FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Computeranimation

Lesson 2 – Keyframeanimation

COMPUTER GRAPHICS • ERLANGEN

Motivation

Topics

- Rigid Transformation
- **Animation**
- Collision
- Dynamic
- Mass-Spring Simulation
- Rigging and Skeletal Animation
- Motion Capturing using RGB-D Sensor

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Introduction

Interpolation

Orientation

Application

COMPUTER GRAPHICS • ERLANGEN

Introduction

Key Frame Animation (also In-Betweening)

- State of an Object is defined at certain points in time (**key frames**)
- Determine Positions in between this key frames
- … using **interpolation**

COMPUTER GRAPHICS • ERLANGEN

Introduction

Object State

- Snap-shot of all relevant object parameters
- Some Examples of animation variables (**avar**s):

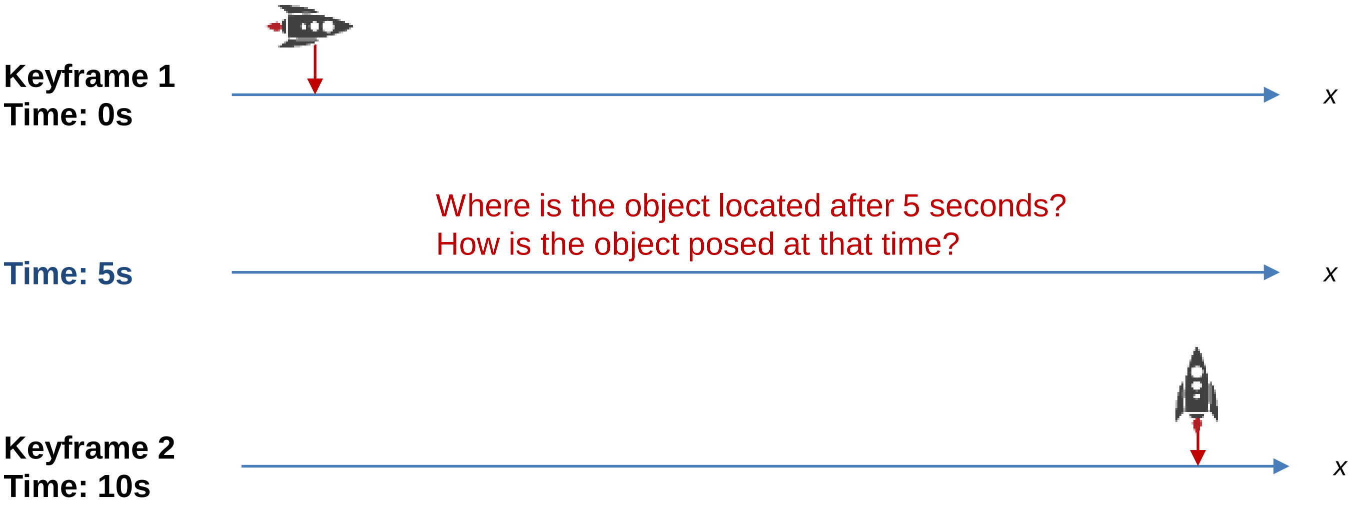
Object's …
… position
… orientation
… shape of an object
… camera parameters
… light information parameters

[…]

"Woody has 712 avars,
212 only for his face…"
(http://www.ee.hawaii.edu/~tep/EE461/Notes/Intro/toystory.html)

Introduction

Object State: Translation And Rotation Example



COMPUTER GRAPHICS • ERLANGEN

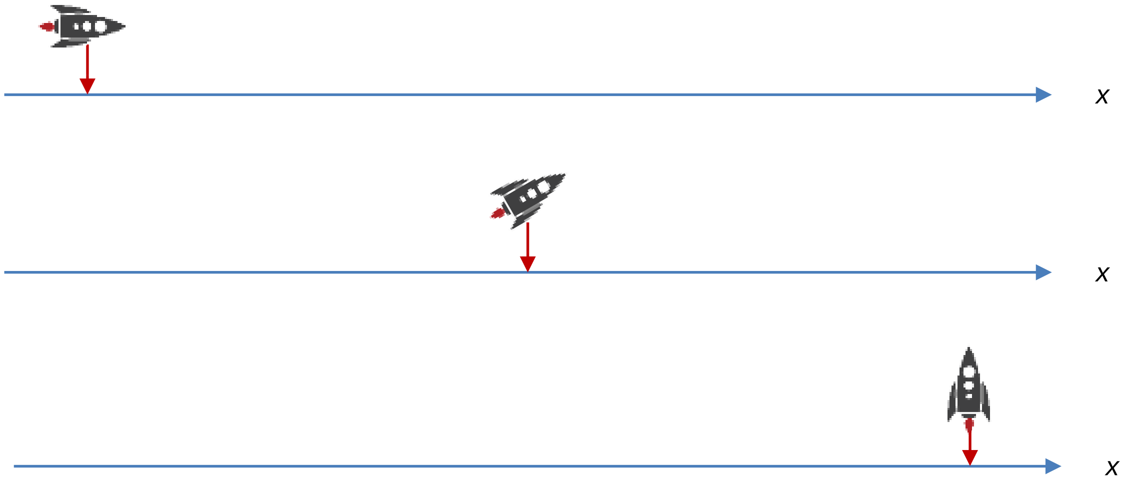
Introduction

Object State: Translation And Rotation Example

Keyframe 1
Time: 0s

Time: 5s

Keyframe 2
Time: 10s



COMPUTER GRAPHICS • ERLANGEN

Introduction

Interpolation

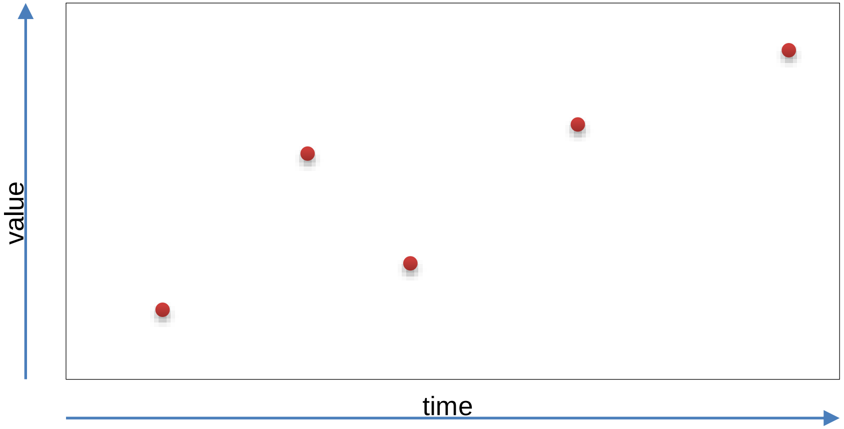
Orientation

Application

Interpolation

Motivation

- Problem: A function (e.g. an *avar*) is given only at some points in time.
- Challenge: How to find valid (or plausible) values in between these **sampling points**.



Interpolation

Local Methods

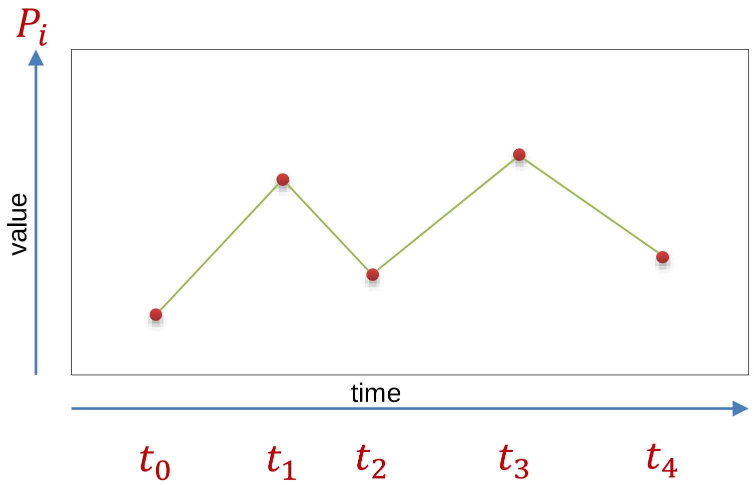
- To interpolate a value only a surrounding sampling points are used
- Some methods:
    - Linear interpolation
    - Hermite interpolation
    - Catmull-Rom interpolation

Global Methods

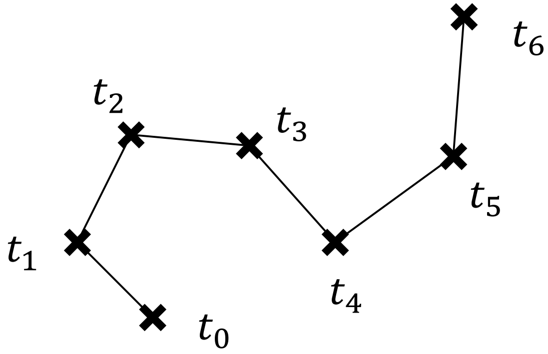
- More than a local neighborhood is is included in the computation
- Some Methods use the whole set of sample points
- Some methods:
    - Polynomial interpolation
    - Bézier curves
    - B-Spline curves

Interpolation

Linear Interpolation



1D



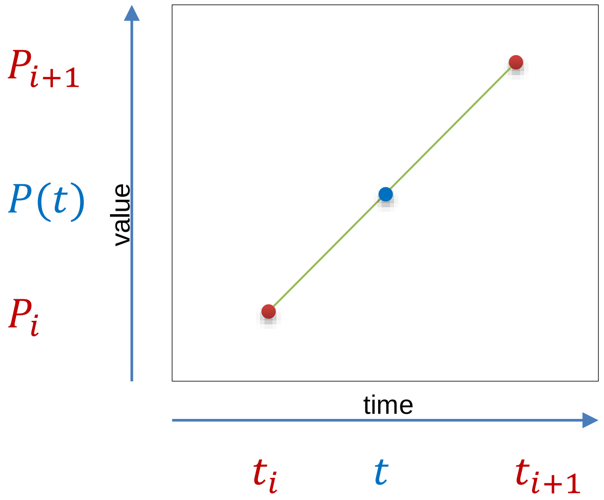
2D

Interpolation

Linear Interpolation

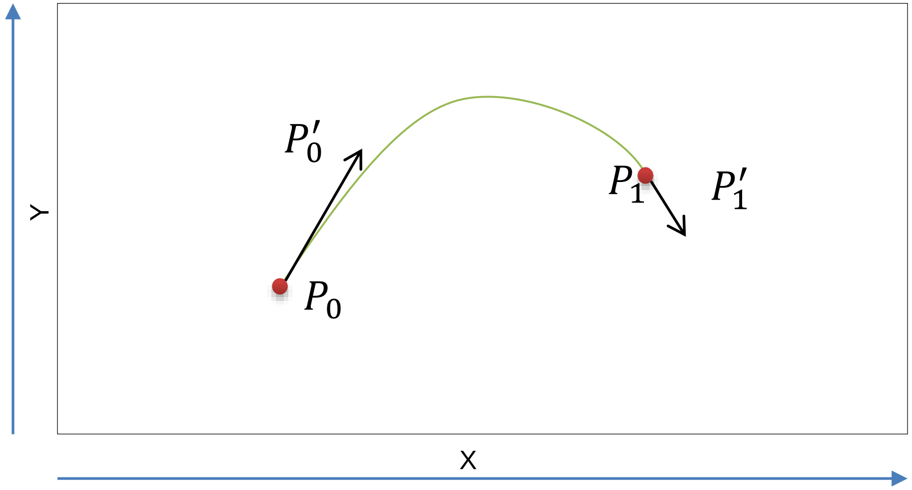
- Seek for the i with $t_i \leq t \leq t_{i+1}$
- Compute the **local parameter** $u(t) = \frac{t - t_i}{t_{i+1} - t_i}$
- The function Value $P(t)$ finally the linearly interpolated value at u between the sampling points (t_i, P_i) and $(\mathrm{t}_{\mathrm{i+1}}, P_{i+1})$:

$$P(u) = (1 - u) \cdot P_i + u \cdot P_{i+1}$$



Interpolation

Hermite-Interpolation



- In Addition to values the 1st derivatives (slope) is defined at each sample

Interpolation

Hermite-Interpolation

- Challenge: Find an interpolating cubic polynomial which slope fits good to the desired curve

$$P(u) = a_3u^3 + a_2u^2 + a_1u + a_0$$

- Consider Hermite Polynomials with their boundary properties:

	$H_i(0)$	$H_i'(0)$	$H_i'(1)$	$H_i(1)$
$H_0(u) = 2u^3-3u^2+1$	1	0	0	0
$H_1(u) = u^3-2u^2+u$	0	1	0	0
$H_2(u) = u^3-u^2$	0	0	1	0
$H_3(u) = -2u^3+3u^2$	0	0	0	1

Interpolation

Hermite-Interpolation

- With the following approach

$$P(u) = P_0 H_0(u) + P_0'(u) H_1(u) + P_1'(u) H_2(u) + P_1 H_3(u), \qquad u \in [0,1]$$

… this leads to the interpolation:

$$P(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} 2 & 1 & 1 & -2 \\ -3 & -2 & -1 & 3 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_0' \\ P_1 \\ P_1' \end{bmatrix}$$

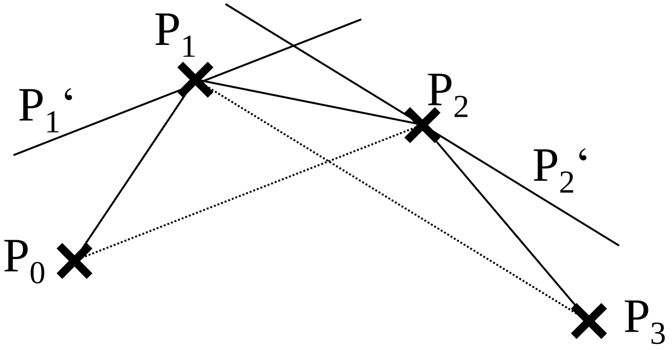
LGDV
COMPUTER GRAPHICS • ERLANGEN

Interpolation

Catmull-Rom Interpolation

- Basically Hermite interpolation
- Tangent (derivative) is calculated by finite differences:

$$P_i' = \frac{P_{i+1} - P_{i-1}}{t_{i+1} - t_{i-1}}$$

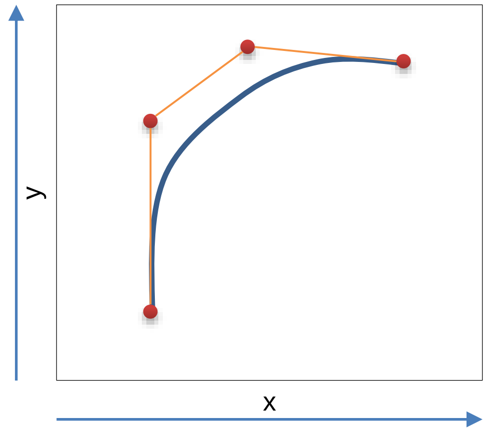


Interpolation

Bézier Curves

- Global Interpolation Method
- Uses all sample points for interpolation

- Geometrical construction
- Algorithm of De' Casteljau



COMPUTER GRAPHICS • ERLANGEN

Introduction

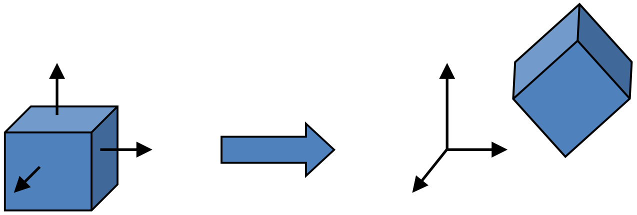
Interpolation

Orientation

Application

Orientation

Introduction



- **Task**: Define an Objects Rotation (Camera, Object)
- Remember the rigid body Transformation

$$\vec{x} \rightarrow \boldsymbol{R} \cdot \vec{x} + \vec{t}$$

- The Rotation Matrix $\boldsymbol{R}$ is orthogonal: $\boldsymbol{R} \cdot \boldsymbol{R}^{\boldsymbol{T}} = \boldsymbol{R}^{\boldsymbol{T}} \cdot \boldsymbol{R} = \boldsymbol{I} \rightarrow \left|\boldsymbol{R} \cdot \boldsymbol{R}^{\boldsymbol{T}}\right|^{2} = \mathbf{1}$
- **The combination of rigid Transformations is a rigid Transformation as well:**

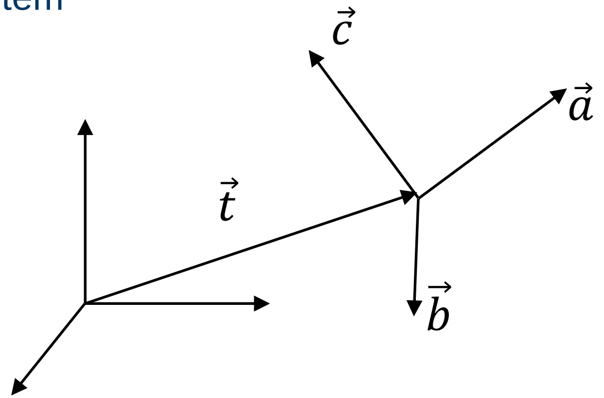
$$\vec{x} \rightarrow \boldsymbol{R_2} \cdot \left(\boldsymbol{R_1} \cdot \vec{x} + \overrightarrow{t_1}\right) + \overrightarrow{t_2} = \underbrace{\boldsymbol{R_2} \cdot \boldsymbol{R_1}}_{\text{is a orthogonal matrix!}} \cdot \vec{x} + \underbrace{\boldsymbol{R_2} \cdot \overrightarrow{t_1} + \overrightarrow{t_2}}_{\text{is a vector!}}$$

Orientation

Rotation Matrix

- The columns of the rotation matrix span a new coordinate system

$$\vec{x} \rightarrow \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \cdot \vec{x} + \vec{t}$$

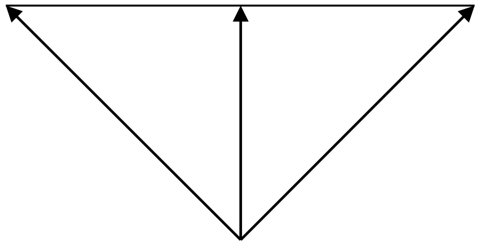


- The matrix cannot be arbitrarily choosen!
- The orthonoality has to be ensured → 3 degrees of freedom!

Orientation

Interpolation of a Rotationmatrix

- $interpolate(\boldsymbol{R_1}, \boldsymbol{R_2}, u) = (1-u) \cdot \boldsymbol{R_1} + u \cdot \boldsymbol{R_2}$



- The interpolated Matrix in general is not a Rotation Matrix
- Unit vectors are no longer unit length, orthogonality is not ensured

$$\frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

→ **It is not a good idea interpolating rotation matrices!**

Orientation

Reorthonomalization of interpolated Rotation Matrices

- $interpolate(\boldsymbol{R_1}, \boldsymbol{R_2}, u) = (1-u) \cdot \boldsymbol{R_1} + u \cdot \boldsymbol{R_2} \rightarrow \boldsymbol{\widetilde{R}_{int}}$
- $\boldsymbol{\widetilde{R}_{int}}$ consists of the column vectors $\{\boldsymbol{\tilde{a}}, \boldsymbol{\tilde{b}}, \boldsymbol{\tilde{c}}\}$, we want to get the adjusted vectors $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$

1. Set vector $\boldsymbol{a} = \frac{\boldsymbol{\tilde{a}}}{||\boldsymbol{\tilde{a}}||}$
2. Set vector $\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{\tilde{c}}$, and normalize it $\boldsymbol{b} = \frac{\boldsymbol{b}}{||\boldsymbol{b}||}$
3. Set vector $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$, and normalize it $\boldsymbol{c} = \frac{\boldsymbol{c}}{||\boldsymbol{c}||}$
4. Set adjusted rotation matrix $\boldsymbol{R_{int}} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$

LGDV
COMPUTER GRAPHICS • ERLANGEN

Orientation

Fixed Angles

- The rotation is described by a chain of Rotations about the global Axis

$$\boldsymbol{R} = \boldsymbol{R_z}(\gamma) \cdot \boldsymbol{R_y}(\beta) \cdot \boldsymbol{R_x}(\alpha)$$

- Note: the rotations are multiplied from right to left
- The avars is the triplet (α, β, γ)
- Rotation axis can combinated (almost) freely:
    - xyz (example above), zyx
    - also xyx, zxz
    - Not: xxy! A subsequent rotation about the same axis leads to a wrong result

Orientation

Fixed Angles

- The rotation is described by a chain of Rotations about the global Axis

$$\boldsymbol{R} = \boldsymbol{R_z}(\gamma) \cdot \boldsymbol{R_y}(\beta) \cdot \boldsymbol{R_x}(\alpha)$$

- The matrices for rotations about the unit axis are:

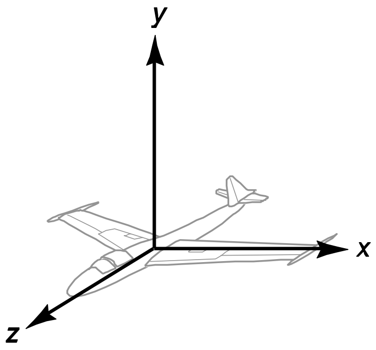
$$\boldsymbol{R_x}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \qquad \boldsymbol{R_y}(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \qquad \boldsymbol{R_z}(\theta) = \begin{bmatrix} cos(\theta) & -sin(\alpha) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

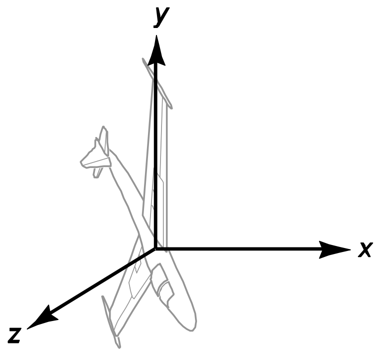
LGDV
COMPUTER GRAPHICS • ERLANGEN

Orientation

Fixed Angles

- Example, order is xyz (see example above), angles are $(10°, 45°, 90°)$





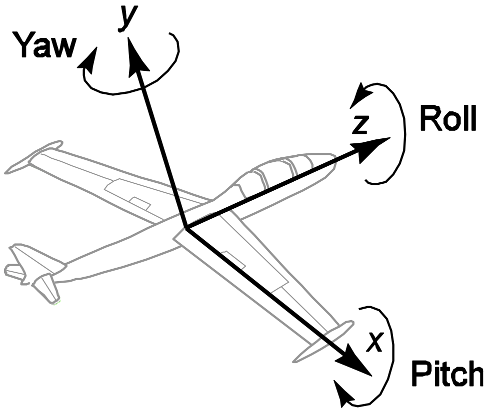
Orientation

Euler Angles

- Instead of rotation about global axis, we rotate the system with the object:

Example

1. Yaw: $R_y(\alpha)$
2. Pitch in local space (reverse yaw, rotate, re-apply yaw): $R_y(\alpha)\ R_x(\beta)\ R_y(-\alpha)$
3. combine: $R_y(\alpha)\ R_x(\beta)\ R_y(-\alpha)\ R_y(\alpha) = R_y(\alpha)\ R_x(\beta)$
4. Roll local space
    1. Revert Yaw und Pitch, apply roll, re-apply yaw, pitch:
    2. $R_y(\alpha)\ R_x(\beta)\ R_z(\gamma)\ R_x(-\beta)\ R_y(-\alpha)$
    3. kombinieren:
       $R_y(\alpha)\ R_x(\beta)\ R_z(\gamma)\ R_x(-\beta)\ R_y(-\alpha)\ R_y(\alpha)\ R_x(\beta) = R_y(\alpha)\ R_x(\beta)\ R_z(\gamma)$



→ Standard in Automotive/Aeronautics is: $R_x(\alpha)\ R_y(\beta)\ R_z(\gamma)$

Orientation

Interpolation of Fixed/Euler Angles

- Simply interpolate the angles
- **Problems**: *flipping angles*, *gimbal lock*

- Gimbal Lock:
  - Pitch (green) is 90° → roll (blue) and yaw (violet) have the same effect
  - In this constellation no roll about the original roll axis possible!

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

LGDV COMPUTER GRAPHICS • ERLANGEN

Introduction

Interpolation

Orientation

Application

Application

In class coding

1. **Add a data structure to main.cpp to hold the originally loaded mesh**
   Hint: Use std::vector<vec3> for the vertices and std::vector<ivec3> for the triangles

2. **Add a Method to main.cpp that transforms the vertices of the originally loaded mesh and updates the render-model. Call this functionality by a key-press event.**

3. **(BONUS) Add functionality to load the transformation from a file.**

COMPUTER GRAPHICS • ERLANGEN