Computergrafik SS 2016

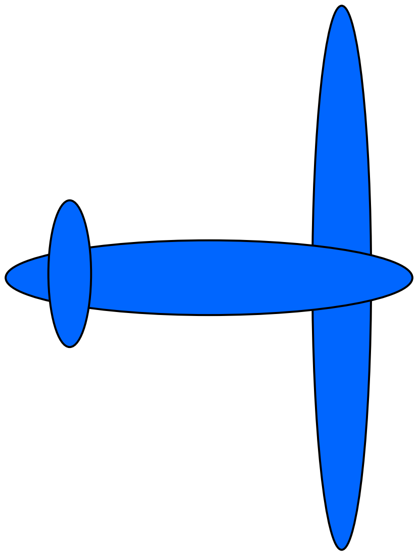
Oliver Vornberger

Kapitel 24: Animation

Animationsarten

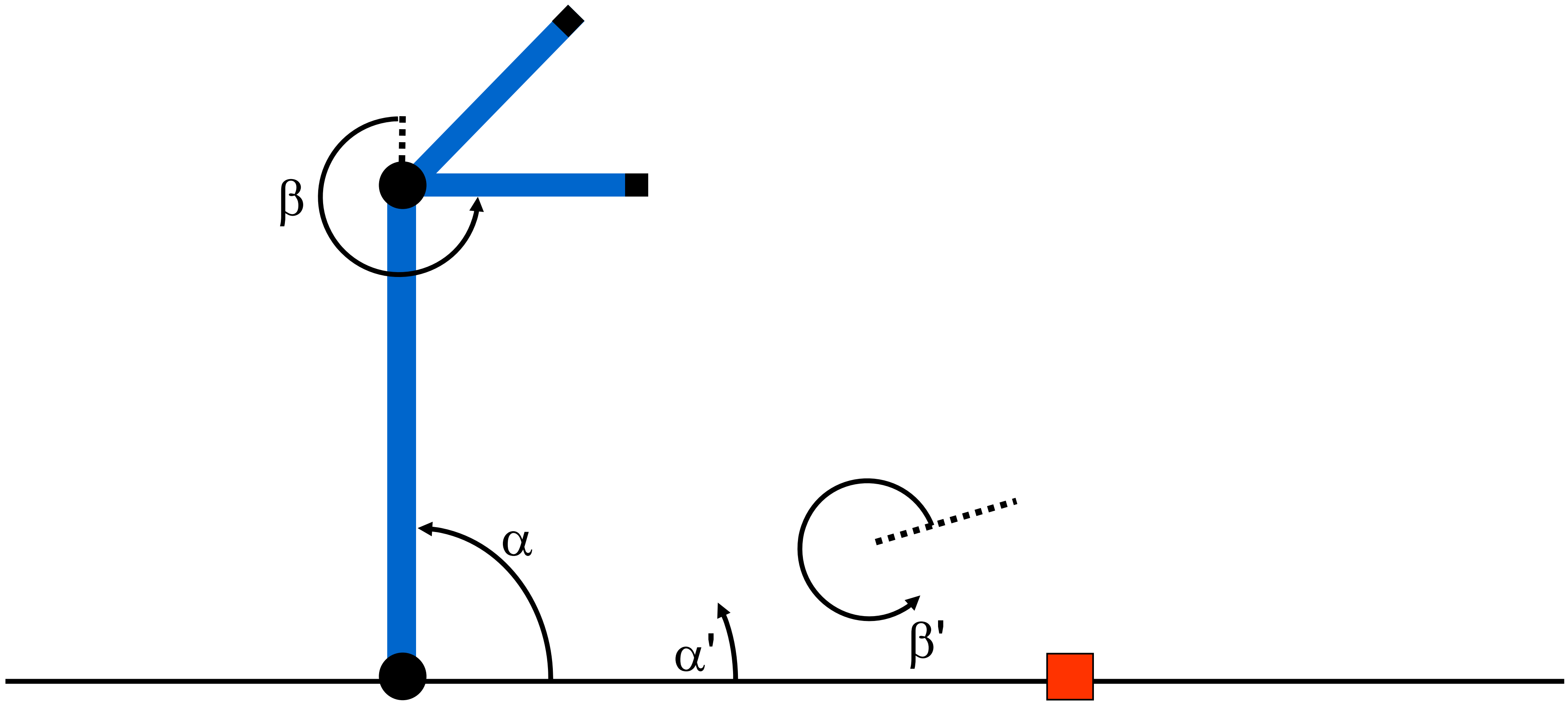
- Key Frame Animation
- Forward Kinematics
- Inverse Kinematics
- Particle Systems
- Verhaltensanimation

Key frame Animation



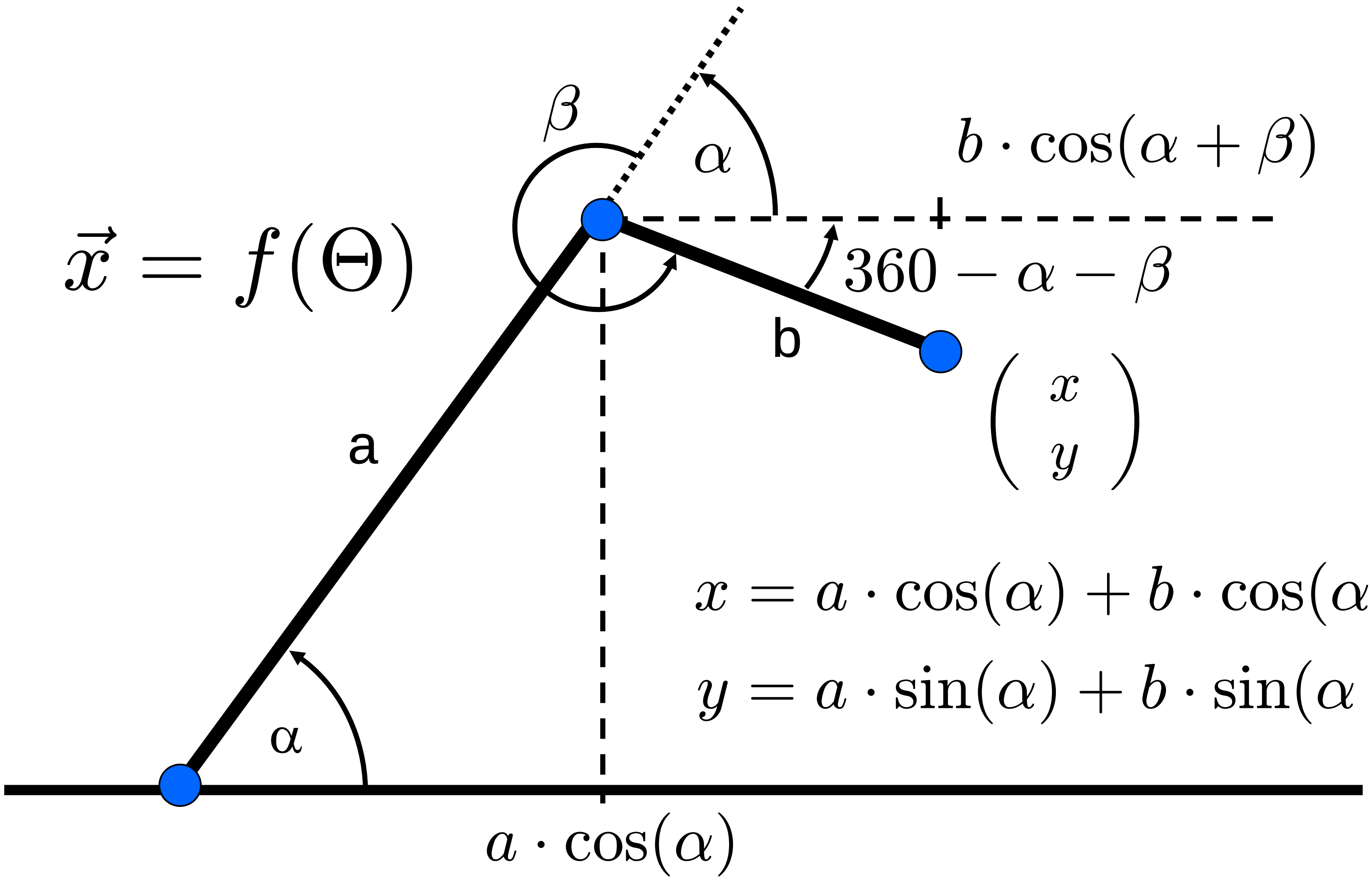
Kinematik

Bewegung im Raum
mit Körperverbindungen



Forward Kinematics

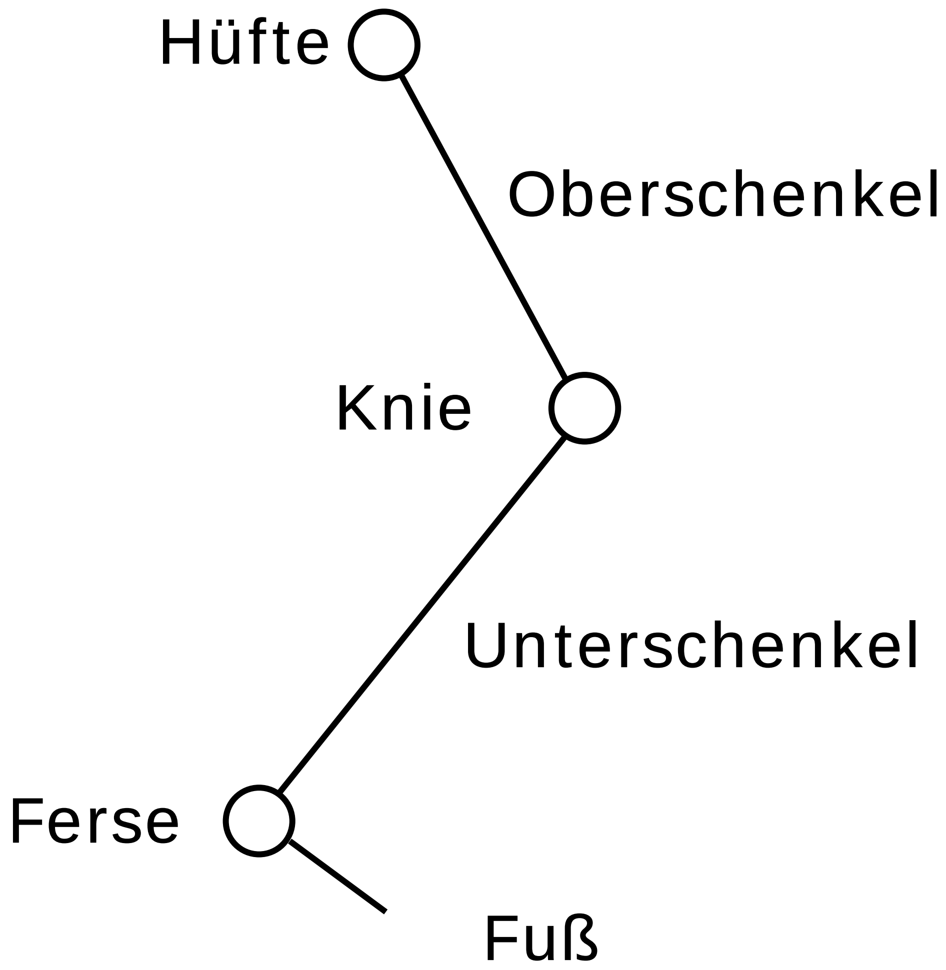
$$\vec{x} = f(\Theta)$$



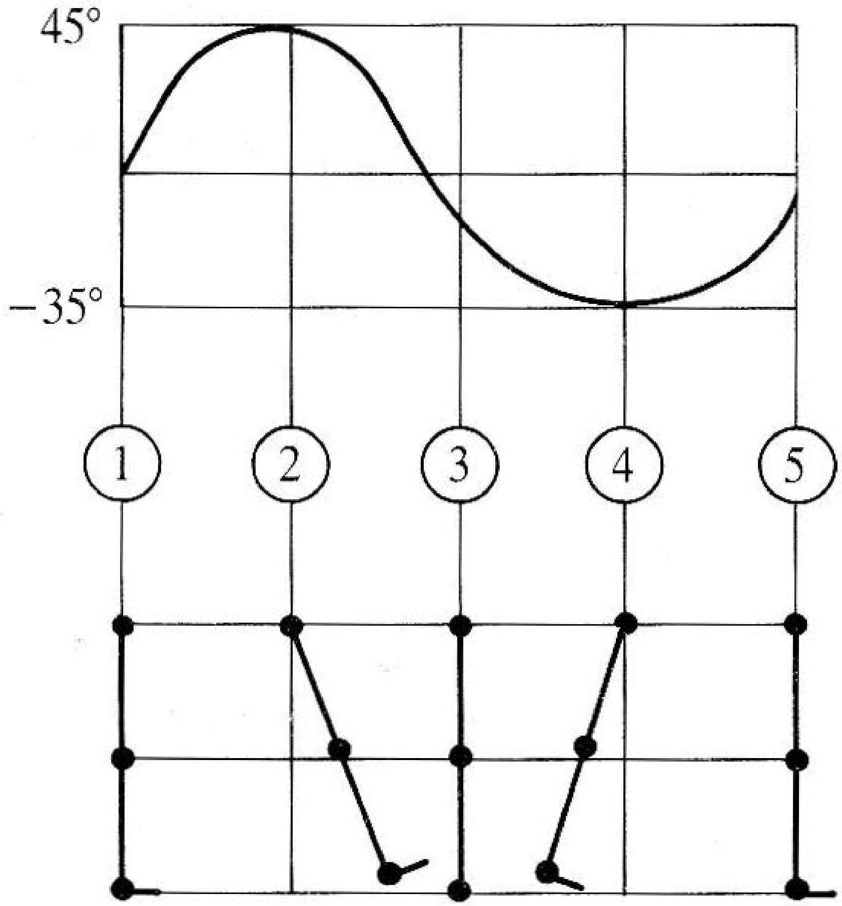
$$x = a \cdot \cos(\alpha) + b \cdot \cos(\alpha + \beta)$$

$$y = a \cdot \sin(\alpha) + b \cdot \sin(\alpha + \beta)$$

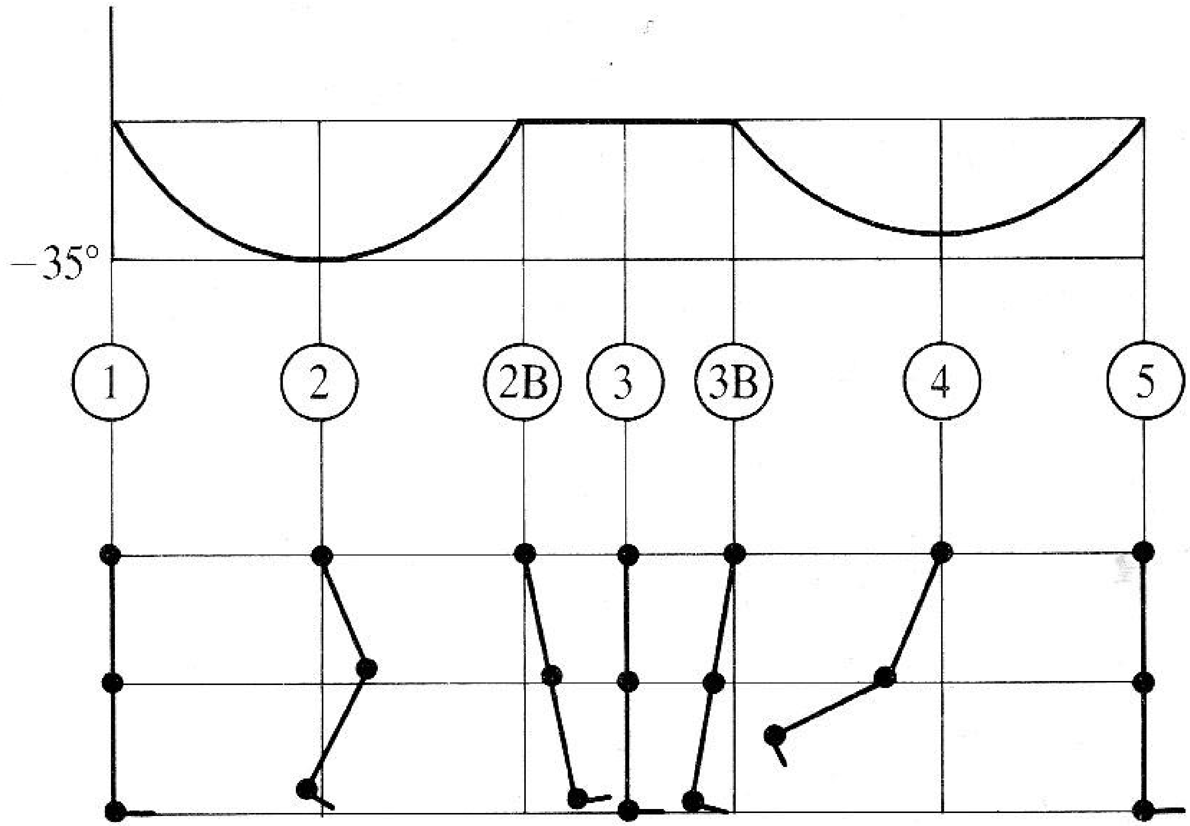
Skript für Forward Kinematics



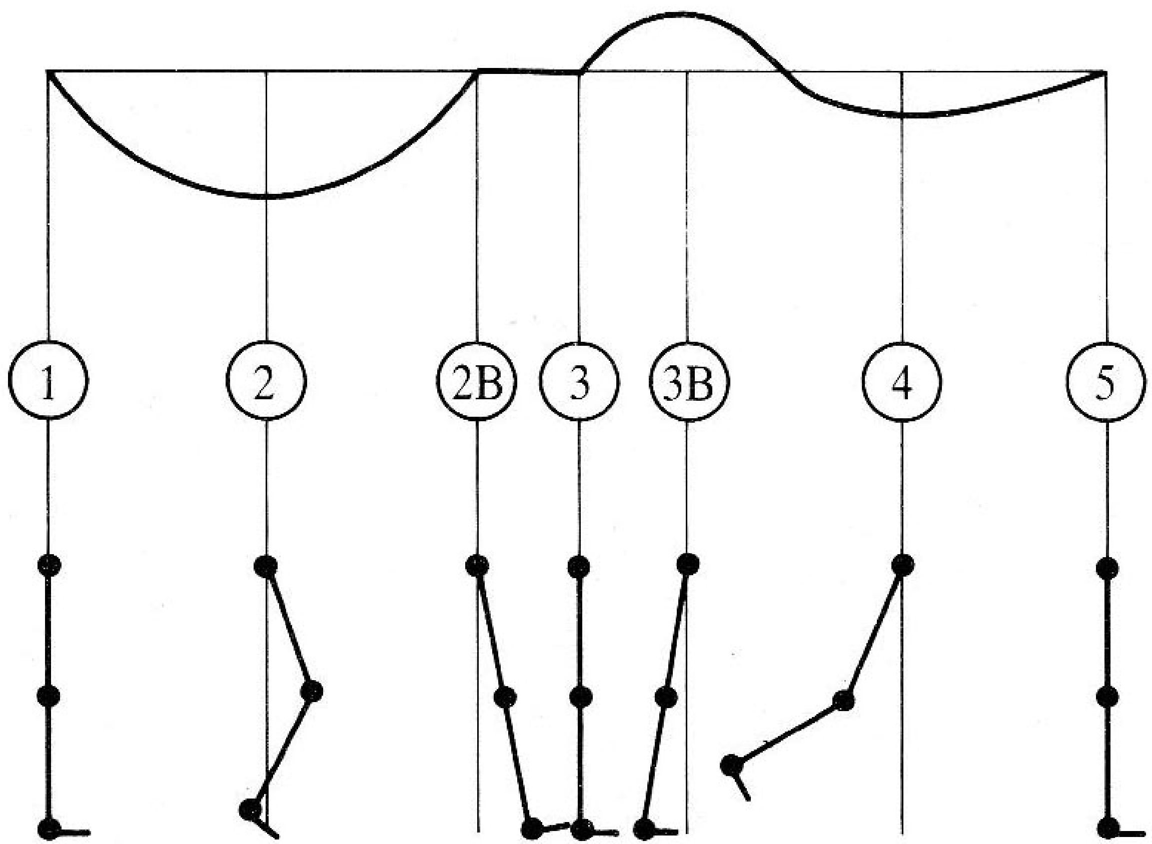
Rotation in der Hüfte



Rotation im Knie



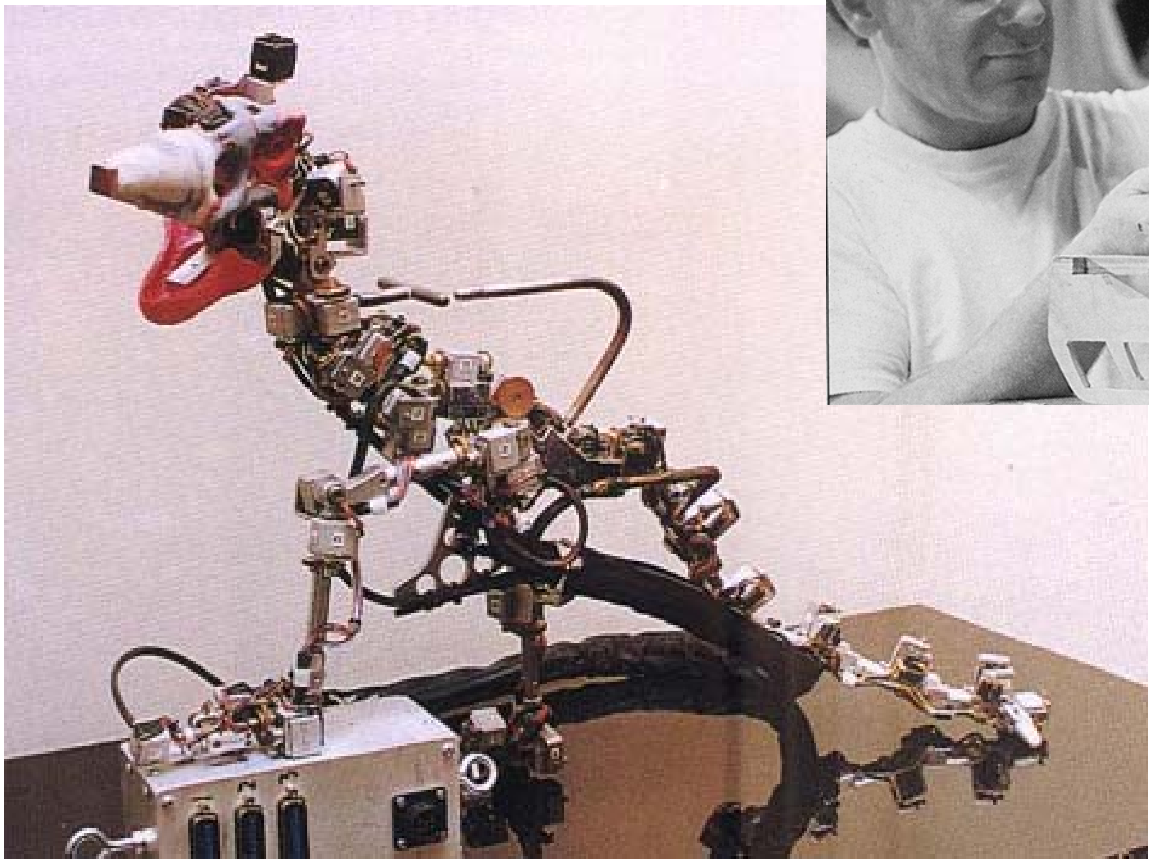
Rotation in der Ferse

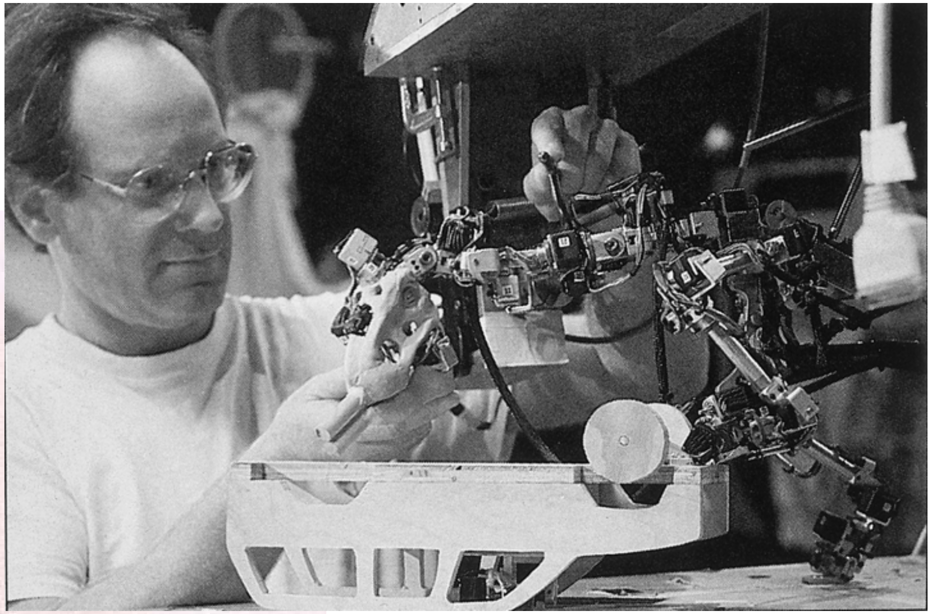


Jurassic Park [1993]

geplant als Stop Motion Film

Jurassic Park



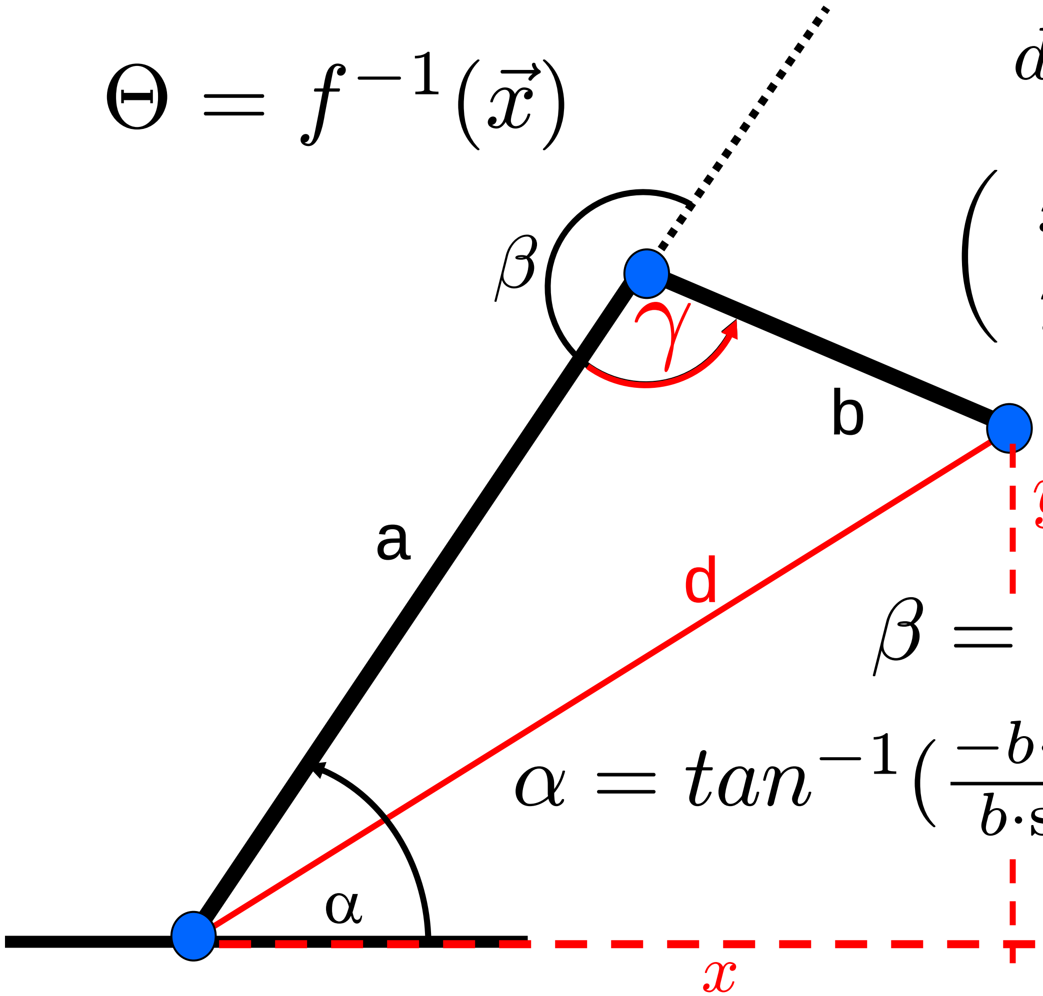


Ron Magid: "After Jurassic Park", American Cinematographer, December 1993.
Abgedruckt in: Alan Watt "3D-Computergrafik", S. 549, Pearson Studium, ein Imprint von Pearson Education Deutschland GmbH, 2001.

realisiert mit Dinosaur Input Device (DID)

Inverse Kinematics

$$\Theta = f^{-1}(\vec{x})$$



$$d^2 = a^2 + b^2 - 2ab\cos(\gamma)$$

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

$$d^2 = x^2 + y^2$$

$$\cos(\gamma) = \frac{a^2+b^2-x^2-y^2}{2ab}$$

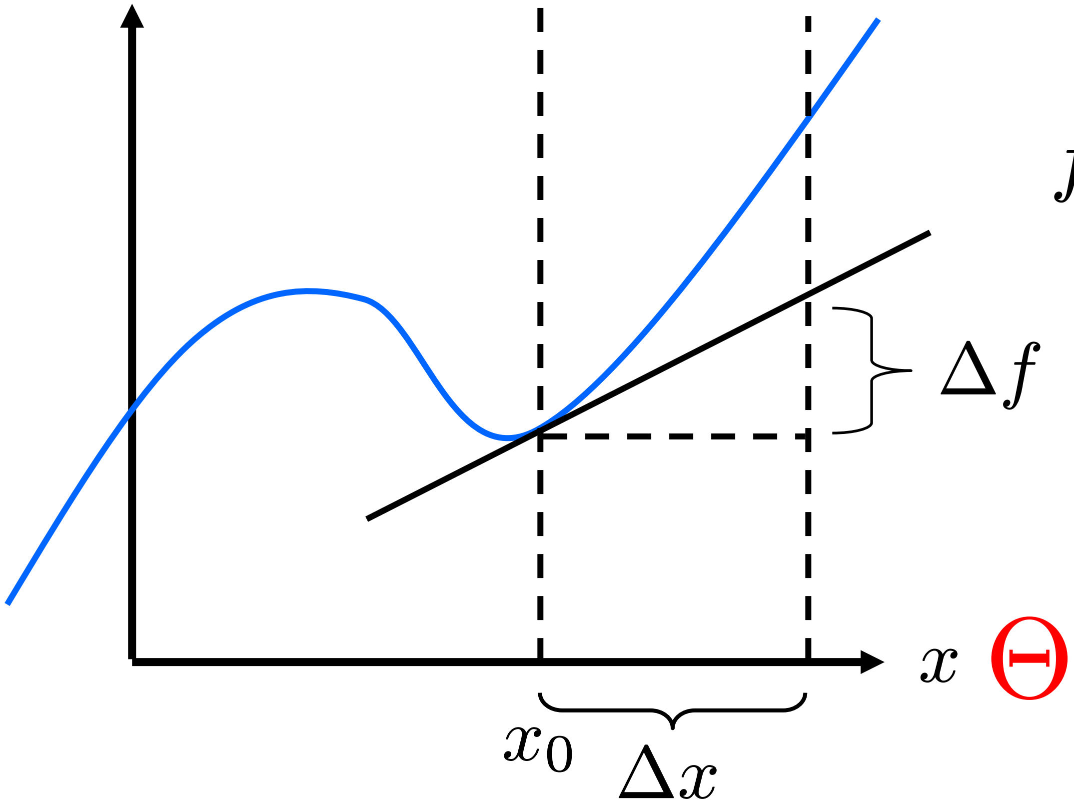
$$\beta = \cos^{-1}\left(\frac{x^2+y^2-a^2-b^2}{2\cdot a\cdot b}\right)$$

$$\alpha = tan^{-1}\left(\frac{-b\cdot\sin(\beta)\cdot x+(a+b\cdot\cos(\beta))\cdot y}{b\cdot\sin(\beta)\cdot y+(a+b\cdot\cos(\beta))\cdot x}\right)$$

Differenzierbarkeit

$\vec{x} = f(\Theta)$

$y = f(x)$



$$f'(x_0) \approx \frac{\Delta f}{\Delta x}$$

$$f'(x_0) \cdot \Delta x \approx \Delta f$$

$$\Delta x \approx \frac{\Delta f}{f'(x_0)}$$

Jakobi-Matrix

Die Jakobi-Matrix einer differenzierbaren Abbildung

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

ist die *m* × *n* Matrix aller partiellen Ableitungen

$$J_f = \frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & & \dots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

$$f(x,y,z) = x^2 + y^2 + z \cdot \sin(x)$$

$$\frac{\partial}{\partial x} f(x,y,z) = 2x + z \cdot \cos(x)$$

$$\frac{\partial}{\partial y} f(x,y,z) = 2y$$

$$\frac{\partial}{\partial z} f(x,y,z) = \sin(x)$$

Abhängigkeit zwischen dx und dΘ

Schwer zu errechnen: $\Theta = f^{-1}(\vec{x})$

Aber: kleine Änderungen im Winkel verursachen kleine Änderungen in der Position

$$J(\Theta) = \frac{\partial \vec{x}}{\partial \Theta}$$

$$J(\Theta)\partial\Theta = \partial\vec{x}$$

$$\partial\Theta = J^{-1}(\Theta)(\partial\vec{x})$$

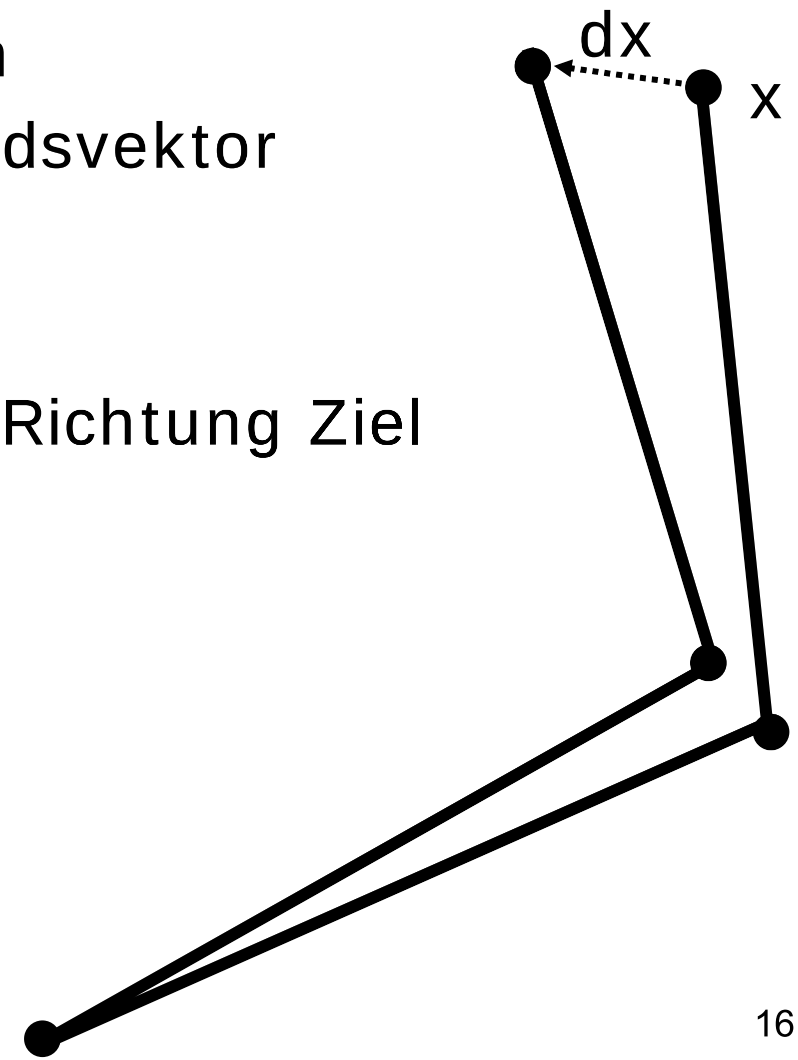
Obacht: invertieren nur bei quadratischen Matrizen möglich !

Iterationsverfahren

Sei x die aktuelle Position

Sei Θ der aktuelle Zustandsvektor

```
while (!fertig) {
  dx := kleine Bewegung Richtung Ziel
  J(Θ) = dx/dΘ
  berechne Inverse von J
  dΘ := J^-1(Θ)(dx)
  x := f(Θ+dΘ)
}
```



Particle Systems

Geeignet für

- Sand
- Funken
- Wasser
- Schnee
- Feuer
- ...

William T. Reeves [1983]:
"Particle Systems - A Technique for Modeling a Class of Fuzzy Objects"
ACM Transaction on Graphics

LucasFilm, Pixar, Oscar für Wiss. & Entw.

Karl Sims [1990]:
″Particle Animation and Rendering using Data Parallel Computation″
ACM Computer Graphics
Connection Machine CM-2, 65.536 Prozessoren

Simulation physikalischer Gesetze
keine Interaktion untereinander

Partikeleigenschaften

- Position
- Geschwindigkeit
- Bewegungsrichtung
- Lebenszeit
- Größe
- Farbe
- Transparenz
- Gestalt

Phasen

- Generierung neuer Partikel
- Zuordnung von Attributen
- Entfernen von Partikeln
- Transformation von Partikeln
- Rendern des neuen Frames

Physik

Position *P*, Geschwindigkeit *V*, Beschleunigung *A*
Masse m_0 im Zentrum *O*, Flächennormale *N*

$$A = g \cdot m_0 \cdot \frac{O-P}{|O-P|^3}$$

$$V' := V + A \cdot \Delta t$$

$$P' := P + \frac{V+V'}{2} \cdot \Delta t$$

$$V' := V - 2 \cdot (V \cdot N) \cdot N$$

$$F = m \cdot A$$

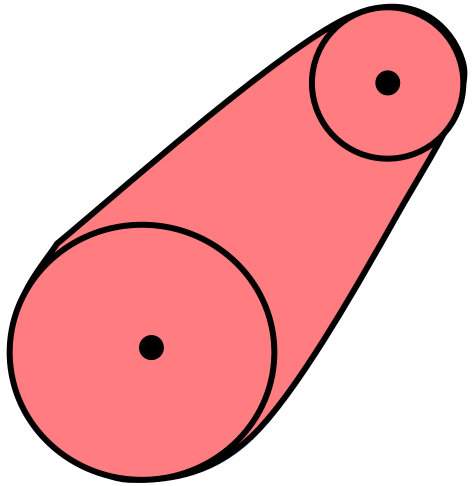
$$A = \frac{F}{m}$$

$$F = \frac{g \cdot m_0 \cdot m_1}{r^2}$$

Particle Rendering

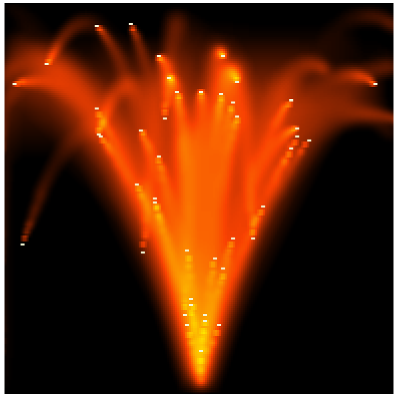
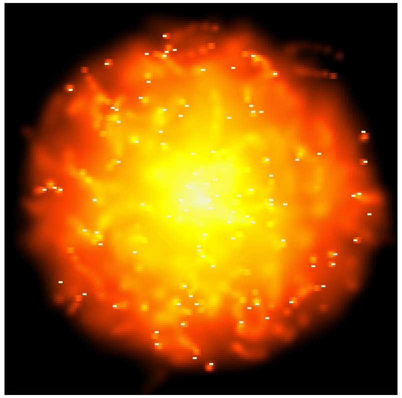
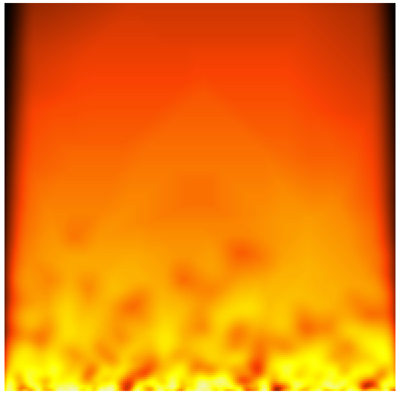
Für Head und Tail:

- Position
- Radius
- Farbverlauf
- Transparenz



Motion Blur berücksichtigen !

Particle Systems Demos



~cg/2016/skript/Applets/Particle/jhlabs.html

http://www.jhlabs.com/java/particles2.htm

http://galloman.github.io/ss2d/samples/particlesWGL.html

http://minimal.be/lab/fluGL/

Pflanzen

white.sand © Alvy Ray Smith, Lucasfilm

Verhaltensanimation

Simple Vehicle Model:

- Masse
- Position
- Fahrtrichtung
- Geschwindigkeit
- Beschleunigung

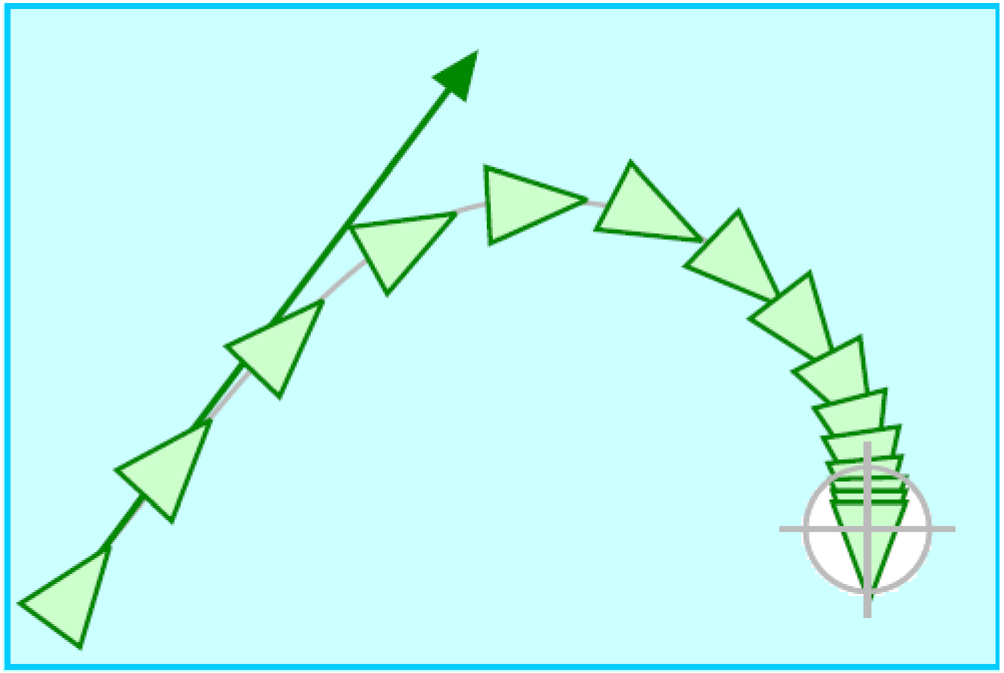
Craig W. Reynolds [1999]:
Steering Behaviors for autonomous Characters
[Sony Computer Entertainment]

http://www.red3d.com/cwr/steer/

Vehikel interagiert

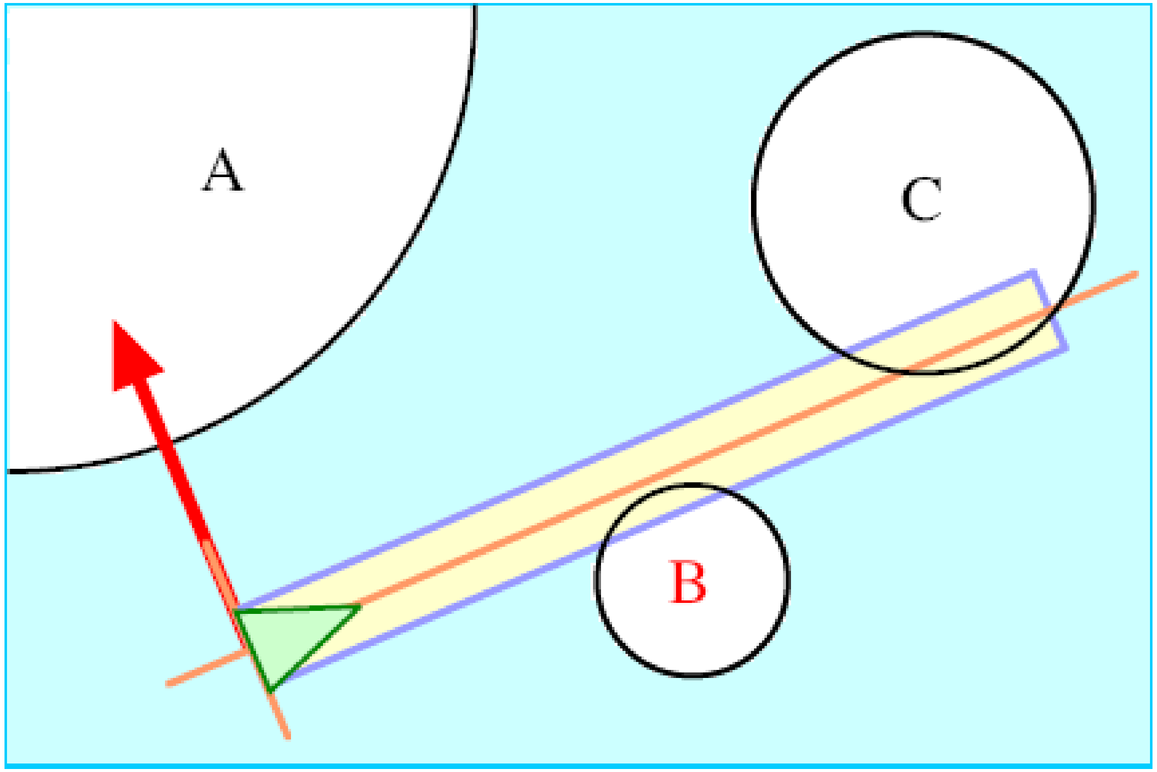
- mit Umweld
- mit anderen Vehikeln

Arrival



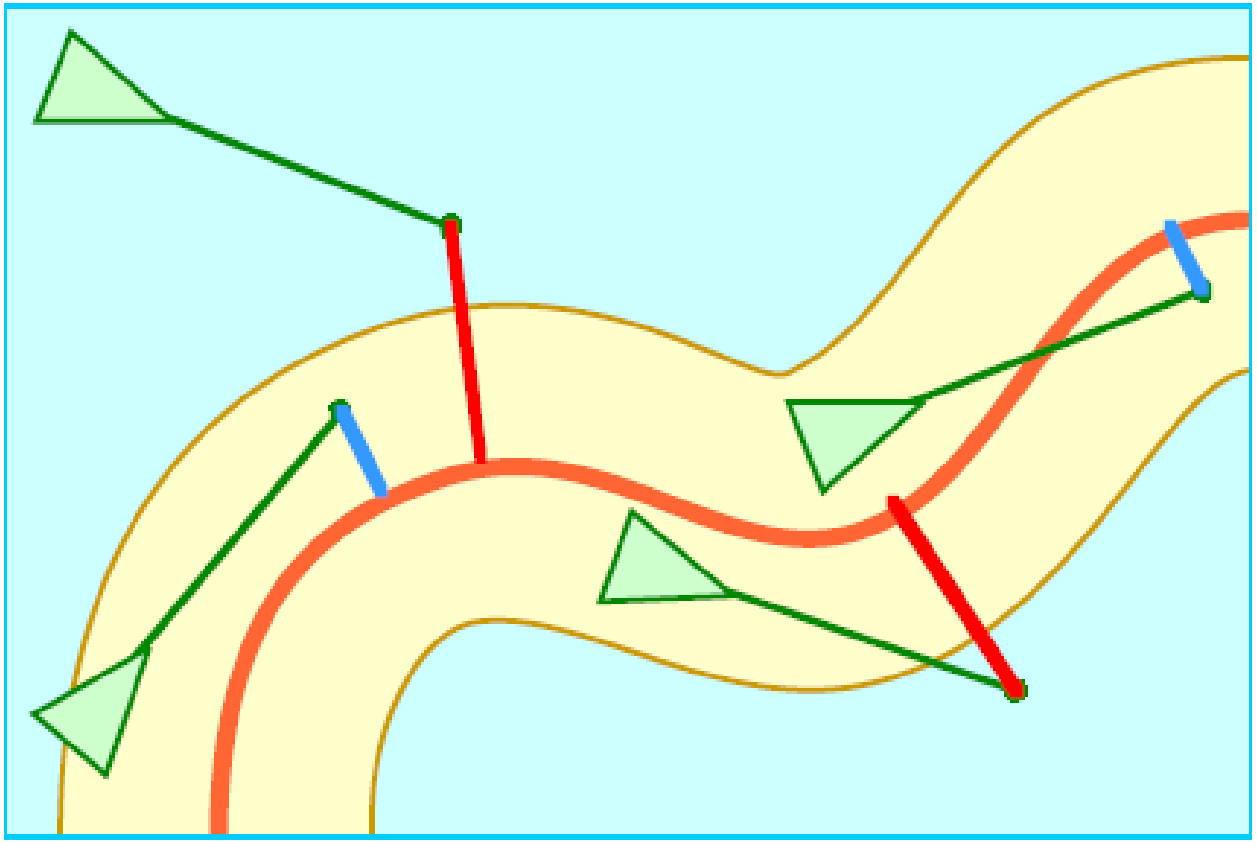
http://www.red3d.com/cwr/steer/Arrival.html

Obstacle Avoidance



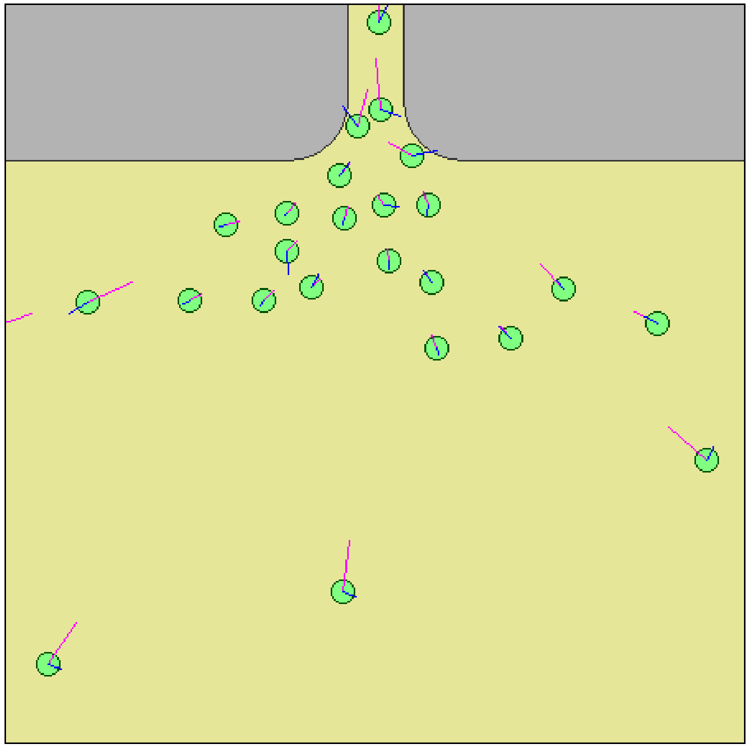
http://www.red3d.com/cwr/steer/Obstacle.html

Path Following



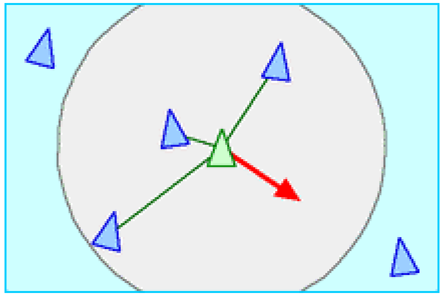
http://www.red3d.com/cwr/steer/PathFollow.html

Queuing Behaviour at door

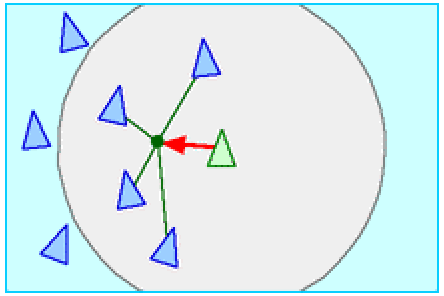


http://www.red3d.com/cwr/steer/Doorway.html

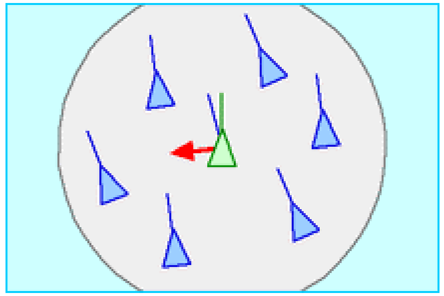
Schwarmverhalten



Separation

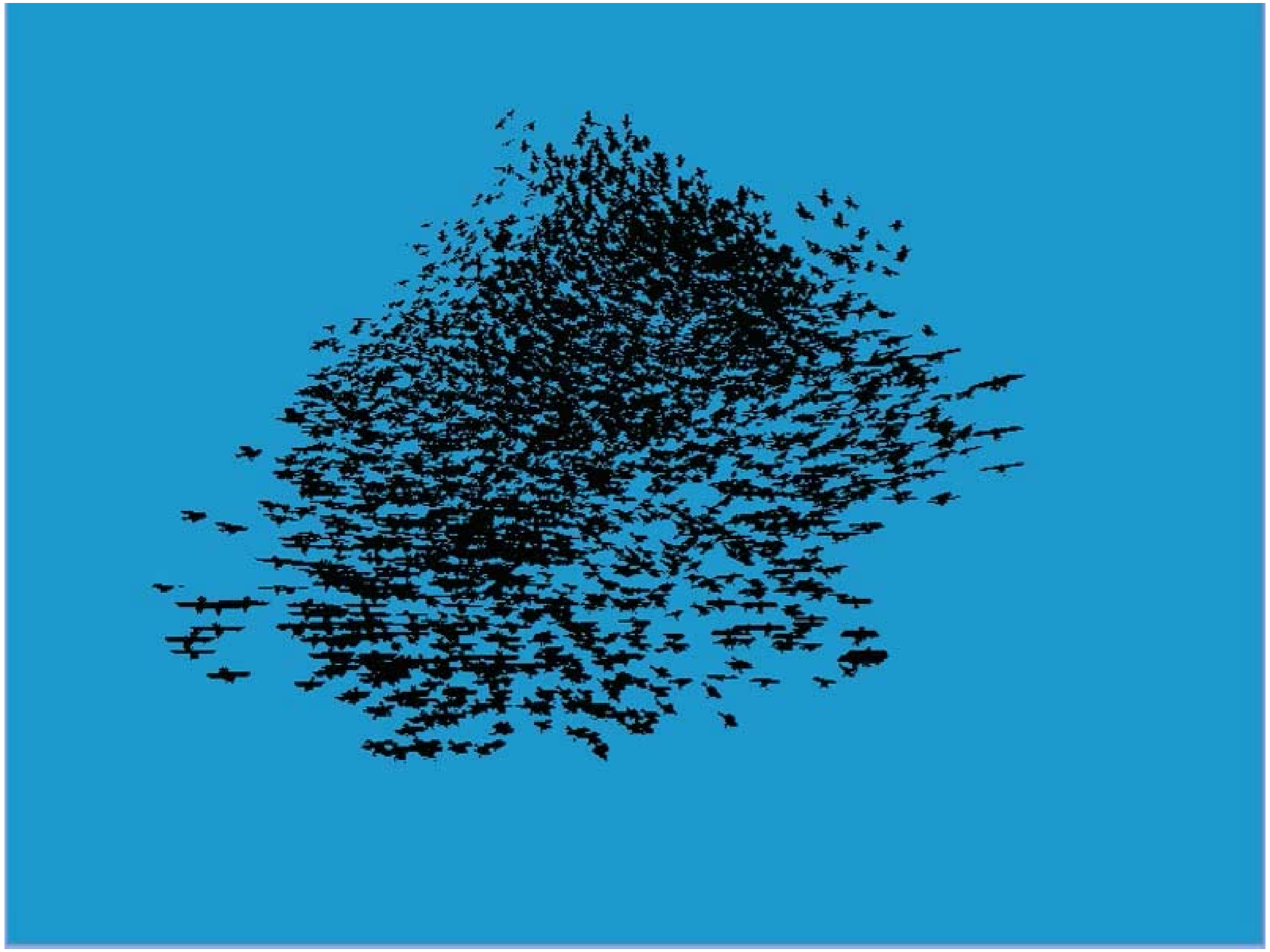


Kohäsion



Ausrichtung

Vogelschwarm von Oliver Tschesche



Scanline Production GmbH, München

TECHNICAL ACHIEVEMENT AWARD der American Academy of Motion Picture

~cg/2016/skript/Applets/Particle/scanline.html