

# **EVOLUTION IN SPIELEN**

Informatik-Seminar: Spiele-KI  
Sommersemester 2007

Fachhochschule Wedel

Autor: Philip Mahler

Erstellt am: 27.06.2007

# INHALTSVERZEICHNIS

<b>Einleitung</b> .....	<b>2</b>
<b>Evolution und Genetik</b> .....	<b>3</b>
1.1    Biologische Evolution.....	3
1.2    Evolutionsgenetik.....	3
1.2.1    Aufbau des Genoms.....	3
1.2.2    Mechanismen der Vererbung.....	4
1.3    Die Evolution als Optimierungsprozess.....	5
1.3.1    Der Suchraum.....	5
1.3.2    Suchstrategien.....	5
<b>Evolutionäre Algorithmen</b> .....	<b>7</b>
2.1    Einleitung.....	7
2.2    Genetische Algorithmen.....	8
2.2.1    Codierung.....	8
2.2.2    Startpopulation.....	8
2.2.3    Bewertung und Fitness der Individuen.....	8
2.2.4    Ausführung des Algorithmus.....	9
2.2.5    Konstruktion der Zwischenpopulation: Selektion.....	9
2.2.6    Konstruktion der nächsten Generation: Rekombination.....	10
2.2.7    Mutation.....	11
2.2.8    Abbruchbedingung.....	11
2.3    Evolutionstrategien.....	12
2.3.1    Codierung.....	12
2.3.2    Selektion.....	12
2.3.3    Mutation.....	12
2.3.4    Selbstadaption.....	12
2.4    Ein Anwendungsbeispiel: GA Racer.....	13
<b>Evolution in Spielen</b> .....	<b>14</b>
3.1    Evolutionäre Algorithmen als Werkzeug in der Spielentwicklung.....	14
3.1.1    Platzoptimierung bei der Texturspeicherung.....	14
3.1.2    Parameteroptimierung bei Rennspielen.....	14
3.2    Verwendung von Evolutionären Algorithmen in Spielen zur Laufzeit.....	15
3.2.1    Creatures.....	15
3.2.2    Cloak, Dagger and DNA.....	16
<b>Zusammenfassung und Ausblick</b> .....	<b>18</b>
<b>Literaturverzeichnis</b> .....	<b>19</b>

## EINLEITUNG

Die Natur hat bislang 500 Milliarden verschiedene Lebensformen hervorgebracht. Sie schafft es, dass sich ständig Milliarden von Lebensformen an ihre wechselnde Umwelt anpassen [Schö]. Der Mensch versucht, diese Prozesse nachzubilden.

Aus Sicht der Mathematik und Informatik stellt die Evolution ein extrem leistungsstarkes Optimierungsverfahren dar. Die Arbeitsweise der Evolution ist bei weitem noch nicht vollständig erschlossen. Aber schon das Verständnis der Grundprinzipien der Evolution wie Selektion, Rekombination und Mutation versetzt uns in die Lage, diese auf Computern zu modellieren, simulieren und als mächtiges Werkzeug zur Lösung schwieriger Probleme dort einzusetzen, wo konventionelle Such- und Optimierungsverfahren scheitern oder aus anderen Gründen nicht sinnvoll eingesetzt werden können.

Prinzipiell bieten sich sehr viele Möglichkeiten, Evolutionäre Algorithmen in Computerspielen einzusetzen, zum Beispiel um für die KI Lösungen zu finden oder das Verhalten von Computergegnern zu optimieren. Im Rahmen der Optimierung tendiert der Einsatz momentan eher noch zur Verwendung als Hilfsmittel bei der Entwicklung von Spielen.

Das erste Kapitel dieser Ausarbeitung beschäftigt sich mit dem Vorbild der Evolution in der Natur sowie den Mechanismen der Vererbung in der Genetik und stellt die Strategien der Evolution beim Finden von Lösungen im diskreten Suchraum vor. Damit werden die theoretischen Grundlagen für das nächste Kapitel geschaffen.

Im zweiten Kapitel werden die mathematischen Modelle der Evolution, die Evolutionären Algorithmen, vorgestellt. Der Genetische Algorithmus wird im Detail besprochen und die gravierenden Unterschiede zu den Evolutionsstrategien dargestellt. Im Anschluss wird eine praktische Anwendung gezeigt.

Im dritten Kapitel werden im ersten Abschnitt zwei Spiele vorgestellt, in denen Evolutionäre Algorithmen das Herzstück bilden. Im zweiten Abschnitt werden zwei Anwendungsbeispiele für Evolutionäre Algorithmen als Hilfe bei der Entwicklung gegeben.

Zum Ende werden die gewonnen Erkenntnisse über den Einsatz von Evolution in Spielen zusammengefasst, um anschließend einen Ausblick auf zukünftige Anwendungen zu geben.

# Kapitel 1

## EVOLUTION UND GENETIK

### 1.1 Biologische Evolution

Die zeitliche Entwicklung aller Lebewesen ist einem beständigen Wandel unterworfen. Über einen längeren Zeitraum betrachtet, bewirkt die Evolution eine immer stärkere und bessere Anpassung der Arten an ihre Lebensbedingungen.

Die Lebewesen einer Spezies ähneln sich zwar sehr, sind aber nie wirklich identisch. Es gibt immer feine Unterschiede zwischen den einzelnen Individuen einer Spezies.

Die Variationen der Individuen sind an die Nachkommen vererbbar. Eigenschaften, die sich im Überlebenskampf bewährt haben, finden sich in den Folgegenerationen bevorzugt wieder.

Es können sich kleine Variationen der Individuen einer Art im Laufe der Generationen addieren und zur Vervollkommnung und Optimierung der Eigenschaften der Lebewesen führen.

Populationen einer Spezies wachsen ohne hindernden Einfluss in der Regel in einem geometrischen Verhältnis an, während die Nahrungsmittel meist nur arithmetisch ansteigen, also wesentlich langsamer zunehmen. Durch dieses Missverhältnis entsteht ein *Selektionsdruck*. Der auf jedes Individuum einwirkende Druck setzt sich im Detail aus den Einwirkungen seiner Umgebung, der Konkurrenz zu anderen Individuen seiner Spezies und der Rivalität seiner Spezies mit anderen Spezies zusammen.

Die Folge ist eine Abnahme der Populationsgröße und damit eine Stabilisierung und relative Konvergenz der Populationsgröße, bis die Nahrungsmittel wieder ausreichen. Dies ist eine wichtige Selbstregulation, die überall in der Natur anzutreffen ist – nur der Mensch setzt sich darüber hinweg.

### 1.2 Evolutionsgenetik

Die Evolutionsgenetik beschäftigt sich mit den Mechanismen der Evolution und den Vererbungsvorgängen. Die folgende Darstellung ist stark vereinfacht, um einen kurzen Überblick als Grundlage für das Verständnis der folgenden Kapitel zu bieten.

#### 1.2.1 *Aufbau des Genoms*

Zellen sind die Grundbausteine des Lebens auf der Erde. Alle Lebewesen sind aus ihnen zusammengesetzt (bis auf Viren, diese sind deshalb definitionsgemäß keine Lebewesen).

In den Zellkernen ist in der DNS (Desoxyribonukleinsäure) die Information kodiert, wie die Zellen eines Lebewesens zu einem funktionierenden Organismus zusammengesetzt werden.

Die DNS Moleküle sind auf Protein basierten Strukturen, den Chromosomen, gespeichert. Menschliche Zellen besitzen 46 Chromosomen, andere Lebewesen weniger oder auch mehr.

Bestimmte DNS Sequenzen enthalten die „Baupläne“ für die Erzeugung von unterschiedlichen Proteinen als Baustoff für die Zellbildung. Jede dieser Sequenzen bildet damit eine Informationseinheit, das *Gen*. Zusammengesetzt sind die unterschiedlich langen Sequenzen, die jeweils ein Gen codieren, aus den Nukleotidbasen Adenin, Cytosin, Guanin und Thymin, die jeweils paarweise im DNS Strang angeordnet sind (Abbildung 1.2.1).

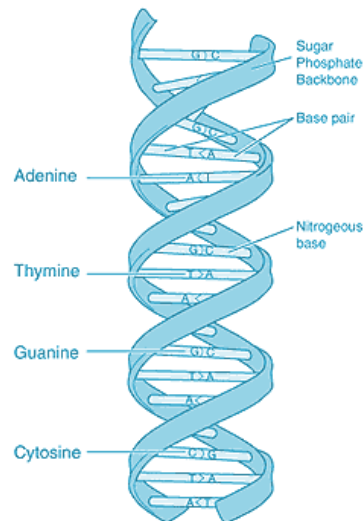


Abbildung 1.2.1: Im DNS Strang werden die Gene gebildet durch Sequenzen von Nukleotidbasenpaaren, bestehend aus Adenin, Cytosin, Guanin und Thymin.

Eine Base auf einem DNS Strang hat einen Informationsgehalt von 2 Bit, da sie  $2^2 = 4$  Zustände (A / T / G / C) annehmen kann. Ausgehend von  $3 \cdot 10^9$  Basenpaaren hat das Genom des Menschen einen Informationsgehalt von etwa 750 Megabyte.

Die individuelle Kodierung der Gene eines Individuums bezeichnet man als den Genotyp. Die physisch ausgeprägten Merkmale des Individuums werden als Phänotyp bezeichnet. Jeder Phänotyp wird aus dem dekodierten Genotyp gebildet.

## 1.2.2 Mechanismen der Vererbung

Bei der zweigeschlechtlichen Fortpflanzung von Lebewesen verschmelzen die Genmaterialien der Eltern zum Genmaterial, aus dem der Nachkomme gebildet wird. Zwei grundsätzliche Mechanismen sorgen dafür, dass im Nachkommen eine genetische Variation mit vermischten Eigenschaften des Eltern-Erbgutes entsteht.

### 1.2.2.1 Rekombination

Während der Fortpflanzung findet eine *Rekombination* der Eltern-Chromosomen statt: die Kind-Chromosomen werden aus sich abwechselnden Gensequenzen der Eltern-Chromosomen gebildet. Dabei berücksichtigt die Natur bestimmte Sequenzen, die bei der Rekombination erhalten bleiben und nicht getrennt werden.

### 1.2.2.2 *Mutation*

In vergleichsweise seltenen Fällen wird das Genmaterial bei der Vererbung zusätzlich durch *Mutation* verändert. So bezeichnet man die zufällige strukturelle Änderung einer DNS Sequenz, die die Ausbildung eines völlig neuen Gens zur Folge haben kann. In den meisten Fällen reduziert die Mutation die Überlebenschancen des betroffenen Nachkommens, sie ist jedoch sehr wichtig, um in bestimmten Fällen eine Anpassung an die Umgebung durch völlig neue Eigenschaften zu ermöglichen oder die Entwicklung einer Spezies in eine andere Richtung zu lenken.

### 1.2.2.3 *Survival of the fittest*

Individuen mit günstigen Mutations- und Rekombinationsergebnissen in ihrem Erbgut tendieren dazu, mehr Nachkommen zu zeugen als die weniger erfolgreichen Mitstreiter. Auf diese Weise verbreiten sich die Gene der in Bezug auf ihre Anpassung erfolgreichen Individuen weiter innerhalb der Population – zu Lasten der schwächeren Kontrahenten. Mutations- und Rekombinationsresultate, die die Chance eines Individuums, in seiner Umgebung zu überleben, reduzieren, verschwinden schnell. Dieser Prozess wird *survival of the fittest*, das „Überleben des Tauglichsten“, genannt.

## 1.3 Die Evolution als Optimierungsprozess

Die Evolution schafft es durch Manipulation der Erbinformationen selbst komplex aufgebaute Lebensformen an ihre sich ständig ändernde Umwelt anzupassen. Das Ziel der Evolution ist es jeweils, diejenigen Erbanlagen aus der Menge aller möglichen Erbanlagen zu finden, die eine Lebensform am besten dazu befähigen, sich im Überlebenskampf zu behaupten.

### 1.3.1 *Der Suchraum*

Die Evolution ist eine Art *Suchprozeß* im Raum der genetisch möglichen Erbanlagen. Der nach dem Optimum zu durchsuchende Raum der genetischen Variationen des Erbgutes kann durch einen diskreten Raum beschrieben werden, dessen Gitterpunkte für alle möglichen Kombinationen der vier Nukleotidbasenpaare aus der DNS stehen. Das menschliche Genom hat wie schon erwähnt eine Gesamtlänge von etwa  $3 \cdot 10^9$  Basenpaaren. Die Anzahl der möglichen Alternativen, also die Anzahl der zu durchsuchenden Gitterpunkte liegt demnach bei  $4^{3.000.000.000}$ . Ein zweifelsohne gigantischer Suchraum, in dem sich das Optimierungsverfahren der Evolution offensichtlich zurechtfindet. Wir Menschen stellen schließlich eine (wenn auch vielleicht nur lokal) optimale Lösung dieses Suchraumes dar.

### 1.3.2 *Suchstrategien*

Das Erstaunliche der Evolution als Optimierungsprozess ist die Einfachheit ihrer Vorgehensweise und das Zusammenwirken der verschiedenen Steuerungsmechanismen Mutation des Erbgutes, Rekombination der Erbinformation und Selektion auf Basis der Fitness.

Analysiert man die drei Prinzipien, so zeigt sich, dass diese eine geschickte Kombination von ungerichteten und gerichteten Suchstrategien sowie seriellen und parallelen Suchprozessen darstellen.

### 1.3.2.1 *Gerichtete und ungerichtete Suchstrategien*

Die Mutation ist ein ungerichteter Prozess und dient einzig dazu, durch Erzeugung von Varianten und Alternativen lokale Optima zu verlassen und damit ein „Einpendeln“ der Evolution bei suboptimalen Lösungen durch zufällige Veränderungen des Erbgutes zu verhindern. Die Mutationswahrscheinlichkeit darf allerdings nur sehr gering sein (ca.  $5 \cdot 10^{-5}$  bis  $5 \cdot 10^{-7}$ ), um ein Einpendeln bei optimalen Lösungen zu ermöglichen.

Die Rekombination hat einen teils gerichteten und teils ungerichteten Suchcharakter. Sie erfolgt an zufälligen Stellen zwischen homologen Chromosomen. Jedoch werden nah beieinander liegende und funktional verbundene Gene seltener getrennt als weiter auseinander liegende Gen-Gruppen. Die Rekombination mischt dadurch zwar das Erbgut zufällig durch, folgt aber gleichzeitig bis zu einem bestimmten Punkt gewissen statistischen Gesetzmäßigkeiten.

Die Selektion ist für die eigentliche Steuerung der Evolution verantwortlich. Sie bestimmt die Richtung, in die sich die Gene verändern, indem sie entscheidet, welche Phänotypen sich stärker vermehren und welche weniger stark. Sie lenkt die Suche immer weiter Richtung Optimum.

### 1.3.2.2 *Kombinierte Suchstrategie: serielle und parallele Suche*

Bei der Anpassung des Genpools einer Spezies an veränderte Lebensbedingungen ist die Zeit der kritische Faktor. Die evolutionäre Suchstrategie ist ein zeiteffizienter Optimierungsprozess. Es existieren zwei Strategien: eine kurze Generationenfolge, um in kurzer Zeit möglichst oft neu zu mutieren, rekombinieren und selektieren oder die Erzeugung möglichst vieler Individuen zur gleichen Zeit, um auf diese Weise die Evolutionsdauer zu minimieren. Die Evolution ist eine vermutlich annähernd optimale Kombination beider Strategien. Sie koppelt die Tiefensuche (geringe Reproduktionszeiten) mit der Breitensuche (hohe Reproduktionsquote).

Die Parallelisierbarkeit des Suchprozesses der Evolution, also die Möglichkeit, mehrere Individuen, die zur gleichen Zeit leben, simultan auf ihre Fitness zu überprüfen, ist besonders wichtig, um in großen, hochdimensionalen Suchräumen die genetischen Kombinationsmöglichkeiten zeitgleich von mehreren Punkten (Genotypen) aus zu durchsuchen. Dies erhöht die Wahrscheinlichkeit, optimale Punkte innerhalb des Suchraums zu erreichen und reduziert gleichzeitig die Wahrscheinlichkeit, suboptimale Pfade zu verfolgen und fehlgeleitet zu werden.

## EVOLUTIONÄRE ALGORITHMEN

### 2.1 Einleitung

Evolutionäre Algorithmen nutzen die Basistechniken der natürlichen Evolution, indem in einer „Lösungs-Population“ die besseren Lösungen eines Problems zu „Eltern“ werden und ihre Eigenschaften an „Kind-Lösungen“ vererben.

Bislang existieren im Wesentlichen zwei algorithmische Modelle der Evolution, die sich für Computersimulationen und Anwendungen in der Informatik eignen:

- Genetische Algorithmen (GA)
- Evolutionsstrategien (ES)

Beide Ansätze wurden etwa zeitgleich in den 60er Jahren weitgehend unabhängig voneinander entwickelt. Die Theorie der Genetischen Algorithmen wurde von John Holland erstmals in seiner Promotionsarbeit an der University of Michigan festgehalten. Das Gebiet der Evolutionsstrategien wurde von Ingo Rechenberg an der TU Berlin begründet. Die Bestreiter der jeweiligen Ansätze haben sich lange Zeit gegenseitig über den anderen lustig gemacht und die eigene Theorie als die einzig richtige beansprucht.

Genetische Algorithmen behandeln allgemein die Fragestellung, wie die Evolution Informationen codiert und verarbeitet und über Generationen hinweg weitergibt. Evolutionsstrategien beschäftigen sich hingegen eher mit der biologischen Evolution als Wegweiser für die Entwicklung eines leistungsstarken Such- und Optimierungsverfahrens. Zusammengefasst werden beide Theorien unter der Bezeichnung Evolutionäre Algorithmen (EA). Zu erwähnen ist noch die Genetische- und Evolutionäre Programmierung, die die Evolution von Programmcode zur Lösungsfindung nutzt.

Für beide Evolutionären Algorithmen gilt: eine erfolgreiche und effektive Anwendung hängt sehr stark von den verwendeten Verfahren (Codierung, Selektions-Schema, etc.) und Parametereinstellungen (Populationsgröße, Mutationsraten, etc.) ab. Werden Verfahren oder Parameter schlecht gewählt, so kann dies dazu führen, dass das Auffinden einer Lösung für das jeweilige Problem erschwert oder gar unmöglich gemacht wird.

Der nächste Abschnitt zeigt detailliert die Codierung und Arbeitsweise der in der Literatur am häufigsten anzutreffenden Genetischen Algorithmen. Im Anschluss werden die Unterschiede zu den Evolutionsstrategien erläutert. Zum Abschluss folgt noch ein einfaches Anwendungsbeispiel.

## 2.2 Genetische Algorithmen

### 2.2.1 Codierung

Die Erbinformation wird in Genetischen Algorithmen (GA) binär codiert unter Verwendung der Grundmenge  $M := \{0,1\}$ . Ein Individuum (oder auch *Chromosom* oder *Genotyp*) wird durch einen binären Vektor  $X$  der Länge  $n$  dargestellt:

$$X = (x_1, x_2, \dots, x_n), \quad x_i \in M.$$

Eine Population von Chromosomen ist immer eine Teilmenge von  $M^n = \{0,1\}^n$ , d.h. eine Teilmenge aller möglichen Tupel aus  $M^n$ .

Die Gene eines Chromosoms können sowohl als einzelne Positionen als auch als zusammenhängende Binärsequenzen innerhalb des Chromosoms codiert werden. Als Beispiel steht an der Position  $i$  eines Chromosoms  $X = (\dots, x_i, \dots) \in M^n$  das  $i$ -te Gen und das Chromosom  $Y = ((1,1,0),1)$  setzt sich aus zwei Genen zusammen: einer Binärsequenz  $(1,1,0)$  und einer einzelnen Position 1.

Die Ausprägung, also der jeweilige Wert eines Gens wird wie in der Biologie Allel genannt und ist nichts anderes als die Belegung der Gen-Variablen mit einem Wert. Zur Veranschaulichung:  $X = (1,0,1,1,1,0)$  ist ein Chromosom aus einer Population von  $M^6$ . Das zweite und sechste Gen haben das Allel 0, alle anderen Gene das Allel 1.

### 2.2.2 Startpopulation

Der erste Schritt bei der Implementierung eines Genetischen Algorithmus ist die Generierung einer Startpopulation. Jedes Individuum dieser Population ist, wie bei der Codierung schon diskutiert wurde, ein Binärvektor der Länge  $n$ . In den meisten Fällen wird die Startgeneration zufällig erzeugt.

### 2.2.3 Bewertung und Fitness der Individuen

Nach Erzeugung der Startpopulation wird jedes Individuum bewertet und bekommt einen Fitnesswert zugewiesen. Die Bewertung misst in diesem Zusammenhang die Performance des einzelnen Individuums unabhängig von dem Rest der Population, während der Fitnesswert den Rang des einzelnen Individuums auf Basis der ermittelten Performance-Bewertung im Vergleich zum Rest der Population angibt.

Die Fitness wird wie folgt definiert:

$$f_i = \frac{p_i}{\bar{p}},$$

wobei  $p_i$  die Performance des Individuums  $i$  ist und  $\bar{p}$  die mittlere Performance aller Individuen der Population. Die Performance eines Individuums ergibt sich aus der Performance seines Phänotyps und die Funktion zur Berechnung muss abhängig von dem Problem, auf das der Genetische Algorithmus angewendet werden soll, spezifisch ermittelt werden.

Die Fitness-Funktion sorgt dafür, dass ein Individuum mit einer durchschnittlichen Performance einen Fitnesswert von  $f_i \approx 1$  bekommt, weniger taugliche Individuen liegen mit ihrer Fitness bei  $f_i < 1$  und die performantesten Individuen haben Werte von  $f_i > 1$ .

#### 2.2.4 Ausführung des Algorithmus

Die Ausführung eines Genetischen Algorithmus kann als zweistufiger Prozess identifiziert werden. Es wird mit der aktuellen Population begonnen. Per Selektion wird eine Zwischenpopulation erzeugt. Durch Anwendung von Rekombination und Mutation auf die Zwischenpopulation wird die nächste Population erzeugt.

Der Prozessablauf von der aktuellen zur nächsten Population beschreibt eine *Generation* in der Ausführung eines Genetischen Algorithmus. Diese hier verwendete Darstellung entspricht der Basis-Implementierung nach Goldberg [Whit] und wird *Simple Genetic Algorithm* (SGA) genannt.

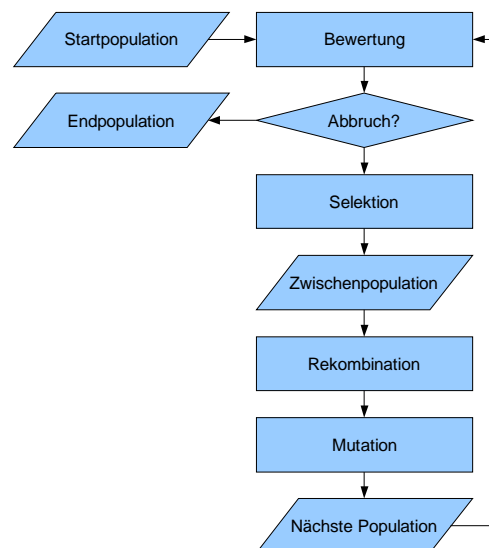


Abbildung 2.2.4: Flussdiagramm des *Simple Genetic Algorithm*.

#### 2.2.5 Konstruktion der Zwischenpopulation: Selektion

Als erstes betrachten wir die Konstruktion der Zwischenpopulation aus der aktuellen Population, welche in der ersten Generation gleichzeitig die Anfangspopulation ist. Nachdem  $f_i = p_i / \bar{p}$  für alle Individuen der aktuellen Population berechnet wurde, wird Selektion angewendet. In den meisten Genetischen Algorithmen ist die Wahrscheinlichkeit für jedes Individuum, als Kopie in die Zwischenpopulation aufgenommen zu werden, proportional zu seinem Fitnesswert.

Es gibt viele Möglichkeiten, Selektion durchzuführen. Ein häufig angewendetes Verfahren ist das *Rouletteverfahren*, bei dem jedes Individuum als Fläche auf einem Rouletterad repräsentiert wird. Die Fläche ist dabei proportional zur jeweiligen Fitness. Durch wiederholtes Drehen des Rouletterades werden die Individuen für die Zwischenpopulation ausgewählt.

Ein Selektionsprozess, der die Fitnesswerte besser berücksichtigt, ist das *Remainder Stochastic Sampling*. Für jedes Individuum  $i$ , bei dem  $f_i = p_i / \bar{p}$  größer ist als 1,0, gibt die Vorkommastelle an, wie viele Kopien des Binärvektors direkt in die Zwischenpopulation übernommen werden. Alle Individuen, auch diejenigen mit  $f_i < 1.0$ , platzieren dann mit einer Wahrscheinlichkeit, die sich aus ihrer Nachkommastelle von  $f_i$  ergibt, weitere Kopien in der Zwischenpopulation.

Als Beispiel kommen von einem Individuum mit  $f_i = 2,05$  zwei Kopien direkt in die Zwischenpopulation und mit einer Chance von 0,05 noch eine weitere Kopie. Ein Individuum mit  $f_i = 0,81$  bringt mit einer Wahrscheinlichkeit von 0,81 eine Kopie in die Zwischenpopulation ein.

Das *Remainder Stochastic Sampling* wird am effizientesten von der *Stochastic Universal Sampling* Methode implementiert. Die Population wird hier in zufälliger Reihenfolge wie in einem Tortendiagramm angeordnet, wobei jedes Individuum eine Fläche auf dem Tortendiagramm proportional zu seiner Fitness belegt. Als nächstes wird ein äußerer Ring mit  $N$  Zeigern gleichen Abstands um das Tortendiagramm gelegt. Eine einfache Drehung des Ringes bestimmt simultan alle  $N$  Mitglieder der Zwischenpopulation. Die so zustande kommende Selektion ist erwartungstreu [Whit].

Eine in vielen Fällen zu empfehlende Technik ist die Anwendung von *Elitismus*. Sie verbessert fast immer die Performance des Genetischen Algorithmus, indem sie hilft, sehr viel schneller gegen eine Lösung zu konvergieren.

Elitismus ist das simple Vorgehen, die  $n$  besten Individuen aus der aktuellen Population unverändert in die nächste Population zu übernehmen. Diese Technik garantiert, dass die bislang performantesten vom Algorithmus gefundenen Individuen in der Population bleiben. Ein guter Wert für  $n$  liegt zwischen 1 und 10 Prozent der Populationsgröße. Bei der Anwendung ist jedoch auch Vorsicht geboten: zu starker Elitismus führt zu einer zu schnellen Konvergenz – und kann damit leicht zum Einpendeln auf einem nur lokalen Maximum führen.

### 2.2.6 Konstruktion der nächsten Generation: Rekombination

Nach der Selektion ist die Konstruktion der Zwischenpopulation komplett und die Rekombination findet statt. Dieses Verfahren erzeugt aus der Zwischenpopulation die nächste Population. Jeweils zwei zufällig aus der Zwischenpopulation ausgewählte Individuen werden mit der Wahrscheinlichkeit  $P_C$  zu zwei neuen Individuen rekombiniert und in die nächste Population eingefügt. Die Wahrscheinlichkeit  $P_C$  ist der Problemstellung entsprechend festzulegen und durch Testläufe anzupassen.

Nehmen wir zwei Binärvektoren

$$X_E = (x_1, x_2, \dots, x_n) \text{ und } Y_E = (y_1, y_2, \dots, y_n) \text{ mit } X_E, Y_E \in M^n.$$

Diese Eltern-Binärvektoren repräsentieren beide eine mögliche Lösung zu einem Parameter-Optimierungsproblem. Durch Rekombination der beiden Eltern-Binärvektoren werden zwei neue Testpunkte im Suchraum erzeugt. Es existieren viele Möglichkeiten, wie die zwei neuen Vektoren durch Kombination der zwei alten entstehen können. Im Folgenden wird die Methode des *Single Point Crossover* beschrieben.

Unter Verwendung einer zufällig bestimmten Position  $i$ , an der das Crossover, also die Vertauschung der Gene innerhalb der Binärvektoren, stattfindet, wird die Rekombination durchgeführt und man erhält die zwei Kind-Binärvektoren

$$X_K = (x_1, x_2, \dots, x_{i-1}, y_i, \dots, y_n) \text{ und } Y_K = (y_1, y_2, \dots, y_{i-1}, x_i, \dots, x_n) \text{ mit } X_K, Y_K \in M^n.$$

Die Methode des *Double Point Crossover* verwendet zwei Stellen auf den Binärvektoren, an denen vertauscht wird und hat sich als die in den meisten Fällen am besten geeignet herausgestellt. Vertauschungen an mehr als zwei Stellen gehören zu den *Multi Point Crossover* Verfahren. In Abbildung 2.2.6 ist eine Darstellung des Single Point Crossover zu sehen.

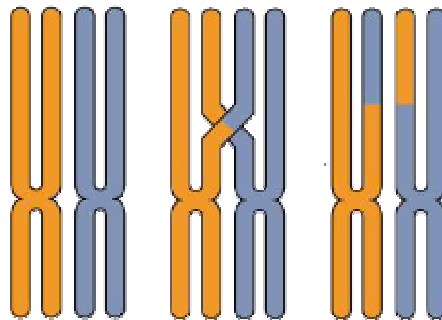


Abbildung 2.2.6: Darstellung des Single Point Crossover.

### 2.2.7 Mutation

Nach der Rekombination wird die Mutation auf die Individuen angewendet: jedes Bit aus der Population mutiert mit einer Wahrscheinlichkeit  $P_M$ . Typischerweise wird Mutation mit einer Wahrscheinlichkeit von  $P_M < 1\%$  angewendet. Die einfachste Art der Mutation ist die Invertierung des betreffenden Bits.

Nachdem die Prozesse der Selektion, Rekombination und Mutation vollständig durchgeführt wurden, wird die gerade erzeugte nächste Population zur aktuellen Population und kann erneut ausgewertet werden. Der Prozess der Bewertung, Selektion, Rekombination und Mutation bildet eine Generation in der Ausführung eines Genetischen Algorithmus.

### 2.2.8 Abbruchbedingung

Genetische Algorithmen werden in der Regel über sehr viele Generationen ausgeführt. Es ist möglich (und erwünscht!), dass innerhalb dieser Zeit das gesuchte Optimum oder eine als hinreichend optimal befundene Lösung unter den Individuen der Population auftaucht. Ab einer bestimmten Generationszahl muss nach jedem Durchlaufen einer Generation das Individuum mit der höchsten Fitness auf Erreichen des Suchziels getestet werden, ein positiver Test führt zur Terminierung des Algorithmus.

In einigen Fällen kann es möglich sein, dass auch nach beliebig vielen Generationen kein Optimum gefunden wird, und für diesen Fall sollte eine Begrenzung der Anzahl der zu durchlaufenden Generationen implementiert werden. Eine zeitliche Beschränkung für die Ausführung des Algorithmus ist ebenfalls anwendbar.

## 2.3 Evolutionsstrategien

Im Folgenden werden die gravierenden Unterschiede der Evolutionsstrategien zu den Genetischen Algorithmen aufgezeigt.

### 2.3.1 Codierung

In der Evolutionsstrategie wird jedes Gen als eine reelle Zahl und nicht als Binärwert oder Binärsequenz dargestellt. Die gesamte relevante Erbinformation eines Individuums kann so einfach durch einen Vektor reeller Zahlen codiert werden. Eine Population von Individuen ergibt dann eine Menge von Vektoren.

Die reelle Codierung stellt die kompakteste Form der Codierung dar. Ein offensichtlicher Vorteil der binären Codierung ist ihre programm-technische Implementierung auf Digitalrechnern, auf denen binäre codierte Individuen kompakt im Speicher untergebracht und verarbeitet werden können. Die letztendliche Entscheidung über die Codierungsform ist jedoch zunächst immer von dem jeweils zu lösenden Problem abhängig.

### 2.3.2 Selektion

Bei Evolutionsstrategien werden Elter-Individuen unabhängig von ihrer Fitness oder Bewertung rein zufällig ausgewählt. Aus den erzeugten Nachkommen werden dann jeweils die Besten ausgewählt. Hier handelt es sich also um ein *survival of the fittest* bei den Nachkommen. Dagegen wird ein *survival of the fittest* bei den Genetischen Algorithmen schon bei der Auswahl der Eltern durchgeführt, wenn die Eltern mit einer zu ihrer Bewertung proportionalen Wahrscheinlichkeit selektiert werden.

### 2.3.3 Mutation

Die Mutation kann bei den Evolutionsstrategien wegen der reellen Codierung nicht einfach als „Bit-Flip“ ausgeführt werden. Eine Möglichkeit ist hier die Addition einer Zufälligen reellen Zahl oder die Multiplikation mit einem Faktor. Auch das Setzen des betroffenen Gens auf einen Minimal oder Maximalwert kann sinnvoll sein.

### 2.3.4 Selbstadaption

Ein weiterer Unterschied liegt in der bei den Evolutionsstrategien integrierten Selbstadaption, durch die Steuerungsparameter, wie die Mutationsschrittweite, selbstständig und automatisch angepasst werden können. Dies führt zu einer gewissen Reaktionsfähigkeit auf verschiedene Faktoren. Eine solche Selbstadaption wäre bei Genetischen Algorithmen nur durch eine recht umständliche Codierung dieser zusätzlichen Parameter möglich und aufgrund des Codierungsproblems nicht problemunabhängig realisierbar.

## 2.4 Ein Anwendungsbeispiel: GA Racer

Dieses Beispielprogramm stammt von dem Beitrag „Building Better Genetic Algorithms“ von Mat Buckland [Rab2] und ist auf der dem Buch beiliegenden CD zu finden. Es zeigt, wie ein Genetischer Algorithmus benutzt wird, um eine Steuersequenz zu entwickeln, mit der ein Fahrzeug über eine einfache Strecke geführt werden soll (Abbildung 2.4).

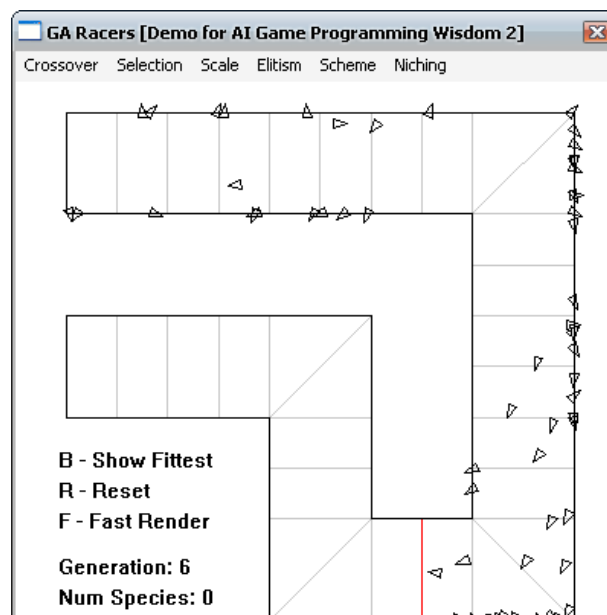


Abbildung 2.4: Screenshot des Beispielprogramms GA Racer. Die Dreiecke geben die Richtung der Fahrzeuge an, die rote Linie markiert die bisher durch das beste Fahrzeug zurückgelegte Strecke.

Die als Dreiecke dargestellten Fahrzeuge versuchen die Strecke so weit wie möglich zu überwinden, indem sie eine Sequenz von Lenk- und Beschleunigungskommandos entwickeln. Sie versuchen zusätzlich, ihre Durchschnittsgeschwindigkeit zu maximieren. Jedes Genom ist in dem folgenden Format codiert:

Richtung | Beschleunigung | Richtung | Beschleunigung | Richtung | Beschleunigung . . . usw.

Jedes Gen wird durch eine reelle Zahl zwischen 0 und 1,0 repräsentiert. Ein Fahrzeug durchläuft sein Genom in Schritten von zwei Genen pro Aktualisierungsdurchgang. Das jeweils erste Gen gibt die Steuerrichtung an, das zweite den Betrag der Beschleunigung. Die Fahrzeuge nutzen diese Werte, um ihre Position zu aktualisieren.

Es gibt mehrere Einstellungsmöglichkeiten zur Verwendung unterschiedlicher Techniken bei der Durchführung des Algorithmus um diese in unterschiedlichen Kombinationen auszuprobieren und sich die Auswirkungen unmittelbar anzuschauen.

## EVOLUTION IN SPIELEN

Der Einsatz von Evolutionären Algorithmen in Computerspielen hat in den letzten Jahren zwar zugenommen, ist aber nach wie vor eher verhalten. Ihre Anwendung findet man meist eher bei der Entwicklung von Spielen, wo sie als Werkzeug eingesetzt werden, um Optimierungsaufgaben zu lösen wie zum Beispiel als Hilfsmittel zur Erstellung eines intelligenten Computergegners, d.h. zum Auffinden der besten Taktik. Die nächsten Abschnitte zeigen anhand von Beispielen, wie Evolutionäre Algorithmen in der Entwicklung und zur Laufzeit von Spielen benutzt werden.

### 3.1 Evolutionäre Algorithmen als Werkzeug in der Spielentwicklung

#### 3.1.1 Platzoptimierung bei der Texturspeicherung

Moderne grafisch orientierte Computerspiele benötigen Texturen für die Darstellung von 3D Objekten. Diese werden im Grafikspeicher abgelegt. Da die Texturen in unterschiedlichen Größen und meist in großer Zahl vorliegen, ist es sinnvoll, mehrere Texturen in einer großen Textur zu speichern – diese sollte allerdings so klein wie möglich sein, um Speicherplatz zu sparen. Das Problem liegt nun darin, die Texturen so anzuordnen, dass sie eine minimale quadratische Fläche mit einer Potenz von 2 als Seitenlänge (diese Restriktionen sind für die Erzielung der besten Grafik-Performance notwendig) ausfüllen.

In seiner Diplomarbeit zeigt Karsten Kaul [Kaul], dass dieses Optimierungsproblem mit einem Genetischen Algorithmus zu lösen ist. Als Besonderheit zeigt Kaul zudem die Implementierung des Algorithmus auf der GPU der Grafikkarte unter Ausnutzung der dort vorhandenen Möglichkeiten zur parallelen Verarbeitung von Teilen des Algorithmus.

Der verwendete Ansatz sieht eine Anordnung der kleinen Texturen in mehreren Spalten vor, in die die große Textur eingeteilt wird. Durch Verschieben und 90°-Drehen der einzelnen Texturen kann die von ihnen belegte Fläche minimiert werden. Die Codierung speichert für jede Textur ein Gen mit den Eigenschaften Breite, Höhe, Spalte und Drehung.

Die Berechnungsdauer für 100 Generationen lag bei den Testfällen mit bis zu 50 Texturen und Individuen im Mittel im 100ms Bereich und führte im Mittel zu sehr guten Ergebnissen bei der platzsparenden Texturanordnung. Die bei sehr großen Mengen an Texturen (>150) exponentiell ansteigende Größe des Suchraumes führte schnell zu sehr viel längeren Laufzeiten.

#### 3.1.2 Parameteroptimierung bei Rennspielen

In seinem Seminarvortrag „KI in Rennspielen“ [Bloc] behandelt Yannick Block in Abschnitt 3.2 das Trainieren der KI für computergesteuerte Fahrzeuge. Es sind an jedem KI-Fahrzeug Feineinstellungen für die Balance, den Bremsweg etc. vorzunehmen. Um die jeweils optimale Parameter-Konfiguration herauszufinden, lässt man die KI einige Runden mit einer Einstellung fahren, bestimmt anhand der Rundenzeit die Güte der

aktuellen Parameter Einstellung, nimmt per Hand Anpassungen an dieser vor und lässt die KI erneut fahren. Anhand der neuen Rundenergebnisse werden die Parameter weiter angeglichen. Dieser Ablauf wird solange wiederholt, bis eine zufrieden stellende Konfiguration erreicht wird.

Die vorliegende Optimierungsaufgabe ist ideal für die Anwendung eines Evolutionären Algorithmus. Jeder Parameter wird auf ein Gen abgebildet und reell codiert. Jedes Gen erhält Beschränkungen hinsichtlich seines Minimal- und Maximalwertes. Dann wird eine Startpopulation von etwa 50 bis 100 Parameterkonfigurationen entweder zufällig erzeugt oder mit dem Vorwissen des Entwicklers angelegt. Nach der Wahl der passenden Mechanismen für die Vererbung wie z.B. Elitismus, Double Point Crossover etc., der Festlegung der Bewertungsfunktion (Performance nach Rundenzeit) und einem Abbruchkriterium (Rundenzeit kleiner vorgegebene Zeit) ist die Arbeit getan und der Algorithmus kann gestartet werden. Dann heißt es nur noch abwarten. Es ist mit einer sehr langen Laufzeit von mehreren Tagen zu rechnen, bis nach mehreren hundert Generationen (und damit zehntausenden von Runden) ein Optimum gefunden wird, jedoch geschieht dies vollständig automatisiert und ohne „Handarbeit“ des Entwicklers.

### 3.2 Verwendung von Evolutionären Algorithmen in Spielen zur Laufzeit

Es gibt bis heute nur eine Handvoll Spiele mit einer Implementierung Evolutionärer Algorithmen. Ein dem allgemeinen Anstieg der Anzahl von Spielen, die Wert auf eine hoch entwickelte Spiele-KI legen, folgender Trend ist jedoch erkennbar.

Die im Folgenden vorgestellten Spiele *Creatures* und *Cloak, Dagger and DNA* sind zwei der wenigen Spiele, in denen Evolutionäre Algorithmen das Herzstück bilden.

Weitere Spiele wie *Die Sims* (Teile 1 bis 3 und zahlreiche Add-Ons), das bislang erfolgreichste Computerspiel was Verkaufszahlen angeht, *Nooks and Crannies* (Entwicklung wurde wahrscheinlich eingestellt), *Return Fire II*, *Sigma*, *Serenity* verwenden ebenfalls mehr oder weniger genetische Mechanismen. Spiele wie *Half-Life* und *Black & White* sollen ebenfalls Genetische Algorithmen einsetzen [Bech], dem Autor war es jedoch nicht möglich, an Informationen zu gelangen, die dies bestätigen.

#### 3.2.1 *Creatures*

##### 3.2.1.1 *Spielbeschreibung*

*Creatures* (Teile 1 bis 3 und Online-Version) ist ein unbefristetes Spiel ohne festgelegtes Ende, es kann so lange gespielt werden, wie gewünscht. Ein Ziel des Spielers kann es sein, eine dauerhafte, gesunde Norm-Population zu erhalten, ein anderes wäre das Züchten einer einzelnen Kreatur.

Die Norms im Spiel sollen lebende Kreaturen darstellen, sie haben eine komplex nachgebildete Biochemie, ein Organsystem, ein lernfähiges Gehirn und ein dies alles beschreibendes Genom.

Norms besitzen zudem einen genetischen Code, vom Hersteller Digital DNA genannt. Das äußerliche Erscheinungsbild, die Fähigkeit Futter zu Verdauen, die Intelligenz und die Gesundheit – alles wird von der DNA festgelegt. Norms können sich fortpflanzen und ihre einmalige DNA an ihre Nachkommen weitergeben. Norms gesund zu halten ist nicht immer leicht, es ist die Aufgabe des Spielers sie durchs Leben zu führen, ihnen richtig und falsch beizubringen und sie wieder aufzupäppeln, wenn alles schief läuft.

### 3.2.1.2 *Evolutionäre Elemente*

Das Spiel *Creatures* ist eine Simulation des Lebens. Die verwendeten Kreaturen besitzen viele Verhaltensmuster von echten Tieren, ohne dass diese gezielt programmiert werden mussten. Die Norns machen nicht immer das, was ihnen gesagt wird und können gänzlich unerwartete Dinge tun. Die Gene der Norns ändern sich von Generation zu Generation, was zum Beispiel an der Änderung ihres Aussehens, ihrer Einstellung (freundlich, böse) oder ihrer Intelligenz zu beobachten ist. Das genetische Material eines Norns enthält Informationen über Aussehen, Biochemie, Gangart, Instinkte und Gehirnstruktur.

In *Creatures 2* besitzen die Norns 771 Gene. Die Hälfte davon codiert das Aussehen und die Haltung, der Rest die Biochemie, Organe und Gehirnstruktur. Gespeichert wird dabei für einige Gene eine Menge von Instruktionen, die Strukturen beschreiben, welche ein System aufbauen, aus dem sich Verhalten herausbilden kann. Jedes Gen besitzt zusätzlich zur codierten Information einen Header, der festlegt, welche Mechanismen beim Erzeugen des Erbgutes für den Nachkommen aus den Eltern Genomen erlaubt sind:

- Duplikation – erlaubt die zufällige Duplizierung des Gens
- Mutation – erlaubt die Mutation von innerhalb des Gens
- Löschen – erlaubt das vollständige Herauslösen eines Gens aus dem Genom

Auf diese Weise wird erreicht, dass lebenskritische Gene nur in zulässigem Maße verändert werden können. Bei der Replikation wird neben dem „normalen“ Single bis Multi Point Crossover und der Mutation auch das Löschen und Verschieben von Genen angewendet. Dadurch sind auch ungewöhnliche Variationen des Erbgutes möglich, zum Beispiel dass ein Norn acht Lungen oder gar keine Lunge besitzt.

### 3.2.1.3 *Fazit*

Evolutionäre Mechanismen sind vorhanden und werden sehr umfangreich umgesetzt, dienen jedoch nicht einem höheren Ziel wie einer Optimierungsstrategie sondern laufen eher auf einer Art Züchtungs-Ebene ab. Selektion wird nicht wirklich auf Basis des Genotyps vollzogen, das Überleben der Norns hängt meist vom Geschick des sie trainierenden Spielers ab. Trotzdem ist diese Implementierung sehr interessant und stellt eine Sonderform bezüglich des Spielablaufs und der verwendeten Technik unter den Computerspielen dar.

## 3.2.2 *Cloak, Dagger and DNA*

### 3.2.2.1 *Spielbeschreibung*

*Cloak, Dagger and DNA* ist ein taktisches Echtzeit-Eroberungsspiel. Die Spielkarte ist in Regionen unterteilt, von denen einige Fabriken enthalten. Bringt ein Spieler eine Fabrik in seinen Besitz, erhöht dies das Einkommen und bildet die Grundlage für die Produktion von mehr Bodeneinheiten. Es gibt zwei Arten von Bodeneinheiten: Armeen für den Angriff und die Verteidigung von Regionen und Spione, die nicht kämpfen oder getötet werden aber das gegnerische Gebiet erforschen können. Der Kampf wird abhängig von der Anzahl an Einheiten in einem Gebiet berechnet, mit einem Bonus für den Verteidiger. Es können bis zu vier Spieler in jeder Mensch / Computer Kombination spielen.

### 3.2.2.2 *Evolutionäre Elemente*

Das Spiel benutzt Genetische Algorithmen, um das Spielverhalten des Computergegners zu lenken. Das Genom eines Gegners besteht aus einem Satz von Regeln, die das Spielverhalten steuern. Nach jedem Kampf wird der jeweilige Regelsatz auf seinen Erfolg geprüft und in einem Genpool abgelegt. Für den nächsten Kampf wird ein anderer Regelsatz aus dem Genpool geholt. Es besteht zusätzlich die Möglichkeit

zur Durchführung von Wettkampfsereien unter den teilnehmenden Spielern zwecks evolutionärer Entwicklung *außerhalb* des Spiels. Es gibt eine Menge von genetischen Mechanismen wie Rekombination, Mutation und Selektion, außerdem können die Regelsätze der Genotypen editiert und in einer Genom-Bibliothek gespeichert werden. Mit der Zeit (>1000 Kämpfe) werden die Computergegner merklich besser.

### 3.2.2.3 *Fazit*

Der im Spiel Cloak, Dagger and DNA eingesetzte Genetische Algorithmus optimiert das Spielverhalten der Computergegner und ist ein Musterbeispiel für den erfolgreichen Einsatz von Evolution in Spielen. Die Möglichkeit zur Verwaltung der Genotyp Regelsätze ist sehr interessant (wenn auch etwas rustikal implementiert).

## ZUSAMMENFASSUNG UND AUSBLICK

Evolutionäre Algorithmen stellen eine robuste Suchmethode für große, komplexe oder wenig bekannte Suchräume und nichtlineare Probleme dar, und können dort weiterhelfen, wo gängige Algorithmen versagen. Sie verlagern die Arbeit auf die Seite des Computers, der durch ein gezieltes Suchen selbst gute Lösungen produziert.

Für die Auswahl der geeigneten Werkzeuge unter den vielen möglichen Varianten (Rekombinationsmethoden, Mutationsvarianten, Selektionsschemata etc.) für ein spezifisches Problem ist jedoch einige Erfahrung mit dem Umgang mit Genetischen Algorithmen erforderlich. Auch wenn die Grundstrukturen immer gleich bleiben, unterscheiden sich doch die Details von Problem zu Problem. Werden lange Gensequenzen und Populationen mit einer hohen Individuenzahl verwendet (und das ist meist der Fall), frisst die Ausführung Ressourcen bei der Anwendung (Zeit, Speicher und Rechenleistung).

Aufgrund des hohen Ressourcenverbrauchs zur Laufzeit finden Evolutionäre Algorithmen kaum Anwendung innerhalb von Computerspielen. Sie werden eher als Werkzeug bei der Entwicklung benutzt, um zum Beispiel das Verhalten eines Computergegners zu trainieren.

Durch geschickte Implementierungen, zum Beispiel die Berechnung im Hintergrund in Spielabschnitten, in denen die Prozessorlast gering ist oder die Nutzung der guten Parallelisierbarkeit der Evolutionären Algorithmen, wie bei dem Beispiel der Ausführung auf dem Prozessor der Grafikkarte, wäre es jedoch durchaus denkbar, Evolutionäre Algorithmen mehr in Spielen zu verwenden.

## LITERATURVERZEICHNIS

- [Bech] *Becher, Felix*  
KE und Lernen in Spielen: Genetic Algorithms  
Seminar, TU Darmstadt, 2006
- [Bloc] *Block, Yannick*  
KI in Rennspielen  
Seminar: Spiele-KI, Fachhochschule Wedel, 2007
- [Heis] *Heistermann, Jochen*  
Genetische Algorithmen: Theorie und Praxis evolutionärer Optimierung  
Teubner, 1994
- [Kaul] *Kaul, Karsten*  
Genetische Algorithmen auf Shaderhardware  
Diplomarbeit, Fachhochschule Wedel, 2005
- [Koel] *Koell, Bastian; Schulz, Sebastian*  
Künstliche Intelligenz in Spielen: Genetische Algorithmen  
Seminar, Universität Kassel, 2004
- [Rab1] *Rabin, Steve*  
AI Game Programming Wisdom  
Thomson Learning, 2002
- [Rab2] *Rabin, Steve*  
AI Game Programming Wisdom 2  
B&T, 2004
- [Rab3] *Rabin, Steve*  
AI Game Programming Wisdom 3  
Thomson Learning, 2006
- [Schö] *Schöneburg, Eberhard*  
Genetische Algorithmen und Evolutionsstrategien: Eine Einführung in Theorie und  
Praxis der simulierten Evolution  
Addison-Wesley, 1994
- [Ste] *Steinbach, Jan Oliver*  
Evolution virtueller Lebewesen und Funktionale Programmierung  
Master Thesis, Fachhochschule Wedel, 2006
- [Whit] *Whitley, Darrell*  
A Genetic Algorithm Tutorial  
Colorado State University, 1994