

# Evolution in Spielen



FH Wedel  
Informatik-Seminar  
„Spiele-KI“

Philip Mahler  
mi2177

# Einleitung

## Thematik:

- Evolution in Spielen
- Evolutionäre Algorithmen
- Evolution in der Natur



# Einleitung

## Gliederung:

- Back to the roots
- Bottom-up



# Gliederung

- Evolution und Genetik
- Evolutionäre Algorithmen
- Evolution in Spielen
- Zusammenfassung und Ausblick



# Gliederung

- **Evolution und Genetik**
  - Biologische Evolution
  - Evolutionsgenetik
  - Die Evolution als Optimierungsprozess



# Gliederung

- **Evolutionäre Algorithmen**
  - **Genetische Algorithmen**
  - **Evolutionsstrategien**
  - **Genetische Programmierung**
  - **Anwendungsbeispiel: GA Racer**



# Gliederung

- **Evolution in Spielen**
  - Evolutionäre Algorithmen als Werkzeug
  - Evolutionäre Algorithmen zur Laufzeit



# Evolution und Genetik

- **Gliederung:**
- **Biologische Evolution**
- **Evolutionsgenetik**
  - Aufbau des Genoms
  - Mechanismen der Vererbung
- **Die Evolution als Optimierungsprozess**
  - Der Suchraum
  - Suchstrategien



# Biologische Evolution

- **Feine Unterschiede zwischen Individuen**
- **Variationen an Nachkommen vererbbar**
- **Aufsummierung über Generationen**



# Biologische Evolution

- **Bewährte Eigenschaften finden sich in Nachfolgegenerationen bevorzugt wieder**
- **Mit der Zeit Vervollkommnung der Eigenschaften**
- **Anpassung an neue Lebensbedingungen**



# Biologische Evolution

- **Selektionsdruck**
  - Einwirkungen der Umgebung
  - Konkurrenz unter Individuen
  - Rivalität einer Spezies mit anderen Spezies
- **Antrieb zur Evolution**
- **Überleben der tauglichsten Individuen  
(survival of the fittest)**

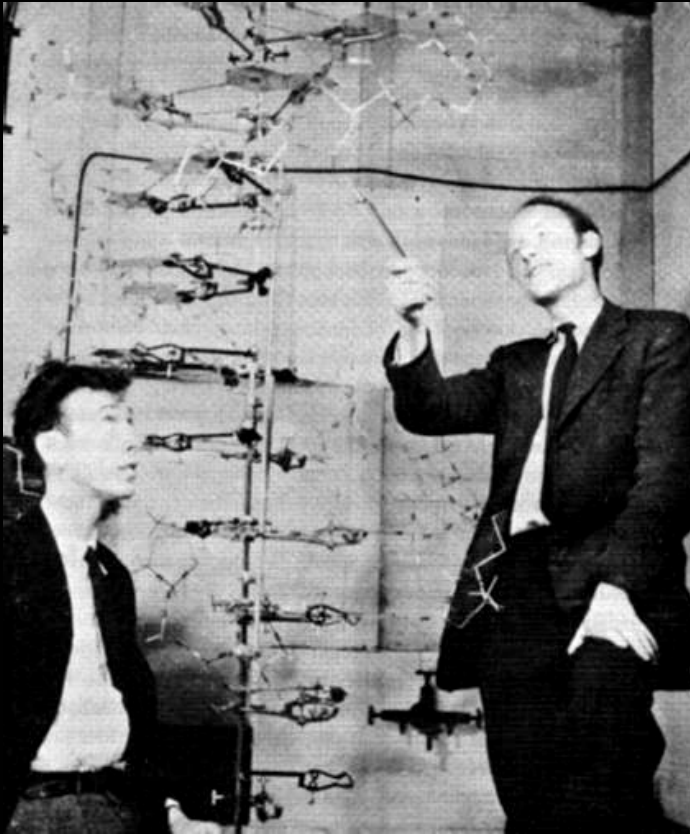


# Evolutionsgenetik

- Aufbau des Genoms
- Mechanismen der Vererbung



# Evolutionsgenetik



- Vorgänge und Strukturen sehr komplex
- Im Folgenden Stark vereinfachte Darstellung



# Aufbau des Genoms

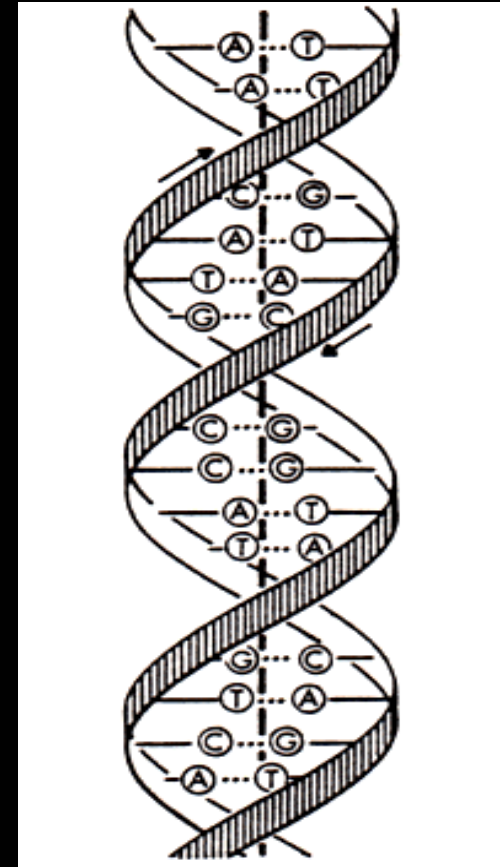
- Zellen sind die Grundbausteine des Lebens
- In jedem Zellkern befinden sich Chromosomen
- Die Chromosomen bestehen aus DNS
- Die DNS enthält Sequenzen von Nukleotidbasenpaaren
- Bestimmte Sequenzen bilden Informationseinheiten, die Gene



# Aufbau des Genoms

Codierung der Gene durch die vier Nukleotidbasen:

- Adenin
- Cytosin
- Guanin
- Thymin



# Aufbau des Genoms

Eine Base auf einem DNS-Strang hat einen Informationsgehalt von 2 Bit, da sie  $2^2 = 4$  Zustände annehmen kann. Ausgehend von  $3 \times 10^9$  Basenpaaren hat das Genom des Menschen einen Informationsgehalt von etwa 750 Megabyte.



# Aufbau des Genoms

- **Genotyp: die individuelle Codierung der Gene**
- **Phänotyp: die physisch ausgeprägten Merkmale**



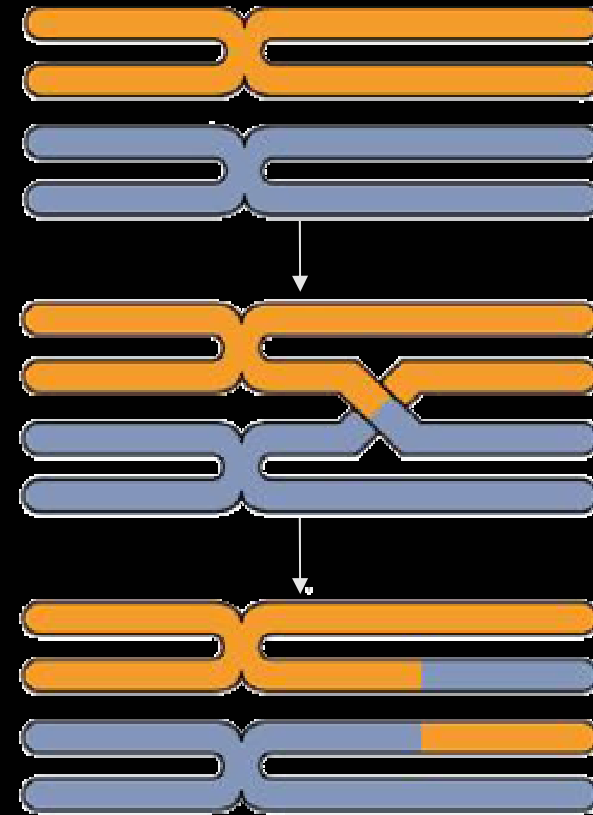
# Mechanismen der Vererbung

- Das Genmaterial der Eltern verschmilzt zum Genom des Nachkommens
- Folgende Mechanismen sorgen für eine genetische Variation
  - Rekombination
  - Mutation



# Rekombination

Die Kind Chromosomen werden aus abwechselnden Gensequenzen der Eltern Chromosomen gebildet (Crossover)



# Mutation

- Zufällige strukturelle Veränderung einer DNS Sequenz
- Reduziert in den meisten Fällen die Überlebenschancen des Nachkommens
- Wichtig, um die Entwicklung einer Spezies in eine andere Richtung zu lenken
- Tritt sehr selten auf



# Selektion

- **Günstige Mutations- und Rekombinationsergebnisse erhöhen die Chance eines Individuums...**
  - **in seiner Umgebung länger zu überleben**
  - **mehr Nachkommen zu zeugen**
  - **seine Gene in der Population zu verbreiten**
- **Prozess wird „survival of the fittest“ genannt**



# Evolution als Optimierungsprozess

## Das Ziel der Evolution:

aus der Menge aller Erbanlagen

diejenigen Erbanlagen finden

die eine Lebensform am besten dazu befähigen,  
sich im Überlebenskampf zu behaupten

→ Die Evolution ist ein Suchprozeß im Raum der  
genetisch möglichen Erbanlagen



# Der Suchraum

- diskreter Raum
- hochdimensional
- Gitterpunkte stellen alle möglichen Kombinationen der 4 Nukleotidbasen dar
  - Mensch:  $4^{3.000.000.000}$  Gitterpunkte
  - Gigantischer Suchraum!



# Suchstrategien

Um bei der Suche im (gigantischen) Suchraum erfolgreich zu sein, kombiniert die Evolution *einfache Strategien*:

- Gerichtete und ungerichtete Suche
- Serielle und parallele Suche



# Ungerichtete Suche

- **Die Mutation**
  - läuft rein zufällig ab
  - dient zur Erzeugung von Variationen um *lokale Optima* zu verlassen
  - verhindert dadurch ein Einpendeln bei suboptimalen Lösungen
  - M-Wahrscheinlichkeit gering, um ein Einpendeln bei optimalen Lösungen zu ermöglichen



# (Un)gerichtete Suche

- **Die Rekombination**
  - gerichteter / ungerichteter Suchcharakter
  - erfolgt an zufälligen Stellen zwischen den Chromosomen
  - nah beieinander liegende und funktional verbundene Gene werden seltener getrennt
  - durchmischt das Erbgut statistisch mit „Gewichtung“



# Gerichtete Suche

- **Die Selektion**
  - **Steuerung der Evolution**
  - **Entscheidet, welche Phänotypen -> Genotypen sich stärker vermehren und weniger stark**
  - **Lenkt die Suche Richtung Optimum**



# Serielle Suche

- Evolution ist ein zeiteffizienter Optimierungsprozess
- Tiefensuche (geringe Reproduktionszeiten)
  - Kurze Generationenfolge, um in möglichst kurzer Zeit möglichst oft zu rekombinieren, mutieren und selektieren
  - Durchdringt den Suchraum möglichst schnell



# Parallele Suche

- Evolution ist ein zeiteffizienter Optimierungsprozess
- Breitensuche (hohe Reproduktionsquote)
  - Erzeugung möglichst vieler Individuen zur gleichen Zeit, um die Evolutionsdauer zu minimieren
  - Breitet sich im Suchraum möglichst großflächig aus



# Parallele Suche

- **Parallelisierbarkeit:**
  - Simultane Überprüfung mehrerer Individuen auf ihre Performance / Fitness
  - Ermöglicht, den Suchraum zeitgleich von mehreren Punkten aus zu durchsuchen
  - P++, optimale Punkte innerhalb des Suchraums zu erreichen
  - P--, suboptimale Pfade zu verfolgen



# Evolutionäre Algorithmen

## Gliederung:

- Motivation / Übersicht
- Genetische Algorithmen
- Evolutionsstrategien
- Anwendungsbeispiel: GA Racer



# Motivation

**Evolutionäre Algorithmen werden verwendet, um Probleme zu lösen, die auf klassischem Wege nicht oder nur schwer lösbar sind, etwa wegen hoher Komplexität, Nichtlinearität oder zu großem Suchraum.**



# Übersicht

- Nutzen die Basistechniken der Evolution
- Bilden einer „Lösungs-Population“
- Lösungen eines Problems werden zu Eltern und erzeugen in ihren Nachkommen neue Lösungen



# Übersicht

**Zwei Modelle:**

- **Genetische Algorithmen**

John Holland, 1960er, University of Michigan

- **Evolutionstrategien**

Ingo Rechenberg, 1960er, TU Berlin



# Übersicht

**Zwei algorithmische Modelle:**

- **Genetische Algorithmen**

Codierung, Verarbeitung und Weitergabe der Informationen durch die Evolution

- **Evolutionstrategien**

Evolution als Vorlage für Entwicklung von Such- und Optimierungsverfahren



# Genetische Algorithmen

- **Basis-Algorithmus ist für alle Probleme gleich**
- **Problemspezifisch:**
  - **Größe der Population**
  - **Codierung der Erbinformationen**
  - **Bewertung der gefundenen Lösungen**
  - **Auswahl der verschiedenen Detail-Implementierungen**



# Codierung

- **Erbinformation: binär codiert**
  - Grundmenge:  $M := \{0, 1\}$
- **Individuum: binärer Vektor der Länge n**
  - $X = (x_1, x_2, \dots, x_n)$  mit  $x_i$  aus  $M$
- **Population: Menge von Binärvektoren**
  - Teilmenge von  $M^n := \{0, 1\}^n$



# Codierung

**Beispiel:**

$$X = (1, 0, 1, 1, 1, 0)$$

$$Y = ((1, 1, 0), 1)$$

→ Gene als Einzelposition und als Sequenz

**Allel: Wert eines Gens (Variablenbelegung)**

$x_2$  hat das Allel 0

$x_5$  hat das Allel 1



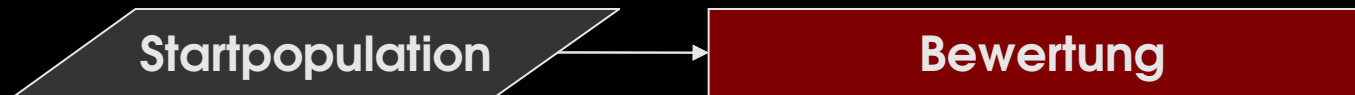
# Startpopulation

Startpopulation

- Es werden  $k$  Individuen zufällig erzeugt
- Jedes Individuum ist ein Binärvektor der Länge  $n$ .



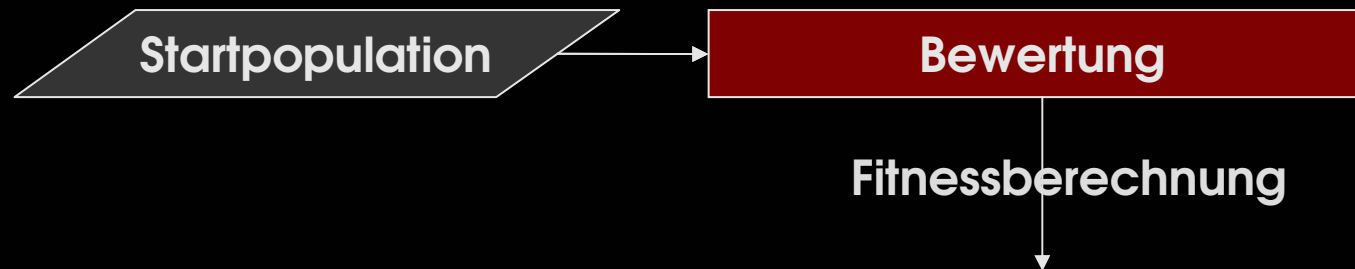
# Bewertung



- **Jedes Individuum wird bewertet**
- **Die Bewertung misst die Performance  $p_i$  unabhängig vom Rest der Population**
- **Performance ergibt sich aus dem Phänotyp**
- **Problemspezifische Funktion zur Ermittlung**



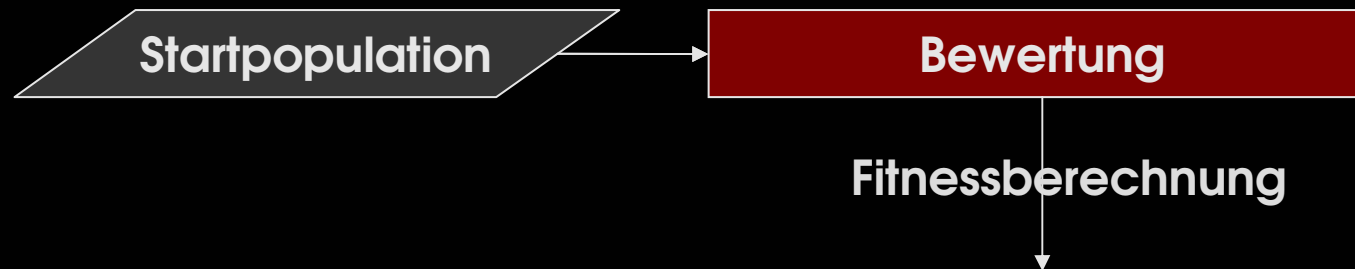
# Fitnessberechnung



- Der Fitnesswert  $f$  gibt für jedes Individuum den Rang in der Population an
- Berechnung:  $f_i = p_i / P$ 
  - $p_i$  : Performance des Individuums  $i$
  - $P$  : mittlere Performance aller Individuen



# Fitnessberechnung



Die Fitnessfunktion teilt ein:

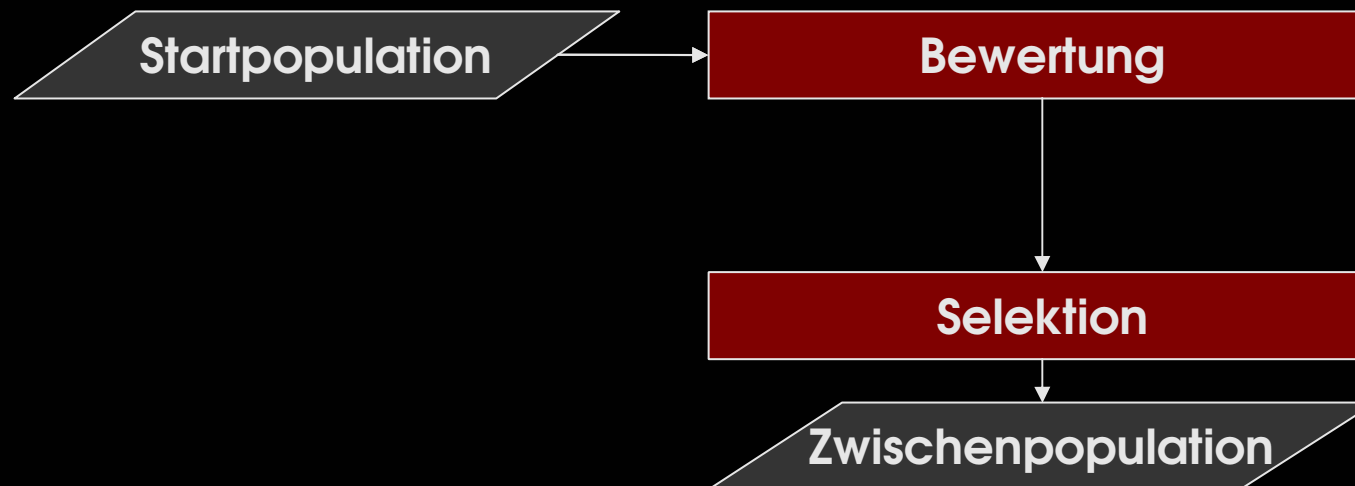
$f_i \sim 1$  : Individuen mit durchschnittl. Performance

$f_i > 1$  : taugliche Individuen

$f_i < 1$  : weniger taugliche Individuen



# Selektion



- **Konstruktion der Zwischenpopulation mit Kopien der Startpopulation**



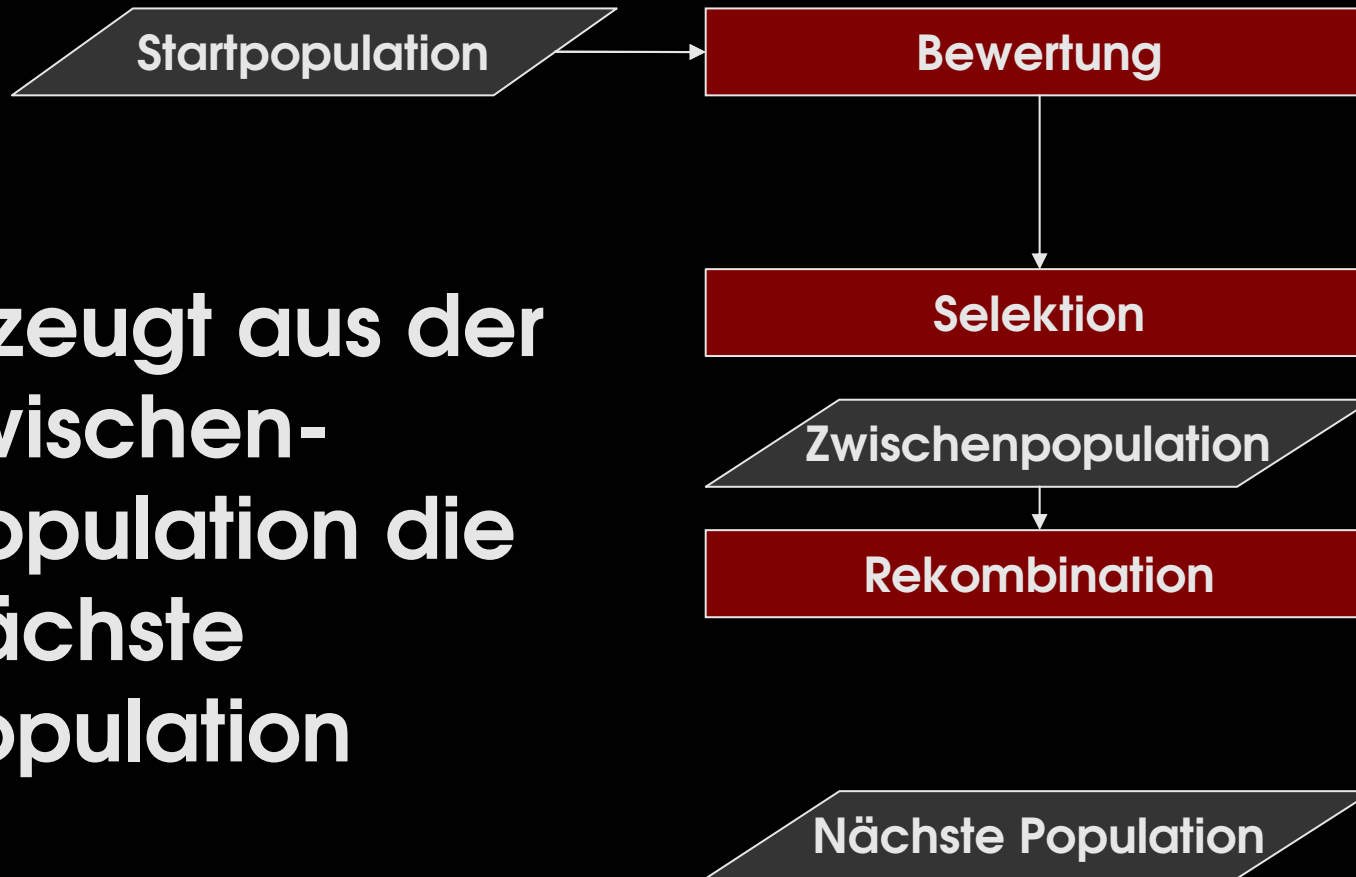
# Selektion

- **Rouletteverfahren**
  - Fläche proportional zur Fitness
- **Remainder Stochastic Sampling**
  - Vorkommastelle von  $f_i$  : direkte Kopien
  - Nachkommastelle von  $f_i$  : Chance für weitere Kopie
- **Elitismus**
  - Die  $n$  besten Individuen werden direkt in die Zwischenpopulation übernommen



# Rekombination

- Erzeugt aus der Zwischenpopulation die nächste Population



# Rekombination

- Zufällige Auswahl von 2 Binärvektoren aus der Zwischenpopulation
- Rekombination mit Wahrscheinlichkeit  $P_c$  zu zwei neuen Binärvektoren
- Einfügen in die nächste Population (vorher: Mutation)



# Rekombination

Verschiedene Verfahren:

Single / Double / Multi Point Crossover

Single Point Crossover an der Position  $i$ :

$$X_E = (x_1, x_2, \dots, x_n) \quad \text{und} \quad Y_E = (y_1, y_2, \dots, y_n)$$



$$X_K = (x_1, x_2, \dots, x_{i-1}, y_i, \dots, y_n) \quad \text{und} \quad Y_K = (y_1, y_2, \dots, y_{i-1}, x_i, \dots, x_n)$$



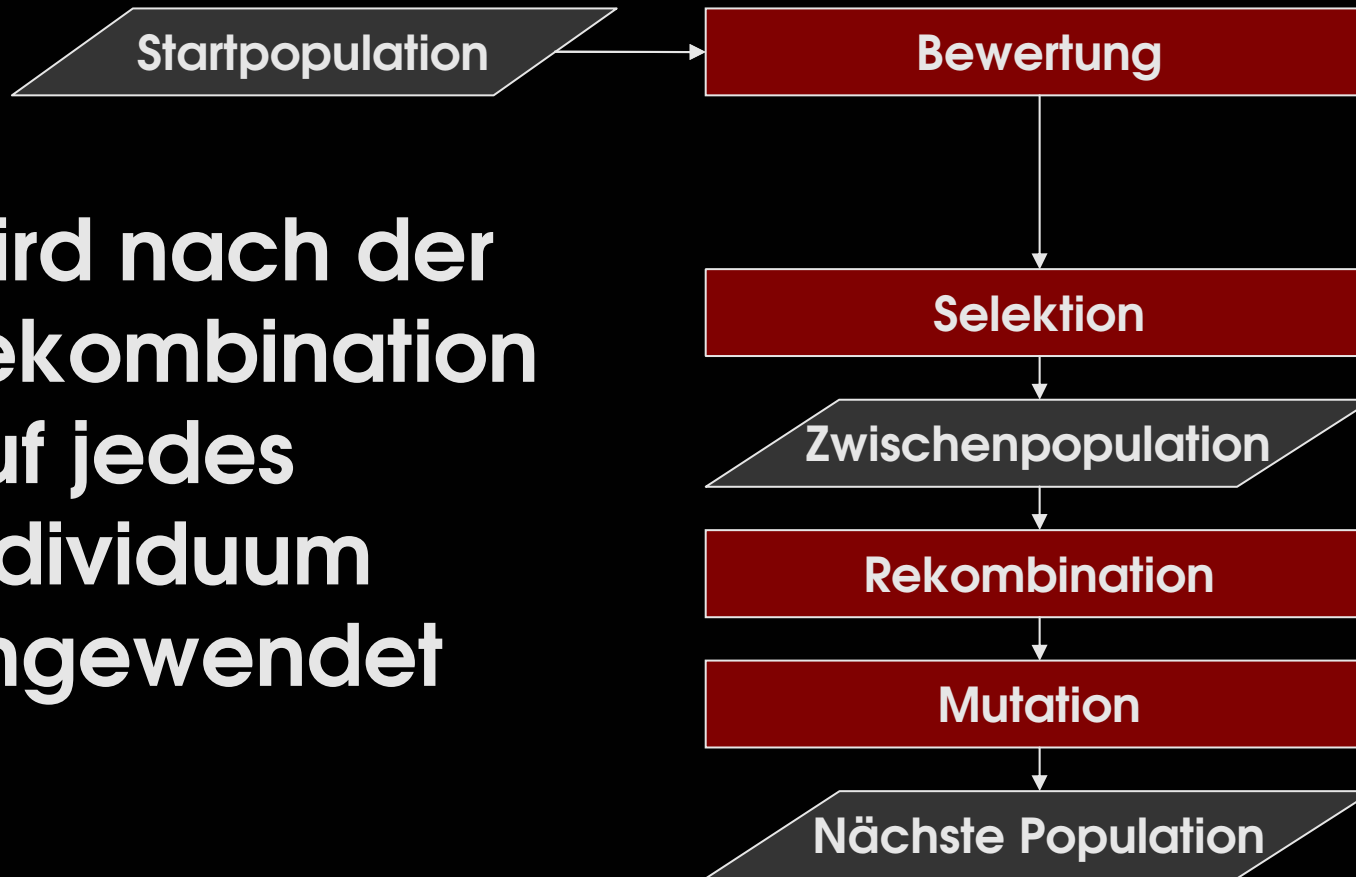
# Rekombination

- Durch Rekombination werden neue Testpunkte im Suchraum erzeugt
- Double Point Crossover erzielt meist beste Ergebnisse



# Mutation

- Wird nach der Rekombination auf jedes Individuum angewendet



# Mutation

- Jedes Bit aus der Population mutiert mit einer Wahrscheinlichkeit  $P_M$
- Typische Mutationsrate:  $P_M < 1\%$
- Einfachste Art: Invertierung des betreffenden Bits

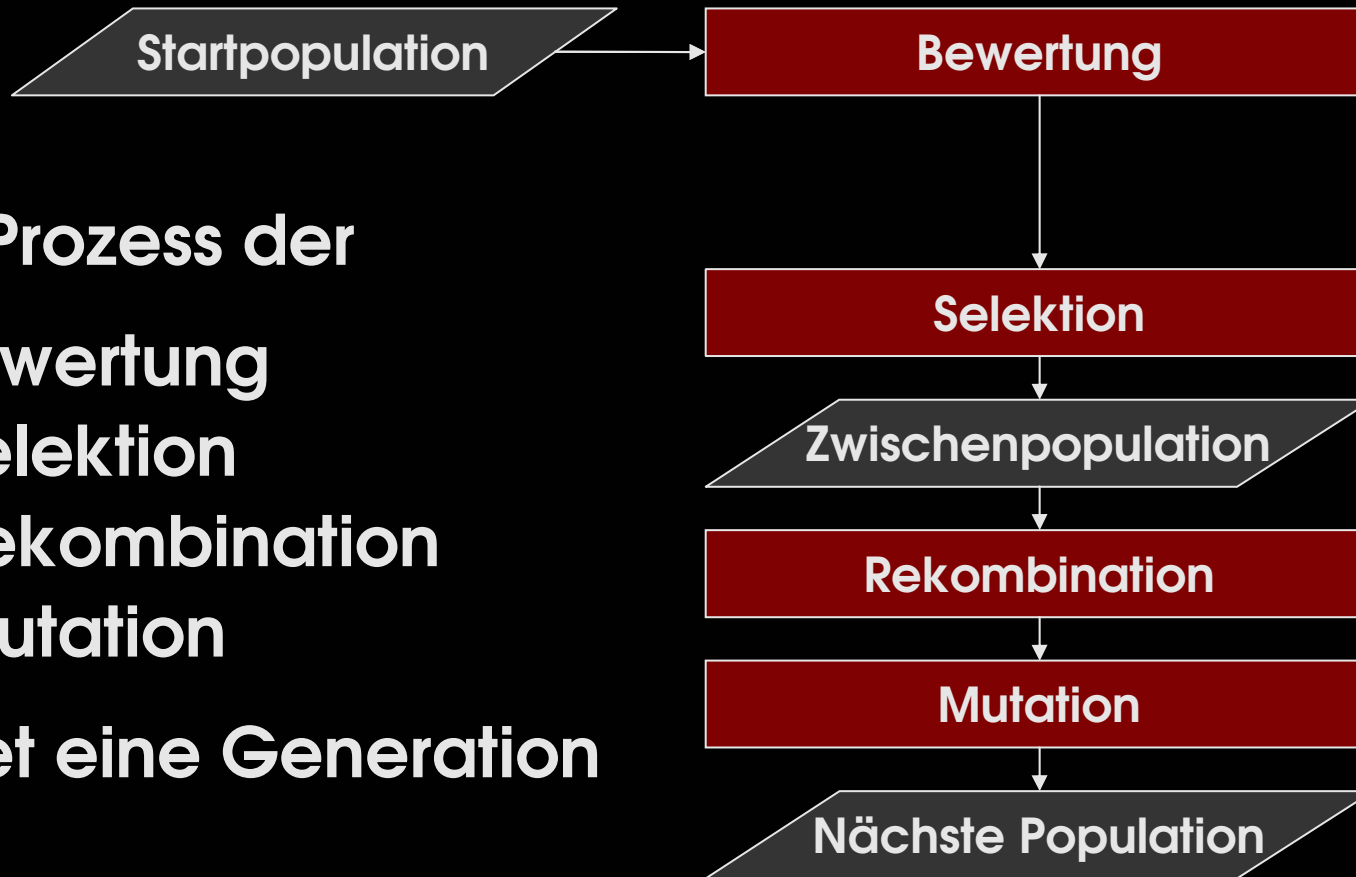


# Generation

Der Prozess der

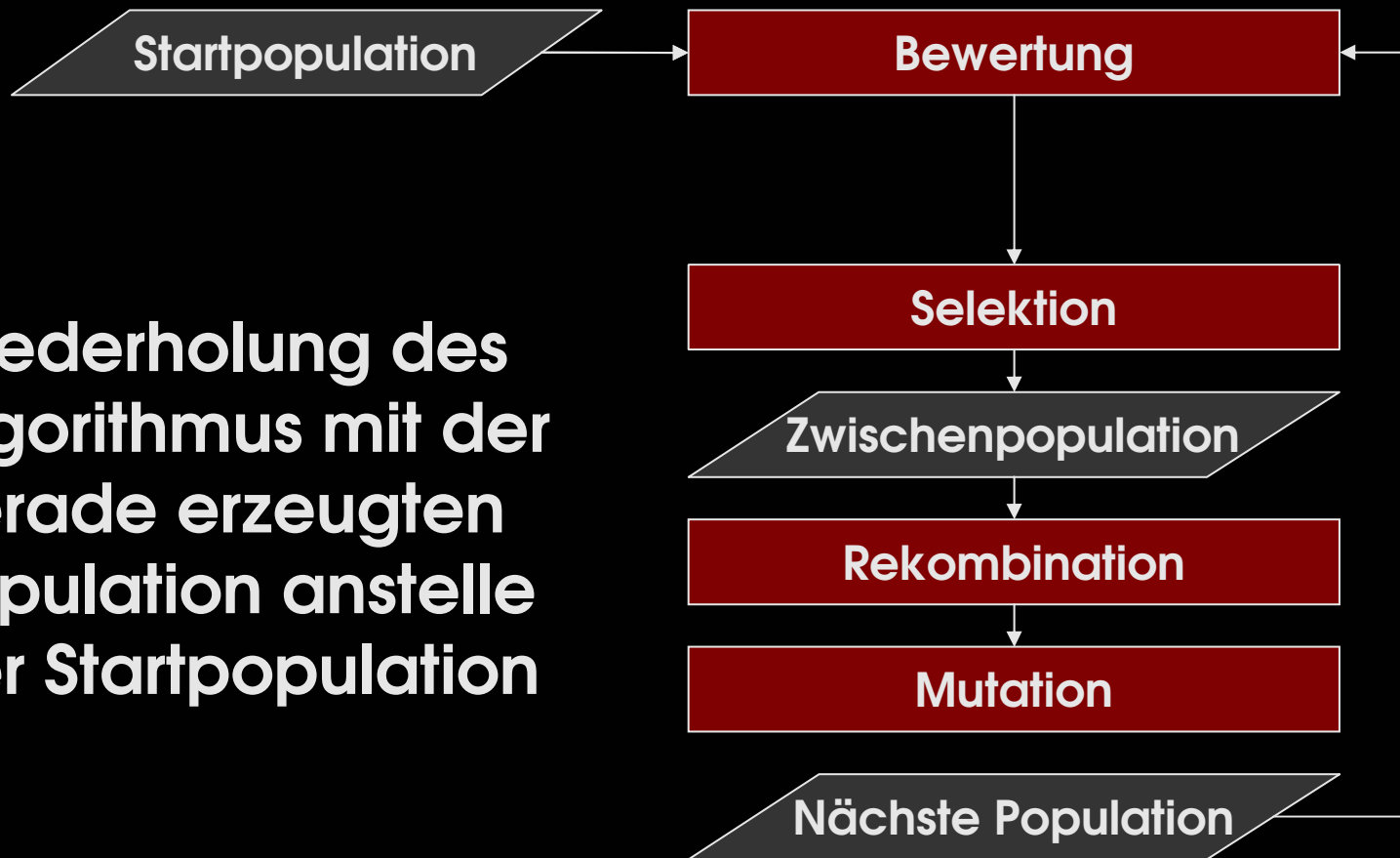
- Bewertung
- Selektion
- Rekombination
- Mutation

bildet eine Generation



# Wiederholung

- Wiederholung des Algorithmus mit der gerade erzeugten Population anstelle der Startpopulation



# Abbruch



## Testen auf:

- Suchziel gefunden
- Generationszahl  $G$  überschritten
- Laufzeit  $T$  überschritten



# Ablaufschema



# Evolutionsstrategien

Unterschiede zu den Genetischen  
Algorithmen:

- Codierung
- Selektion
- Rekombination
- Mutation
- Selbstadaption



# Codierung

## Unterschiede zu den Genetischen Algorithmen:

- Jedes Gen wird als reelle Zahl dargestellt
- Stellt die kompakteste Form der Codierung dar



# Selektion

## Unterschiede zu den Genetischen Algorithmen:

- Elter-Individuen werden zufällig gewählt, unabhängig von ihrer Fitness oder Bewertung
  - aus den erzeugten Nachkommen werden die besten ausgewählt
- „Survival of the fittest“ bei den Nachkommen



# Mutation

## Unterschiede zu den Genetischen Algorithmen:

- Kann nicht einfach als Bit-Flip ausgeführt werden
- Möglichkeiten:
  - Addition einer Zufallszahl
  - Multiplikation mit einer Zufallszahl
  - Setzen auf einen Minimal- oder Maximalwert



# Rekombination

## Unterschiede zu den Genetischen Algorithmen:

- In vielen Formen der Evolutionsstrategien nicht vorgesehen
- Mutation spielt dann die Hauptrolle, häufig mit höheren Raten



# Selbstadaption

## Unterschiede zu den Genetischen Algorithmen:

- Selbstständige Anpassung der Mutationsschrittweite
- Reaktionsfähigkeit auf verschiedene Faktoren, z.B.
  - Beschleunigung der Konvergenz in Optimumsnähe
  - Neuorientierung im Suchraum
- Bei GA nicht ohne umständliche Codierung und nicht problemunabhängig realisierbar



# Genetische Programmierung

- **Grundlegende Arbeit von John Koza**
- **Individuen werden als Programme interpretiert**
- **Suchraum voller Programme**
- **Allgemeinerer Ansatz als bei anderen EAs**
- **Suchraum in der Regel sehr viel größer**
- **Häufiges Problem: größter Teil des Suchraums enthält „unsinnige“ Programme**



# Genetische Programmierung

Geeignete Programmiersprachen:

- LISP (Programm liegt als Baumstruktur vor)
- Assembler (lineares Programm)



# Anwendungsbeispiel

## GA Racer

- Fahrzeuge versuchen, eine Strecke so weit wie möglich zu überwinden
- Fahrzeuge entwickeln dazu eine Sequenz von Lenk- und Beschleunigungskommandos



# Anwendungsbeispiel

## GA Racer

- Kommandosequenz:  
Richtung | Beschl. | Richtung | Beschl. | ...
- Jedes Gen wird durch eine reelle Zahl zwischen 0 und 1 repräsentiert
- Fahrzeug durchläuft sein Genom in 2er Schritten
- Fahrzeug passt seine Position in jedem Aktualisierungsdurchgang an



# Evolution in Spielen

## Motivation: Anwendung von EA

- bei der Entwicklung, um Optim.aufgaben zu lösen
  - KI eines Gegners trainieren
  - Trainieren von neuronalen Netzen
  - ...
- zur Laufzeit, um KI zu implementieren
  - Strategiespiele
  - Spiele mit „Evolutionfaktor“
  - ...



# Evolution in Spielen

## Gliederung

- EA als Werkzeug
  - Platzoptimierung bei der Texturspeicherung
  - Parameteroptimierung bei Rennspielen
- EA zur Laufzeit
  - Creatures
  - Cloak, Dagger and DNA

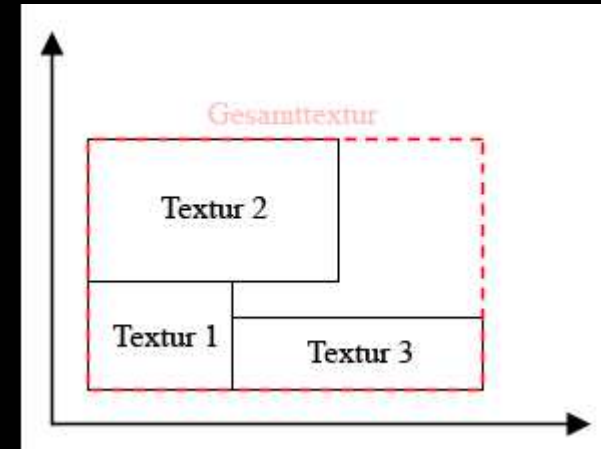


# EA als Werkzeug



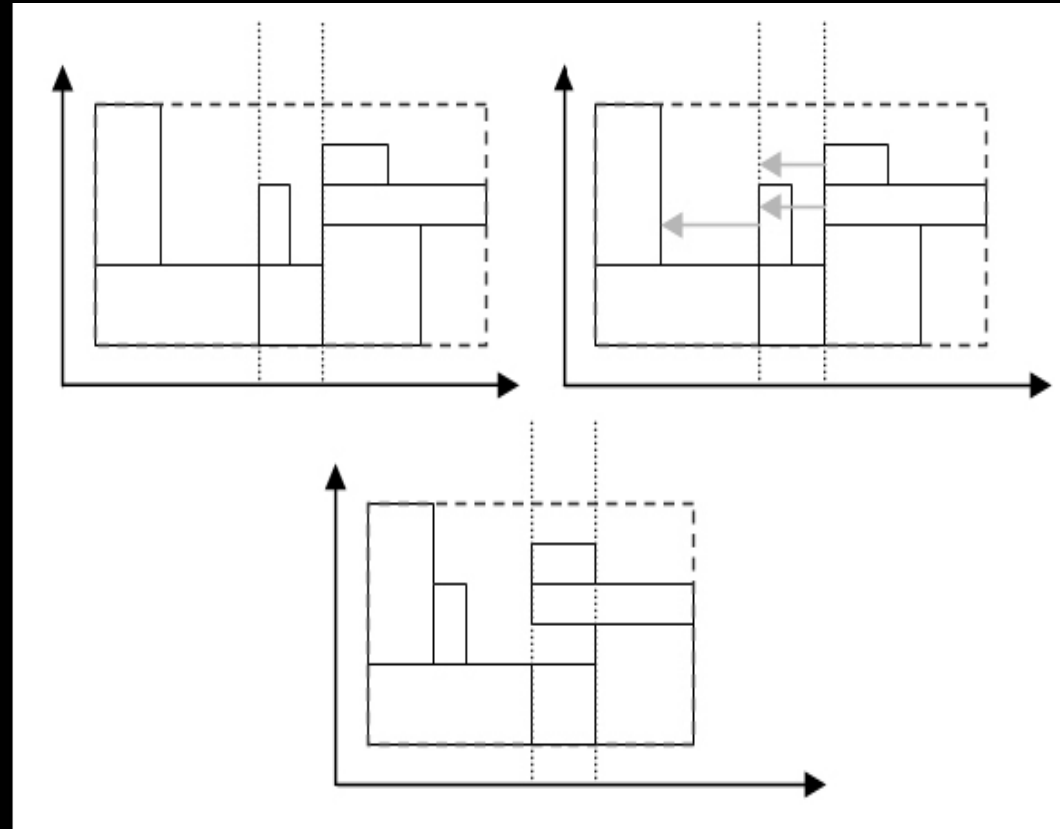
# Texturspeicherung

- Diplomarbeit von Karsten Kaul, FH Wedel 2005
- Texturen zur Darstellung 3D-Objekten
- Speicherung mehrerer kleiner Texturen in einer großen
  - Problem: Platzsparende Anordnung
  - Lösung: Evolutionärer Algorithmus



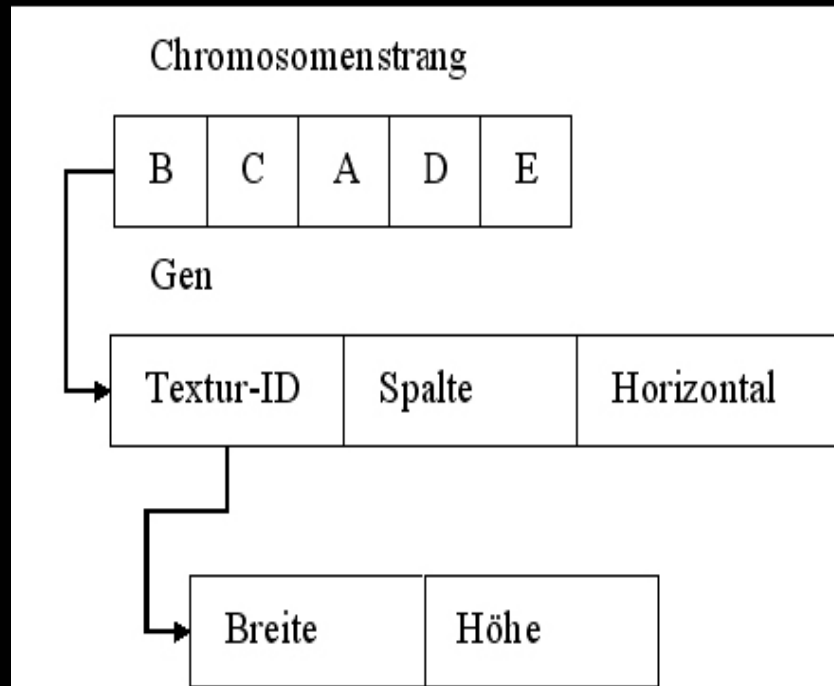
# Texturspeicherung

- Anordnung in Spalten
- Optimierung durch Verschieben und Drehen



# Texturspeicherung

**Codierung:**



**Besonderheit:**

**Implementierung  
von Teilen des  
Algorithmus auf  
Shaderhardware  
unter Ausnutzung der  
Parallelisierbarkeit  
von EA**



# KI in Rennspielen

- Seminarvortrag von Yannick Block
- Trainieren der KI für computergesteuerte Fahrzeuge
- Problem: Feineinstellungen für Balance, Bremsweg etc. finden



# KI in Rennspielen

- Lösung ohne EA:
  - Parameter per Hand einstellen
  - KI fahren lassen
  - erneute Anpassung anhand Rundenzeit
  - Wiederholung bis Rundenzeit zufriedenstellend



# KI in Rennspielen

- Lösung mit EA:
  - Jeder Parameter wird auf ein Gen abgebildet (reell)
  - Jedes Gen erhält Beschränkungen (Min- / Maximalwert)
  - Startpopulation mit zufällig erzeugten Individuen
  - Auswahl der passenden Mechanismen (Elitismus, ...)
  - Aufstellen der Bewertungsfunktion (Rundenzeit)
  - Festlegen des Abbruchkriteriums
  - Starten und was anderes machen (eine Woche lang)



# EA zur Laufzeit



# Creatures

- **Norns:**
  - sollen lebende Kreaturen darstellen
  - besitzen komplex nachgebildete Biochemie, Organsystem, lernfähiges Gehirn
  - haben ein dies alles beschreibendes Genom
- **Mögliche Ziele des Spiels:**
  - eine dauerhafte, gesunde Norn-Population erhalten
  - gezieltes Züchten einer einzelnen Kreatur



# Creatures

## Genetischer Code: „Digital DNA“

- 771 Gene (Creatures 2)
- Die Hälfte codiert das Aussehen und die Haltung / Bewegung
- Der Rest codiert Biochemie, Organe, Gehirnstruktur
- Gespeichert wird eine Menge von Instruktionen, die Strukturen beschreiben, welche ein System aufbauen, aus dem sich Verhalten herausbilden kann



# Creatures

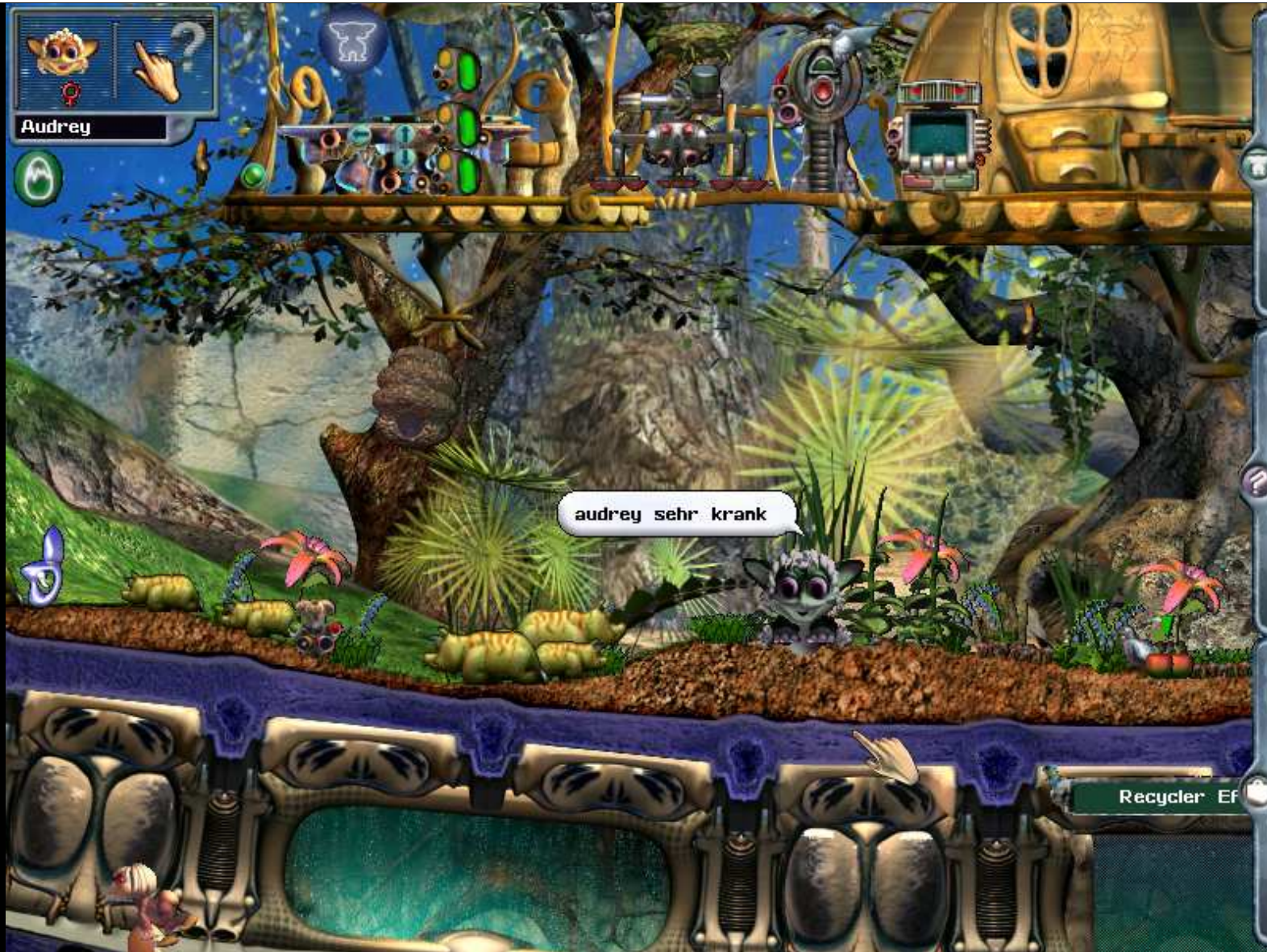
**DNA wird bei der Fortpflanzung weitergegeben**

- **Alle Genetische Mechanismen vorhanden**
- **Unerwartete Variationen möglich, z.B. acht Lungen...**

**Selektion nicht nur auf Basis des Genotyps:**

- **Überleben und Entwicklung der Norns hängt vom Geschick des trainierenden Spielers ab**





# Cloak, Dagger & DNA

- Taktisches Echtzeit-Eroberungsspiel
- Spielkarte ist in Regionen unterteilt, von denen einige Fabriken enthalten
- Fabriken im Besitz eines Spielers ermöglichen die Produktion von Einheiten
- Zwei Einheiten:
  - Armeen für Angriff und Verteidigung
  - Spione zur Erforschung gegnerischer Gebiete
- Kampf abhängig von der Einheitenzahl in einem Gebiet
- Bis zu vier Spieler (Comp/Mensch) in jeder Kombination



# Cloak, Dagger & DNA

- Jeder Gegner besitzt Satz aus Regeln, die das Spielverhalten steuern
- Nach jedem Kampf Bewertung des verwendeten Regelsatzes und Ablegen im Genpool
- Menge von Mechanismen wie Rekombination, Mutation und Selektion
- Regelsätze der Genotypen können editiert werden
- Verbesserung der Computergegner nach >1000 Kämpfen...



# Zusammenfassung

- EA stellen robuste Suchmethode für große, komplexe oder wenig bekannte Suchräume und nichtlineare Probleme dar
- Verlagern die Arbeit auf die Seite des Computers, der durch ein gezieltes Suchen selbst gute Lösungen produziert.
- Vergleichsweise einfache Grundmechanismen, leicht zu verstehen



# Zusammenfassung

- Auch wenn die Grundstrukturen immer gleich bleiben, unterscheiden sich doch die Details von Problem zu Problem
- Erfahrung nötig, um geeignete Werkzeuge unter den vielen möglichen Varianten für ein spezifisches Problem auszuwählen
- Lange Gensequenzen und Populationen mit hoher Individuenzahl (meist der Fall): hoher Ressourcenverbrauch (Zeit, Speicher und Rechenleistung)



# Ausblick

- Aufgrund des hohen Ressourcenverbrauchs zur Laufzeit finden EA bislang kaum Anwendung innerhalb von Computerspielen
- Deshalb häufig Nutzung als Werkzeug bei der Entwicklung
- Denkbar, EA mehr in Spielen zu verwenden, wenn geschickte Implementierungen verwendet werden, z.B. :
  - Berechnung in Spielabschnitten, in denen die Prozessorlast gering ist
  - Nutzung der guten Parallelisierbarkeit der EA, wie bei Nutzung der Shaderhardware



# Ende des Vortrags

