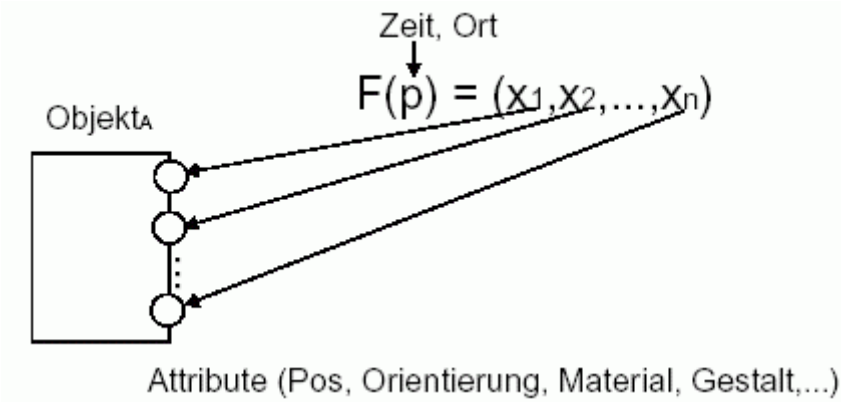
Prozedurale Animation

Idee: Modifizieren der Objekte (und Erzeugen, Löschen) mittels algorithmischer Vorschriften
(vgl. prozedurale Modellierung, Kap. 10)



Art des verwendeten Algorithmus: sehr unterschiedliche Ansätze

- stochastische Animation (Partikelsysteme)
- verhaltensbasierte Animation (Fische, Vögel, Insekten)
- wissensbasierte Ansätze, z.B. genetische Algorithmen

typisch: emergentes globales Verhalten durch lokale Regeln

Anwendungen:

Partikelsysteme
: Pflanzen, Feuer, Wassertropfen

verhaltensbasierte Ansätze
: Schwärme, agentenorientierte Modelle, Artificial Life

Modellierung von Strukturen
: Textilien, elastische Oberflächen

Partikelsysteme
(vgl. Kap. 10)

Partikel: größere Anzahl an betrachteten Einheiten mit Attributen, die Aussehen festlegen
oft grafische Primitive wie Punkte, Linien, einfache 3D-Körper

- stochastische Komponente: Kontrolleinheit, die die Partikel beeinflusst
- nichtdeterministische Modifikation der Attribute (z.B. Farbe, Geschwindigkeit), meist innerhalb vordefinierter Grenzen
- Verwendung zuerst von Reeves in "Star Trek II" zur Animation der Entwicklung eines Planeten

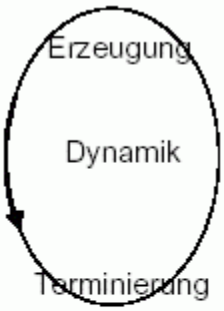
5 Schritte für die Berechnung eines Frames:
(1) erzeuge neue Partikel im System
(2) Zuordnung von initialen Attributwerten für jedes neue Partikel
(3) jedes Partikel, das seine maximale Lebensdauer erreicht hat, wird gelöscht
(4) aktuelle Partikel werden gemäß der Berechnungsvorschrift bewegt
(5) Alle Partikel innerhalb vordefinierter Grenzen werden gerendert

```
while (time < duration)
    {
    generate_particles
    update_particles
    remove_dead_particles
    draw_particles
    time++
    }
```

Vorteile des Ansatzes:
- komplexe Objekte mit nicht eindeutig definierbaren Konturen und komplexer Bewegung können mit wenig Aufwand modelliert werden
- Algorithmen können mit anderen Systemen gekoppelt werden, z.B. Simulationen
- Umfang der Parametrisierung erlaubt Kontrolle über den Aufwand in Relation zur Wirksamkeit

Attribute eines Partikel:

- Position
- Velocity (speed and direction)
- Color
- Lifetime
- Age
- Shape
- Size
- Transparency



- Darstellung als Punkt, Linie, Fläche, 3D-Objekt
- Partikel kollidieren nicht mit anderen Partikeln
- Partikel werfen keine Schatten
- Partikel reflektieren kein Licht, sondern sind als Punktlichtquellen modelliert

Erzeugung der Partikel:

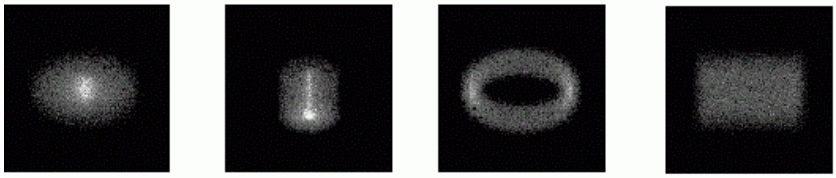
Partikel werden durch einen globalen, kontrollierten stochastischen Prozeß erzeugt,
Zeitabhängigkeit und Varianz können Ausdehnung der Wolke steuern

$$N(t) = averageN(t) + rand(r) * \text{variance}(t)$$

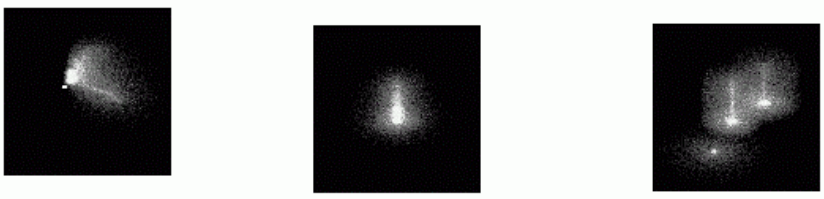
2. Methode: Bildschirmgröße als Parameter verwenden

$$N(t) = (averageN(t) + rand(r) * \text{variance}(t)) * \text{screenSize}$$

Ort der Erzeugung (Generation shape) kann variieren



Initialwerte der Partikel können variieren
DefaultValue(i,t)=MeanValue(i,t)+Rand()*Var(i,t) mit t als Zeit und i=(Position, Orientierung, Größe, Farbe, Transparenz, Geschwindigkeit,...)



Dynamik:

- lokale Skripte können die Partikelattribute modifizieren
- kinematische oder dynamische Gesetze können auf diese Weise eingebaut werden

z.B. Explosion: von Kern aus nach oben, dann aufgrund der Schwerkraft nach unten, dabei wechselt die Farbe

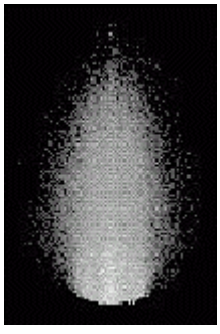
Termination von Partikeln:
Lebensalter und maximale Lebensdauer als zentrale Attribute
Partikel werden gelöscht, wenn

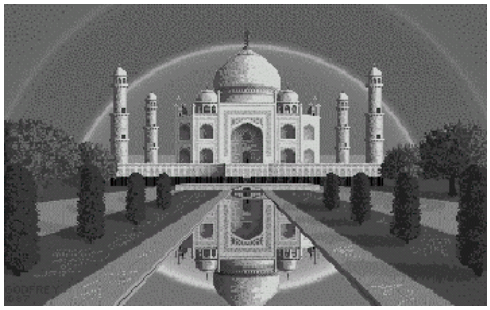
- ihre max. Lebensdauer erreicht ist
- sie sich aus dem relevanten Bereich entfernt haben und nicht zurückkehren können
- Attributwerte bestimmte Schwellenwerte überschritten haben (z.B. Farbwert nahe Schwarz oder transparent, Partikel zu klein etc.)
- besondere Ereignisse eintreffen (z.B. Kollision mit Szenenobjekt)

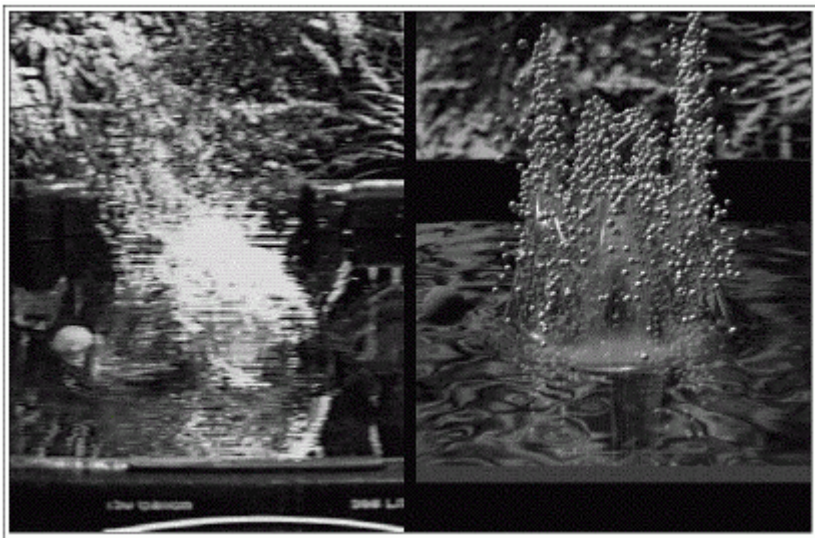
Rendern von Partikeln:
unterschiedliche Ansätze

- geometriebasiert: kleine polygonale 3D-Modelle für individuelle Partikel (aufwändig!)
- bildbasiert: Partikel werden direkt als Pixel dargestellt, Farbwerte werden addiert (schnelle Berechnung der Partikel, separates Rendering, Komposition mit restlicher Szene)

Beispiele:







Vergleich von Realität (links) und Simulation mit Partikelsystem (rechts), nach O'Brien et al.: Balls in a pond

Verhaltensbasierte Animation

Flocks, Herds, Schools
basiert auf Partikelansatz,
aber:

„Each boid is an entire polygonal object rather than a graphical primitive
Each boid has a local coordinate system.
There are a fixed number of boids - they are not created or destroyed.
Traditional rendering methods can be used because there are a small number of boids.
Boids behavior is dependent on external as well as internal state. In other words, a boid reacts to what other boids are doing around it. "

Boids (Bird objects)

Idee: Durch einfache lokale Regeln komplexes globales Verhalten

C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", Computer Graphics, vol. 21, no. 4, pp 25-34, 1987.

Regeln für Boids

3 Basisprinzipien für Boids:

(1) *Kollisionsvermeidung*: Ein Boid versucht, Kollisionen zu vermeiden
(2) *Geschwindigkeit annähem*: Ein Boid versucht, seine Geschwindigkeit der seiner Nachbarn anzupassen
(3) *Schwarmzentrierung*: Ein Boid versucht, sich stets in der Nähe von anderen Boids aufzuhalten.

Kollisionsvermeidung:
mit anderen Schwarmobjekten und mit "fremden" Szenen-objekten – gewisser Mindestabstand wird während der Bewegung gewährleistet
Strategien zur Kollisionsvermeidung (Reynolds 1988):

- wegsteuern von Oberflächen (virtuelles Kraftfeld benutzen), Weg durch Öffnung suchen oder an Oberfläche entlang fliegen
- von Objekt-Mittelpunkten wegsteuern
- virtuelle Fühler ausstrecken, um mögliche Oberflächen direkt auf Kollisionskurs zu entdecken
- Ansteuern der in die eigene Sichtebene projizierten Umriss-kanten (bzw. mit sicherem Abstand daran vorbei)

Schwarmverhalten (flock centering):

- jedes Boid will beim Schwarm bleiben
- mittlere Boids bleiben bei ihrer Richtung, äußere drängen nach innen
- wichtig: Wahrnehmungsbereich der Boids (endliche Distanz)
- keine Information über globalen Schwarm-Mittelpunkt,
- keine Gefolgschaft eines bestimmten "Führers"!

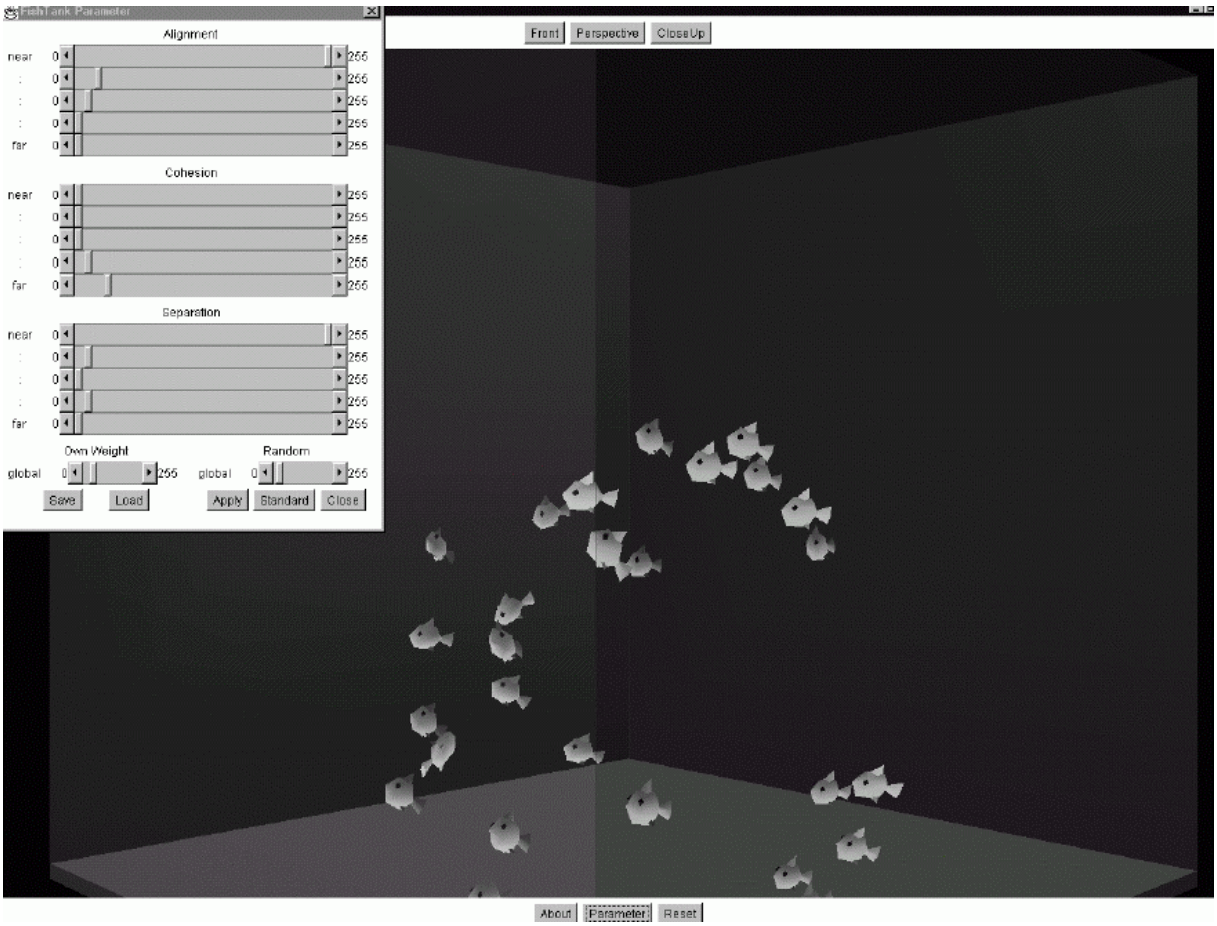
Navigation und Geschwindigkeit:

- Boid hat nur kleine Zahl von "wahrnehmbaren" Nachbarn, die zur Ermittlung der Durchschnittsgeschwindigkeit benutzt werden, an die das Boid sich anpasst

- Navigationsmodul kombiniert die 3 low-level Steuerprinzipien:
  Kollisionsvermeidung > Geschwindigkeitsannäherung > Schwarmzentrierung
- Soll-Geschwindigkeit wird benutzt, um Ist-Geschwindigkeit allmählich anzupassen, ebenso für die Richtung: Vermeidung abrupter Änderungen
- je nach den Gewichtungsfaktoren im Navigationsmodul können unterschiedliche Verhaltensweisen modelliert werden

Komplikation: Einführung von Raubvögeln
– zusätzliche Verhaltensregeln (Fernhalten von den Raubvögeln, Vermeiden ihrer Flugrichtung)

interaktive Systeme mit Möglichkeit der Regelung der Verhaltenspräferenz-Parameter

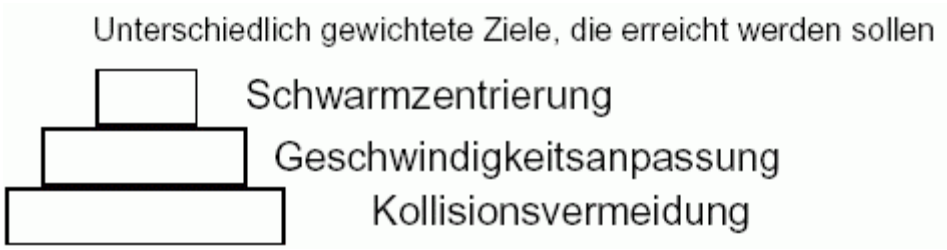


Agentenbasierte Animation

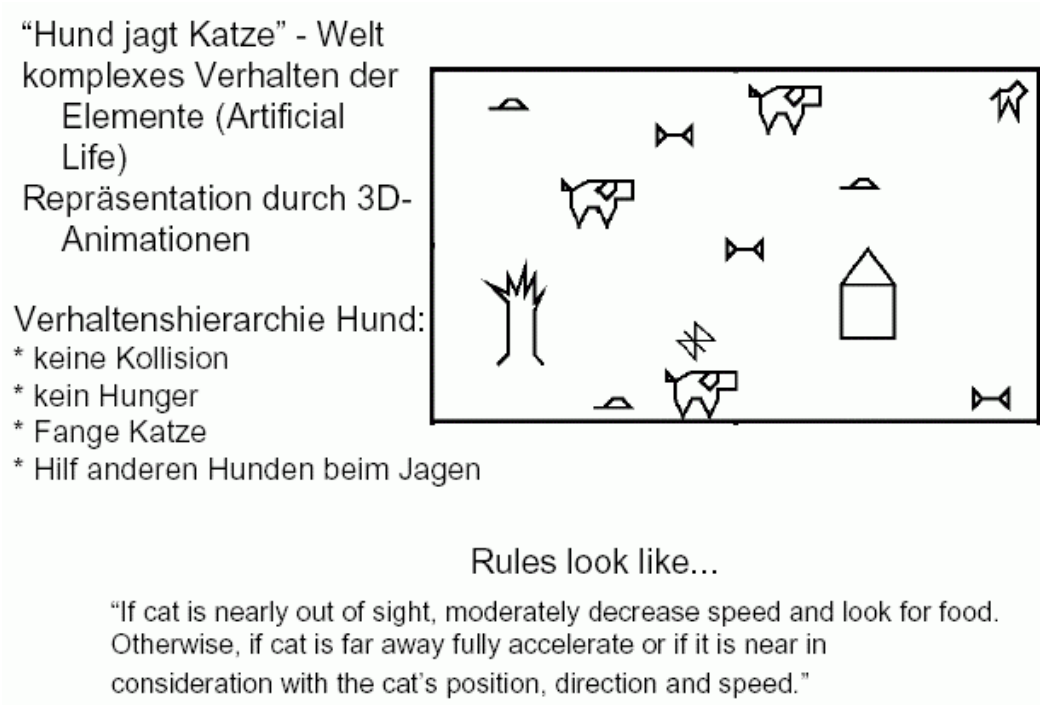
wenige Einheiten mit sehr komplexen Verhaltensweisen

- *autonom*: Verhalten wird durch den Agenten selbst bestimmt, ohne Kontrolle von außen
- *reaktiv*: Agent kann Ereignisse wahrnehmen und auf dieser Grundlage seine Aktionen abstimmen
- *proaktiv*: Agent reagiert nicht nur auf Reize, sondern kann von sich aus die Initiative ergreifen
- *sozial*: strukturierte Kommunikation mit anderen Agenten ist möglich

Orientierung u.a. an psychologischen Modellen (z.B. hierarchische Bedürfnisstrukturen), Rückgriff auf KI-Methoden und wissensbasierte Ansätze (Regelsysteme, neuronale Netze)



Beispiel:



Vergleich Partikel, Boids, Agents

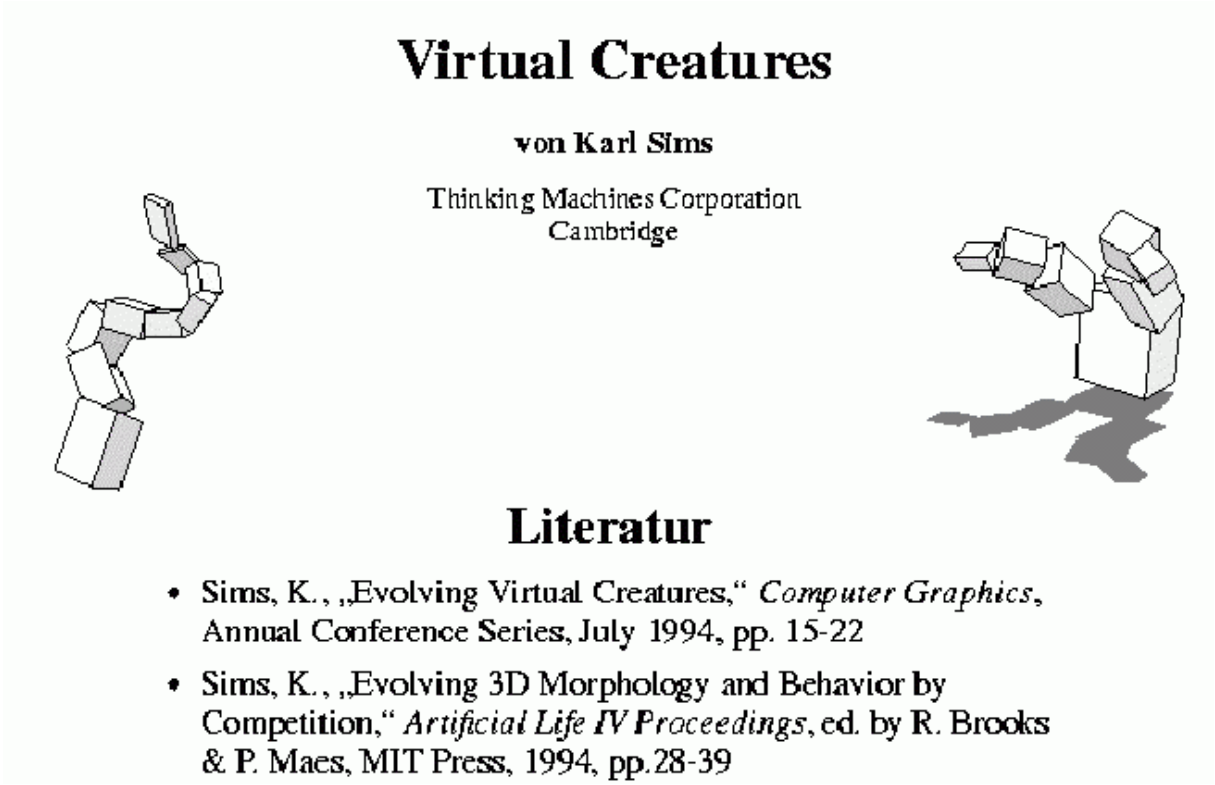
	Anzahl	Physikalische Regeln	Kollision	Kontrolle	Art der Strategien
Partikel	viele	ja	Nicht mit anderen Partikeln, sonst ja	niedrig	meist nur initiales Setzen von Parametern
Boids	mittel	teilweise	Vermeidung	mittel	meist durch einfache Regeln
Agenten	wenig	Weniger, meist eigenes Modell	Vermeidung, ggf. Planung für optimierte Wege	hoch	Komplexe Regeln, ggf erweitert durch Hierarchien und lernende Ansätze

"Virtual Creatures"

Anwendung wissensbasierter Techniken für Modellierung und Animation von virtuellen Kreaturen

- genetische Algorithmen
- künstliche neuronale Netze
- "Artificial Life" (AL)

wegweisende Artikel von Karl Sims (SIGGRAPH 1994 und Artificial Life IV - Konferenz 1994)



Ziel:
Es sollen 3D-Individuen entwickelt werden, die in einer virtuellen Welt spezielle Aufgaben besonders gut bewältigen

Struktur (Morphologie) und Verhalten (Regeln, Kontroll-strukturen) sind durch den genetischen Algorithmus variierbar

Grundschritte des genetischen Algorithmus:

1. Generiere Startpopulation mit Default-Parametern und Zufallsstreuung
2. Bestimme Qualität ("Fitness") der einzelnen Individuen
3. Selektion der Besten für Schritt 4
4. Erzeuge Nachkommengeneration – Anwendung von Mutationen und Crossing-Over (Modifikation und Mischung der Eltern-Parametersätze)
5. Gehe zu Schritt 2.

zu klärende Fragen:

- wie werden Morphologie und Funktionsweise (Verhalten) der Individuen codiert?
- wie wird die Fitness der Individuen ermittelt?
- wie werden die Codes an die Nachkommengeneration vererbt?

verschiedene Varianten für Morphologie und Verhalten möglich.

Kodierung der Kreaturen:Gestalt

Genotyp: Strukturbeschreibung
Phänotyp: Ausprägung

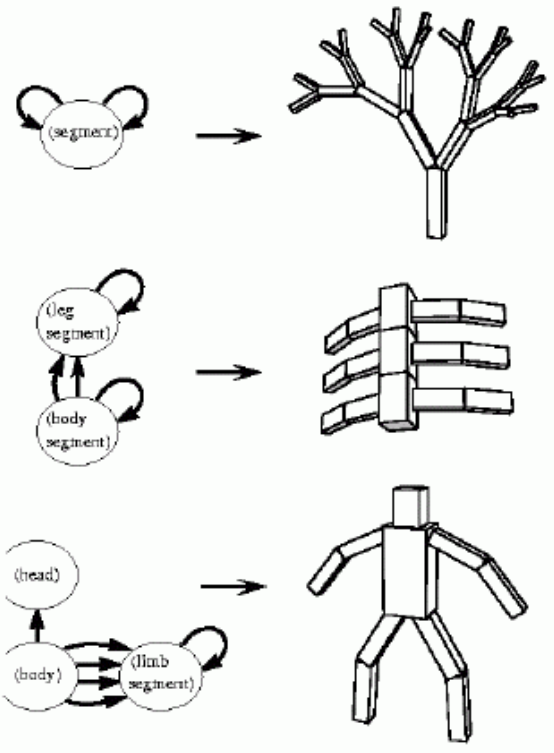
Die Erbinformationen über die Gestalt der Lebewesen wird als *gerichteter Graph* gespeichert.

Die **Knoten** entsprechen je einem **Körperteil** und enthalten folgende Informationen:

- Größe
- Gelenktyp
- Rekursionslimit
- Neuronen
- Verbindungen

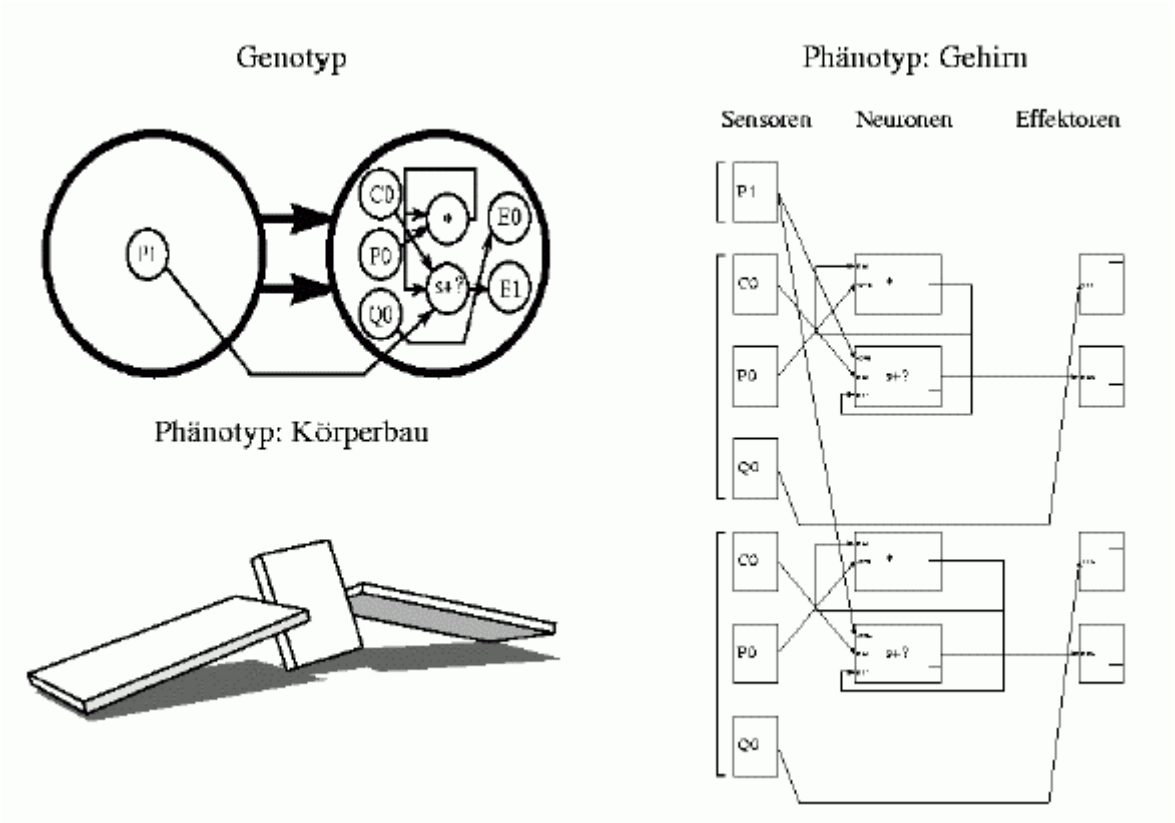
Die **Kanten** entsprechen den **Gelenken** und enthalten folgende Informationen:

- Position
- Orientierung
- Skalierungsfaktor
- Nur-Ende-Flag



Aufbau einer Kreatur:

- segmentiertes System mit Gelenken
- Winkelsensoren und Effektoren für jedes Gelenk
- neuronale Schaltkreise in jedem Segment + "Gehirn" (neuronales Netz)



Codierung durch Graphen:

- "äußerer Graph": Anordnung der Segmente (Glieder) und Gelenke
- "innerer Graph": Verschaltung der Sensoren, Neuronen und Effektoren

Evolution

Ein Genotyp wird in folgenden Schritten mutiert:

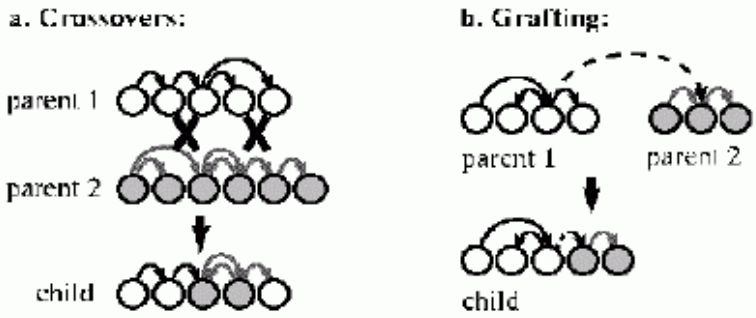
1. Interne Parameter der Knoten werden z. T. verändert.
2. Ein neuer Knoten wird hinzugefügt.
3. Parameter der Kanten werden z. T. verändert; evtl. zeigt die Kante anschließend auf einen anderen Knoten.
4. Neue Kanten werden evtl. hinzugefügt.
5. Nicht (mehr) befestigte Knoten werden entfernt

Das Verfahren besteht aus zwei Phasen:

Zuerst wird der äußere Graph bearbeitet, dann der innere.

Methoden:

1. [40%] Kopieren des Genotyps; dabei treten **Mutationen** auf
2. [30%] **Crossover** (evtl. mit Mutationen)
3. [30%] **Grafting** (evtl. mit Mutationen)

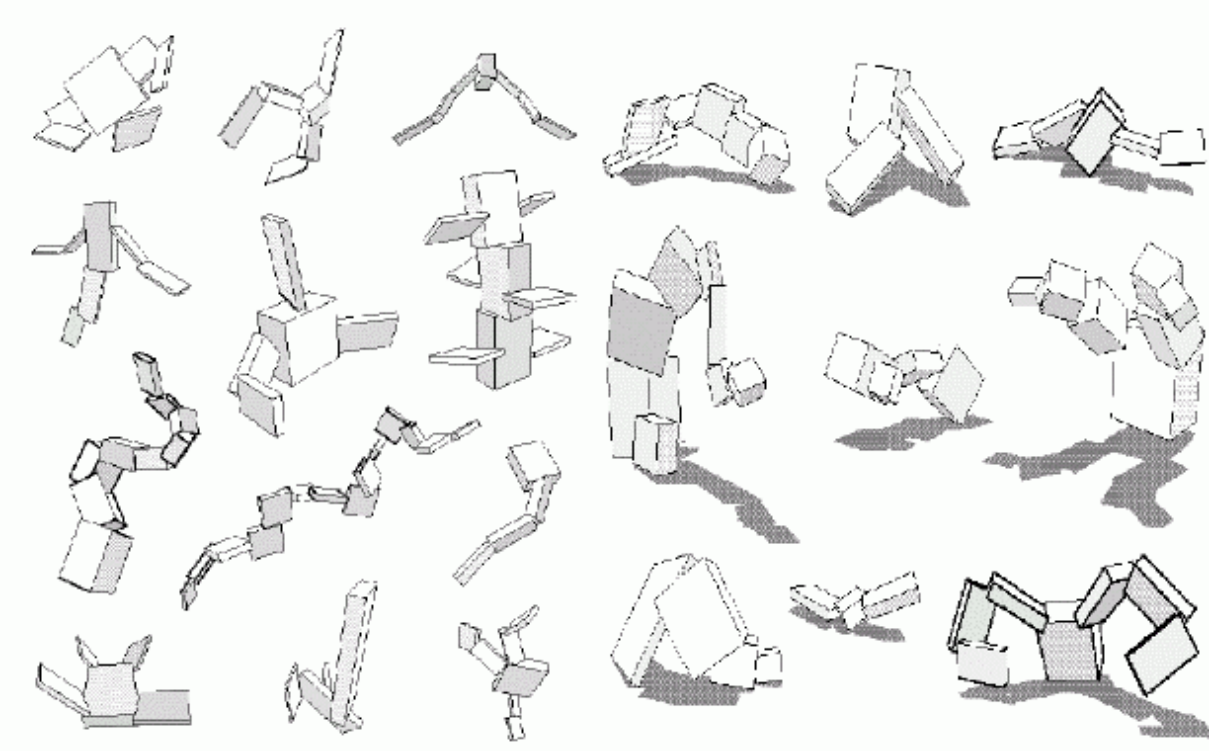


Der beschriebene genetische Algorithmus wurde angewendet, um Lebewesen zu generieren, die eine der folgende Aufgaben besonders gut beherrschen:

1. **Schwimmen**
2. **Fortbewegen an Land**
3. **Springen**
4. **Folgen** einer Lichtquelle
5. Sich im **Wettkampf** gegen ein anderes Individuum durchsetzen

Zur Implementierung:

- Populationsgröße: 300
- jeweils 50 - 100 Generationen pro Evolution
- jeweils max. 10 sec. pro Individuum und Generation zur Qualitätsbewertung
- **Laufzeit**: 3 - 4 Std. pro Evolution mit 100 Generationen auf einer CM-5 mit 32 Prozessoren



Aufgreifen des Grundprinzips und Umsetzung in ein Spiel:

Stephen (Steve) Grand 1996, "Creatures" (Cyberlife Inc.)

http://www.creatures.co.uk/