# Computer Graphics II
# 7: Computer Animation

Computer Graphics and
Multimedia Systems Group

University of Siegen



## 7: Computer Animation



### Structure of Chapter

- *Subsection 1: Keyframe Animation*
  - *Subsection 1: Interpolation of the Position*
  - *Subsection 2: Representation of Orientations*
  - *Subsection 3: Quaternions*
  - *Subsection 4: Interpolation of Orientation*
- *Subsection 2: Spline-Based Animation*
  - *Subsection 1: Path and Speed*
  - *Subsection 2: Form Control*
  - *Subsection 3: Camera Animation*
- *Subsection 3: Deformations and Morphing*
  - *Subsection 1: Freeform Deformations*
  - *Subsection 2: Blend Shapes and Morphing*

# 7: Computer Animation

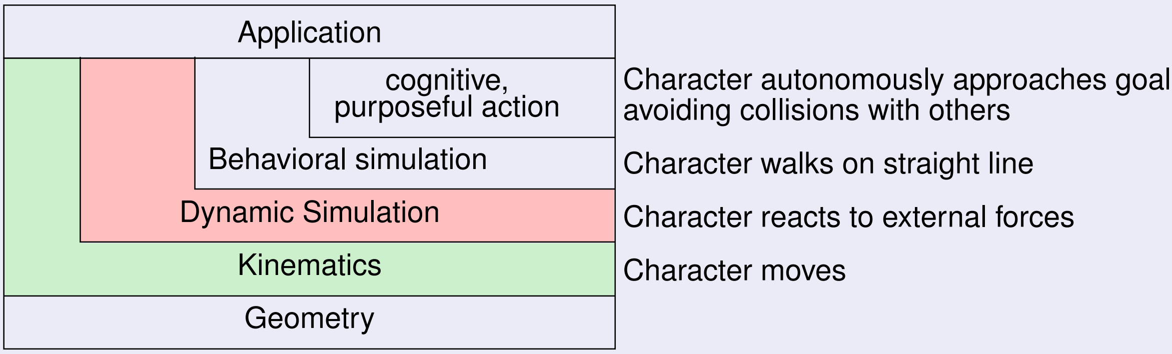


## Motivation

**Animation =** *"objects changing in time", e.g.*

1. *global position and orientation*
2. *Motion/change of the object in itself and in relation to other objects, resp.*

**Hierarchy of approaches to motion-control:**




**Kinematics:**
- *Study of motion (physics); motion = direct modification of geometry*
- *Application dependent time-steps ⇒ geometry must be determinable at each point in time.*



# General Considerations



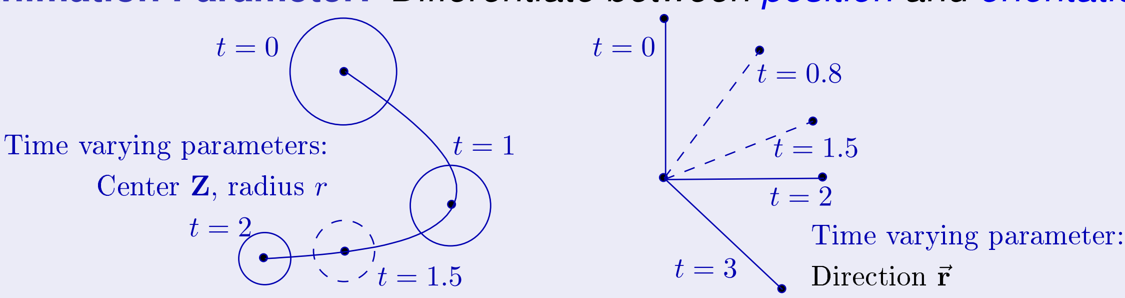
## Notation

**Compared to Animated Cartoons:** *Interpolation of given keyframes:*

| | Cartoon Animation | Computer Animation |
|---|---|---|
| keyframe | artist/choreographer | animator |
| in-betweens | drawing-artist | algorithm |

**Basic Procedure:**
1. *Description of the scene $\mathcal{O}$ by time-dependent parameters (animation parameters) $\phi_1, \phi_2, \dots, \phi_m$*
2. *Interval Point $t$: Interpolation of Parameters $\mathcal{O}(\phi_1(t), \dots, \phi_m(t))$*

**Animation Parameter:** *Differentiate between position and orientation*




**Orientation:** *Results from a rotation with reference to a starting position*

## 7.1.1: Position Interpolation

### Approach

**The interpolation of position parameters** *is carried out on the basis of suitable curve classes.*

**Specific Task:**

*Given:* *Time-position-pairs* $(t_i, \mathbf{P}_i),\ i = 1, \dots, N$

*Wanted:* *Curve* $\mathbf{C} : \mathbf{C}(t_i) = \mathbf{P}_i,\ i = 1, \dots, N$

**Catmull-Rom Splines:**

- $C^1$*-continuous cubic polynomial curves*
- *heuristic determination of the tangent (manual adjustment if necessary)*

**Alternative:** *Using a B-Spline, which interpolate values at the knots* $t_i$

- *requires the solution of a* $N \times N$ *linear system*
- *piecewise cubic polynomial curves*
- $C^2$*-continuous curves*





## Euler Angles

### Definition (Euler Angles)

**Sequential rotation** around $x-, y-$ & $z-$axis in *local coordinates*
*Pitch, Yaw, Roll* Rotation $R_x(\phi_x), R_y(\phi_y), R_z(\phi_z)$ about $x-, y-, z-$ axis

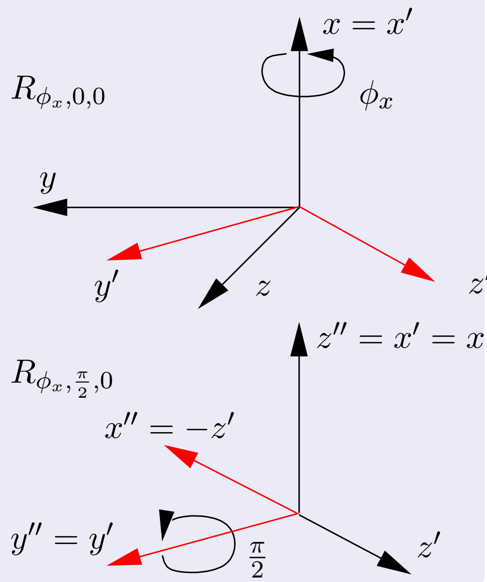
**Sequence:** Influences the result $\Rightarrow$ constant sequence, e.g. $x, y, z$.

**Full Rotation** for processing sequence $x, y, z$ as matrix multiplication:

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

**Ambiguity problem:** e.g. *Gimbal-lock*

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



# Ambiguity of Euler Angles



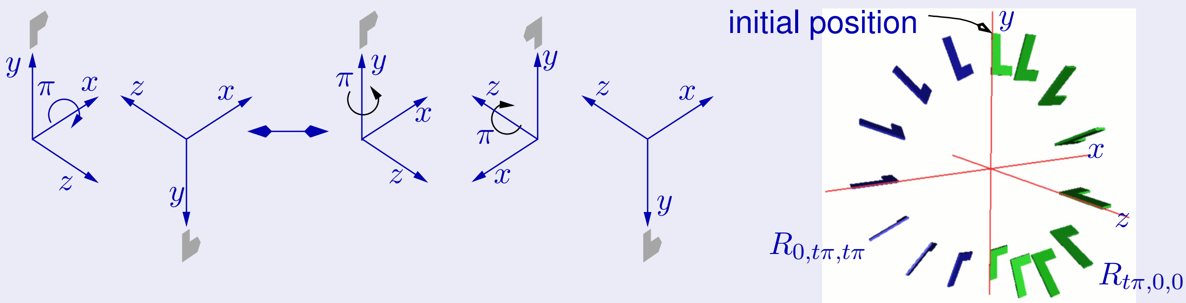
## Property

**Note:** *For animation the representation of the orientation must be unambiguous*

**Because:** *otherwise "random" interpolation results arise*

**Example:**
- *$R_{\pi,0,0}$ and $R_{0,\pi,\pi}$ describe the same orientation*
- *Interpolation paths between $R_{0,0,0}$ and $R_{\pi,0,0} = R_{0,\pi,\pi}$ are different*

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




**Finding:** *Euler angles are unsuitable representation for animation*



# Exkursion: Complex Numbers



## Definition (Complex Numbers)

**General:** Complex numbers extend real numbers, so that the root of negative numbers is defined.

**Imaginary Unit:** $i := \sqrt{-1}$ as the solution of $x^2 = -1$

**Representation of complex numbers:** $c = x + i \cdot y$, with $x, y \in \mathbb{R}$
The real part of $c$: $\mathfrak{Re}(c) = x$, the imaginary part of $c$: $\mathfrak{Im}(c) = y$

**Set symbol:** The set of complex numbers is called $\mathbb{C}$

**Calculation rules** are valid as usual, considering $i^2 = -1$

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

**Multiplication:**

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

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

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

*Addition und multiplication of complex numbers is commutative!*

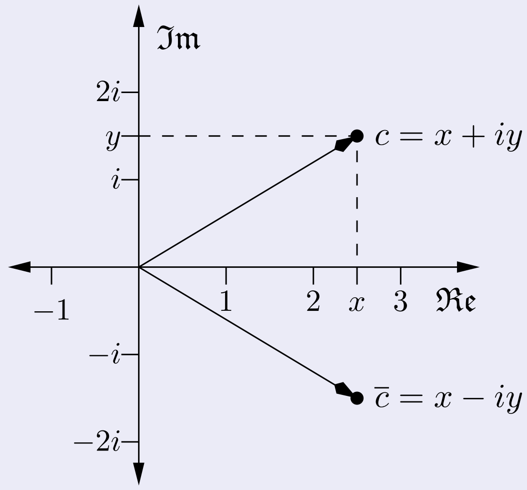
## Property (Complex Plane)

**Complex plane:** *One can imagine the complex numbers as points in the complex plane*(dt: Gau"s'sche Zahlenebene) *plane:*

- *"Basis vector" are* $1$ *and* $i$

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

**Length:** *Analogous to the vector length,* $|c|^2 = x^2 + y^2$




**Conjugation:** *"Reflection" of the vector at the real axis*

$$c = x + i \cdot y \Rightarrow \bar{c} := x - i \cdot y \text{(complex conjugate)}$$

*This results in:* $c \cdot \bar{c} = (x + iy) \cdot (x - iy) = x^2 - (iy)^2 = x^2 + y^2 = |c|^2$



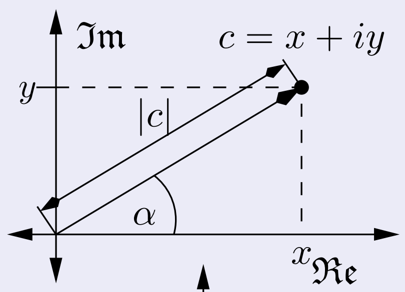


## Remark

**Each** $c = x + iy \in \mathbb{C}$ *can be written as:*
$c = r \cdot \cos\alpha + r \cdot i \sin\alpha,$ *with* $r = |c|, \ \alpha \in [0, 2\pi[$

**Because** *of* $\cos^2\alpha + \sin^2\alpha = 1$ *(Pythagoras) is*

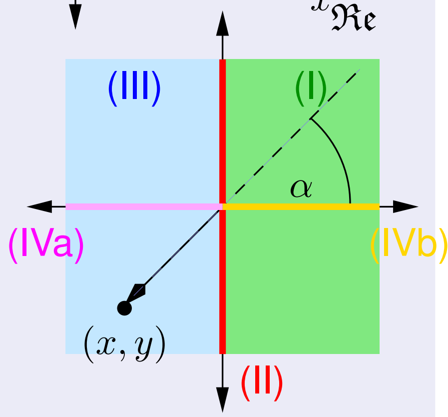
$$\begin{aligned} |c|^2 &= r^2\cos^2\alpha + r^2\sin^2\alpha \\ &= r^2(\cos^2\alpha + \sin^2\alpha) = r^2 \end{aligned}$$




**Calculation:** *For* $c = x + iy$ *is* $r = \sqrt{x^2 + y^2}$ *and*

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

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




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

## Rotation with Complex Numbers

### Approach

**Consider** *the following subset:* $\mathcal{Z} = \{c \ : \ |c| = 1\}$

- *Valid is:* $1 = |c| = x^2 + y^2 \Rightarrow \exists_1 \alpha \in [0, 2\pi[ : \cos\alpha = x \wedge \sin\alpha = y$
- *For* $c_1, c_2 \in \mathcal{Z}$ *with* $c_i = \cos\alpha_i + i \cdot \sin\alpha_i$ *is (addition theorems)*

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

**Rotation** *around* $\phi$ *using* $c_\phi = \cos\phi + i \cdot \sin\phi \in \mathcal{Z}$*:*

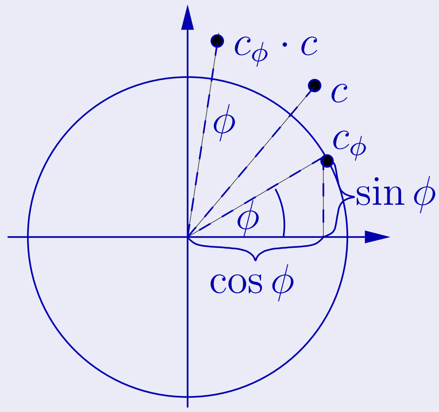
$$R_\phi : \begin{array}{ccc} \mathbb{C} & \longrightarrow & \mathbb{C} \\ c & \mapsto & R_\phi(c) = c_\phi \cdot c \end{array}$$

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

**Exponential Notation** *of polar coordinates:*

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

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








## Exponential Notation for Polar Coordinates

### Remark

**Functions** *like* $\sin, \cos, \exp$ *are defined by* ***series****(dt: Reihen):*

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

*Thus complex parameters can be inserted into this function without any problem.*

**Insertion** *of* $i\phi$ *as parameter into the exponential function produces:*

$$e^{i\phi} = \sum_{k=0}^{\infty} \frac{(i\phi)^n}{n!} = \sum_{k=0}^{\infty} \frac{(i\phi)^{2k}}{(2k)!} \text{(even expon.)} + \sum_{k=0}^{\infty} \frac{(i\phi)^{2k+1}}{(2k+1)!} \text{(odd Expon.)}$$
$$= \sum_{k=0}^{\infty} \frac{(i^2)^k \cdot \phi^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{i(i^2)^k \cdot \phi^{2k+1}}{(2k+1)!} = \sum_{k=0}^{\infty} \frac{(-1)^k \phi^{2k}}{(2k)!} + i \sum_{k=0}^{\infty} \frac{(-1)^k \phi^{2k+1}}{(2k+1)!}$$
$$= \cos\phi + i \sin\phi$$

## Algorithm

**Given:** *Point* $\mathbf{P} = \binom{x}{y} \in \mathbb{R}^2$ *in the plane and rotation angle* $\phi$

**Execution** *of the rotation*

1. *Interpret* $\mathbf{P}$ *as a complex number in* $\mathbb{C}$: $z = x + iy$
2. *Rotate* $z$ *in* $\mathbb{C}$: $z' = z \cdot e^{i\phi}$
3. *Interpret* $z'$ *again as a point in* $\mathbb{R}^2$: $\binom{x'}{y'} \in \mathbb{R}^2$

**Classic Procedure** *with matrices in 2D:* $\binom{x'}{y'} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \binom{x}{y}$

*Rotation with complex numbers in 2D offers no advantages here*

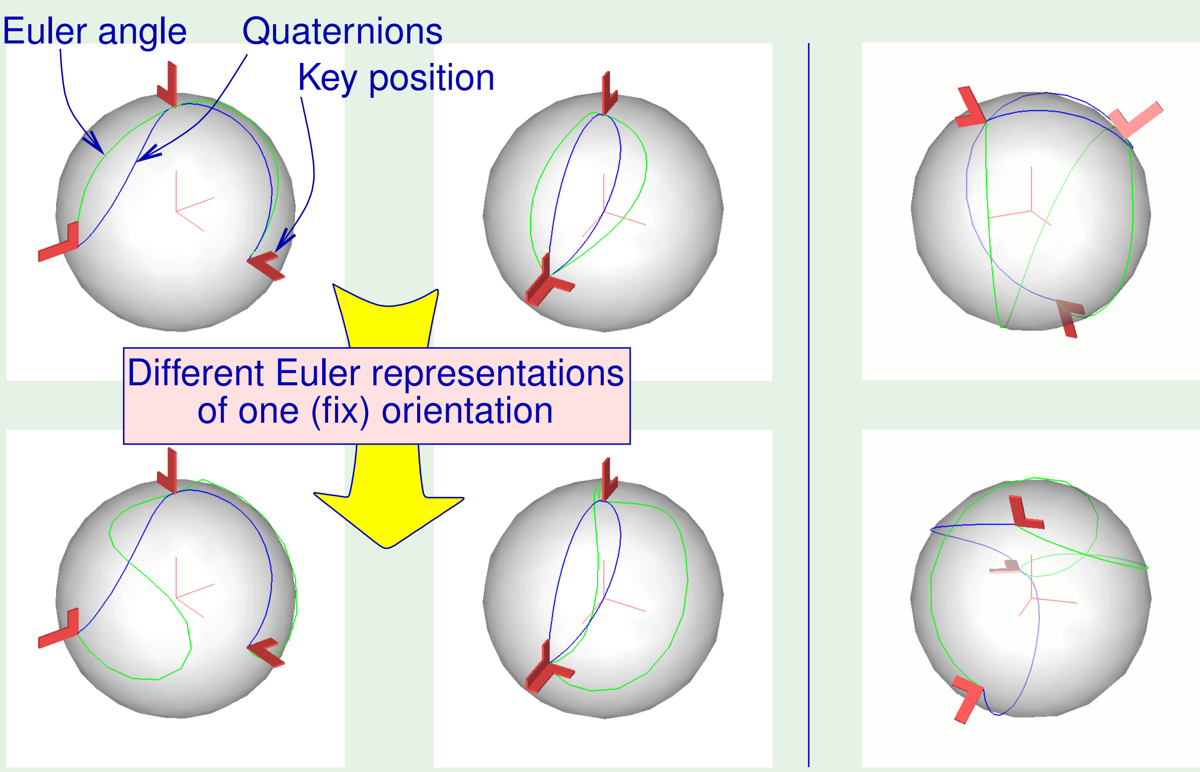
**...but:** *Sir William Hamilton (1805-1865) found out:*

- *an extension of complex numbers to* $3D$ *is not possible*
- *wheras a generalization for* $2n$*-dimensional spaces exist*
- *Quaternions are generalized complex numbers for* $4D$*-rotations*





## Example



## Definition (Quaternions)

**Quaternions** have one real and three imaginary components $s$ and $x, y, z$, resp.

**Notation:** With imaginary units $i, j, k$:

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

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

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

**Multiplication:** From application of the above rule follows:

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

**Asymmetry** $ij = -ji, jk = -kj, ki = -ik \Rightarrow$ multipl. it *not commutative*.





## Property (Quaternions)

**Conjugate:** $\overline{\underline{\mathbf{q}}} := (s, -\vec{\mathbf{v}})$; *conjugation is not commutative:*

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

**Length:**

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

**Unit Quaternion:** *Quaternion with a length of* $1$, *i.e.* $\left|\underline{\mathbf{q}}\right| = 1$

**Angle:** *For* $\left|\underline{\mathbf{q}}_1\right| = \left|\underline{\mathbf{q}}_2\right| = 1$ *we get:*

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

*This corresponds to the inner product of the respective 4D-vectors* $(s_1, x_1, y_1, z_1)$ *und* $(s_2, x_2, y_2, z_2)$

### Approach

**Problem:** *Unit quaternions describe rotations in 4D*

**Questions** *when using quaternions in 3D*

- *How is 3D embedded into 4D?*
- *How can 4D-rotations be transferred to 3D?*

**The Following Quaternion** *represents 3D-rotation (axis $\hat{\mathbf{v}}$, angle $\phi$):*

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

**Computation of the Rotation** *for a given point $\mathbf{P} \in \mathbb{R}^3$:*

1. *Transformation of $\mathbf{P}$ in 4D: $\underline{\mathbf{q}}_{\mathbf{P}} = (0, \vec{\mathbf{p}})$, $\vec{\mathbf{p}}$ is the position vector of $\mathbf{P}$*
2. *Rotation computation: $\underline{\mathbf{q}}'_{\mathbf{P}} = \underline{\mathbf{q}}\underline{\mathbf{q}}_{\mathbf{P}}\overline{\underline{\mathbf{q}}}$*
3. *Interpretation in 3D: $\underline{\mathbf{q}}'_{\mathbf{P}} = (0, R_{\hat{\mathbf{v}},\phi}(\mathbf{P}))$, i.e. $s' = 0$, $R_{\hat{\mathbf{v}},\phi}(\mathbf{P}) = (x', y', z')$*

**Composition** *of the rotations corresponds to the quaternion multiplication:*

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





### Example

**Rotation** of $\mathbf{P} = (1, 2, 0)$ by $\hat{\mathbf{v}} = (1, 0, 0)$ with angle $\pi$ (result: $(1, -2, 0)$).

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

**Calculation** of $R_{\underline{\mathbf{q}}}(\mathbf{P})$ in two steps:

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

**Result:** Quaternion $(0, (1, -2, 0))$ corresponds to the point $(1, -2, 0)$

**Note:** The resulting quaternion must always have a real part of $0$.

## Observation

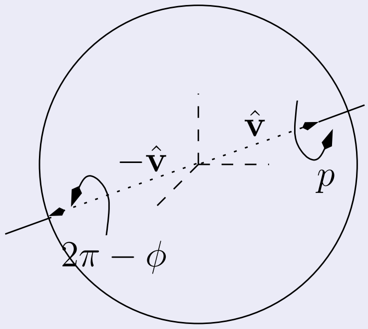
**Obviously** $\underline{\mathbf{q}}$ *and* $-\underline{\mathbf{q}}$ *produce the same result:*

1. *Multiplication with regard to real factors is commutative, e.g.* $a \cdot (s, \vec{\mathbf{v}}) \cdot b = (ab\, s, ab\vec{\mathbf{v}}) = ab(s, \vec{\mathbf{v}})$
2. *Thus:* $R_{\underline{\mathbf{q}}}(\mathbf{P}) \hat{=} \underline{\mathbf{q}}\underline{\mathbf{q}}_{\mathbf{P}}\overline{\underline{\mathbf{q}}} = (-1)^2 \underline{\mathbf{q}}\underline{\mathbf{q}}_{\mathbf{P}}\overline{\underline{\mathbf{q}}} = (-\underline{\mathbf{q}})\underline{\mathbf{q}}_{\mathbf{P}}(-\overline{\underline{\mathbf{q}}}) \hat{=} R_{-\underline{\mathbf{q}}}(\mathbf{P})$

**Analysis:** $-\underline{\mathbf{q}} = (-\cos\frac{\phi}{2}, -\sin\frac{\phi}{2}\hat{\mathbf{v}})$ *corresponds to rotation around* $-\hat{\mathbf{v}}$ *with an angle of* $2\pi - \phi$*, since*

$$\cos(\pi - \tfrac{\phi}{2}) = -\cos(\tfrac{\phi}{2}) \text{ and } \sin(\pi - \tfrac{\phi}{2}) = \sin(\tfrac{\phi}{2})$$

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




**Ambiguity:** *Both rotations deliver the same result! There are no further ambiguities!*





## Example (Ambiguity with Euler Angles)

**Reminder:** For Euler angles applies: $R_{\pi,0,0} = R_{0,\pi,\pi}$

**Question:** What does to look like for quaternions?

$$\begin{aligned}
R_{\pi,0,0} &\hat{=} R_{\underline{\mathbf{q}}_x} \text{ with } \underline{\mathbf{q}}_x = (0,(1,0,0)), \text{ since } \sin\frac{\pi}{2} = 1, \cos\frac{\pi}{2} = 0 \\
R_{0,\pi,0} &\hat{=} R_{\underline{\mathbf{q}}_y} \text{ with } \underline{\mathbf{q}}_y = (0,(0,1,0)), \quad R_{0,0,\pi} \hat{=} R_{\underline{\mathbf{q}}_z} \text{ with } \underline{\mathbf{q}}_z = (0,(0,0,1)) \\
R_{0,\pi,\pi} &\hat{=} R_{\underline{\mathbf{q}}_z} \cdot R_{\underline{\mathbf{q}}_y} = R_{\underline{\mathbf{q}}_z\underline{\mathbf{q}}_y} = R_{-\underline{\mathbf{q}}_x}
\end{aligned}$$

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

which defines the same orientation as $R_{\underline{\mathbf{q}}_x} = (0,(1,0,0))$

**Result:** $R_{\pi,0,0}$ and $R_{0,\pi,\pi}$ correspond to the same unit quaternion $\underline{\mathbf{q}} = (0,(1,0,0))$

### Algorithm

**Conversion** *of the unit quaternion* $\underline{\mathbf{q}} = (\cos\frac{\phi}{2}, \sin\frac{\phi}{2}\hat{\mathbf{v}}) = (s,(x,y,z))$ *into the rotation matrix* $R$*:*

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

**Conversion** *of the rotation matrix* $R = (r_{ij})_{i,j=1,\ldots,3}$ *in unit quaternion. From the eq.* (1) *we get*

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

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

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

**Notice:** *Positive and accordingly negative* $s$ *deliver two quaternions* $\underline{\mathbf{q}}^+, \underline{\mathbf{q}}^-$, *but* $\underline{\mathbf{q}}^- = -\underline{\mathbf{q}}^+$ *describe the same orientation!*





### Problem

**Goal:** *Interpolation of orientations, in Euler notation:*

*Given: Time-orientation-pairs* $(t_i, R_{\vec{\phi}^i})$, $\vec{\phi}^i = \left(\phi_x^i, \phi_y^i, \phi_z^i\right)$, $i = 1,\ldots,N$

*Wanted: Interpolation function* $R$ *with* $R(t_i) = R_{\vec{\phi}^i}$, $i = 1,\ldots,N$

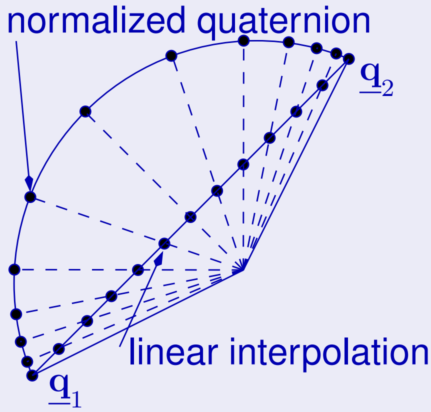
**Interpolation with Euler Angles:**
- *Direct interpolation e.g. with B-splines*
- *Problem: Result depends on the representation of the orientation.*

**Interpolation with Quaternions:**

**Reminder:** *Rotations are described by* $4D$ *unit quaternions*

**Problem:** *Interpolated quaternions must be again unit quaternions.*
*Bézier, B-Splines do not achieve this, i.e.:*
$\mathbf{C}(t_i) \in \mathbb{R}^4, \|\mathbf{C}(t_i)\| = 1 \not\Rightarrow \|\mathbf{C}(u)\| = 1.$

**Normalization** *delivers non-uniform step size*



# Spherical Linear Interpolation (slerp)



## Approach (Spherical Linear Interpolation)

**Goal:** *"Linear" interpolation on a sphere resp. a great circle arc*

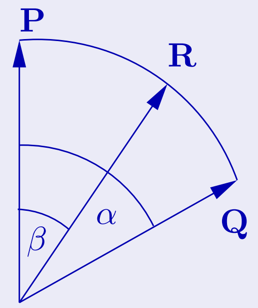
**Given:** $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ *on the unit circle (i.e. on the great circle),* $\mathbf{R}$ *"between"* $\mathbf{P}$ *and* $\mathbf{Q}$*:* $\alpha = \angle(\mathbf{P}, \mathbf{Q}), \beta = \angle(\mathbf{P}, \mathbf{R})$.

**Solution:** *Relation between angle and point:*

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

*Therefore, for* $\beta = t\alpha,\ t \in [0, 1]$ *we get*

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




**Linear Interpolation of Rotations** *by spherical interpolation of unit quaternions:*

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



# 7.1.4: Interpolation of Orientation



## Remark (Details to Interpolation on circular arcs)

**Given:** $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ *points on the unit circle,* $\mathbf{R}$ *"between"* $\mathbf{P}$ *and* $\mathbf{Q}$, $\alpha = \angle(\mathbf{P}, \mathbf{Q})$

**Assumption:** *Points lie in the plane and* $\mathbf{P} = (1, 0)$
$\Rightarrow \mathbf{Q} = (\cos\alpha, \sin\alpha)$ *and* $\mathbf{R} = (\cos\beta, \sin\beta)$

**Wanted:** *Linear combination* $\mathbf{R} = a\mathbf{P} + b\mathbf{Q}$

**Verify,** *whether* $a = \frac{\sin(\alpha-\beta)}{\sin\alpha}$ *and* $b = \frac{\sin(\beta)}{\sin\alpha}$ *"do the job"*

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

## Algorithm

**Evaluation** *of Bézier- and B-spline curves is based on successive affine combinations from control points (de Casteljau, de Boor)*

**Question:** *Is this also true for* Catmull-Rom-Splines *with the interpolation points of* $\mathbf{P}_i$, *i.e. are Bézier-control points affine combinations of the* $\mathbf{P}_i$*?*

**Specifically:** $i$*-th Bézier-segment with control points* $\mathbf{C}_0^i, \ldots, \mathbf{C}_3^i$ *and interpolation points* $\mathbf{P}_{i-1}, \ldots, \mathbf{P}_{i+2}$*:*

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

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

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

| |
|---|
| *Evaluation of Catmul-Rom splines involves only affine combinations* |

**Spherical Catmull Rom Spline:** *Use slerp-operation instead of affine combinations yields a curve that proceeds on the sphere.*





## Approach

**Given:** *A quaternion sequence that is to be interpolated:* $\underline{\mathbf{q}}_1, \ldots, \underline{\mathbf{q}}_n$

**Question:** *Which representative of* $R_{\underline{\mathbf{q}}_i}$ *is to be chosen,* $\underline{\mathbf{q}}_i$ *or* $-\underline{\mathbf{q}}_i$*?*

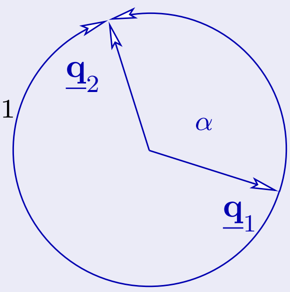
**Analogy to the Unit Circle:** *The smaller angle represents the "short" distance.*

**Rule:** *Adopt the quaternion* $\underline{\mathbf{q}}_{i+1}$ *if*

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

*in sequential manner* $i = 1, i = 2, \ldots, i = n-1$

**Result:** *The resulting sequence has the property* $\angle(\underline{\mathbf{q}}_i, \underline{\mathbf{q}}_{i+1} \leq \tfrac{\pi}{2}$



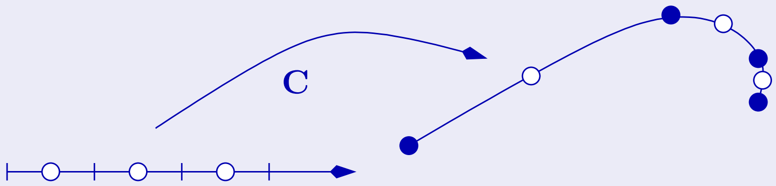
## Approach (Spline Based Animation)

**Keyframe Animation:** *Parameter states defined at key-times, calculation of interpolation functions (splines)*

**Spline Based:** *Direct defintion and manipulation of spline curves*

**Questions:**

1. *Where is spline based animation useful?*
   - **+:** *Motion of a body through space (intuitive trajectory)*
   - **–:** *The fingertip of a swinging arm of a walking character in world space (no intuitive motion path)*
2. *How does time-reference come into play (no more keyframe!)?*
3. *Important: Uniform sampling of $u$ does not create uniform motion along a curve*
   $\Rightarrow$ *Goal: Positioning along the curve by means of the path length*








**Goal:** Motion of an object along a curve

**Spatial Motion Path** of the object

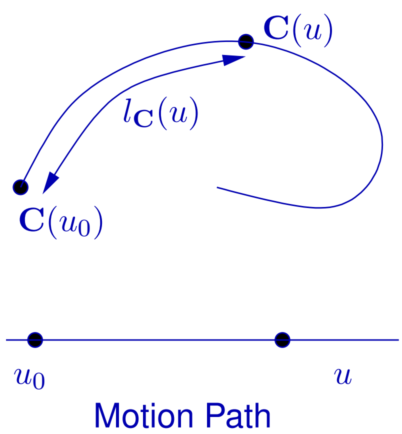
$$\mathbf{C}(u) : \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R}^3 \\ u & \mapsto & (x(u), y(u), z(u)) \end{array}$$

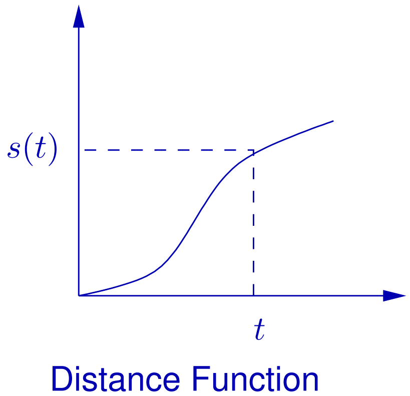
**Distance Function:** Distance travelled on the path at time $t$: $s(t) : \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ t & \mapsto & s(t) \end{array}$

**Goal:** Motion of the object along $\mathbf{C}$, so that until time $t$ the distance $s(t)$ has been travelled.

**Arc Length Function** $l_{\mathbf{C}}(u)$ measures the length of the path $\mathbf{C}$ from starting point $\mathbf{C}(u_0)$ until $\mathbf{C}(u)$

**Distinguish:** Modeling parameters $u$ (motion path) and time $t$ (distance function)



## Algorithm

**Given:** *Motion path $\mathbf{C}(u)$ with start and end parameters $u_0$ and $u$, resp.*

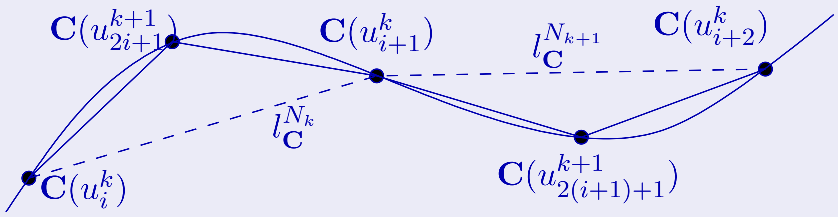
**Estimation** *of the arc length $l_{\mathbf{C}}(u)$ using the length of a polygonal line*

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

**Exact Solution** *via limit, i.e. infinite many sample points*

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

**Implementation:** *Generally integral not solvable $\Rightarrow$ estimation with $N_k = 2^k$*




*Abort, in case of small accuracy gain:* $l_{\mathbf{C}}^{N_{k+1}} - l_{\mathbf{C}}^{N_k} < \varepsilon$





## Approach (Determination of Parameter $u$ (Fixed Path-Curve))

**Observation:** *The computation of $l_{\mathbf{C}}$ is dominated by the evaluation of $\mathbf{C}$*

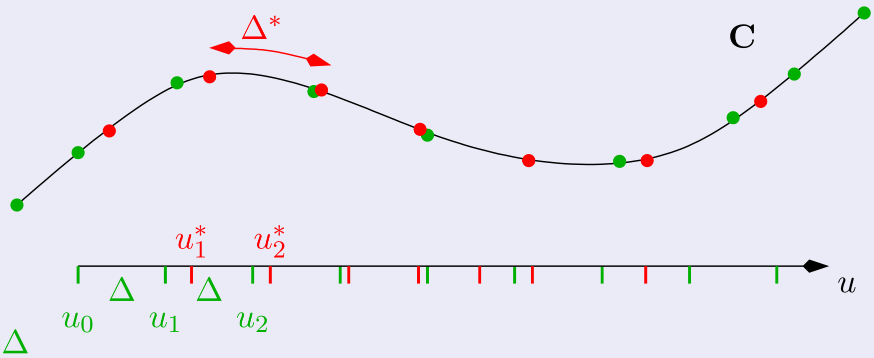
**Approach for a fixed path-curve:** *Storage of $l_{\mathbf{C}}(u)$ in Look-Up Table*

1. *Storage of $l_{\mathbf{C}}(u_i)$ for equidistant parameters $u_i$*
2. *Storage of $u_i^*$ for equidistante arc length*
   *$\Rightarrow$ more efficient access to desired $u$-parameter*

**Hint:** *Calculation of $u_i^*$ by bisection via the values $u_i, l_i$*

Variante 1:

| | |
|---|---|
| $u_0$ | $0$ |
| $u_1$ | $l_1$ |
| $u_2$ | $l_1 + l_2$ |
| | |




Variante 2:

| | |
|---|---|
| $0$ | $u_0^*$ |
| $l_1^*$ | $u_1^*$ |
| $l_2^*$ | $u_2^*$ |
| | |

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

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

# Parameter Determination for a Curve-Point



### Approach (Determination of the parameter $u$ (general case))

**Task:** *For the given path function $s$ and time $t$: For which parameter value $u$ does $l_{\mathbf{C}}(u) = s(t)$ apply*

**Approach:** *Seek root of $g(u) = l_{\mathbf{C}}(u) - s(t)$*
*Note: $l_{\mathbf{C}}(u)$ is monotonically increasing $\Rightarrow$ unique root of $g(u)$.*
**Newton's Method** *determines root of $g(u)$:*

*Initial: Parameter $u_0$*

*Iteration: Determine tangent at $g(u_i)$:*

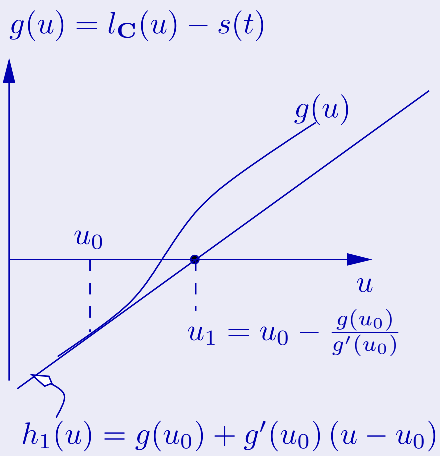
$$h_i(u) = g(u_i) + g'(u_i)(u - u_i)$$

$u_{i+1}$ *is root of* $h_i(u)$:

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

*since*

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






# 7.2.2: Form Control



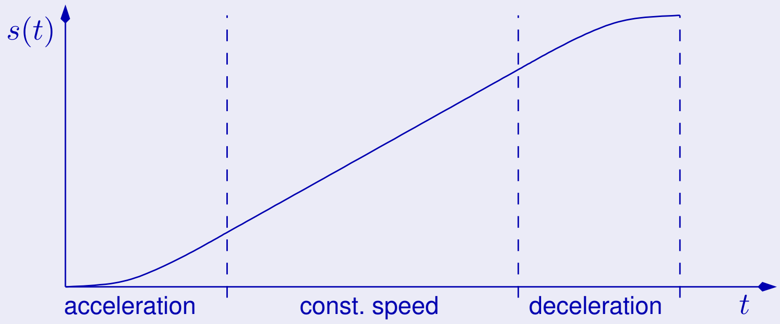
### Remark

**So Far:** *Control of the curve progression limited to*

**Bézier-curves/B-spline:** *Placement of control points*
**Bézier-splines/Catmull-Rom splines:** *Placement of interpolation points and choice/estimation of tangents*

**Problem:** *Many applications require other means of control, e.g. the definition of a* path function

**Example:** *Constant acceleration - constant speed - constant delay*




**Goal:** *Improved possibilities for animation-spezific cpntrol*

# Ease-In/Ease-Out Functions



## Objective

*Definition of an Ease-In/Ease-Out function:*

$$\mathrm{ease}(t) = \begin{cases} \text{“soft acceleration”} & \text{for } t \in [0, \frac{1}{2}] \\ \text{“soft deceleration”} & \text{for } t \in [\frac{1}{2}, 1] \end{cases}$$

## Approach (Sinus-Ease-Function)

**Concept:** *Sinus displays on* $[-\frac{\pi}{2}, \frac{\pi}{2}]$ *the desired behaviour*

**Definition:** *Adopted to* $t \in [0,1]$*:* $\mathrm{ease}_{\sin}(t) = \frac{1}{2}\sin(t\pi - \frac{\pi}{2}) + \frac{1}{2} \in [0,1]$

**Adoption** *to specific time interval and/or speed by scaling of parameters*

**Example:** *Acceleration from* $0$ *to* $v_0$ *in the period from* $t = 0$ *to* $t = t_0$*:*

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



# Parabolic Ease-Function



## Approach

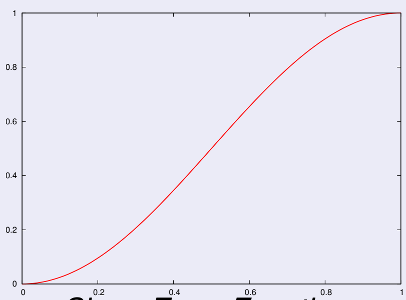
**Concept:** *Use a* constant acceleration, *i.e.*

*constant acceleration* ($a = 1$)$\Longrightarrow$ *linear velocity* $\Longrightarrow$ *quadratic path function*

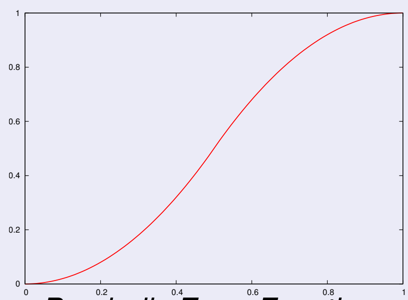
$$\mathrm{ease}_{parabol}(t) = \begin{cases} 2t^2 & \textit{falls } t \in [0, \frac{1}{2}] \\ 1 - 2(1-t)^2 & \textit{falls } t \in [\frac{1}{2}, 1] \end{cases}$$

**Notice:** *Adaptation of the acceleration/delay must consider that*
$(\mathrm{ease}_{\sin})'(0.5) = 1 \neq (\mathrm{ease}_{parabol})'(0.5) = 2$

**Comparison:**




*Sinus-Ease-Function*

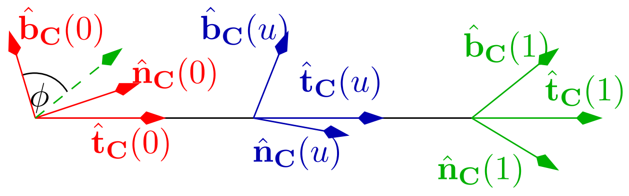



*Parabolic Ease-Function*

**Reminder:** Frenet-Serret frame definies local coordinates for curves (camera path)

**Problem:** Straight lines have no curvature, thus the Frenet frame is undefined.

**Interpolation of Frames** for $u \in [0, 1]$, if the frame for $u \in \{0, 1\}$ is known



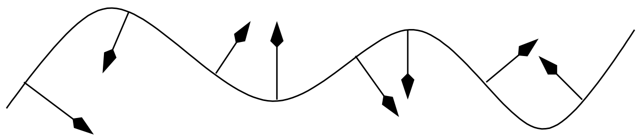

Tangent: $\hat{\mathbf{t}}_{\mathbf{C}}(u) = \hat{\mathbf{t}}_{\mathbf{C}}(0) = \hat{\mathbf{t}}_{\mathbf{C}}(1)$

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

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

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

**Problem:** Extreme motions, e.g. meanders like paths, exhibit many points of inflection(dt: Wendepunkte), which lead to flipped normals
Example of a curve with flipped normal vectores:



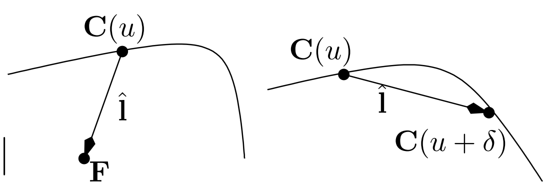




**Local coordinates:** Orthogonal vectores $\{\hat{\mathbf{l}}, \hat{\mathbf{u}}, \hat{\mathbf{w}} = \hat{\mathbf{l}} \times \hat{\mathbf{u}}\}$ with viewing direction $\hat{\mathbf{l}}$ and up-vector $\hat{\mathbf{u}}$ in the curve point $\mathbf{C}(u)$

**Determination of $\hat{\mathbf{l}}$:**

- Focal point $\mathbf{F}$:
  $\hat{\mathbf{l}} = (\mathbf{F} - \mathbf{C}(u)) / \|\mathbf{F} - \mathbf{C}(u)\|$
- Look-ahead direction (secant):
  $\hat{\mathbf{l}} = (\mathbf{C}(u+\delta) - \mathbf{C}(u)) / \|\mathbf{C}(u+\delta) - \mathbf{C}(u)\|$




focal point look-ahead

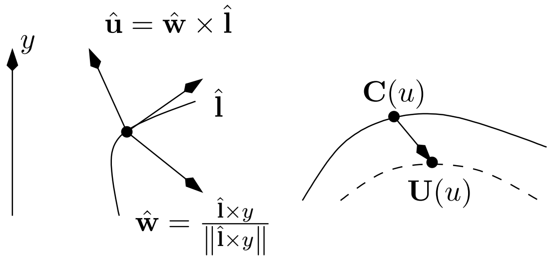
**Determination of $\hat{\mathbf{u}}$:**

- Global up-vector e.g. $y$-axis

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

- Up vector through 2nd path $\mathbf{U}(u)$

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

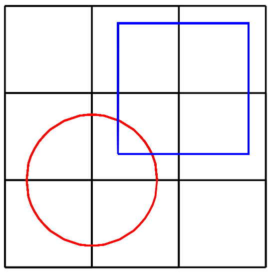



global up-vector up-vector path

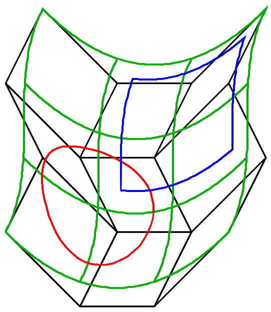
## Notation (Deformationen, Warping, Morphing)

**Deformation:** *Modification of complex objects (e.g. bending or twisting) by non-affine transformation of the surrounding space.*
*Image Warping refers to in-plane deformations of images.*

**Morphing:** *Combination of several objects (or modified variations of an object) resulting in mixed geometries/shapes.*
*Blend shapes: Morphing applied to several instances of a modified geometry.*

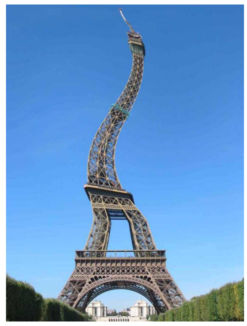


Original objects



Deformed objects
(black: deform. control mesh)
(green: deformatied space)

Original



Warped image





## Notation

**Deformations (also called space-warp)** *map a space onto itself:*

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

**Reference cuboid $Q$:** *Deformation $D$ is applied relative to a reference cuboid (respectively quad in 2D):*

$$Q = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \text{ in 2D resp. in 3D}$$
$$Q = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$$

**Normalized Coordinates:** *Transform $\mathbf{P} = (x, y, z) \in Q$ into $T_N(\mathbf{P}) = (u, v, w) \in [0,1]^3$:*

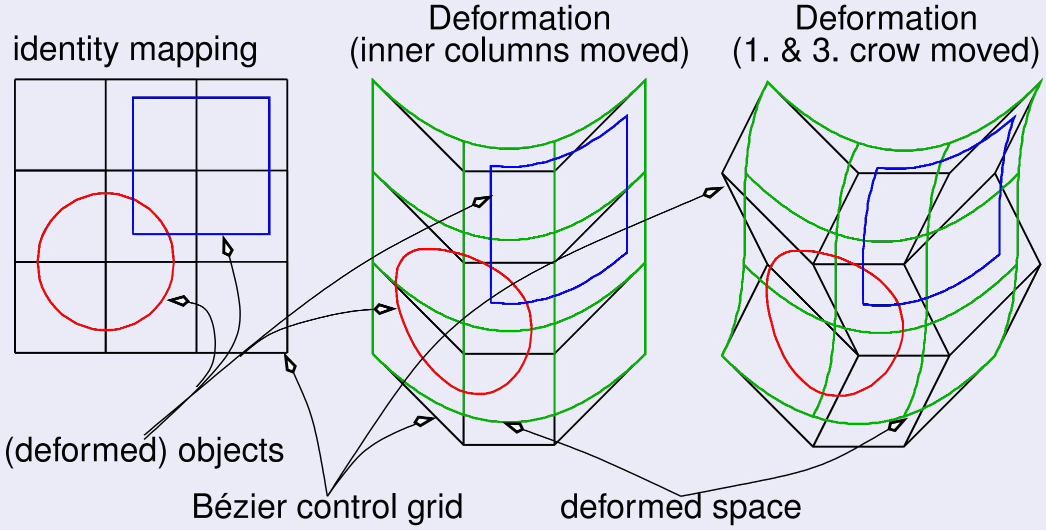
$$\mathbf{P} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in Q \Rightarrow T_N(\mathbf{P}) = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} (x - x_{min})/(x_{max} - x_{min}) \\ (y - y_{min})/(y_{max} - y_{min}) \\ (z - z_{min})/(z_{max} - z_{min}) \end{pmatrix} \in [0,1]^3$$

### Approach (General Idea of FFDs)

**Aim:** *Free distortion of space i.e. of objects in $Q$*

**Initially:** *Map the reference cuboid $Q$ identical to itself using a planar Bézier surface $\mathbf{C}(u, v)$.*

**Deformation:** *Modification of the control points in the plane defines the deformation*








### Approach (Setting the Initial Control Points)

**Goal:** *Set (2D) Bézier control points $\mathbf{C}_{ij}$, so that no deformation occurs, i.e.*

$$D(\mathbf{P}) = D(x, y) = \sum_{i,j=0}^{n} \mathbf{C}_{ij} B_i^n(u) B_j^n(v) = \mathbf{P} \text{ for } \mathbf{P} \in Q, \; (u, v) = T_N(\mathbf{P})$$

*Note: Here $(u, v) = T_N(\mathbf{P}) \in [0, 1]$ are used as Bézier-parameters*

**2D and 3D Bézier Control Points** *are initially set to:*

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

*resulting in the required identity mapping (no deformation)*

**3D-case:** *Analog to the 2D Bézier surfaces we can set up tri-variat Bézier-volumes*

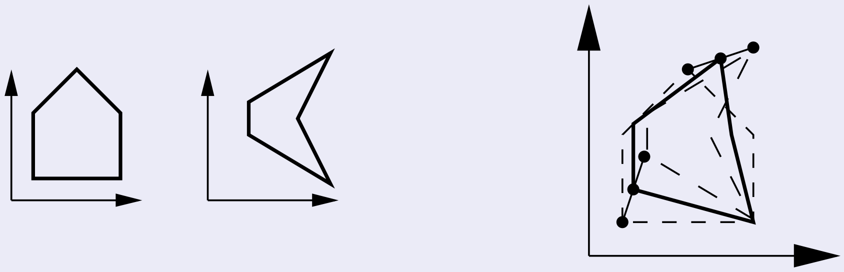
# 7.3.2: Blend Shapes and Morphing

## Approach (Temporal Shape Blending)

**Concept:** *Direct application of the interpolation techniques to geometries* $\Rightarrow$ *Floating transition between shapes*

**Initial Situation:** *Control points* $\mathbf{P}_i$ $i = 1, \ldots, k$ *of an object are known for several points in time* $t_1, \ldots, t_n$*:* $\mathbf{P}_i(t_1), \ldots, \mathbf{P}_i(t_n),\ i = 1, \ldots, k$

**Procedure:** *Interpolation delivers* $\mathbf{P}_i(t)$ *for any* $t$*, thus the geometry can be determined at any time, resulting in a blending over time.*







# Morphing using Affine Combinations

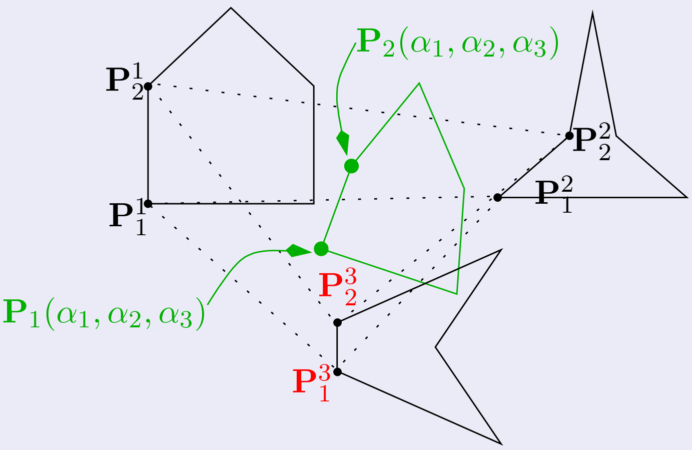
## Approach (Parameter Based Shape Bedning)

*Use shape prototypes (blend shapes) and combine/blend them by means of affine-combinations.*

**Initial Situation:** *Control points* $\mathbf{P}_i$ *of an object which are known for* key positions $1, \ldots, k$*:* $\mathbf{P}_i^1, \ldots, \mathbf{P}_i^k$.

**Approach:** *Affine combination: For blend weights* $\alpha_j$ *with* $\sum_{j=0}^{n} \alpha_j = 1$ *we get:*

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

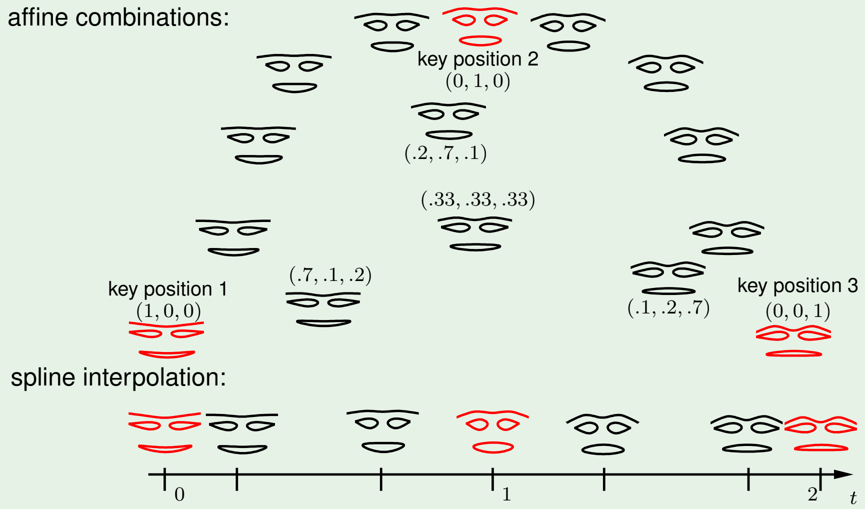



**Animation:** *Vary the bleind weights* $\alpha_k$ *over the time:* $\alpha_i(t)$

## Example

**Initial Situation:** 3 *prototype facess* with (“cheerful”, “frightened”, “scared”)

**Animation:** Modification of the blend weights over time, from cheerful ($t = 0$) over frightened ($t = 1$) to scared ($t = 2$).




**Note:** Key positions have been placed at different locations in order to visualize the animation in one image.





## Application (Image Morphing)

**Color:** *Besides form, color etc. can also be blended/morphed*

**Example:** *Morphing of an image*

1. *Warping (FFD) puts pixels in relation:* $(x, y) = D^{\textit{init}}(x, y) \rightarrow D^{\textit{end}}(x, y)$
   $\Rightarrow$ *Bézier-control points for initial and final images* $\mathbf{C}_{ij}^{\textit{init}}$ *(initial),* $\mathbf{C}_{ij}^{\textit{end}}$
2. *Intermediate image (time $t$) by interpolation of control points & color:*

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