{C}_0^i, \ldots, \mathbf{C}_3^i$ und Interpolationspunkte $\mathbf{P}_{i-1}, \ldots, \mathbf{P}_{i+2}$:

$$\mathbf{C}_0^i = \mathbf{P}_i, \quad \mathbf{C}_3^i = \mathbf{P}_{i+1} \text{(Endpunkte)}, \qquad \vec{\mathbf{t}}_i = \tfrac{1}{2}\left(\mathbf{P}_{i+1} - \mathbf{P}_{i-1}\right) \text{(Tangenten)}$$

$$\mathbf{C}_1^i = \mathbf{P}_i + \tfrac{1}{3}\vec{\mathbf{t}}_i = \mathbf{P}_i + \tfrac{1}{6}\left(\mathbf{P}_{i+1} - \mathbf{P}_{i-1}\right) = \tfrac{1}{6}\left[2\left(\tfrac{1}{2}\mathbf{P}_i + \tfrac{1}{2}\mathbf{P}_{i+1}\right) - \mathbf{P}_{i-1}\right] + \tfrac{5}{6}\mathbf{P}_i$$

$$\begin{aligned}\mathbf{C}_2^i &= \mathbf{P}_{i+1} - \tfrac{1}{3}\vec{\mathbf{t}}_{i+1} = \mathbf{P}_{i+1} - \tfrac{1}{6}\left(\mathbf{P}_{i+2} - \mathbf{P}_i\right) \\ &= \tfrac{1}{6}\left[2\left(\tfrac{1}{2}\mathbf{P}_{i+1} + \tfrac{1}{2}\mathbf{P}_i\right) - \mathbf{P}_{i+2}\right] + \tfrac{5}{6}\mathbf{P}_{i+1}\end{aligned}$$

Sphärische Spline Interpolation:
Verwende slerp-Operation anstelle linearer Interpolation!

Sphärische Spline Interpolation (Forts.)

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

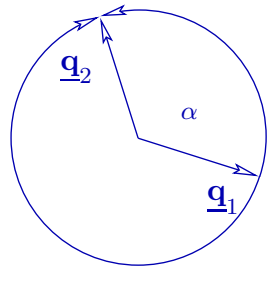
Frage: Welcher Repräsentant von $R_{\underline{\mathbf{q}}_i}$ ist zu wählen, $\underline{\mathbf{q}}_i$ oder $-\underline{\mathbf{q}}_i$?

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

Regel:

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

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



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

Idee: Spline-basierte Animation

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

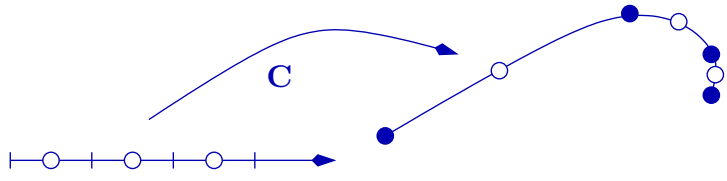
Spline-basiert: Direkte Vorgabe der Spline-Kurven

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

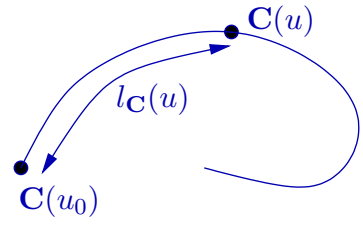


9.2.1 Pfad und Geschwindigkeit

Ziel: Bewegung eines Körpers entlang einer Kurve

Bewegungspfad des Objekt im Raum

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



Wegfunktion: Zurückgelegter Weg auf dem Pfad zum Zeitpunkt t:

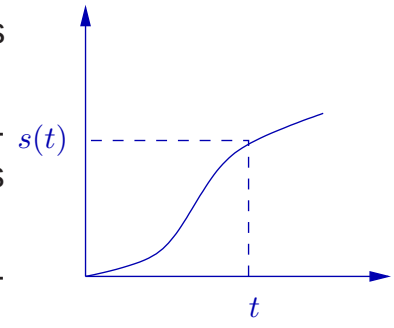
$$s(t): \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ t & \mapsto & s(t) \end{array}$$

Bewegungspfad

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

Bogenlängenfunktion $l_{\mathbf{C}}(u)$ misst Länge des Kurvenstückes für Pfad $\mathbf{C}$ vom Startpunkt $\mathbf{C}(u_0)$ bis $\mathbf{C}(u)$

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



Wegfunktion

Ermittlung der Bogenlänge

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

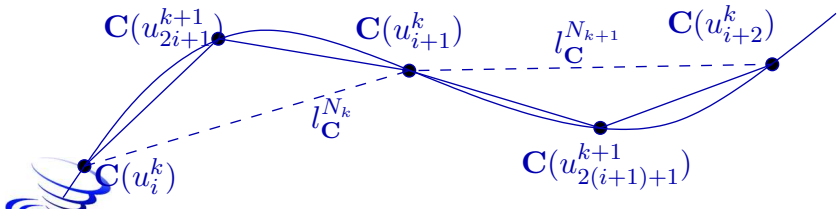
Abschätzung der Bogenlänge $l_{\mathbf{C}}(u)$ durch Länge eines Kantenzuges:

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

Exakte Lösung durch Grenzbetrachtung:

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

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



Abbruch, falls wenig Genauigkeitszuwachs: $l_{\mathbf{C}}^{N_{k+1}} - l_{\mathbf{C}}^{N_k} < \varepsilon$

Ermittlung des Parameters u (feste Pfad-Kurve)

Beobachtung: Ermittlung von $l_{\mathbf{C}}$ wird durch Auswertung von $\mathbf{C}$ dominiert

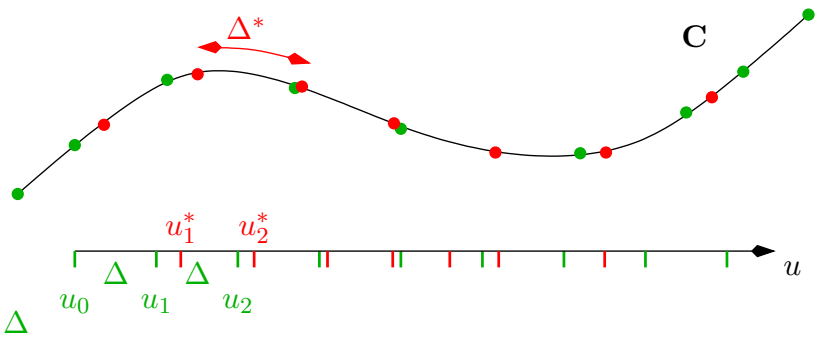
Ansatz für feste Pfad-Kurve: Speicherung von $l_{\mathbf{C}}(u)$ in **Look-Up Tabelle**

1. Speicherung von $l_{\mathbf{C}}(u_i)$ 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	$l_1 + l_2$



Variante 2:

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

$u_i = u_0 + i \cdot \Delta$
$l_i = \|\mathbf{C}(u_i) - \mathbf{C}(u_{i-1})\|$

$l_{i+1}^* - l_i^* = \Delta^*$
$l_i^* = l_{\mathbf{C}}(u_i^*)$

Ermittlung des Parameters u (allgemeiner Fall)

Aufgabe: Zu gegebener Wegfunktion s und Zeitpunkt t: Für welchen Parameterwert u gilt $l_{\mathbf{C}}(u) = s(t)$

Ansatz: Suche Nullstelle von $g(u) = l_{\mathbf{C}}(u) - s(t)$

Beachte: $l_{\mathbf{C}}(u)$ ist monoton wachsend $\Rightarrow$ eindeutige Nullstelle von $g(u)$.

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

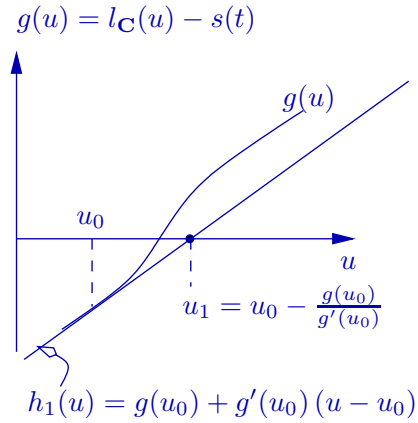
Initial:Startparameter u_0

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 $h_i(u)$:

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



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

9.2.2 Formkontrolle

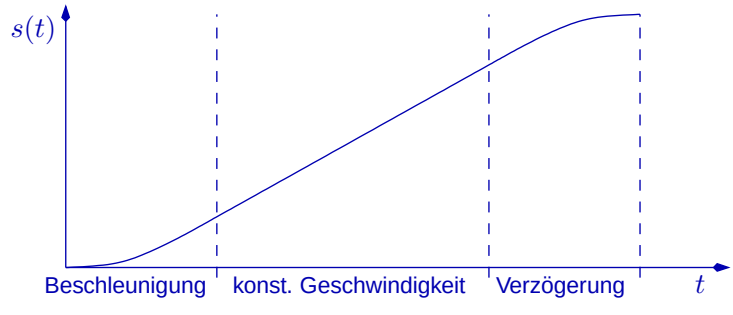
Bislang: Kontrolle des Kurvenverlaufs beschränkt auf

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

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 $t = t_0$:

$$\begin{aligned} \text{ease}_{\sin}^{v_0,t_0}(t) &= \frac{4v_0t_0}{\pi}\,\text{ease}_{\sin}\left(\frac{t}{2t_0}\right), \quad \text{d.h. } t \in [0,t_0] \Rightarrow \frac{t}{2t_0} \in [0,\tfrac{1}{2}] \\ \text{Geschwindigkeit } v(t) &= \left(\text{ease}_{\sin}^{v_0,t_0}\right)'(t) = \frac{4v_0t_0}{2\pi}\cos\left(\frac{t\pi}{2t_0} - \frac{\pi}{2}\right)\frac{\pi}{2t_0} \\ &= v_0\cos\left(\frac{\pi}{2t_0}(t-t_0)\right) \quad \Longrightarrow \; v(t_0) = v_0 \end{aligned}$$

Geschwindigkeit und Beschleunigung

Bislang: Bewegungskontrolle über Wegfunktion

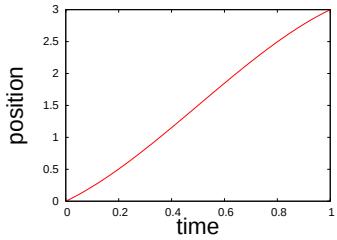
Interaktion: Meist Kontrolle über Geschwindigkeit bzw. Beschleunigung

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

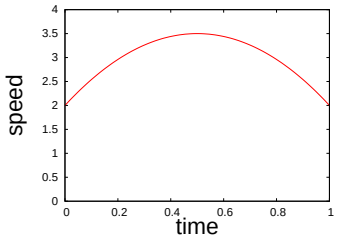
Konst. v : $s(t) = s(t_0) + (t - t_0) \cdot v$ Konst. a : $v(t) = v(t_0) + (t - t_0) \cdot a$

Variables v : $s(t) = s(t_0) + \int_{t_0}^{t} v(\tau)\, d\tau$ Variables a : $v(t) = v(t_0) + \int_{t_0}^{t} a(\tau)\, d\tau$

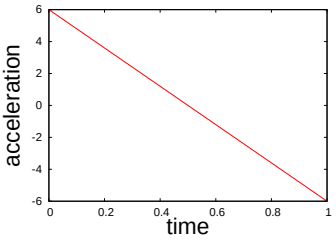
$\Rightarrow v(t) = s'(t)$ $\Rightarrow a(t) = v'(t)$



Position



Geschwindigkeit



Beschleunigung

Ansatz: Parabolische Ease-Funktion

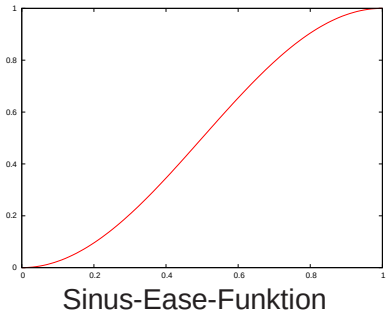
Idee: Verwende eine *konstante Beschleunigung*, damit

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

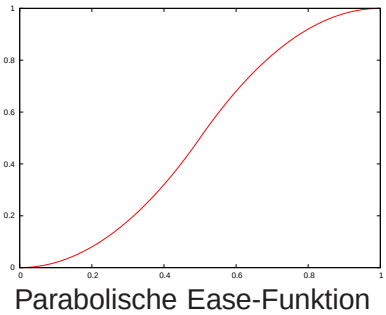
$$\text{ease}_{parabol}(t) = \begin{cases} 2t^2 & \text{falls } t \in [0, \frac{1}{2}] \\ 1 - 2(1-t)^2 & \text{falls } t \in [\frac{1}{2}, 1] \end{cases}$$

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

Vergleich:



Sinus-Ease-Funktion



Parabolische Ease-Funktion

9.2.2 Formkontrolle ...

TCB-Splines

Ansatz: Intuitive Kontrollparameter zur Tangentenabschätzung von Catmull-Rom-Splines

Kontrollparameter:

Interpolationspunkte	$\mathbf{P}_i,\ i = 1, \ldots, n$	
Tangentenparameter	*Tension* t:	Tangentenlänge
	Continuity c:	Stetigkeit benachbarter Kurvensegm.
	Bias b:	Einfluss der Kanten $\overline{\mathbf{P}_{i-1}\mathbf{P}_i}$ und $\overline{\mathbf{P}_i\mathbf{P}_{i+1}}$

Tangentenberechnung: Bézier-Kurve $\mathbf{C}^i$ verbindet $\mathbf{P}_i, \mathbf{P}_{i+1}$

1. $\mathbf{C}_0^i = \mathbf{P}_i,\ \mathbf{C}_3^i = \mathbf{P}_{i+1}$
2. Tangenten bei $\mathbf{P}_i$: Für $t, c, b \in [-1, 1]$ wird gesetzt:

$$\text{links: } \vec{\mathbf{t}}_i^- = \tfrac{1-t}{2}\left((1-b)(1-c)\left(\mathbf{P}_i - \mathbf{P}_{i-1}\right) + (1+b)(1+c)\left(\mathbf{P}_{i+1} - \mathbf{P}_i\right)\right)$$

$$\text{rechts: } \vec{\mathbf{t}}_i^+ = \tfrac{1-t}{2}\left((1-b)(1+c)\left(\mathbf{P}_i - \mathbf{P}_{i-1}\right) + (1+b)(1-c)\left(\mathbf{P}_{i+1} - \mathbf{P}_i\right)\right)$$

und $\mathbf{C}_1^i = \mathbf{P}_i + \frac{1}{3}\vec{\mathbf{t}}_i^+ \quad \mathbf{C}_2^i = \mathbf{P}_{i+1} - \frac{1}{3}\vec{\mathbf{t}}_{i+1}^-$

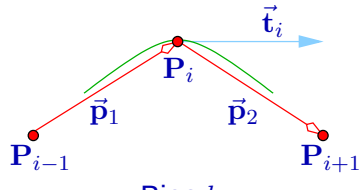
Einfluss der Kontrollgrößen t, c, b

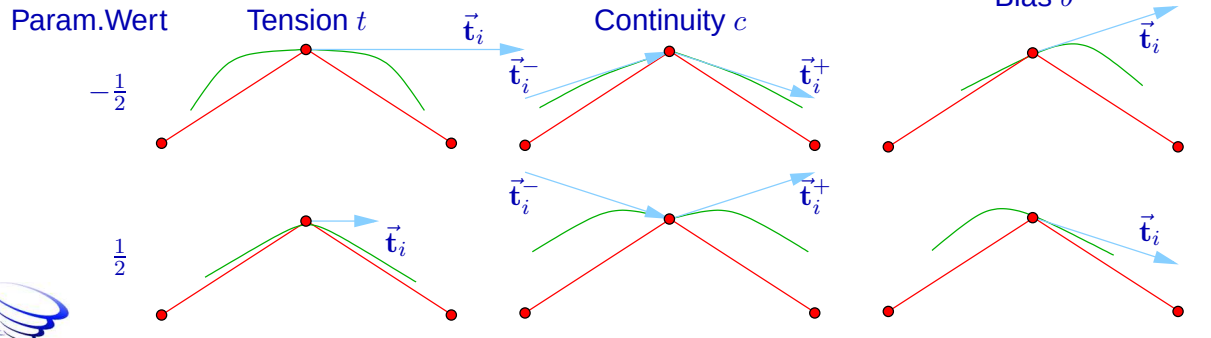
Darstellung in lokalen Koordinaten:

$$\vec{\mathbf{t}}_i = (a_1, a_2) \hat{=} a_1 \underbrace{(\mathbf{P}_i - \mathbf{P}_{i-1})}_{=\vec{\mathbf{p}}_1} + a_2 \underbrace{(\mathbf{P}_{i+1} - \mathbf{P}_i)}_{=\vec{\mathbf{p}}_2}$$

Ausgehend von $t = c = b = 0$:

	Tension t	Continuity c	Bias b
0	$\vec{\mathbf{t}}_i = (\frac{1}{2}, \frac{1}{2})$	$\vec{\mathbf{t}}_i = (\frac{1}{2}, \frac{1}{2})$	$\vec{\mathbf{t}}_i = (\frac{1}{2}, \frac{1}{2})$
$-\frac{1}{2}$	$\vec{\mathbf{t}}_i = (\frac{3}{4}, \frac{3}{4})$	$\vec{\mathbf{t}}_i^- = (\frac{3}{4}, \frac{1}{4}), \vec{\mathbf{t}}_i^+ = (\frac{1}{4}, \frac{3}{4})$	$\vec{\mathbf{t}}_i = (\frac{3}{4}, \frac{1}{4})$
$\frac{1}{2}$	$\vec{\mathbf{t}}_i = (\frac{1}{4}, \frac{1}{4})$	$\vec{\mathbf{t}}_i^- = (\frac{1}{4}, \frac{3}{4}), \vec{\mathbf{t}}_i^+ = (\frac{3}{4}, \frac{1}{4})$	$\vec{\mathbf{t}}_i = (\frac{1}{4}, \frac{3}{4})$



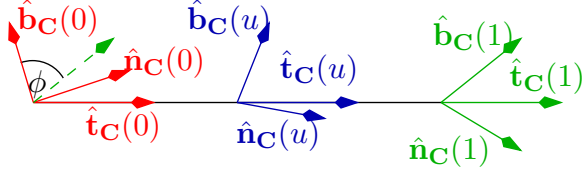


9.2.3 Kamera-Animation

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

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

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



Tangente: $\hat{\mathbf{t}}_\mathbf{C}(u) = \hat{\mathbf{t}}_\mathbf{C}(0) = \hat{\mathbf{t}}_\mathbf{C}(1)$

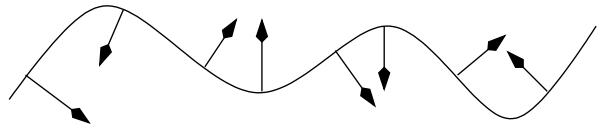
Normale: $\hat{\mathbf{n}}_\mathbf{C}(u) = R_{\hat{\mathbf{t}}_\mathbf{C}}(u \cdot \phi)\hat{\mathbf{n}}_\mathbf{C}(0)$

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

wobei $\cos\phi = (\hat{\mathbf{n}}_\mathbf{C}(0) \cdot \hat{\mathbf{n}}_\mathbf{C}(1))$

Problem extremer Bewegungen z.B. meanderförmig

Beispiel einer Kurve mit Normalenvektoren:



Alternative Ansätze zur Kamera-Ausrichtung

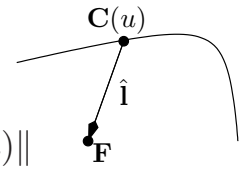
Lokale Koordinaten: Orthogonale Vektoren $\{\hat{\mathbf{l}}, \hat{\mathbf{u}}, \hat{\mathbf{w}} = \hat{\mathbf{l}} \times \hat{\mathbf{u}}\}$ mit Blickrichtung $\hat{\mathbf{l}}$ und Up-Vektor $\hat{\mathbf{u}}$ im Kurvenpunkt $\mathbf{C}(u)$

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

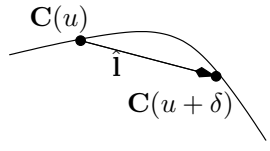
$$\hat{\mathbf{l}} = (\mathbf{F} - \mathbf{C}(u)) / \|\mathbf{F} - \mathbf{C}(u)\|$$

❍ Vorausschauend (Sekante):

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



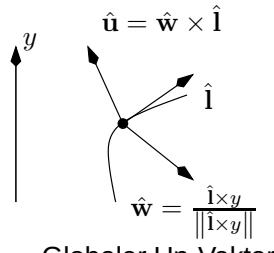
Fokuspunkt



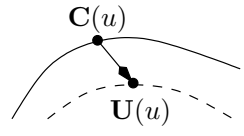
Vorausschauend

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 $\mathbf{U}(u)$

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

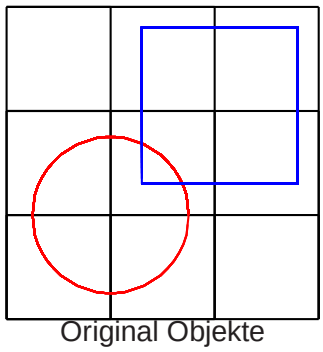
9.3 Deformationen, Morphing und Warping

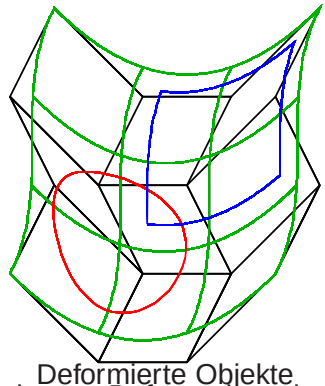
Bezeichnung:

Deformation: Veränderung komplexer Objekte (z.B. Verbiegen oder Verdrehen) durch **punktweise 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:

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

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

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

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

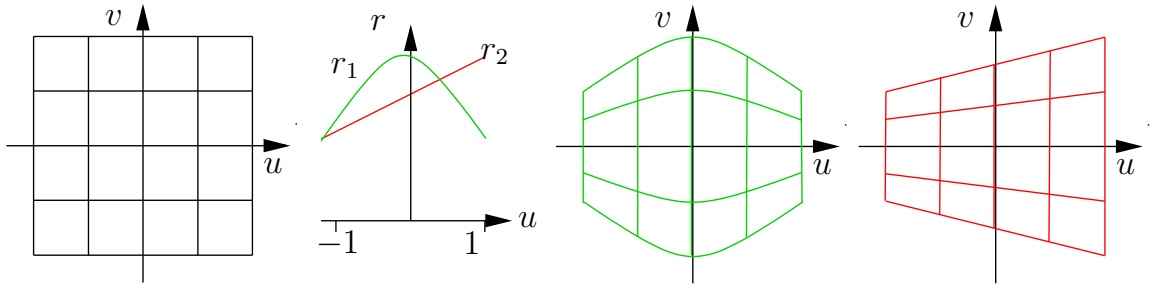
Vorgehen (2D):

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:



Ansatz: Twisting(Verdrehen), $3D$

Idee: Verdrehung abhängig von einer konstanten Koordinate

Vorgehen (3D):

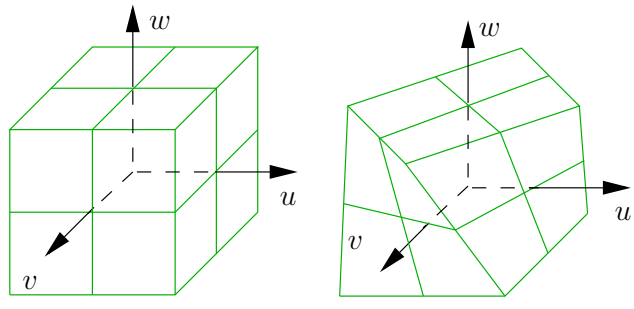
Konst. Koordinate: beispielsweise w

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



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) & Winkelextrema ϕ_{min}, ϕ_{max}

Biegewinkel (u-Parameter):

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

Biegeradius (v-Parameter):

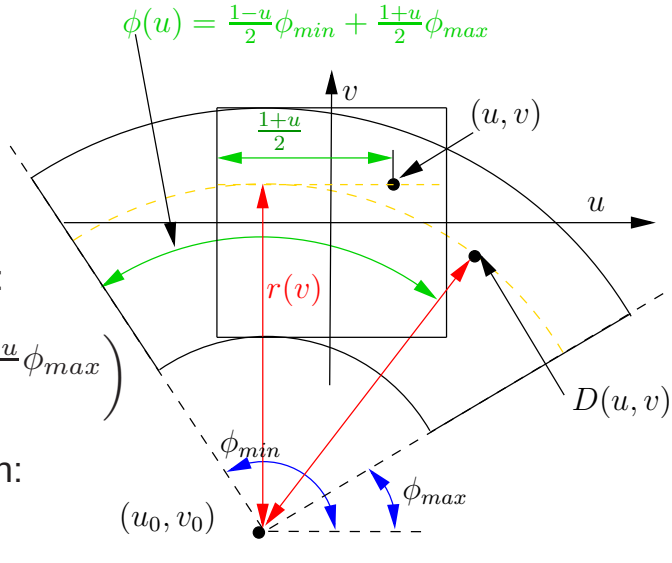
$$r(v) = v - v_0$$

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:

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

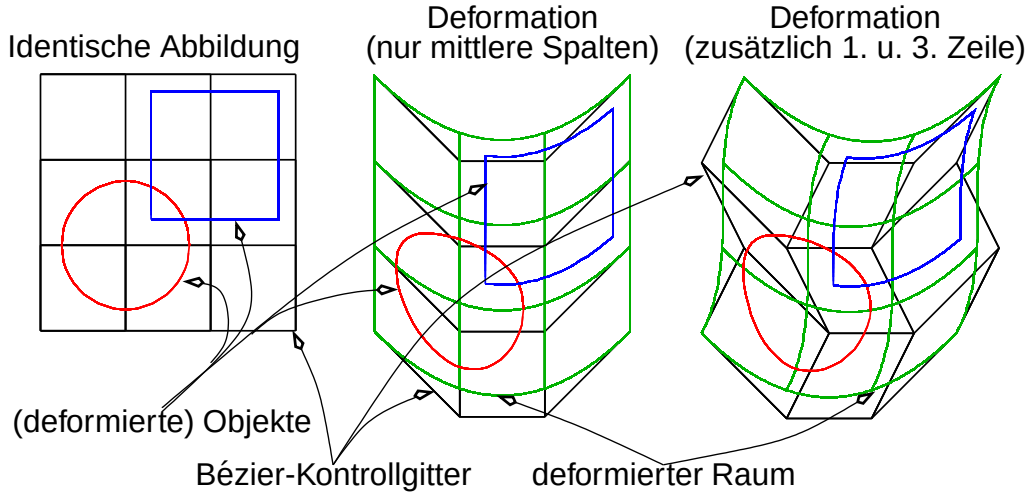


Freiform Deformation, $2D$

Ziel: Freie Verzerrung des Raumes

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

Deformation: Veränderung der Kontrollpunkte in der Ebene definiert Deformation



9.3.1 Deformationen ...

Freiform-Deformation (Forts.)

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

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 $\mathbf{C}_{ij},\ i,j = 0,\ldots,3$ für kubische Bézier-Fläche:

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

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

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

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

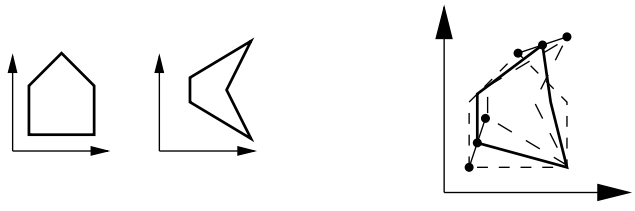
Idee: Direkte Anwendung der Interpolationstechnik auf Geometrien

$\Rightarrow$ Fließender Übergang zwischen den Formen

Grundsätzlicher Ansatz

Ausgangssituation: Kontrollpunkte $\mathbf{P}_i$ $i = 1, \ldots, k$ eines Objektes die zu Zeitpunkten $t_1, \ldots, t_n$ bekannt sind: $\mathbf{P}_i(t_1), \ldots, \mathbf{P}_i(t_n),\ i = 1, \ldots, k$

Vorgehen: Interpolation liefert $\mathbf{P}_i(t)$ für beliebiges t, womit die Geometrie zu jedem Zeitpunkt bestimmbar ist.



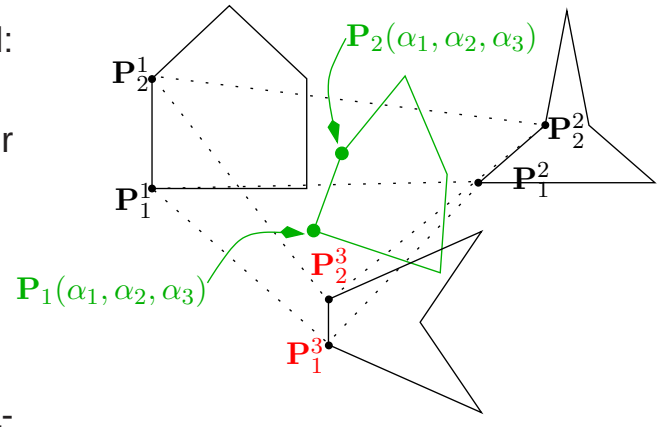
Morphing mit Affin-Kombinationen

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

Ausgangssituation: Kontrollpunkte $\mathbf{P}_i$ eines Objektes die für *Key-Positionen* $1, \ldots, k$ bekannt sind: $\mathbf{P}_i^1, \ldots, \mathbf{P}_i^k$.

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

$$\mathbf{P}_i = \mathbf{P}_i(\alpha_1, \ldots, \alpha_k) = \sum_{j=0}^{k} \alpha_k \mathbf{P}_j$$

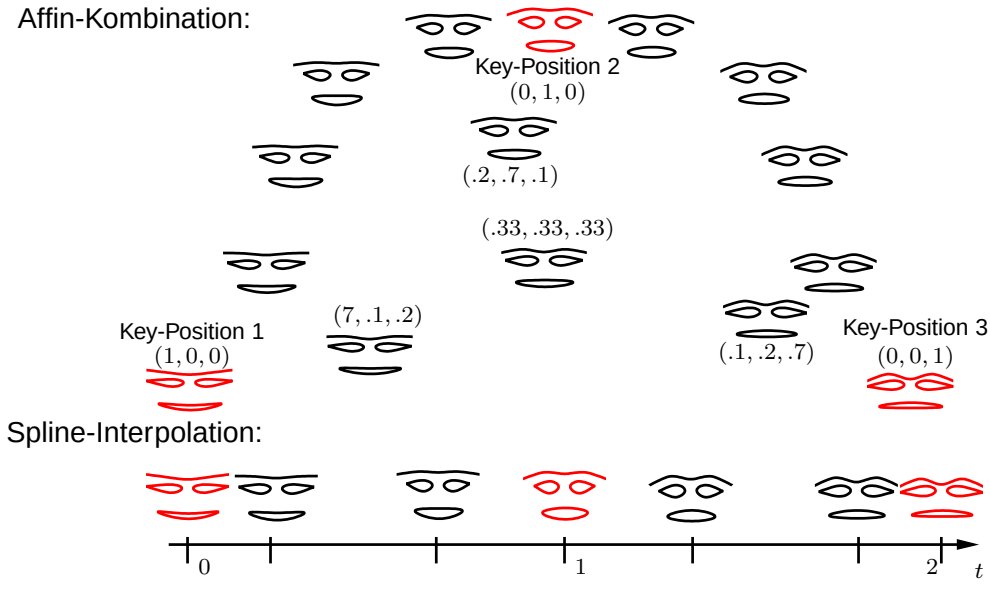


Animation: Variiere die Gewichtsparameter α_k über die Zeit: $\alpha_i(t)$

Morphing-Beispiel

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

Animation: *Veränderung der Gewichte* über die Zeit



9.3.2 Morphing ...

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 $\mathbf{C}_{ij}^{\text{init}}$ (initial), $\mathbf{C}_{ij}^{\text{end}}$
2. Zwischenbild (Zeit t) durch Interpolation von Kontrollpunkten & Farbe:

$$\mathbf{C}_{ij}^{\text{t}} = (1-t) \cdot \mathbf{C}_{ij}^{\text{init}} + t \cdot \mathbf{C}_{ij}^{\text{end}} \Rightarrow D^t$$
$$\mathbf{I}(D^t(x,y)) = (1-t) \cdot \mathbf{I}(D^0(x,y)) + t \cdot \mathbf{I}(D^1(x,y))$$