



School of  
Engineering

IAMP Institut für Angewandte  
Mathematik und Physik

## **Bachelor- /Projektarbeit (Informatik)**

### Design von artifiziellen Tieren mit evolutionären Algorithmen

<b>Autoren</b>	Fabian Hediger Florian Tanner
<b>Hauptbetreuung</b>	Prof. Dr. Rudolf Füchslin
<b>Nebenbetreuung</b>	Prof. Dr. Olaf Stern
<b>Datum</b>	09.06.2016

## **Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering**

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

**Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.**

Ort, Datum:

Unterschriften:

.....

.....

.....

.....

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

# Zusammenfassung

Das Optimieren vieler, praktisch relevanter und interessanter Probleme in der Informatik und Robotik sind aufwendige Aufgaben, die viel Rechenleistung benötigen. Wird der Lösungsraum eines Problems zu komplex, stossen formale Lösungsmethoden an ihre Grenzen. Evolutionäre Algorithmen sind der natürlichen Evolution nachempfunden und produzieren durch “trial and error” neue Lösungen. Mit dem Prinzip “survival of the fittest” werden immer bessere Annäherungen produziert.

Diese Arbeit versucht sechsbeinigen künstlichen Tieren mit Hilfe eines evolutionären Algorithmus, das Fortbewegen durch einen Parcours beizubringen. Es wird untersucht, wie die Geometrie und Bewegungsablauf eines Tieres evolviert werden können. Anschliessend wird analysiert, wie der Bewegungsablauf sowie die Form eines evolvierten Tieres aussehen. Dabei werden folgende Forschungsfragen beantwortet:

1. Wie kann eine Steuerung der Bewegung implementiert werden?
2. Wie kann diese Steuerung evolviert werden?
3. Wie kann die Geometrie der Tiere evolviert werden?
4. Wie sehen der Bewegungsablauf und die Geometrie eines evolvierten Tieres aus?
5. Liefert kleine Mutationswahrscheinlichkeiten bessere Fitnesswerte als grosse?

Die Fragestellungen werden mit Ergebnissen verschiedener Simulationen untersucht.

Die Resultate zeigen, dass sich drei Typen von Bewegungsabläufen entwickelt haben. Entstanden sind Hüpf-, Ruder- und Rollbewegungen. Dabei zeigen sich je nach Bewegung verschiedene Tendenzen zu Körperperformen. Rollende Individuen neigen zu kugelförmigen Körpern, Ruderer zu flachen Formen. Hingegen Hüpfer zeigen keinen eindeutigen Trend. Optimierungsbedarf besteht bei der Reaktion der Individuen auf Steigungen oder Gefälle im Parcours.

# Abstract

Optimization of many practically relevant and interesting problems in informatics and robotics are expensive tasks with respect to computing resources. Solving these problems requires expertise and costs time and resources. If a solution space grows too complex, formal solution methods reach a limit. Evolutionary algorithms produce through trial and error new approximate solutions. Following the principle of the survival of the fittest, these approximations should improve over time.

In this paper we try to teach artificial animals with six legs how to move through a course. It is explored how the geometry and movement sequence can be evolved. Afterwards, an analysis is done on how the movement sequence and the shape of an evolved individual looks. We postulate a number of hypotheses: 1. How can a movement controller be implemented? 2. How can this controller be evolved? 3. How can the shape of these animals be evolved? 4. What looks the movement sequence of evolved animals like? 5. Do small mutation probabilities yield better fitness than big probabilities? The hypotheses are examined based on the results of different simulations.

As a main result we show that three different types of movement sequences were developed. A jump, row and roll movement. Animals which use a rolling as movement to have a spherical body. When animals use rowing, their body resembles a flat shape. However, animals which jump don't show a clear trend. As further optimization, a feedback system can be utilized to tackle ascending and descending parts of a course.

# **Vorwort**

Diese Arbeit ist als Teil des Bachelor-Studiums des Informatik-Studiengangs an der Zürcher Hochschule für angewandte Wissenschaften am Institut für angewandte Mathematik und Physik zwischen Februar und Juni 2016 entstanden.

Wir möchten uns bei allen denen bedanken, die uns während dem Verfassen der Arbeit stets unterstützt und motiviert haben.

Besonders möchten wir unseren Dank an unsere Betreuer Herrn Prof. Dr. Rudolf Füchslin und Herrn Prof. Dr. Olaf Stern aussprechen, welche uns mit Rat und Tat zur Seite standen und allen unseren Anliegen mit Verständnis und konstruktiven Diskussionen begegneten.

Herrn Dr. Dandolo Flumini danken wir für die initiale Idee zum Thema der Arbeit.

# Inhaltsverzeichnis

<b>Erklärung</b>	i
<b>Zusammenfassung</b>	ii
<b>Abstract</b>	iii
<b>Vorwort</b>	iv
<b>1 Einleitung</b>	1
1.1 Ausgangslage . . . . .	1
1.2 Aufgabenstellung . . . . .	2
1.3 Zielsetzung . . . . .	2
1.4 Anforderungen . . . . .	3
1.4.1 Fitnessfunktion . . . . .	3
1.4.2 Parcours . . . . .	3
1.4.3 Limiten . . . . .	4
<b>2 Grundlagen</b>	5
2.1 Natürliche vs. artifizielle Evolution . . . . .	5
2.2 Artifizielle Evolution . . . . .	5
2.2.1 Individuum . . . . .	5
2.2.2 Genotyp . . . . .	5
2.2.3 Genom . . . . .	5
2.2.4 Phänotyp . . . . .	5
2.2.5 Genetische Repräsentation . . . . .	6
2.2.6 Initiale Population . . . . .	7
2.2.7 Fitnessfunktion . . . . .	7
2.2.8 Diversität . . . . .	7
2.2.9 Selektionsoperation . . . . .	7
2.2.10 Rekombinationsfunktion . . . . .	9
2.2.11 Mutationsfunktion . . . . .	11
2.2.12 Arten von evolutionären Algorithmen . . . . .	12
2.3 Frozen Accident . . . . .	13
<b>3 Methoden</b>	14
3.1 Design der Individuen . . . . .	14
3.1.1 Körper . . . . .	14
3.1.2 Beine . . . . .	14
3.1.3 Genotyp . . . . .	15
3.1.4 Phänotyp . . . . .	17
3.2 Bewegungsablauf . . . . .	18
3.2.1 Funktionsprinzip . . . . .	18
3.2.2 Vorgegebener Bewegungsablauf . . . . .	19
3.3 Auswahl des evolutionären Algorithmus . . . . .	21

3.4	Eingesetzte Technologien . . . . .	22
3.4.1	Auswahl der Physik-Engine . . . . .	22
3.4.2	Fehler in der Physik-Engine . . . . .	22
3.4.3	Laufzeitumgebung . . . . .	23
3.5	Konfiguration . . . . .	24
3.5.1	Technische Parameter . . . . .	24
3.6	Ablauf der Simulation . . . . .	25
3.6.1	Initiale Population . . . . .	27
3.6.2	Parcours-Generierung . . . . .	27
3.6.3	Evaluation der Fitnessfunktion . . . . .	27
3.6.4	Reporting . . . . .	28
3.6.5	Selektion . . . . .	28
3.6.6	Mutation . . . . .	29
3.6.7	Abbruchkriterium . . . . .	29
3.6.8	Domänenmodell . . . . .	29
<b>4</b>	<b>Resultate</b>	<b>32</b>
4.1	Allgemeine Lösung — erster Simulationslauf . . . . .	32
4.1.1	Konfiguration . . . . .	32
4.1.2	Auswertung . . . . .	32
4.2	Allgemeine Lösung — zweiter Simulationslauf . . . . .	34
4.2.1	Konfiguration . . . . .	34
4.2.2	Auswertung . . . . .	35
4.3	Allgemeine Lösung — dritter Simulationslauf . . . . .	36
4.3.1	Konfiguration . . . . .	36
4.3.2	Auswertung . . . . .	37
4.4	Allgemeine Lösung — vierter Simulationslauf . . . . .	39
4.4.1	Konfiguration . . . . .	39
4.4.2	Auswertung . . . . .	40
4.5	Ervolvieren auf Evolvierbarkeit — fünfter Simulationslauf . . . . .	44
4.5.1	Konfiguration . . . . .	44
4.5.2	Auswertung . . . . .	44
<b>5</b>	<b>Diskussion und Ausblick</b>	<b>46</b>
5.1	Diskussion der Resultate . . . . .	46
5.1.1	Wie kann eine Steuerung der Bewegung implementiert werden? . . . . .	46
5.1.2	Wie kann diese Steuerung evolviert werden? . . . . .	47
5.1.3	Wie kann die Geometrie der Tiere evolviert werden? . . . . .	48
5.1.4	Nimmt die Diversität mit zunehmenden Generationen stetig ab? . . . . .	48
5.1.5	Wie sieht der Bewegungsablauf und die Geometrie eines evolvierten Tieres aus? . . . . .	49
5.1.6	Individuum: Allgemeine Lösung vs. evolvieren auf Evolvierbarkeit . . . . .	50
5.1.7	Hypothese Körperfunkte . . . . .	51
5.1.8	Hypothese Mutationswahrscheinlichkeiten . . . . .	51
5.2	Ausblick . . . . .	52
5.2.1	Feedback an den Bewegungsmotor . . . . .	52
5.2.2	<i>N</i> -beinige Tiere . . . . .	52
5.2.3	Austauschen der Physik-Engine . . . . .	53

## *Inhaltsverzeichnis*

5.2.4 Hypothese Selektionsstrategie . . . . .	53
5.2.5 Hypothese Allgemeine Lösung vs. evolvieren auf Evolvierbarkeit	53
5.2.6 Parcours . . . . .	54
<b>Literaturverzeichnis</b>	<b>55</b>
<b>Glossar</b>	<b>57</b>
<b>Abbildungsverzeichnis</b>	<b>59</b>
<b>Tabellenverzeichnis</b>	<b>61</b>
<b>Anhang</b>	<b>62</b>
1 Projektmanagement . . . . .	62
1.1 Protokolle . . . . .	66

# 1 Einleitung

In der Informatik und Robotik ist das Lösen von praktisch relevanten und interessanten Problemstellungen eine aufwendige Aufgabe. Das Lösen dieser Problemstellungen erfordert viel Fachwissen, kostet Zeit und Ressourcen. Sobald der Lösungsraum eines Problems zu komplex wird, stoßen die formalen Lösungsmethoden an ihre Grenzen. Oft spielt dabei die Anzahl der Freiheitsgrade eine entscheidende Rolle. Alternativ kann eine Lösung durch evolutionäre Algorithmen gesucht werden. Die Algorithmen produzieren Populationen, welche die möglichen Lösungen beinhalten. Das Konzept ist der natürlichen Evolution nachempfunden, wo durch "trial and error" neue Lösungen generiert werden. Mit dem Prinzip "survival of the fittest" werden immer bessere Näherungen für ein Problem produziert. Die Kombination aus Reproduktion, Vererbung, Variation und Selektion übernimmt dabei die Rolle des Lösens und Optimierens einer Problemstellung.

Dawkins [1] definiert die vier Begrifflichkeiten (auch universeller Darwinismus genannt) wie folgt:

- Reproduktion: Die Individuen müssen fähig sein, sich fortzupflanzen.
- Vererbung: Der Nachwuchs muss fortpflanzungsfähig sein und erbt Eigenschaften der Eltern.
- Variation: Es muss ein Mechanismus existieren (Mutation Abschnitt 2.2.11 auf Seite 11), welcher neue Variationen in die Nachkommen einführen kann.
- Selektion: Die vererbten und varierten Merkmale müssen die Reproduktionsfähigkeit beeinflussen. Die Selektion bestimmt die Individuen, welche sich reproduzieren können.

In dieser Arbeit wird versucht mit evolutionären Algorithmen Individuen zu entwickeln, welche sich lernen fortzubewegen und somit einen Parcours bewältigen können.

## 1.1 Ausgangslage

Evolutionäre Algorithmen sind bereits in vielen Bereichen im Einsatz. Besonders bei Problemstellungen, wo eine approximative Lösung ausreichend ist, wird dieses Verfahren eingesetzt.

Forscher der NASA verwenden evolutionäre Algorithmen, um Antennen zu entwerfen, die für den Einsatz im Weltraum gedacht sind [2]. Zum ersten Mal wurden dabei durch evolutionäre Algorithmen entworfene Teile im Weltraum eingesetzt. Die Anforderung an die Antenne ist ein optimales Verhältnis, zwischen Energiebedarf und Empfangsleistung zu bieten. Im Vergleich zu den formalen Methoden wurden durch die evolutionären Algorithmen viele Antennenstrukturen gefunden, die von Ingenieuren nicht entworfen worden wären. Dabei wurde der Zeitaufwand, der für

den Entwurf und das Testen benötigt wird, stark reduziert. Ebenfalls ermöglicht der Einsatz von evolutionären Algorithmen Szenarien zu prüfen, die wegen Zeitmangels nicht hätten geprüft werden können.

Auch das Verhalten von Robotern kann mit Hilfe von evolutionären Algorithmen entwickelt werden. Floreano [3] zeigt, wie die Steuerung und Kommunikation von Robotern entwickelt werden kann. Dabei zeigen sich Ähnlichkeiten in der Evolution von künstlichen Robotern und natürlichen Organismen, wenn es um Verhaltensmuster geht. Ein grosses Problem der evolutionären Robotik ist, wenn Individuen eine spezifische Eigenschaft einer Umgebung ausnützen. Sobald aber die Umgebung gewechselt wird, wo diese Eigenschaft fehlt, sinkt die Fitness [3, S.6]. Diese Einschränkung wird durch die verwendete künstliche Umgebung hervorgerufen.

Als theoretische Grundlagen werden die Bücher “Bio-inspired artificial intelligence” [4] und “Evolutionäre Algorithmen” [5] verwendet.

## 1.2 Aufgabenstellung

Im Folgenden geben wir die Aufgabenstellung, wie sie zu Beginn der Arbeit definiert wurde wieder: “Ziel der BA ist es, in einer virtuellen Umwelt mit Hilfe evolutionärer Algorithmen künstliche Tiere entstehen zu lassen. Evolvieren werden die Geometrie und die Steuerung der Tiere, d.h. die Anordnung der Beine und deren Kontrolle. Als Mass für die Fitness wird eine Funktion der Zeit, welche das Tier braucht, um einen gegebenen Parcours zurückzulegen, verwendet. Als Randbedingungen vorgegeben sind: erstens die Forderung einer physikalisch sinnvollen Bewegung, zweitens eine obere Grenze für die total aufgewandte Energie und drittens eine Beschränkung der abgegebenen Leistung. Damit die Forderung nach einer physikalischen Bewegung erfüllt werden kann, wird die Bewegung mit Hilfe einer Physik-Engine simuliert. Diese Engine wird von den Studierenden ausgewählt und kann als Blackbox verwendet werden.

Die BA beinhaltet folgende Forschungsfragen: 1) Wie kann eine Steuerung der Bewegung implementiert werden? 2) Wie kann diese Steuerung evolvieren? 3) Wie kann die Geometrie der Tiere evolvieren?

Erfüllungskriterien: A) Die Steuerung von Tieren mit vorgegebener Geometrie (6 Beine) wird evolviert, sodass ein Parcours mit vorgegebenen Eigenschaften bewältigt werden kann. B) In einem zweiten Schritt wird sowohl die Geometrie als auch die Steuerung evolutiv weiterentwickelt; dies mit dem Ziel, die für einen Parcours benötigte Zeit zu reduzieren.” [6]

## 1.3 Zielsetzung

Das Ziel dieser Arbeit ist, sechsbeinigen künstlichen Tieren mit Hilfe eines evolutionären Algorithmus in einer zweidimensionalen Umgebung eine Fortbewegungsart beizubringen. Es soll nicht nur die Steuerung der Beine evolvieren, sondern auch die Form des Körpers und die Abmessung der Beine.

Folgende Fragestellungen werden untersucht:

1. Wie kann eine Steuerung der Bewegung implementiert werden?
2. Wie kann diese Steuerung evolvieren?

3. Wie kann die Geometrie der Tiere evolviert werden?
4. Wie sieht der Bewegungsablauf und die Geometrie eines evolvierten Tieres aus?
5. Nimmt die Diversität (Abschnitt 2.2.8 auf Seite 7) mit zunehmenden Generationen stetig ab?
6. Unterscheidet sich ein Individuum, welches mit dem Ziel der allgemeinen Lösung gefunden wurde, von einem das mit dem Evolvieren auf Evolvierbarkeit gefunden wurde (Abschnitt 3.5 auf Seite 24)?

Eine bestehende *Physik-Engine* darf für das Simulieren der Tiere in einer virtuellen Umgebung eingesetzt werden. Der Hauptfokus liegt auf dem Evolvieren der artifiziellen Tiere und nicht auf dem Ausbau der verwendeten *Physik-Engine*. Ein Werkzeug in Form einer selbst programmierten Applikation soll erstellt werden, um den evolutionären Algorithmus zu implementieren.

Weitere Hypothesen (Abschnitt 3.1.1.1 auf Seite 14 und Abschnitt 3.6.6.1 auf Seite 29) werden im Kapitel Methoden (Kapitel 3 ab Seite 14) erwähnt, da die erforderlichen Begrifflichkeiten dann eingeführt sind.

## 1.4 Anforderungen

### 1.4.1 Fitnessfunktion

Die Fitness ist ein Mass für den Grad der Anpassung eines Individuums an die Umgebung. In frühen Generationen können Tiere existieren, die sich nicht fortbewegen können. Die Zeit, welche ein Individuum für die Bewältigung eines Parcours (Abschnitt 1.4.2) brauchen würde, wäre somit unendlich. Um dieses Problem zu umgehen, wird als Mass für die Fitness die zurückgelegte Strecke verwendet. Für die Bewältigung des Parcours steht dem Individuum eine fixe Anzahl Zeiteinheiten zur Verfügung.

### 1.4.2 Parcours

Die Individuen müssen während der Simulation eine Strecke zurücklegen. Diese Strecke und ihre Begebenheiten werden als Parcours bezeichnet. Die Abb. 1.1 zeigt eine Skizze des Parcours. Mit zunehmenden Generationen soll die Schwierigkeit des Parcours steigen. Die Schwierigkeit eines Parcours wird durch die maximale Steigung definiert. Der Parcours soll zufällig generiert werden.

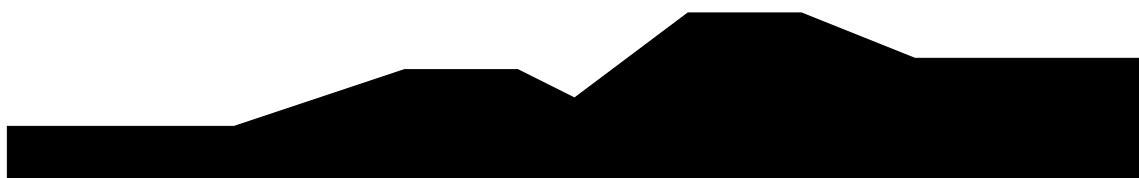


Abbildung 1.1: Skizze Parcours

### 1.4.3 Limiten

Limiten verhindern, dass der Parcours durch triviale Lösungen wie z.B. unendlich grosse Beine bewältigt werden kann. Ebenfalls schränken sie den Lösungsraum ein und helfen gewinnbringende Lösungen zu finden.

Für folgende Komponenten werden Limiten eingeführt:

- Winkelgeschwindigkeit auf den Gelenken: Verhindert Beschleunigung ins unendliche.
- Beinhöhe: Verhindert die triviale Überwindung von Hindernissen.
- Beinbreite: Verhindert unbrauchbare Lösungen
- Masse des gesamten Individuums: Vereinfachung der Problemstellung
- Fläche des Körpers (Körperpunkte dürfen nicht ausserhalb des Konstruktionskreises (Abschnitt 3.1.3.3 auf Seite 16) liegen): Verhindert unendlich grosse Individuen

# 2 Grundlagen

## 2.1 Natürliche vs. artifizielle Evolution

Natürliche Evolution hat kein vordefiniertes Ziel und ist ein sogenannter “open-ended” Anpassungsprozess. Unter einem “open-ended” Anpassungsprozess in der natürlichen Evolution versteht man die Anpassung an die natürliche Umgebung. Da sich die natürliche Umgebung stetig ändert, ist der Prozess immer in Gange und endet nie. Artifizielle Evolution jedoch ist ein Optimierungsprozess, welcher versucht Lösungen zu vordefinierten Problemen zu finden [4, S.1].

## 2.2 Artifizielle Evolution

### 2.2.1 Individuum

Als Individuum wird die zu evolvierende künstliche Kreatur bezeichnet. Ein Individuum hat einen zugehörigen Genotyp (Abschnitt 2.2.2) und einen Phänotyp (Abschnitt 2.2.4).

### 2.2.2 Genotyp

Das genetische Material eines Individuums wird als Genotyp bezeichnet [4, S.5]. Es beinhaltet alle wichtigen Informationen zur Reproduktion des Individuums.

### 2.2.3 Genom

Das Genom ist eine Repräsentation des Genotyps. Es beinhaltet nur Werte, jedoch keine Information was diese bedeuten. Erst durch den Geno- und Phänotyp werden den Werten einen Sinn gegeben.

### 2.2.4 Phänotyp

Der Phänotyp ist das Erscheinungsbild eines Individuums. Ein Individuum welches durch den Genotyp beschrieben wird, nimmt durch den Phänotyp eine sichtbare Form an.

Wenn der Genotyp die Länge und Breite eines Rechtecks beschreibt, so ist das grafisch gezeichnete Rechteck auf dem Monitor der Phänotyp (Abb. 2.1).

Genotyp	Phänotyp
Länge	3
Breite	2

Abbildung 2.1: Genotyp und Phänotyp

## 2.2.5 Genetische Repräsentation

Der erste Schritt bei der Definition eines evolutionären Algorithmus ist die Auswahl der genetischen Repräsentation. Nicht alle Arten von evolutionären Algorithmen harmonieren mit jeder Repräsentation. Die genetische Repräsentation beschreibt die Elemente eines Genoms und wie diese auf einen Phänotyp abgebildet werden [4, S.16].

### 2.2.5.1 Diskrete Repräsentation

Bei der diskreten Repräsentation kann das Genom als binären String dargestellt [0111110] werden. Dieser binäre String kann anschliessend in einen Phänotyp übersetzt werden. Zum Beispiel kann eine Bitsequenz, direkt zu einer Zahl als Phänotyp übersetzt werden [0011] -> 3.

Eine diskrete Repräsentation kann ebenfalls eine Folge von beliebigen Zeichen annehmen [ABCDEF]. D. Floreano und C. Mattiussi [4, S.18] zeigen dies anhand des “Travelling Sales Man Problems”: Jeder Buchstabe in der Sequenz repräsentiert dabei einen Ort, welchen es zu besuchen gilt (Abb. 2.2) [4, S.18].

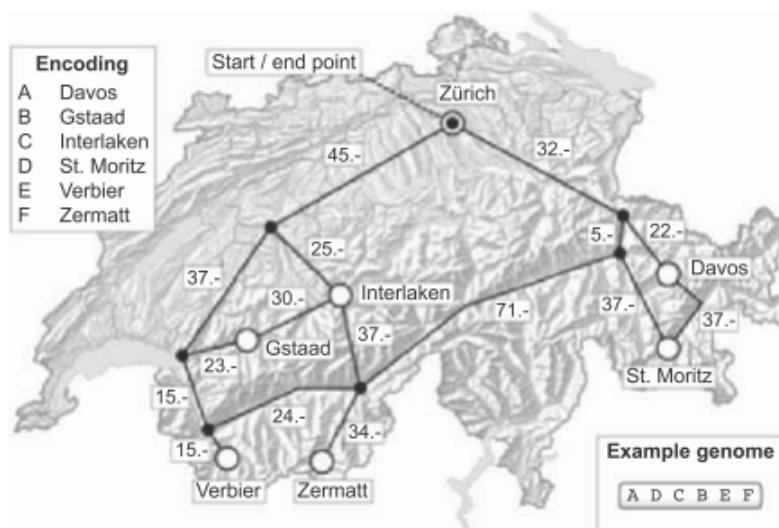


Abbildung 2.2: Beispiel Diskrete Repräsentation

### 2.2.5.2 Reale-Werte-Repräsentation

Weiter kann die Reale-Werte-Repräsentation gewählt werden. Das Genom (Abb. 2.3) wird hierbei durch reelle Zahlen repräsentiert. Beispielsweise kann die optimale Position eines Zimmers (beste Flächennutzung) in einem Haus durch reale Zahlen dargestellt werden.

Reale-Werte-Repräsentation	2.2	3.5	-1.2	4.5	1.7
----------------------------	-----	-----	------	-----	-----

Abbildung 2.3: Beispiel Reale-Werte-Repräsentation

### 2.2.5.3 Baum-Repräsentationen

Das zu evolvierende Objekt kann durch einen Baum dargestellt werden. Baum-Repräsentationen (Abb. 2.4 auf der nächsten Seite) werden eingesetzt um

hierarchische Strukturen mit Verzweigungen und Bedingungen zu beschreiben [4, S.19]. Baumstrukturen haben den Vorteil, dass sie sehr gut rekursiv durchlaufen werden können. Viele Probleme aus der Informatik lassen sich einfacher rekursiv als iterativ lösen.

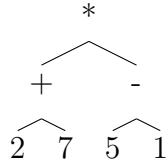


Abbildung 2.4: Darstellung einer Rechnung als Baum

## 2.2.6 Initiale Population

Die Grösse der zu evolvierenden Population kann selber bestimmt werden. Jedoch muss beachtet werden, dass eine grössere Population mehr Rechenaufwand bedeutet. Auch die Eigenschaften des Suchraums dürfen dabei nicht ignoriert werden. Wichtig bei der Erstellung einer initialen Population ist, möglichst diverse (Abschnitt 2.2.8) Individuen zu generieren, damit nicht wertvolle Lösungen verspielt werden.

## 2.2.7 Fitnessfunktion

Mit Hilfe der Fitnessfunktion lassen sich Individuen beurteilen, wie gut geeignet ihre Gene sind, um die Problemstellung zu bewältigen. In der natürlichen Evolution ist die Fitness des Tieres, wie viele Nachkommen es erzeugen kann.

In der technischen Welt jedoch muss der Anwender sie jeweils selbst definieren und anhand der Problemstellung anpassen. Oft ist die Evaluation der Fitness-Funktion die rechenintensivste Teilaufgabe eines evolutionären Algorithmus [4, S.22].

## 2.2.8 Diversität

Die Diversität einer Population von Individuen beschreibt, wie verschieden die Individuen zueinander sind. Durch diese Kennzahl alleine lässt sich jedoch noch keine Aussage über die Diversität treffen. Erst wenn mehrere Generationen von Populationen vorhanden sind, kann diese Kennzahl benutzt werden um die Diversität zu beurteilen. Eine nummerisch grössere Zahl deutet darauf hin, dass die Individuen divers sind.

## 2.2.9 Selektionsoperation

Eine Selektionsoperation hilft die geeigneten Individuen einer Generation zu selektieren. Die Selektierten bilden die Basis für die nächste Generation. Eine grosse Herausforderung beim Selektieren ist die Erhaltung der Diversität.

### 2.2.9.1 Selektionsdruck

Als Selektionsdruck wird der Prozentsatz der Individuen der aktuellen Generation bezeichnet, welche man verwendet um Nachkommen zu erzeugen. Ein hoher Selektionsdruck bedeutet, dass nur wenige Individuen zur Reproduktion selektiert werden [4, S.23].

### 2.2.9.2 Fitnessproportionale Selektion

Bei der fitnessproportionalen Selektion wird die Reproduktionsrate proportional zur Fitness gewählt. D. Floreano und C. Mattiussi [4, S.23] beschreiben die proportionale Selektion als Rouletterad, in dem jedes Individuum ein Stück des Rouletterads für sich beanspruchen (Abb. 2.5). Das Stück welches sie beanspruchen, ist so gross, wie ihre Fitness im Vergleich zu der Fitness ihrer Konkurrenten.

Das Individuum mit der grössten Fitness wird auch das grösste Stück des Rouletterads für sich beanspruchen und hat somit die grösste Wahrscheinlichkeit, Nachkommen zu erzeugen. Wenn die Population  $N$  Individuen gross ist, wird das Rad  $N$  mal gedreht um die Nachkommen zu bestimmen.

Proportionale Selektion funktioniert schlecht, wenn alle Individuen sehr ähnliche Fitnesswerte aufweisen oder es nur wenige/einen Ausreisser gibt. Denn dann haben alle Individuen die gleiche Chance selektiert zu werden.

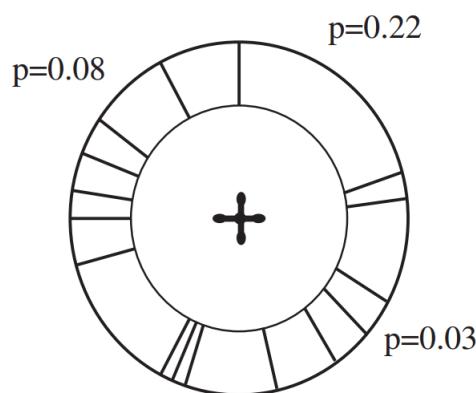


Abbildung 2.5: Rouletterad der proportionalen Selektion

### 2.2.9.3 Rangselektion

Es wird zuerst eine Rangliste nach Fitness erstellt und dann werden die Reproduktionswahrscheinlichkeiten proportional zum Rang zugeordnet. Anstatt das Rouletterad (Abb. 2.5) nach Fitness der Individuen zu unterteilen, wird es anhand des Ranges unterteilt. Diese Selektionsstrategie hat den Vorteil gegenüber der proportionalen Selektion, dass die Fitness der Individuen ähnlich sein darf, jedoch wird mit hoher Wahrscheinlichkeit das Bessere selektiert werden.

### 2.2.9.4 Gekürzte Rangselektion

Es wird eine Rangliste für die Individuen der Population erstellt. Anstatt alle Individuen zu berücksichtigen, werden nur die besten der Rangliste selektiert und diese produzieren Nachkommen (Abb. 2.6).

Schritt 1: Rangliste	1	2	3	4	5	6	7	8	9	10
Schritt 2: Selektion	1	2	3	4	5	6	7	8	9	10
Schritt 3: Reproduktion	1	1	1	1	1	2	2	2	2	2

Abbildung 2.6: Schritte einer gekürzten Rangselektion

### 2.2.9.5 Turnierselektion

Es werden  $k$  zufällig ausgewählte Individuen selektiert, diese Individuen tragen untereinander ein Turnier (Abb. 2.7) aus. Das Turnier gewinnt jenes Individuum, welches den höchsten Fitnesswert aufweist. Dies wird solange wiederholt, bis so viele Nachkommen vorhanden sind wie in der vorherigen Generation.

Ein grosser Vorteil von Turnierselektion ist die gute Balance zwischen Selektionsdruck und genetischer Diversität, da alle Individuen in der Population die gleiche Wahrscheinlichkeit haben, für das Turnier selektiert zu werden. Somit können auch Individuen mit niedrigem Fitnesswert ein Turnier gewinnen.

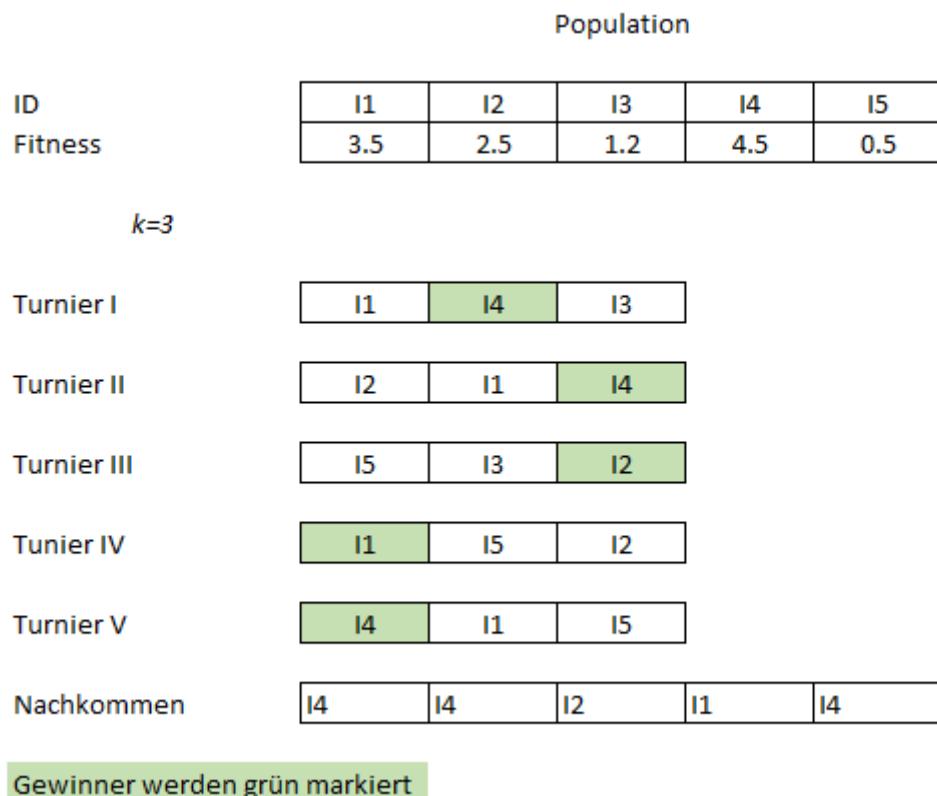


Abbildung 2.7: Turnierselektion

### 2.2.10 Rekombinationsfunktion

Bei der Rekombination werden jeweils zwei Individuen selektiert und deren Genpaare neu kombiniert (untereinander vertauscht). Die Abb. 2.8 auf der nächsten Seite veranschaulicht diesen Prozess.

Nicht bei jedem Typ von evolutionären Algorithmen (Abschnitt 2.2.12 auf Seite 12) und jeder Problemstellung ist Rekombination sinnvoll. Als Beispiel kann ein Haus mit Fenstern und Türen genommen werden. Dabei führt die Rekombination der Gene von Türen mit denen der Fenster zu keinen verwendbaren Resultaten.

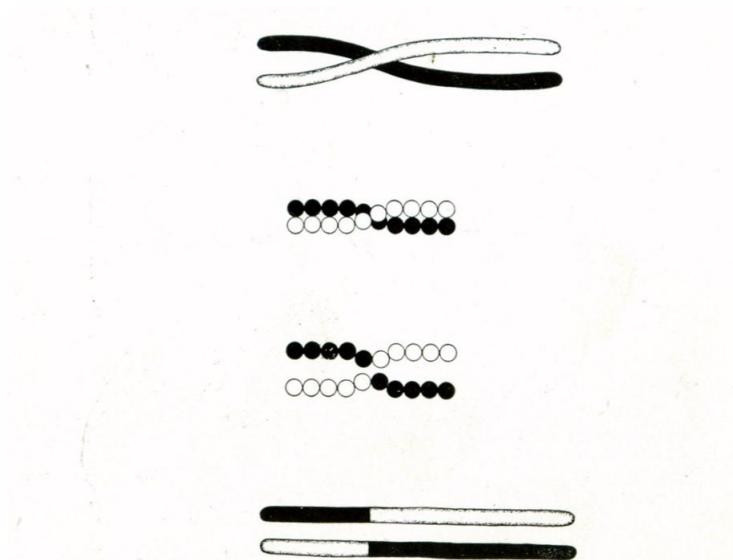


FIG. 64. Scheme to illustrate a method of crossing over of the chromosomes.

Abbildung 2.8: Illustration des Crossover

### 2.2.10.1 One-Point-Crossover

Es wird ein zufälliger Cross-Over-Punkt bestimmt, an dem die Gene des Paares vertauscht werden (a) Abb. 2.9 auf der nächsten Seite. Anwendbar ist diese Strategie bei Diskret- und Reale-Werte-Repräsentationen.

### 2.2.10.2 Multi-Point-Crossover

Dieser Ansatz funktioniert ähnliche wie One-Point-Crossover. Statt nur an einem Punkt, werden mehrere Punkte definiert an denen die Gene ausgetauscht werden.

### 2.2.10.3 Uniform-Crossover

Die Gene werden an  $n$  zufälligen Stellen vertauscht ( b) Abb. 2.9 auf der nächsten Seite).

### 2.2.10.4 Arithmetic-Crossover

Der Durchschnitt von den Genen an  $n$  zufälligen Positionen wird gebildet. Aus diesem Durchschnitt wird dann ein Nachkomme erzeugt ( c) Abb. 2.9 auf der nächsten Seite). Da arithmetische Operationen nur auf Zahlen anwendbar sind, kann diese Art des Crossovers nur bei Reale-Werte-Repräsentationen verwendet werden.

### 2.2.10.5 Sequenzen

Bei Sequenzen muss die Regel eingehalten werden, dass alle Einträge nur einmal vorkommen dürfen. Es wird ein Multi-Point-Crossover durchgeführt unter Einhaltung der oben genannten Regel ( d) Abb. 2.9 auf der nächsten Seite).

### 2.2.10.6 Bäume

Bei Bäumen wird ein zufälliger Teil eines Baumes, mit einem anderen zufälligen Teilbaum eines fremden Individuums vertauscht ( e) Abb. 2.9).

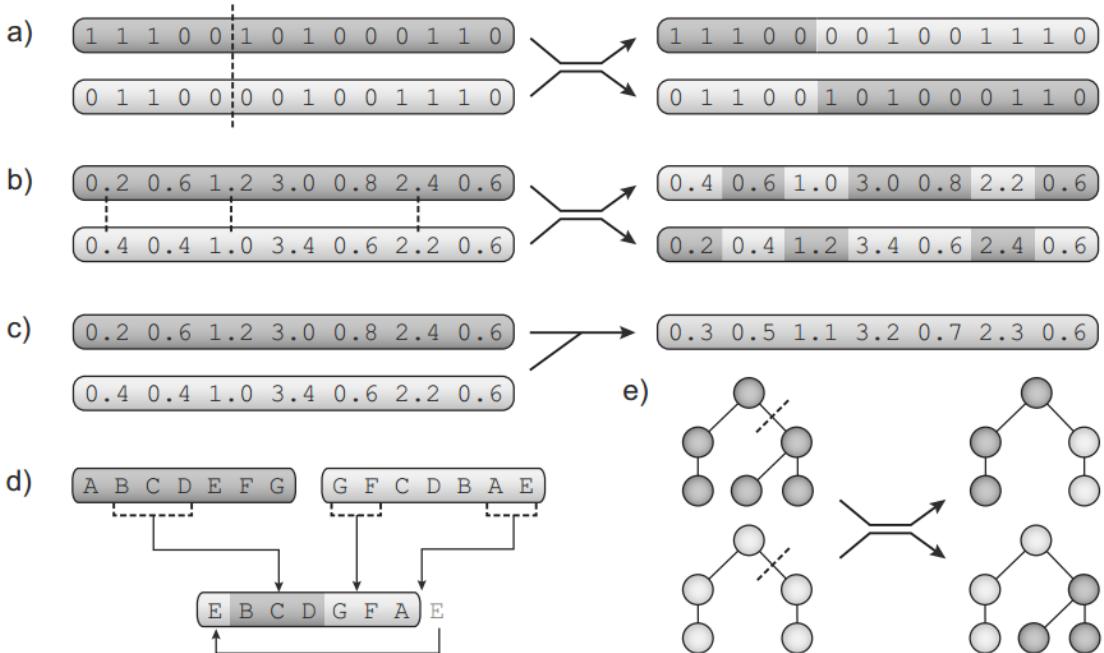


Abbildung 2.9: Typen von Crossovers

## 2.2.11 Mutationsfunktion

Mutationen operieren auf dem Genotyp des Individuums. Die Positionen eines Genoms werden mit einer bestimmten Wahrscheinlichkeit  $p_m$  mutiert.

Es ist jedoch Vorsicht geboten, da gewisse Lösungen durch Mutationen verloren gehen können [7]. Wenn zu viel mutiert wird, kann die Selektion und Evolution der Individuen nicht stattfinden. Es wird dann eher eine Zufallssuche durchgeführt als eine richtige Evolution. Darum sollten Mutationswahrscheinlichkeiten klein gewählt werden.

### 2.2.11.1 Binäre Repräsentationen

Wenn die Repräsentation aus binären Werten besteht, werden die Bits invertiert (oder bestimmte Segmente). Invertieren bedeutet, dass aus einem 0 eine 1 wird und umgekehrt (Abb. 2.10).

Bits des Genoms	1	0	0	1	1
Invertierte Bits des Genoms	0	1	1	0	0

Abbildung 2.10: Invertieren der Bits

### 2.2.11.2 Reale-Werte-Repräsentationen

Bei Realen-Werte-Repräsentation wird jeweils zu einem bestehenden Wert ein numerischer Wert addiert. Der Wert sollte zufällig berechnet werden. Die Zufallsfunktion, welche dafür verwendet wird, sollte normalverteilt sein. Zudem sollte die Mutation innerhalb eines Wertebereichs stattfinden (Abb. 2.11).

Genom	2.5	1.5	3.5	4.2	8.1
	+	+	+	+	+
zufall(-10,10)	1.1	-2.4	6.1	1.2	-1
	=	=	=	=	=
Genom nach Mutation	3.6	-0.9	9.6	5.4	7.1

Abbildung 2.11: Mutation von Reale-Werte-Repräsentationen

### 2.2.11.3 Sequenzen

Zufällige Positionen in Sequenz werden miteinander vertauscht (Abb. 2.12).

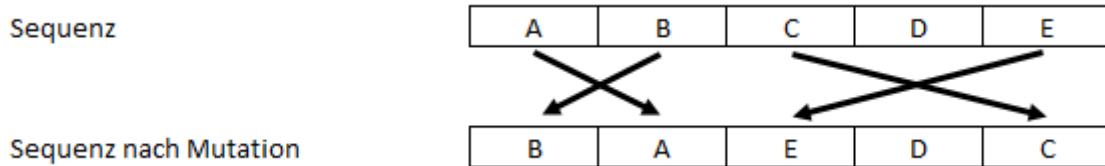


Abbildung 2.12: Mutation von Reale-Werte-Repräsentationen

### 2.2.11.4 Baum

Bei Bäumen werden Teilabschnitte mutiert, wie in Abb. 2.13 erkennbar ist.

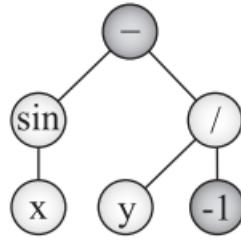
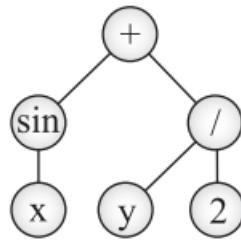


Abbildung 2.13: Mutation von Bäumen

### 2.2.12 Arten von evolutionären Algorithmen

Es gibt verschiedene Typen von evolutionären Algorithmen, welche sich hauptsächlich in der Wahl ihrer genetischen Repräsentation und Operationen unterscheiden [8]. Die Auswahl des evolutionären Algorithmus hängt von den Eigenschaften der Problemstellung ab. Es gibt keinen Algorithmus der alle Probleme optimal lösen kann [9]. Die am häufigsten verbreiteten Algorithmen sind: genetische Algorithmen (Abschnitt 2.2.12.1), genetische Programmierung (Abschnitt 2.2.12.2), evolutionäre Programmierung (Abschnitt 2.2.12.3) und evolutionäre Strategien (Abschnitt 2.2.12.4).

### 2.2.12.1 Genetische Algorithmen

Die Genetischen Algorithmen [10] arbeiten mit binären genetischen Repräsentationen. Es wird rekombiniert und selten mutiert [5, S.128]. Die Mutationen werden, falls eingesetzt, nur mit einer sehr kleinen Wahrscheinlichkeit durchgeführt [5, S.128]. Fitnessproportionale Selektion wird meistens als Selektionsoperator eingesetzt [5, S.128].

### 2.2.12.2 Genetische Programmierung

Bei der genetischen Programmierung [11] werden Bäume als Repräsentation des Genoms eingesetzt. Wie bei den genetischen Algorithmen wird Rekombination eingesetzt. Als Nebenoperator kann Mutation eingesetzt werden [5, S.147].

### 2.2.12.3 Evolutionäre Programmierung

Evolutionäre Programmierung versucht, die Evolution auf einer verhaltensbestimmten Ebene zu berücksichtigen, d.h. die Genetik wird nicht berücksichtigt [5, S.140]. Das bedeutet, dass bei der evolutionären Programmierung [12] ausschliesslich Mutation und keine Rekombination eingesetzt wird. Das Genom kann durch eine beliebige genetische Repräsentation dargestellt werden. Als Selektionsoperator wird oft Turnierselektion eingesetzt [4, S.33].

### 2.2.12.4 Evolutionäre Strategien

Evolutionäre Strategien [13] setzen ausschliesslich Reale-Werte-Repräsentationen ein [5, S.134]. Die Mutation wird als primärer Operator eingesetzt, während Rekombination teilweise zum Einsatz kommt [5, S.134].

## 2.3 Frozen Accident

Der Begriff “frozen accident” wird erstmals von F. H. C. Crick [14] beschrieben. Er beschreibt, dass der genetische Code ein Unfall (“accident”) ist, weil er nicht für Optimalität entworfen ist. Der Code gilt als eingefroren (“frozen”), weil viele Teile eines Organismus sich an einen anderen Code oder System anpassen müssten.

Alternative Systeme können effizienter sein, als das momentan verwendete. Doch es ist nicht möglich, dass das System in einem Schritt geändert wird. Nur einzelne, kleine Schritte in die entsprechende Richtung können unternommen werden. Dies wiederum führt zu Individuen mit geringerer Fitness. Da viele Individuen existieren, wird ein Individuum mit geringer Fitness, eine kleinere Chancen haben selektiert zu werden (Abschnitt 3.6.5 auf Seite 28).

# 3 Methoden

## 3.1 Design der Individuen

Der Ansatz für das Design der Individuen ist an *Hexapods* angelehnt. Dabei sind viele Eigenschaften der Ameise in den Entwurf eingeflossen. Sie bieten mit ihren sechs Beinen eine gute Kombination aus Stabilität und Einfachheit der Steuerung.

### 3.1.1 Körper

Die Form des Körpers soll frei evolvierbar sein. Deshalb wird der Körper eines Individuums als Polygon mit  $n$  Punkten beschrieben.

Als Einschränkung wird festgelegt, dass ein Polygon aus minimal vier und maximal acht Punkte generiert wird. Diese Bedingung beschränkt sowohl den Rechenaufwand der Berechnung des Körpers während der Simulation (Abschnitt 3.6.3 auf Seite 27), als auch in der Mutationsphase (Abschnitt 3.6.6 auf Seite 29).

Weiter wird festgelegt, dass das Polygon ein *einfaches Polygon* sein muss. Ohne diese Bedingung könnten Körper generiert werden, deren Kanten sich überkreuzen. Solche Körper werden als ungültig erachtet, da diese in der Natur nicht vorkommen.



Abbildung 3.1: Konzept Körper

#### 3.1.1.1 Hypothese zu Körperpunkten

Es wird die Hypothese aufgestellt, dass sich ein Individuum mit acht Körperpunkten schneller durch den Parcours bewegt, als ein Individuum mit weniger Körperpunkten. Die Vermutung dabei ist, dass durch die komplexere Form des Individuums die Balance besser gehalten werden kann. Im Kapitel 4 ab Seite 32 und Kapitel 5 ab Seite 46 wird diskutiert, ob dies zutrifft.

### 3.1.2 Beine

Da als Grundkonzept ein *Hexapod* verwendet wird, wird die Eigenschaft von sechs Beinen übernommen. Analog zu einer Ameise werden die Beine in drei symmetrische Beinpaare aufgeteilt.

Technisch bietet sich die Möglichkeit, die Beine nicht in Paare zu gliedern und asymmetrisch am Körper anzubringen. Doch in der Natur existiert eine solche Anordnung nicht, weshalb darauf verzichtet wird.

Ein Bein ist unterteilt in Oberschenkel  $T$  und Unterschenkel  $S$ . Die Höhe  $h_l$  und Breite  $w_l$  des Beins sind limitiert. Andernfalls ist anzunehmen, dass ungewünschte Resultate entstehen Abschnitt 1.4.3 auf Seite 4.

Im Genom bestimmt der Wert des Höhenfaktors  $h_f$ , wie das Verhältnis der Länge von Unterschenkel  $h_s$  zu Oberschenkel  $h_t$  ist.

Die Schenkel sind durch das Kniegelenk  $j_k$  miteinander verbunden. Mit dem Hüftgelenk  $j_h$  wird das Bein mit dem Körper verbunden. Für die Position des Hüftgelenkes existiert die Bedingung, dass das Gelenk innerhalb der Körperfläche angebracht sein muss. Zur Vereinfachung wird die Höhe jedes Gelenkes auf 0 festgelegt.

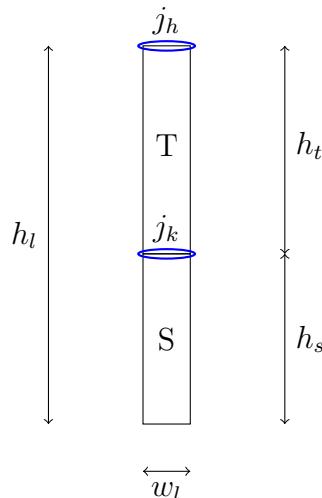


Abbildung 3.2: Konzept Bein

### 3.1.2.1 Masse

Das ganze Individuum hat eine konstante Masse  $m$  von 1. Diese Masse wird auf die einzelnen Körperteile aufgeteilt. Als Verteilschlüssel dient die Fläche eines Teiles.

#### Beispiel

Sei die Fläche eines Körpers  $a_b = 2$  und die eines Beines jeweils  $a_l = 1$ , dann werden die Masse des Körpers  $m_b$  und eines Beines  $m_l$  wie folgt berechnet:

$$\begin{aligned} m &= 1, a_l = 1, a_b = 2 \\ a &= a_l * 6 + a_b = 8 \\ x &= m/a = 0.125 \\ m_l &= x * a_l = 0.125 \\ m_b &= x * a_b = 0.25 \end{aligned}$$

### 3.1.3 Genotyp

#### 3.1.3.1 Design

Der Anspruch an das Design des Genotyps ist möglichst modular und einfach zu sein. Als Grund für ein modulares Design spricht die Erweiterbarkeit und Wieder-verwendung der bestehenden Teile. Deshalb wurde der Genotyp in Teil-Genotypen unterteilt.

Ein Teil-Genotyp repräsentiert dabei eine beliebige Eigenschaft des Individuums. Weiter können Teil-Genotypen wiederum in verschiedene weitere Teile aufgespalten werden. Damit entsteht die Möglichkeit, Individuen nach einem dynamisch generierten Bauplan zu erstellen und beliebige Kombinationen aus vorhandenen Teil-Genotypen innerhalb eines Genotyps zu bilden, ohne dass die Implementation der Seed-Funktion (Abschnitt 3.1.3.2) angepasst werden muss.

#### 3.1.3.2 Seeding

Das zufällige Erstellen eines Genotyps wird “seeding” genannt. Zu Beginn der Simulation wird die initiale Population (Abschnitt 3.6.1 auf Seite 27) mit zufällig generierten Individuen erstellt. Dabei ist es hilfreich manche Parameter nicht zufällig generieren zu lassen, sondern mit einem fixen Wert zu initialisieren.

So kann der Bewegungsablauf von Beginn an festgelegt werden. Deshalb wurde beim Seeding ein Ansatz gewählt, der es erlaubt flexibel jeden Teil-Genotyp mit spezifizierten Werten zu initialisieren.

#### 3.1.3.3 Generierung Körperpunkte

Als erster Ansatz für die Generierung von Körperpunkten wurde ein Quadrat evaluiert. Dieser Ansatz wurde nicht weiterverfolgt, da der Ansatz mit einem Kreis für beliebig viele und ungerade Anzahlen von Punkten einfacher lösbar ist.

Die Generierung eines zufälligen Körpers läuft in mehreren Schritten ab. Als erstes wird definiert aus wie vielen Punkten  $n$  der Körper besteht.

$$4 \leq n \leq 8$$

Anschliessend wird der Einheitskreis in  $n$  Sektoren unterteilt. Der Winkel  $\rho$  eines jeden Sektors entspricht:

$$\rho = \frac{2 * \pi}{n}$$

Für jeden Sektor  $s_i$  werden ein zufälliger Radius  $r$  und ein zufälliger Winkel  $\phi$  gewählt.

$$\begin{aligned} r &= \text{random}(0, 1) \\ \phi &= \text{random}(\rho * i, \rho * (i + 1)) \\ P &= (r, \phi) \end{aligned}$$

Damit die Koordinaten in der *Physik-Engine* verwendet werden können, müssen die Koordinaten in kartesischer Form sein. Die Polar-Koordinaten  $P$  werden in kartesische Koordinaten transformiert und miteinander verbunden.

Die folgende Abbildung Abb. 3.3 auf der nächsten Seite zeigt die Konstruktion der Form eines Individuums mit sechs Körperpunkten.

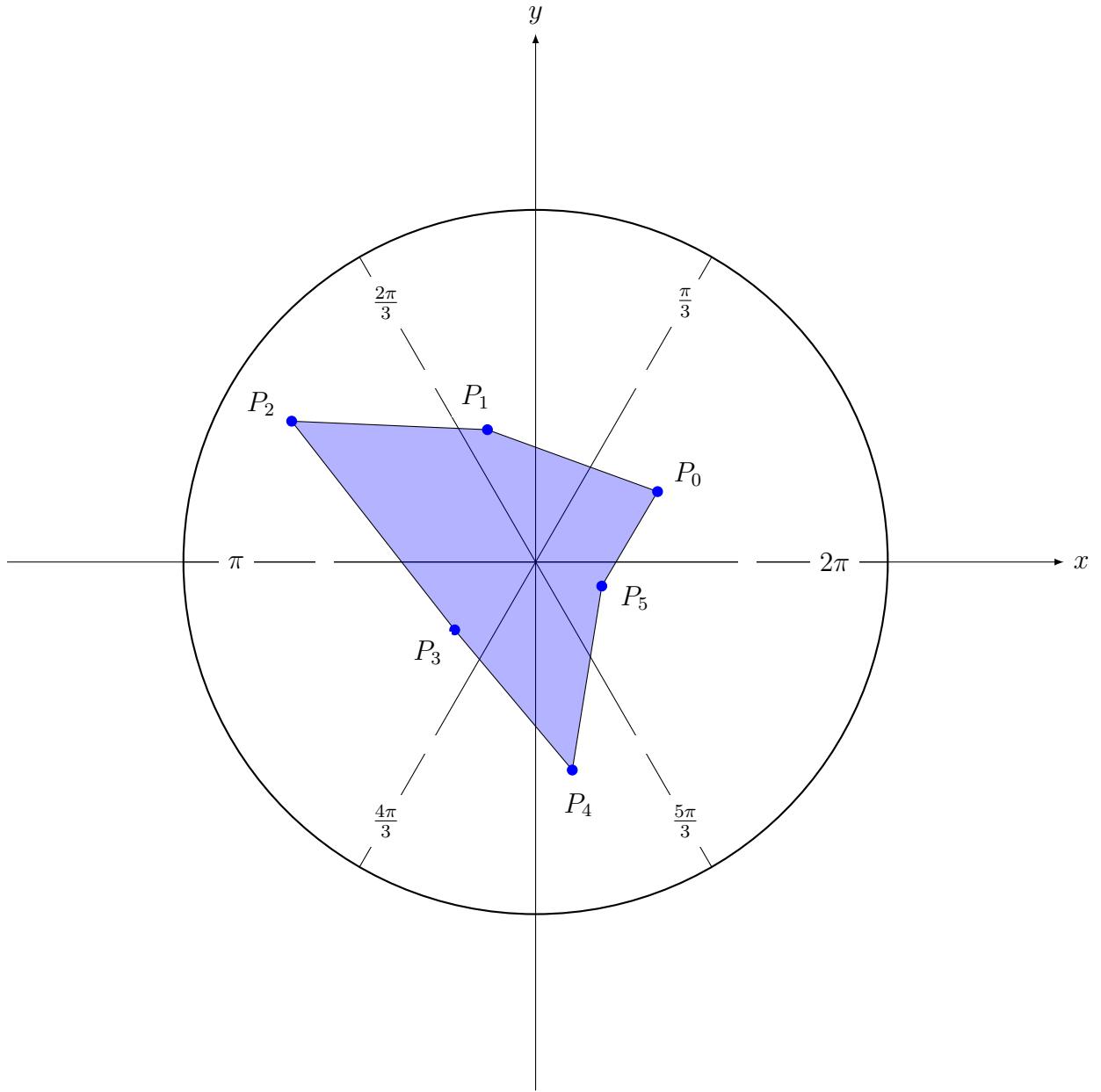


Abbildung 3.3: Berechnung der Körperpunkte veranschaulicht

### 3.1.4 Phänotyp

Aus den realen Werten des Genotyps wird der Phänotyp in der *Physik-Engine* erstellt. Dieser Prozess wird auch als Abbilden des Genotyps auf einen Phänotyp bezeichnet (Abschnitt 2.2.4 auf Seite 5). Der Phänotyp enthält alle Daten, welche die *Physik-Engine* braucht, um das Individuum zu simulieren. Er setzt sich aus mehreren geometrischen Figuren und Gelenken (*Constraints*) zusammen.

Die Beine des Individuums werden mit Hilfe eines Drehgelenks an den Körper gebunden. Mit Hilfe von Constraints wird diese Bindung an den Körper erreicht. Auf allen Gelenken ist ein Rotationsmotor vorhanden, der verwendet wird um eine Bewegung in Gang zu setzen.

## 3.2 Bewegungsablauf

Der Motor sorgt zusammen mit den Informationen zum Bewegungsablauf im Phänotyp für die Bewegung eines Individuums. Im Genotyp ist verankert, wann eine Bewegung (Abschnitt 3.2.2.1 auf Seite 20) vom Motor ausgeführt wird. Diese Informationen werden im Prozess übernommen, wo der Genotyp auf den Phänotyp abgebildet wird. Auf dem Phänotyp ist festgehalten, welches die aktuelle Bewegung im Ablauf ist.

### 3.2.1 Funktionsprinzip

Die Implementation des Motors ist ein *Endlicher Automat* (EA). Dabei repräsentieren die Bewegungen die Zustände des Motors zwischen denen gewechselt wird. Durch diese Art der Implementation kann ein zyklischer Bewegungsablauf einfach abgebildet werden.

Solange die aktuelle Bewegung  $M_i$  nicht vollendet ist, wird kein Zustandswechsel ausgeführt. Ist die Bewegung  $M_i$  vollendet, wird der Zustand gewechselt und die Bewegung  $M_{i+1}$  ausgeführt.

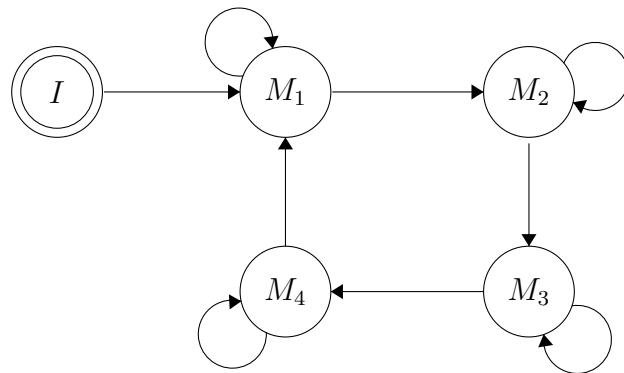


Abbildung 3.4: *Endlicher Automat* veranschaulicht

Aufgrund der Schnittstellen und bereits vorhandener Funktionen in der *Physik-Engine* ist ein Motor, der *joint-driven* arbeitet gewählt worden. Er führt die Bewegungen, die Körperteile in Bewegung setzen, über die Drehgelenke des Individuums aus. Mit der Kraft des Motors wird der Winkel eines Drehgelenkes verändert.

Am Beispiel des Kniegelenks ist ersichtlich wie der Motor arbeitet. Das Kniegelenk ist am Oberschenkel fixiert und verbindet den Unterschenkel mit dem Oberschenkel. Setzt der Motor das Gelenk in Bewegung wird der Unterschenkel bewegt.

Mit den Informationen des Phänotyps kann der Motor den gesamten Bewegungsablauf kontrollieren. Der Motor überwacht die Übergänge zwischen den Bewegungen. Nach jedem Übergang zu einer neuen Bewegung, führt der Motor diese aus und prüft, ob der Zustand gewechselt werden muss. Ein Zustandswechsel ist gleichzusetzen mit einer neuen Bewegung auszuführen.

#### 3.2.1.1 Ausführung in der Simulation

Die *Physik-Engine* führt die Simulation iterativ durch. D.h. der Zustand der Objekte in der Simulationswelt wird pro Schritt berechnet. Diese Eigenschaft nutzt der Motor, indem in während jedem Schritt die aktuelle Bewegung aller Individuen

bearbeitet. Damit ist die feinst mögliche Auflösung für die Bewegungssteuerung erreicht.

Als erstes bringt der Motor ein Individuum immer in die Ausgangsposition (Abb. 3.5a auf der nächsten Seite). Die Ausgangsposition wird definiert durch den initialen Bewegungsablauf. Dieser ist unabhängig vom Ablauf, der das Gehen (oder eine andere Form der Fortbewegung) definiert. Sobald die Ausgangsposition erreicht ist, beginnt der Motor mit dem Ausführen des Bewegungsablaufs.

#### 3.2.2 Vorgegebener Bewegungsablauf

Die Individuen der initialen Population Abschnitt 3.6.1 auf Seite 27 besitzen einen vorgegebenen Bewegungsablauf. Der Grund dafür ist, dass damit die Wahrscheinlichkeit steigt, dass die Evolution lauffähige Individuen hervorbringt.

Der vorgegebene Ablauf ist der Ameise nachempfunden, wegen der Einfachheit und Stabilität. Eine Ameise bewegt pro Schritt immer drei Beine. Alternierend wird auf einer Seite eines und auf der anderen zwei Beine gehoben und nach vorne bewegt. Diese Bewegung erlaubt es der Ameise stets stabil zu stehen, da ein Dreibein immer steht während die anderen bewegt werden.

Mit dem vorgegebenen Muster werden die Bewegungen für eine Seite erstellt. Anschliessend können die Bewegungen für die andere Körperseite mit kleinen Anpassungen übernommen werden.

0. Der Motor setzt die Beine und Gelenke in ihre Ausgangsposition (Abb. 3.5a auf der nächsten Seite)
  1. In der ersten Phase werden die Beine der linken Seite bis zum maximalen Winkel bewegt (Abb. 3.5c auf der nächsten Seite).
  2. Alle rechten Beine schieben nun den Körper nach vorne (Abb. 3.5d auf der nächsten Seite) bis der minimale Winkel des Hüftgelenks erreicht wurde (Abb. 3.5e auf der nächsten Seite).
  3. Die linken Beine werden nun gestreckt (Abb. 3.5f auf der nächsten Seite).
  4. Anschliessend werden die rechten Beine nach vorne bewegt (Abb. 3.5g auf der nächsten Seite und (Abb. 3.5h auf der nächsten Seite)) bis zum maximalen Winkel (Abb. 3.5i auf der nächsten Seite).
- Der Bewegungsablauf wird gespiegelt für die Beine der anderen Körperseite vollzogen.

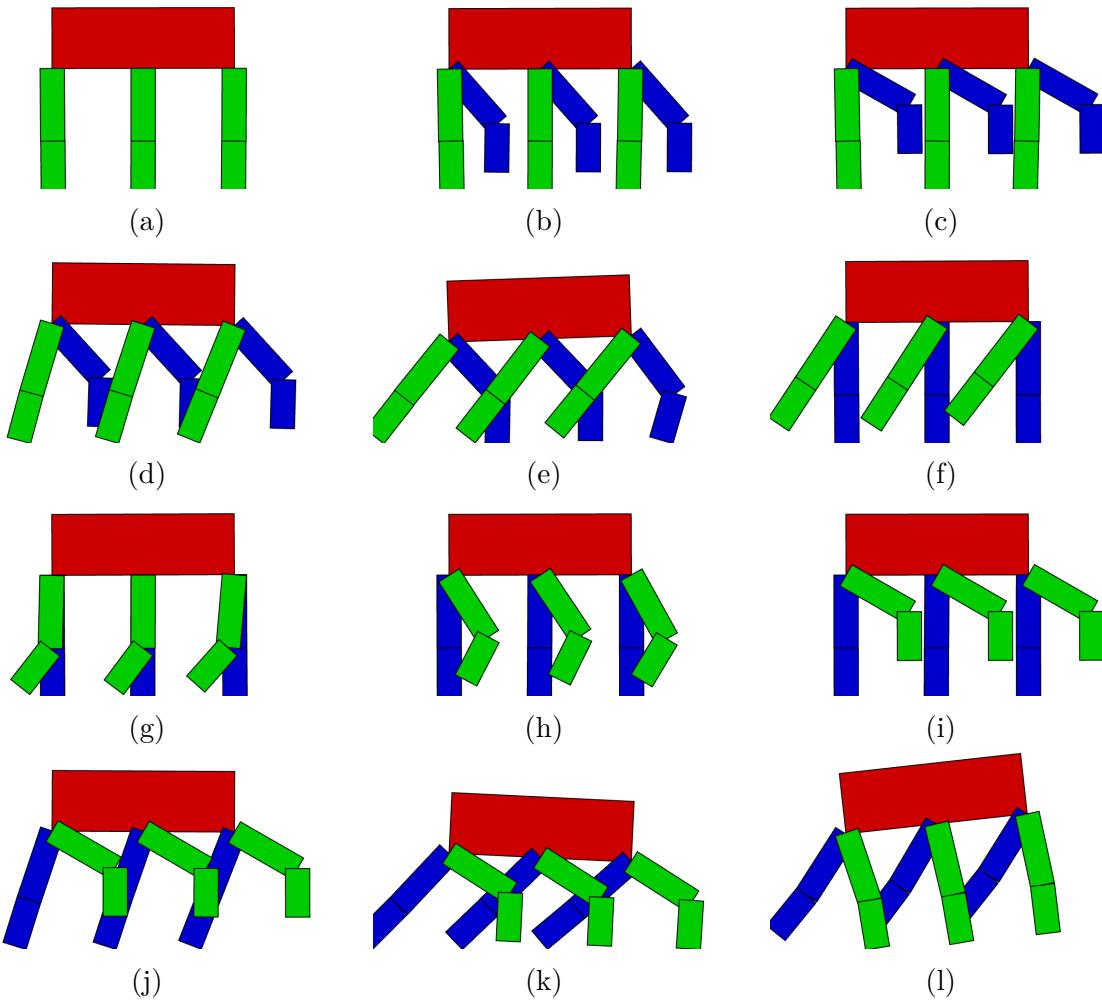


Abbildung 3.5: Bewegungsablauf im Zeitraffer

### 3.2.2.1 Bewegung

Unter dem Begriff Bewegung werden alle Aktionen, die der Motor ausführt verstanden. Es gibt neben den klassischen Bewegungen, wie ein Drehgelenk bewegen, auch solche die Attribute eines Gelenkes ändern oder den aktuellen Winkel eines Gelenkes abfragen.

Eine Bewegung ist mit mehreren Attributen beschrieben. Es wird festgelegt welche Art der Bewegung es ist, welches Gelenk gesteuert wird und was die Parameter für die Bewegung sind. Mit dieser Aufgliederung lassen sich einzelne Teile einer Bewegung mutieren. Insbesondere ist es möglich verschiedene Wahrscheinlichkeiten für Teile zu verwenden.

Sobald der Motor eine Bewegung ausführt, interpretiert er die Daten aus dem Phännotyp. Er liest die Beschreibung der aktuellen Bewegung und führt diese dann aus.

### 3.2.2.2 Feedback

Zur weiteren Verbesserung des Bewegungsablaufs, kann der Motor Rückmeldungen von der Simulationswelt anfordern. Der Motor wird von der *Physik-Engine* benachrichtigt, sobald eine Kollision zwischen einem Körperteil und dem Untergrund stattfindet.

Das Feedbacksystem ist nicht vollständig ausgearbeitet. Aus zeitlichen Gründen ist

dies nicht umgesetzt worden. Es ist nur die Rückmeldung an den Motor vorhanden, jedoch wird die Rückmeldung nicht verarbeitet. Im Kapitel Abschnitt 5.2.1 auf Seite 52 wird erläutert, was zusätzlich umgesetzt werden muss.

## 3.3 Auswahl des evolutionären Algorithmus

Es stehen 4 Typen von evolutionären Algorithmen zur Auswahl:

- *Genetische Algorithmen (GA)* (Abschnitt 2.2.12.1 auf Seite 13)
- *Genetische Programmierung (GP)* (Abschnitt 2.2.12.2 auf Seite 13)
- *Evolutionäre Programmierung (EP)* (Abschnitt 2.2.12.3 auf Seite 13)
- *Evolutionäre Strategien (ES)* (Abschnitt 2.2.12.4 auf Seite 13)

Das Austauschen der Gene (Rekombination) ist wenig hilfreich bei der Lösung der Problemstellung. Man stelle sich dazu vor, dass das Gen welches den Körper definiert, mit dem des Motors vertauscht wird. So kann keine sinnvolle Evolution stattfinden.

Rekombination kann für die Verwendung somit ausgeschlossen werden. Dies stellt schon den ersten Indikator dar, dass Evolutionäre Programmierung eingesetzt werden kann.

Ein weiteres Kriterium ist die genetische Repräsentation. Da sich das Individuum aus Punkten im einem Koordinatensystem, verschiedenen Winkeln und einem Bewegungsablauf definiert, wird die Reale-Werte-Repräsentation eingesetzt. Ein Punkt oder Vektor kann durch zwei reale Werte ausgedrückt werden. Dies gilt ebenso für Winkel oder den Bewegungsablauf.

Die binäre Repräsentation ist für die vielen realen Werte nicht geeignet. Theoretisch wäre es möglich alle realen Werte binär darzustellen, jedoch würde eine unnötige Komplexität hinzugefügt werden.

Die Repräsentation als Baum ist laut K. Weicker [5] eine Möglichkeit die Steuerung eines Roboters abzubilden. Da aber mehr als nur die Steuerung evolviert wird, bleibt die Entscheidung bei der Reale-Werte-Repräsentation.

Die Problemstellung würde anders bewältigt werden, würde sich die Thematik um reale Kreaturen drehen und nicht um virtuelle Roboter, welche in einer künstlichen Umgebung (virtuelles Koordinatensystem) das Laufen lernen. Bei realen Kreaturen spielt die Genetik eine zentrale Rolle, jedoch kann sie bei den virtuellen Robotern vernachlässigt werden. Schlussendlich wurde entschieden die Problemstellung, mit evolutionärer Programmierung zu lösen.

## 3.4 Eingesetzte Technologien

Zur Auswahl standen mehrere Programmiersprachen und *Physik-Engines*. Die Evaluationskriterien dabei waren:

- Plattform-Interoperabilität (Linux, Windows und Mac OS X)
- Einfache Handhabung
- Stabilität
- Funktionsumfang der *Physik-Engine*
- Ökosystem der Programmiersprache (verfügbare Bibliotheken)

*F#* und diverse populäre .NET *Physik-Engines* wurden evaluiert. Jedoch gestaltete sich die Konfiguration und Plattform-Interoperabilität schwierig. Eine plattformübergreifende Konfiguration, konnte nicht erstellt werden.

Dies lag nicht an den *Physik-Engines*, sondern an der Open-Source-Implementation von “Mono Game”. “Mono Game” und Mono werden verwendet, um die .NET-Umgebung auf Linux und Mac zu nutzen.

Als weitere Option wurde Javascript und die *Physik-Engines* in Betracht gezogen. Die Popularität von Javascript hat in den letzten Jahren stark zugenommen. Auch wurde mit dem neuen Standard ECMA2015 die Sprache enorm verbessert und viele neue Funktionalitäten eingeführt. Das Javascript-Ökosystem ist eines der grössten überhaupt.

Da Javascript ursprünglich eine Webtechnologie ist und auf allen Plattformen problemlos läuft, wurde das Kriterium der Plattform-Interoperabilität ohne zusätzliche Konfiguration erfüllt.

### 3.4.1 Auswahl der Physik-Engine

Im Zentrum der Anwendung steht die *Physik-Engine*, welche die ganze Simulation berechnet. Für die Auswahl der *Physik-Engine* ist es besonders wichtig, dass diese die notwendige Funktionalität bereitstellt.

Die erste Option war “Matter.js”, welche als nicht genug ausgereift eingestuft wurde. Viele Funktionen haben zum Zeitpunkt der Evaluation gefehlt.

Deshalb wurde als zweite Option auf “p2.js” zurückgegriffen. Diese *Physik-Engine* bietet eine ausführliche Dokumentation und Beispiele zu sämtlichen Aspekten.

### 3.4.2 Fehler in der Physik-Engine

Für die Erstellung des Terrains wird ein Höhenfeld von p2.js benutzt. Die Kollisionserkennung zwischen einem Höhenfeld und einem Polygon funktioniert nicht einwandfrei. Manche Polygonecken werden nicht berücksichtigt in der Kollisionserkennung.

Das hat zur Folge, dass Körperteile der Individuen im Boden einsinken. Die Abb. 3.6a auf der nächsten Seite zeigt ein Individuum vor dem Einsinken und Abb. 3.6b auf der nächsten Seite zeigt ein Individuum mit einer eingesunkenen Spitze.

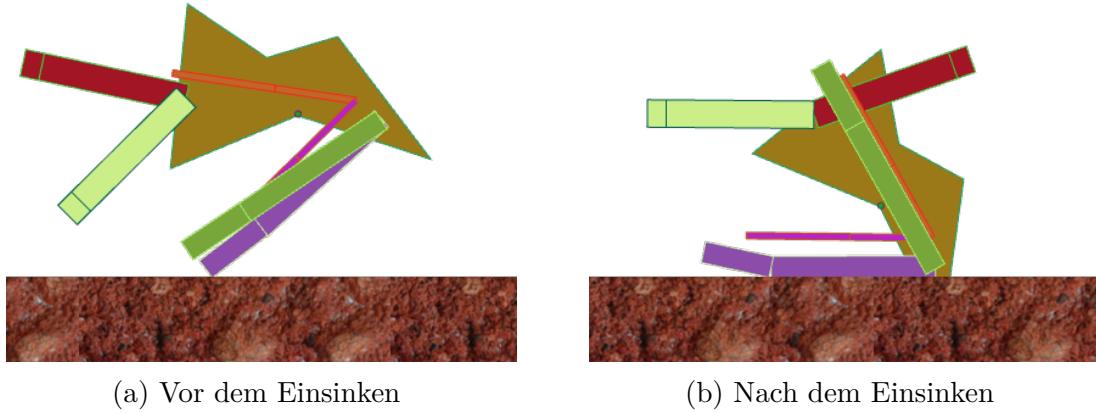


Abbildung 3.6: Vor Korrektur

Die Kollision mit einem Höhenfeld wird am besten erkannt mit einem Kreis. In Abb. 3.7a sind an allen Ecken des Körpers und der Beine kleine Kreise angebracht. Diese besitzen eine vernachlässigbare Masse, damit keine Nebeneffekte auftreten. So wird die Kollisionserkennung zwischen den Phänotypen und dem Untergrund verbessert.

Wie in Abb. 3.7b erkennbar, sinkt die Spitze des Individuums nicht mehr in den Boden ein. Die zusätzlichen Objekte, welche für diese Optimierung verwendet wurden, erhöhen den Rechenaufwand der Simulation.

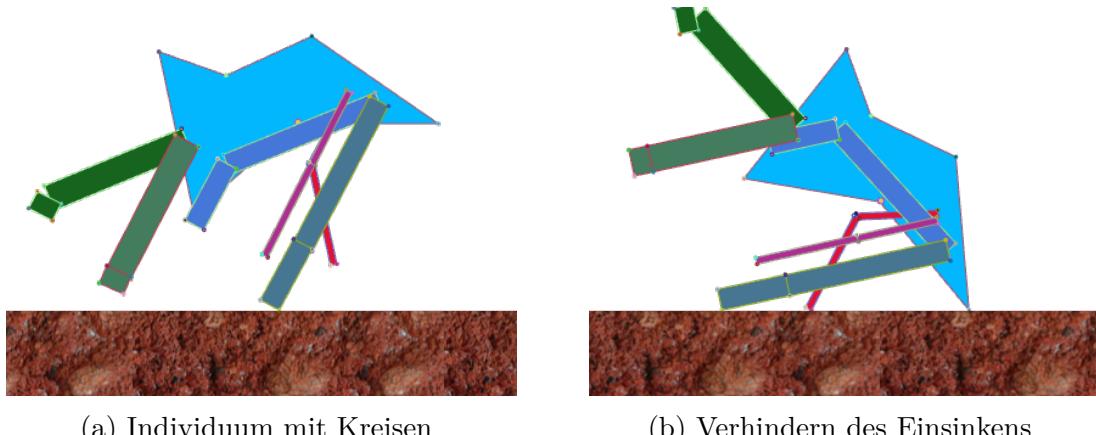


Abbildung 3.7: Nach Korrektur

### 3.4.3 Laufzeitumgebung

Javascript ist eine Script-Sprache. Diese benötigt ein Programm, welches den Quellcode interpretiert und ausführt. Als Plattform wurde deshalb Electron eingesetzt. Electron besteht aus node.js und Google Chrome.

Node.js ist eine Umgebung in der server-seitige Javascript-Anwendungen aufgeführt werden. Durch den Paketmanager *Node Package Manager (NPM)* werden externe Abhängigkeiten und Bibliotheken verwaltet. *NPM* bietet Zugriff auf eine Vielzahl von Bibliotheken.

Google Chrome wird verwendet für die Darstellung der Anwendung. Die Inhalte der Applikation werden wie eine Webseite erstellt und dargestellt.

Electron verbindet beide in einer Anwendung. Damit kann eine Web-Applikation in eine Desktop-Applikation verpackt werden.

## 3.5 Konfiguration

Die Applikation lässt sich mit diversen Parametern konfigurieren. Es wird eine Konfigurationstabelle (Tabelle 3.1) eingeführt, welche im Kapitel Resultate (Kapitel 4 ab Seite 32) verwendet wird, um die jeweilige Konfiguration der Simulation festzuhalten. In der Konfigurationstabelle werden nur die Parameter erwähnt, welche relevant für den Algorithmus sind.

Der Schwierigkeitsgrad des Parcours wird alle 100 Generationen erhöht. Bei der allgemeinen Lösung wird der Parcours nach jeder Generation neu generiert. Beim Evolvieren auf Evolvierbarkeit wird der Parcours nur dann neu generiert, wenn sich die Schwierigkeit des Parcours ändert.

Simulationsparameter	
Populationsgrösse	Grösse der Population
Selektionsstrategie	Eingesetzte Selektionsstrategie
Iterationen	Anzahl Simulationsläufe
Iterationsdauer	Dauer Simulationslauf
Schwierigkeit erhöhen pro Läufen	Erhöhung Schwierigkeit des Parcours ab x Läufen
Steigungszunahme	Zunahme Steigung zwischen zwei Punkten im Parcours
Maximale Steigung	Limitierung der Steigung
Höchste Y-Koordinate Zuwachs	Zuwachs höchstmögliche Y-Koordinate des Parcours
Ziel	Allgemeine Lösung oder evolvieren auf Evolvierbarkeit
Wahrscheinlichkeit Mutation	
Allgemein	Generelle $P$ Mutation
Engine	
add	$P$ Bewegung hinzuzufügen
remove	$P$ Bewegung zu entfernen
lens	$P$ Mutation der Gelenkbeschreibung
id	$P$ Mutation des Bewegungstyps
parameter	$P$ Mutation der Bewegungsparameter

Tabelle 3.1: Konfigurationstabelle Simulation

### 3.5.1 Technische Parameter

Ein wichtiger technischer Parameter ist die Anzahl der eingesetzten Simulationsprozesse (Abschnitt 3.6 auf der nächsten Seite). Durch die Erhöhung dieses Parameters kann von mehreren Prozessor-Kernen profitiert werden. Damit lässt sich die Simulation (Abschnitt 3.6.3 auf Seite 27) in der *Physik-Engine* und die Mutation (Abschnitt 3.6.6 auf Seite 29) parallelisieren. Optimal ist es diesen Wert auf die Anzahl verfügbarer Prozessorkerne zu setzen.

Die Grösse des Zeitschritts [15] der *Physik-Engine* lässt sich einstellen.

Die “relaxation” (Entspannung) kann auch konfiguriert werden. Dieser Parameter beschreibt die Anzahl von Zeitschritten, um eine Constraint-Gleichung [16] zu stabilisieren.

Die “stiffness” (Steifheit) ist der letzte wichtige technische Parameter. Ein Beispiel für diesen Parameter ist die Steifheit einer Feder. Er beschreibt, wie steif die Objekte innerhalb der Simulation sich verhalten sollen.

## 3.6 Ablauf der Simulation

Die Applikation setzt sich aus mehreren Komponenten zusammen. Jede dieser Komponenten kann dem Haupt- oder Simulationsprozess zugeordnet werden (Abb. 3.8). Während es nur eine Instanz vom Hauptprozess gibt, ist die Anzahl Simulationsprozesse frei wählbar (Abschnitt 3.5.1 auf Seite 24).

Die Kommunikation zwischen Simulationsprozessen und Hauptprozess erfolgt über *asynchrones Messaging*. In Abschnitt 3.6.8 auf Seite 29 sind die jeweiligen Zusammenhänge der Klassen und Objekte ausführlicher beschrieben. Die Abb. 3.9 auf der nächsten Seite beschreibt den generellen Ablauf als Flussdiagramm

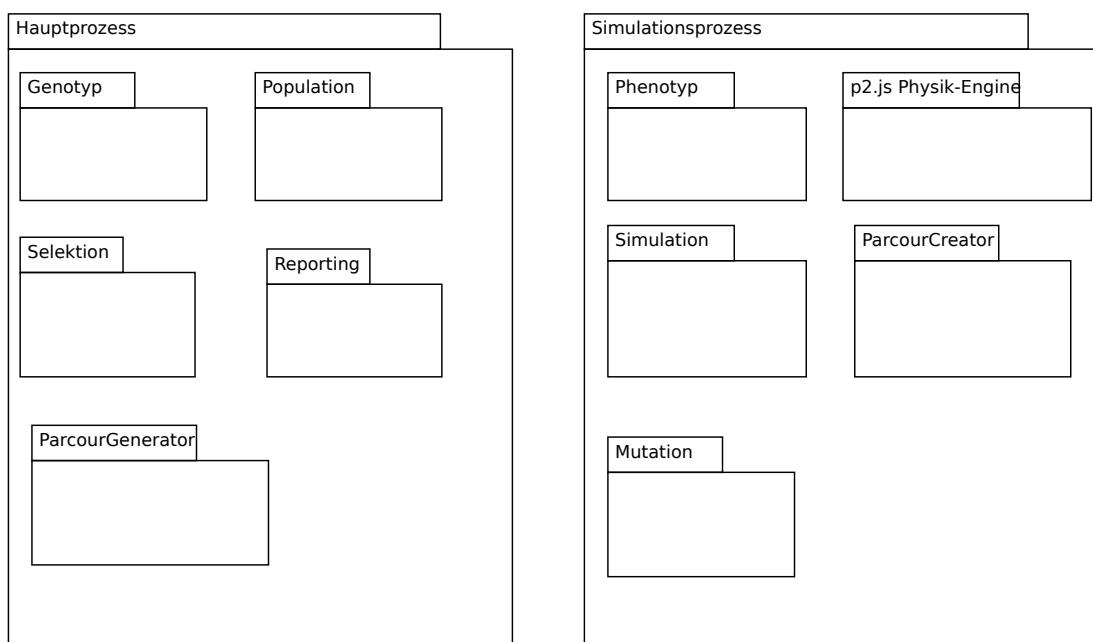


Abbildung 3.8: Haupt- und Simulationsprozess

### 3 Methoden

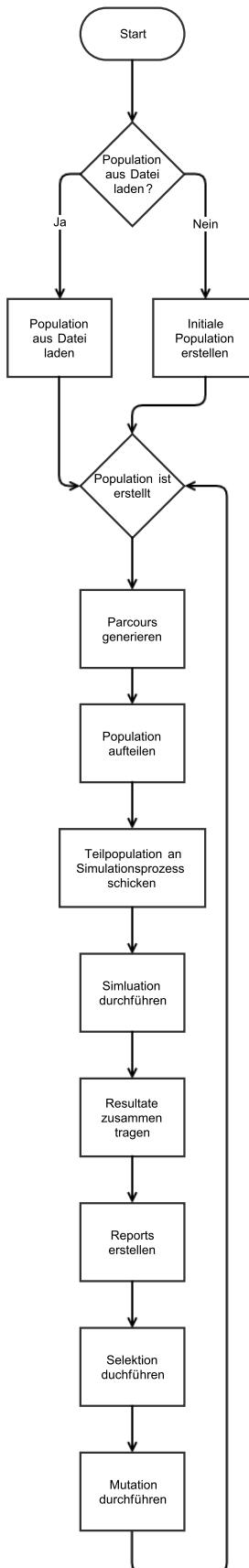


Abbildung 3.9: Ablauf der Simulation

### 3.6.1 Initiale Population

Als erster Schritt muss eine initiale Population erstellt werden. Die Populationsgrösse kann frei gewählt werden. Pro Simulationsprozess können etwa 15 Individuen flüssig berechnet werden. Dies bedeutet das die optimale Populationsgrösse  $Anzahl(CPUKerne) * 15$  beträgt. Dieser Wert wurde durch mehrere Versuche mit unterschiedlichen Prozessoren gefunden.

Es werden jeweils Individuen mit vier, fünf, sechs, sieben und acht Körperpunkten (Abschnitt 3.1.1 auf Seite 14) zufällig generiert. Die verantwortliche Klasse in Javascript ist der InitialPopulationGenerator. Das Endprodukt dieses Schrittes ist eine Population bestehenden aus den Genotypen der Individuen.

### 3.6.2 Parcours-Generierung

Der Parcours wird zufällig generiert. Am Anfang wird ein einfacher und flacher Parcours generiert. Mit zunehmenden Iterationen steigt die Schwierigkeit des Parcours, das heisst es werden höhere Steigungen und Punkte mit höheren Y-Koordinaten vorkommen. In Abb. 3.10, Abb. 3.11 und Abb. 3.12 ist ersichtlich, wie der Parcours mit zunehmenden Generationen schwieriger wird.

Der ParcourGenerator ist die verantwortliche Klasse in Javascript, welche mit Hilfe von Obergrenzen von Werten einen Parcours generieren kann.

Wie unter Abschnitt 3.5 auf Seite 24 erwähnt, wird je nach Einstellung der Parcours nach jeder Generation neu generiert (allgemeine Lösung) oder er wird wiederverwendet (evolvieren auf Evolvierbarkeit). Der Parcours wird im ParcourGenerator generiert (Hauptprozess) und von den jeweiligen Simulationsprozessen mit Hilfe vom ParcourCreator in der *Physik-Engine* erstellt.



Abbildung 3.10: flacher Start-Parcours 1. Generation



Abbildung 3.11: Parcours 100. Generation



Abbildung 3.12: Parcours 1000. Generation

### 3.6.3 Evaluation der Fitnessfunktion

Nachdem eine neue Population und ein neuer Parcours erstellt worden ist. Muss eine Evaluation der Fitnessfunktion durchgeführt werden. Die verantwortliche Klasse ist

die SimulationWorld. Zuerst muss jeder Genotyp zu einem Phänotyp abgebildet werden. Anschliessend muss der Parcours in der *Physik-Engine* erstellt werden. Erst dann kann mit dem Simulieren der Individuen begonnen werden.

Die SimulationWorld merkt sich zu jedem Individuum die Position. Individuen welche sich während vier Sekunden kaum bewegt haben, werden aus der laufenden Simulation entfernt und ihr Fitnesswert wird abgespeichert. Das Simulieren der Individuen kann wie unter Abschnitt 3.5 auf Seite 24 schon erwähnt, parallelisiert werden.

Jedem Simulationsprozess wird eine Teil-Population zugewiesen.

#### 3.6.4 Reporting

Das Reporting-Modul hilft nach einer Simulation alle wichtigen Daten festzuhalten. Mithilfe einer Funktion welche eine Population als Parameter entgegennimmt, werden die Reports erstellt.

Es existieren folgende Typen von Reports:

- Fitness Graph Average Report: Enthält Koordinaten, um einen Graphen über die durchschnittliche Fitness pro Generation zu erstellen.
- Fitness Graph Best Report: Enthält Koordinaten, um einen Graphen über das beste Individuum der Generation zu erstellen.
- Genotype Blueprint Report: Enthält pro Generation ein *JSON*-Objekt, das Informationen über die ganze Population enthält.
- Fitness Graph Average Report bp x: Für diesen Typ von Report gibt es jeweils einen für alle Individuen mit der Körperpunktanzahl x. Ansonsten gleich wie der Fitness Graph Average Report.
- Diversity Report: Enthält die Berechnung der Punktstreuung für eine Generation. Die Punktstreuung berechnet sich wie folgt:
  - Erstellen von Vektoren der Genome. Jede Eigenschaft des Genoms wird in einen numerischen Wert konvertiert.  $V_d$
  - Alle Vektoren werden summiert. Die Summe wird als  $V_s$  bezeichnet.
  - Der Schwerpunktvektor wird gebildet:  $V_s / \text{Anzahl}(V_d) = V_{schwer}$
  - Von jedem  $V_d$  wird  $V_{schwer}$  subtrahiert  $V_d - V_{schwer} = V_{d2}$
  - Nun werden alle  $V_{d2}$  normiert, quadriert  $\text{norm}(V_{d2})^2 = d$
  - Abschliessend werden alle  $d$  zusammengezählt und durch  $\text{Anzahl}(V_d)$  dividiert. So wird ein Mass für die Punktstreuung gewonnen.

#### 3.6.5 Selektion

D. Floreano und C. Mattiussi [4, S.33] schreiben, dass Turnierselektion oft bei evolutionärer Programmierung eingesetzt wird. Turnierselektion hat den Vorteil, dass die Diversität erhalten bleibt, bei gleichzeitig guter Selektion der fittesten Individuen. Aufgrund der Empfehlung von D. Floreano und C. Mattiussi und der Erhaltung der Diversität, wurde entschieden Turnierselektion einzusetzen.

### 3.6.6 Mutation

Auf die Selektionsstrategie folgt die Mutation der Individuen. Bei der Mutation wird über jede Eigenschaft des Individuums iteriert. Diese Eigenschaften werden mit einer Wahrscheinlichkeit und um einen Wert, die Schrittweite, verändert. Die Wahrscheinlichkeit kann Werte zwischen 0 und 1 annehmen. Falls eine Limite definiert wurde, darf die Summe des aktuellen Werts und der Schrittweite die Limite nicht überschreiten. Limiten (Abschnitt 1.4.3 auf Seite 4) werden definiert, um unvorteilhafte Lösung zu verhindern.

Die Position eines Beins wird unter einer zusätzlichen Bedingung mutiert. Falls sich der Körper so verändert hat, dass ein Bein ausserhalb des Körpers ist, wird die Position zufällig neu bestimmt. Die Mutation läuft wie die Simulation parallel ab.

#### 3.6.6.1 Hypothese Mutationswahrscheinlichkeiten

Es wird die Hypothese aufgestellt, dass kleinere Mutationswahrscheinlichkeiten bessere Fitnesswerte liefern als grosse. Die Annahme besteht darin, dass die Evolution der Individuen gebremst wird, falls grosse Mutationswahrscheinlichkeiten eingesetzt werden.

### 3.6.7 Abbruchkriterium

Der evolutionäre Algorithmus wird solange ausgeführt, bis der Anwender sich entscheidet ihn abzubrechen. Theoretisch wäre möglich abzubrechen, wenn ein bestimmter Fitnesswert erreicht wird. Ohne automatischen Abbruch ist mehr Flexibilität gewährleistet, der Anwender kann die Reports immer wieder überprüfen und entscheiden, ob die Individuen den Ansprüchen genügen.

### 3.6.8 Domänenmodell

Das Domänenmodell beschreibt den Zusammenhang der verschiedenen Klassen und Objekten. Die Abb. 3.13 auf der nächsten Seite beschreibt den Haupt- und Abb. 3.14 auf Seite 31 den Simulationsprozess.

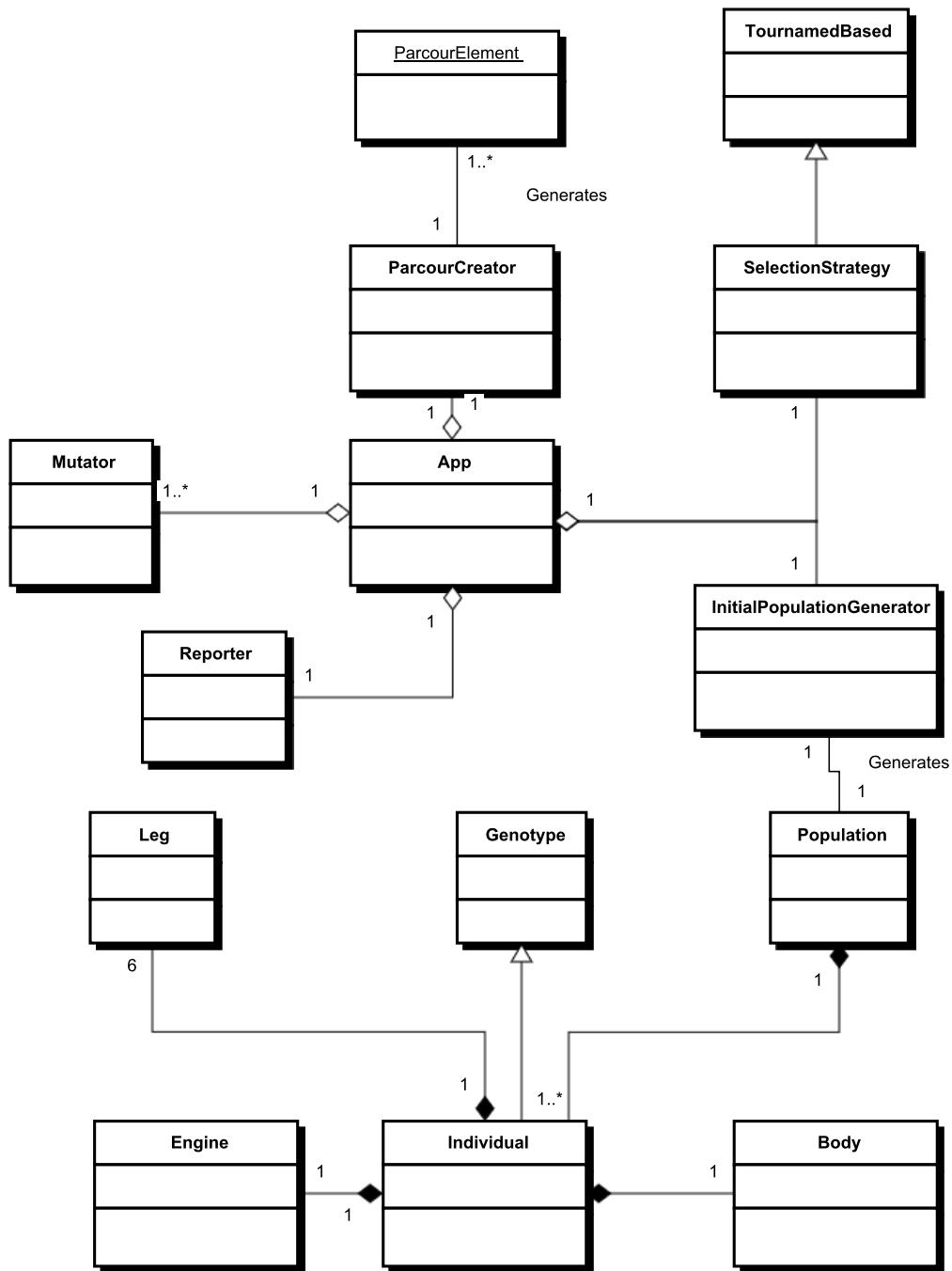


Abbildung 3.13: Domänenmodell Hauptprozess

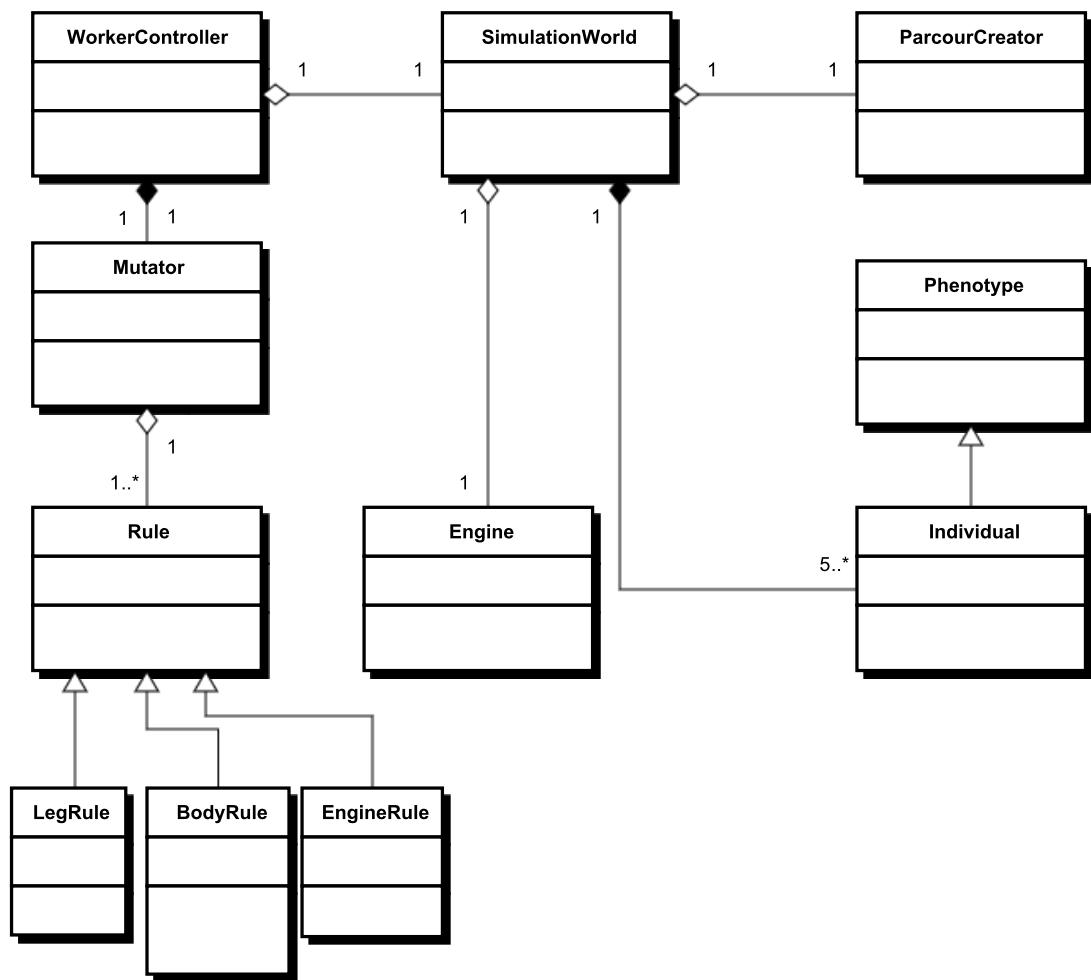


Abbildung 3.14: Domänenmodell Simulationsprozess

# 4 Resultate

## 4.1 Allgemeine Lösung — erster Simulationslauf

### 4.1.1 Konfiguration

Simulationsparameter	
Populationsgrösse	120
Selektionsstrategie	Turnierselektion
Iterationen	ca. 3100
Iterationsdauer	30s
Schwierigkeit erhöhen pro	100 Generationen
Steigungszunahme	0.02
Maximale Steigung	1.0
Höchste Y-Koordinate Zuwachs	0.05
Wahrscheinlichkeit Mutation pro Attribut	0.1
Ziel	Allgemeine Lösung

Tabelle 4.1: Simulationsparameter

Aufgrund der unvollständigen Mutation des Bewegungsablaufs sind die entsprechenden Konfigurationen in der Tabelle nicht vorhanden.

### 4.1.2 Auswertung

Es ist ein klarer Aufwärtstrend der Fitness zu erkennen bis etwa zur 1000. Generation (Abb. 4.1 auf der nächsten Seite). Danach ist ein Abfall der Fitness bei den Individuen zu beobachten.

Die Vermutung dabei ist, dass der Parcours zu schwierig ist und die Individuen mit der grossen Steigung nicht mehr zureckkommen.

Ausreisser sind dadurch zu erklären, dass ein bestimmter Parcours sehr einfach für die Individuen zu bewältigen war oder ihnen Schwierigkeiten bereitet hat.

In dieser Simulation ist der Bewegungsablauf nur wenig mutiert worden. Der Bewegungsablauf gleicht dem Vordefinierten.

Es wird vermutet, dass das Feedbacksystems der Steuerung hier helfen wird. Ausserdem müssen die Parameter des Parcours für die nächste Simulation anders konfiguriert werden. Der Steigungszuwachs sollte auf 0.01 limitiert werden und der Zuwachs der höchsten Y-Koordinate sollte auf 0.02 gesetzt werden. Mit der Herabsetzung dieser Werte werden weniger Mutation durchgeführt und die Lösungsmenge nicht zu stark eingegrenzt werden.

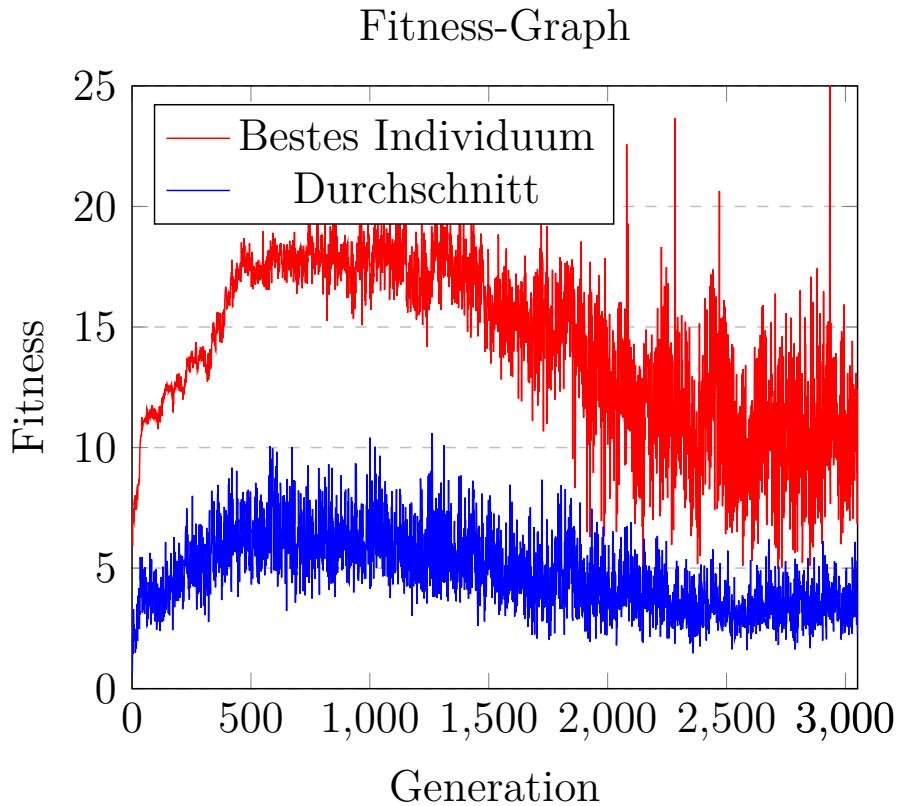


Abbildung 4.1: Bestes Individuum und durchschnittliches Individuum

#### 4.1.2.1 Körperfunkte

Die Auswertung der Individuen nach der Anzahl ihrer Körperfunkte ist ernüchternd (Abb. 4.2 auf der nächsten Seite). Bereits ab der 14. Generation existieren nur noch Individuen mit vier Körperfunkten. Einerseits könnte dies auf das fehlende Feedback zurückzuführen sein oder andererseits könnten die Individuen in der initialen Population mit mehr als vier Körperfunkten unvorteilhaft generiert worden sein. Eine weitere Simulation zeigt, ob vier Körperfunkten am geeignetsten sind für die Bewältigung der Problemstellung, oder ob es sich um einen “frozen accident” handelt.

### Fitness-Graph nach Körperpunkten aufgeteilt

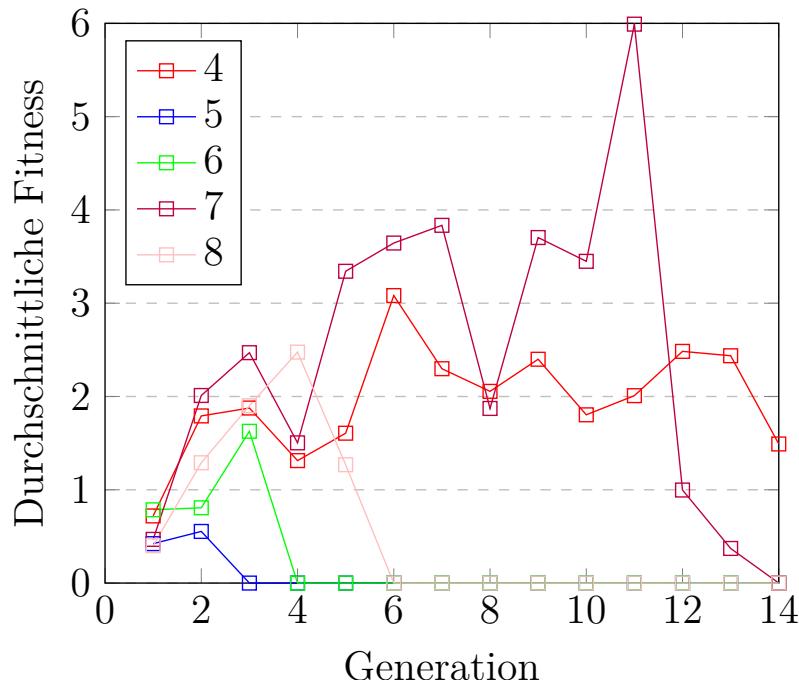


Abbildung 4.2: Durchschnittliches Individuum pro Anzahl Körperpunkte

## 4.2 Allgemeine Lösung — zweiter Simulationslauf

### 4.2.1 Konfiguration

Simulationsparameter	
Populationsgrösse	120
Selektionsstrategie	Turnierselektion
Iterationen	ca. 13900
Iterationsdauer	30s
Schwierigkeit erhöhen pro	100 Generationen
Steigungszuwachs	0.01
Maximale Steigung	1.0
Höchste Y-Koordinate Zuwachs	0.02
Wahrscheinlichkeit Mutation pro Attribut	0.1
Ziel	Allgemeine Lösung

Tabelle 4.2: Simulationsparameter

Aufgrund der unvollständigen Mutation des Bewegungsablaufs sind die entsprechenden Konfigurationen in der Tabelle nicht vorhanden.

Wie in Abschnitt 4.1.1 auf Seite 32 besprochen wurde die Parameter der Parcourschwierigkeit angepasst:

- Steigungszuwachs: 0.01
- Zuwachs der höchsten Y-Koordinate: 0.02

## 4.2.2 Auswertung

Beim zweiten Versuch ist interessant zu beobachten, dass die Fitness der Individuen erst ab der 1400. Generation (Abb. 4.3) wieder abzunehmen beginnt. Hier ist ebenfalls ein klarer, jedoch nicht gleich deutlich ausgeprägter Abwärtstrend zu erkennen. Eine weitere Vermutung, neben dem fehlenden Feedback des Motors, warum die Fitness wieder abzunehmen beginnt, ist die zu hohe Wahrscheinlichkeit der Mutation. In einem nächsten Versuch muss getestet werden, ob niedrigere Mutationswahrscheinlichkeiten einen positiven Effekt auf die langfristige Fitness der Individuen haben.

Ab ca. der 2000. Generation beginnt die Fitness wieder rapide zu steigen, jedoch ist dies durch einen Fehler der *Physik-Engine* zu erklären. Die Individuen verhaken sich im Boden und beginnen grosse Distanzen durch Hüpfbewegungen zurückzulegen. Dieser Fehler kann bereinigt werden indem die Relaxation (Abschnitt 3.5.1 auf Seite 24) erhöht und gleichzeitig der Zeitschritt (Abschnitt 3.5.1 auf Seite 24) verringert wird.

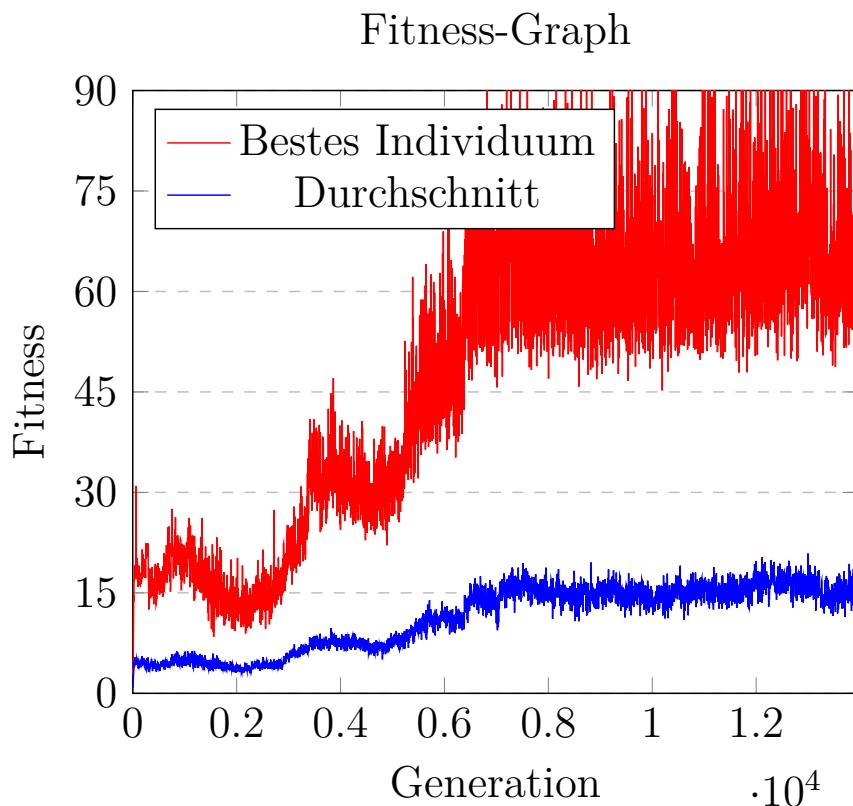


Abbildung 4.3: Bestes Individuum und Durchschnittliches Individuum

### 4.2.2.1 Körperpunkte

Ein ähnliches Bild ist zu beobachten, wie bei Abschnitt 4.1.2.1 auf Seite 33. Bereits ab der 9. Generation existieren nur noch Individuen mit sieben Körperpunkten (Abb. 4.4 auf der nächsten Seite). Es wird angenommen, dass hier ein “frozen accident” geschehen ist. Damit eine Aussage über den Einfluss der Körperpunkte getroffen werden kann, müsste während der ganzen Simulation die Anzahl der Körperpunkte mutiert werden. Stattdessen wird momentan zu Beginn der Simulation die Anzahl der Körperpunkte für die Individuen festgelegt.

### Fitness-Graph nach Körperpunkten aufgeteilt

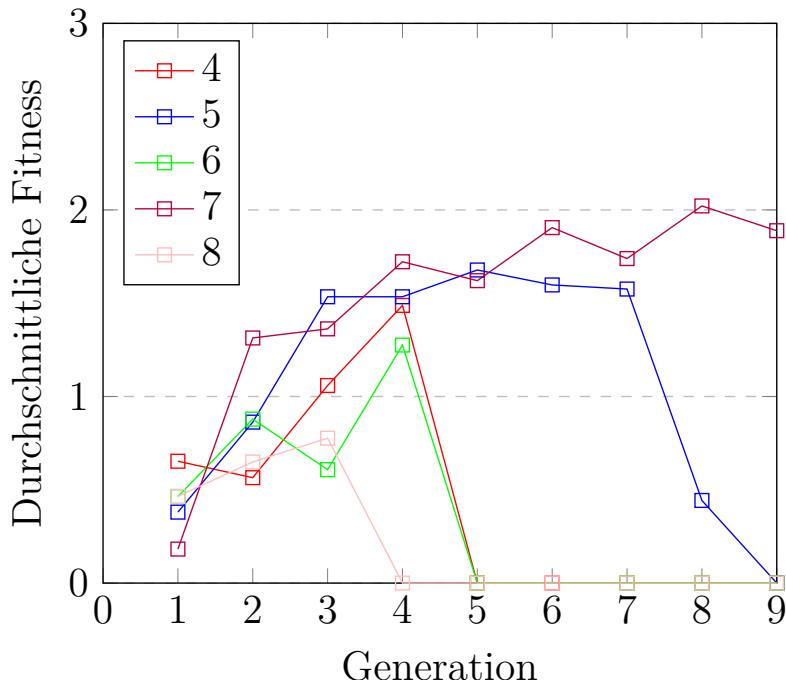


Abbildung 4.4: Durchschnittliches Individuum pro Anzahl Körperpunkte

## 4.3 Allgemeine Lösung — dritter Simulationslauf

### 4.3.1 Konfiguration

Simulationsparameter	
Populationsgrösse	120
Selektionsstrategie	Turnierselektion
Iterationen	ca. 1200
Iterationsdauer	30s
Schwierigkeit erhöhen pro	100 Generationen
Steigungszuwachs	0.01
Maximale Steigung	1.0
Höchste Y-Koordinate Zuwachs	0.02
Ziel	Allgemeine Lösung
Wahrscheinlichkeit Mutation	
Allgemein	0.05
Engine	
add	0.005
remove	0.001
lens	0.01
id	0.001
parameter	0.5

Tabelle 4.3: Simulationsparameter

Die Implementation des Feedbacks musste wegen mangelnder Zeit eingestellt werden. In dieser Version ist die detaillierte Konfiguration der Mutation verfügbar. Die Konfigurationstabelle (Tabelle 4.3 auf Seite 36) stellt die erweiterten Einstellungsmöglichkeiten dar.

### 4.3.2 Auswertung

Leider dauert es mit dieser Version der Applikation sehr lange eine Generation zu evolvieren. Der Grund dafür ist einerseits die ineffiziente Mutation der Individuen. Andererseits ist die Wahrscheinlichkeit eine neue Bewegung hinzuzufügen grösser, als eine Bewegung zu entfernen. Die Datenstrukturen wachsen, da mehr Bewegungen hinzukommen als entfernt werden, was wiederum den Rechenaufwand erhöht. Darum wird dieser Versuch nach ca. 1200 Generationen abgebrochen, damit Verbesserungen an der Geschwindigkeit der Mutation implementiert werden können und die Wahrscheinlichkeit anders konfiguriert werden kann.

Die Individuen haben bei diesem Versuch eine Art rollende Bewegung entwickelt. Sie können die guten Fitnesswerte halten, obwohl der Parcours schwerer wird (Abb. 4.5).

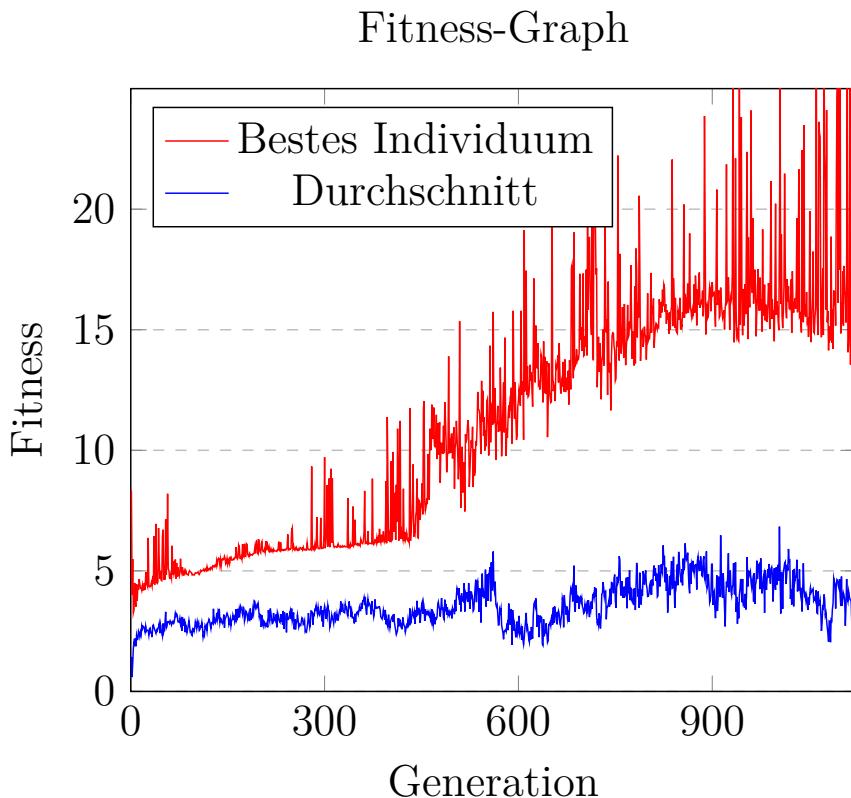


Abbildung 4.5: Bestes Individuum und durchschnittliches Individuum

#### 4.3.2.1 Diversität

In dieser Version der Simulation ist der Diversitätsreport verfügbar.

Wie in Abb. 4.6 auf der nächsten Seite zu sehen ist, bewegt sich die Diversität kurz auf einem hohen Niveau, da am Anfang alles zufällig generiert wird. Die Turnierselektion bevorzugt Individuen mit einer grossen Fitness, deshalb sinkt die Diversität nach der Selektion. Darum ist kurz nach Beginn der Simulation ein starker Abfall der Diversität zu beobachten.

#### 4 Resultate

Interessant zu beobachten ist, dass die Diversität ab der 400. Generation wieder zu steigen beginnt. Die Vermutung dabei ist, dass sich mehr verschiedene Individuen mit einer guten Fitness finden lassen.

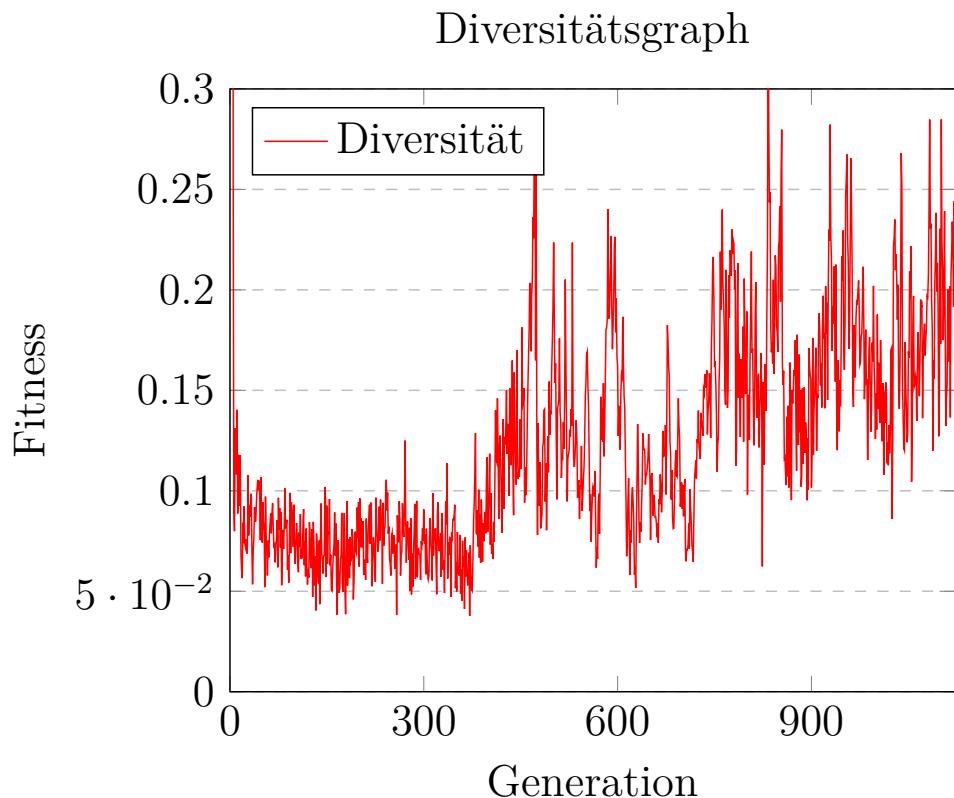


Abbildung 4.6: Diversität pro Generation

## 4.4 Allgemeine Lösung — vierter Simulationslauf

### 4.4.1 Konfiguration

Simulationsparameter	
Populationsgrösse	120
Selektionsstrategie	Turnierselektion
Iterationen	ca. 12000
Iterationsdauer	30s
Schwierigkeit erhöhen pro	100 Generationen
Steigungszuwachs	0.01
Maximale Steigung	1.0
Höchste Y-Koordinate Zuwachs	0.02
Ziel	Allgemeine Lösung
Wahrscheinlichkeit Mutation	
Allgemein	0.05
Engine	
add	0.005
remove	0.005
lens	0.01
id	0.001
parameter	0.05

Tabelle 4.4: Simulationsparameter

Wie beabsichtigt in Abschnitt 4.3.2 auf Seite 37, wurde die Wahrscheinlichkeit Bewegungen hinzuzufügen oder zu entfernen gleichgesetzt. Dies sollte verhindern, dass die Datenstrukturen durch die hinzukommenden Bewegungen zu gross werden und den benötigten Rechenaufwand in die Höhe treiben.

#### 4.4.2 Auswertung

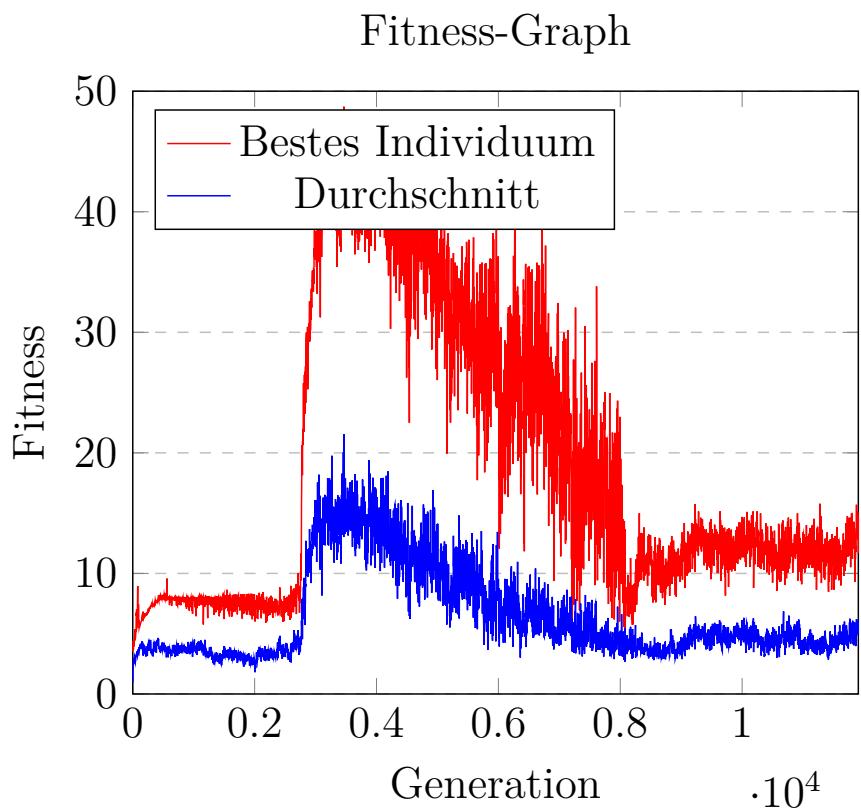


Abbildung 4.7: Bestes Individuum und durchschnittliches Individuum

##### 4.4.2.1 200. Generation

Wie in Abb. 4.8 erkennbar, haben sich die Individuen zu einer Art Rolle entwickelt. Diese Art von Fortbewegung ist nicht optimal, da sich die Individuen genau einmal um sich selber drehen und dann zum Stillstand kommen. Dies setzt sich fort bis etwa zur 3000. Generation.

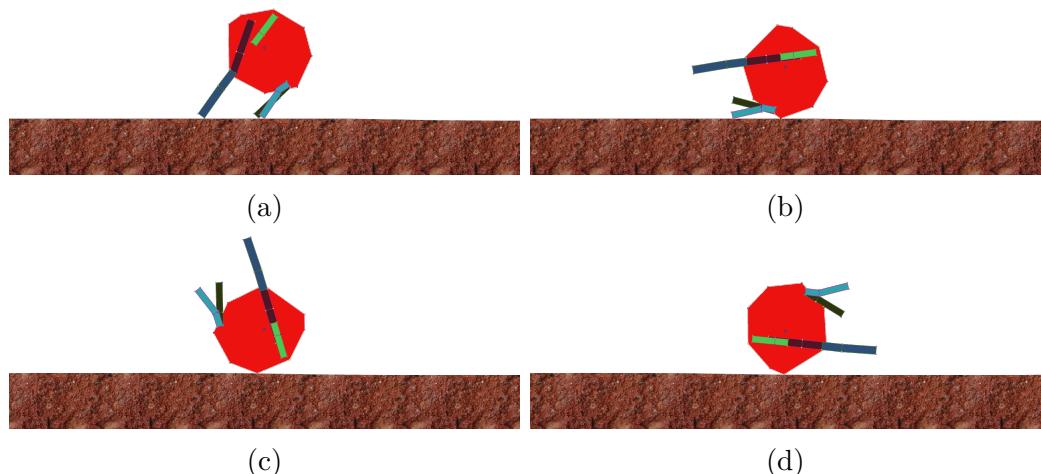


Abbildung 4.8: 200. Generation

#### 4.4.2.2 3000. Generation

Ab der 3000. Generation haben die Individuen eine Hüpf-Bewegung entwickelt. Sie fixieren alle Beine ausser eines, welches sie benutzen um sich vom Boden abzustossen (Abb. 4.9). Diese Art von Fortbewegung bringt eine bessere Fitness hervor, wie in Abb. 4.7 auf Seite 40 erkennbar ist. Im Gegensatz zu den Individuen in Abschnitt 4.2.2 auf Seite 35 ist dies jedoch eine valide Hüpfbewegung. Da die Individuen sich vom Boden korrekt mit einem Bein abstossen, sich nicht im Boden verhaken und nicht durch die zu späte und fehlerhafte Kollisionserkennung weggestossen werden.

Das Herabsetzen der Mutationswahrscheinlichkeiten (Hypothese in Abschnitt 3.6.6.1 auf Seite 29) hat langfristig einen positiven Effekt auf die Fitness, wie in Abb. 4.7 auf Seite 40 im Vergleich zu Abb. 4.1 auf Seite 33 erkennbar ist.

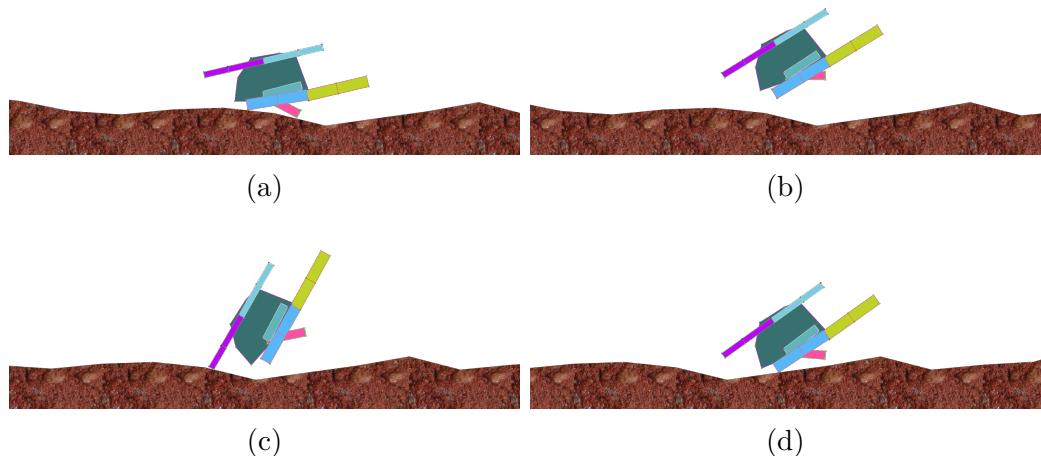


Abbildung 4.9: 3000. Generation

#### 4.4.2.3 7000. Generation

Mit zunehmender Schwierigkeit des Parcours scheint das Hüpfen nicht mehr die richtige Art von Fortbewegung zu sein. Die Individuen bleiben häufig an Berghängen und in Tälern stecken (Abb. 4.10 auf der nächsten Seite).

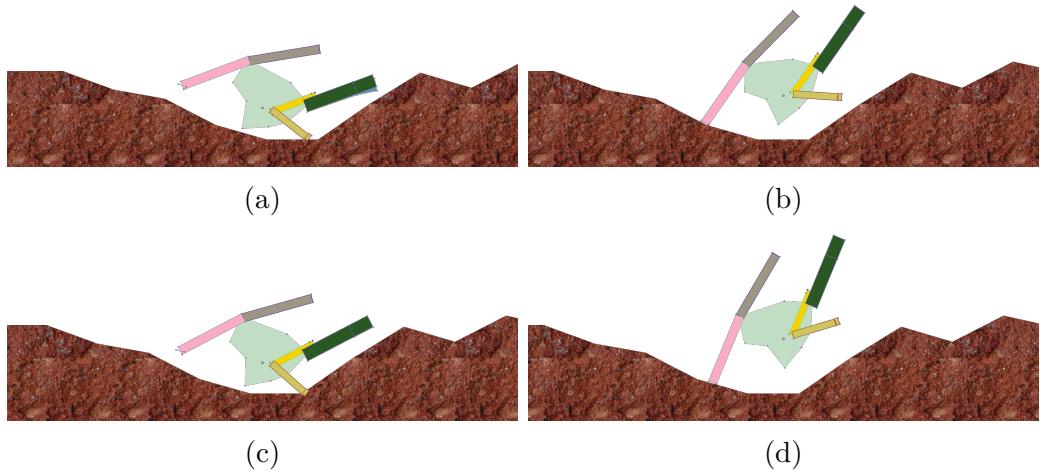


Abbildung 4.10: 7000. Generation

#### 4.4.2.4 11000. Generation

Die Individuen haben sich zu Rudern entwickelt wie in Abb. 4.11 zu sehen ist. Die Masse der anderen Beinpaare, welche nicht für die Ruderbewegung gebraucht werden, ist sichtbar kleiner geworden.

Bei der Analyse dieser Generation ist ein Fehler aufgefallen. Die Individuen werden nach vier Sekunden entfernt, wenn sie sich währenddessen nicht um einen bestimmten Wert vorwärtsbewegen. Dieser Wert sollte allerdings grösser gewählt werden, da es in späteren Generationen und zunehmender Parcours-Schwierigkeit sein kann, dass die Individuen sich lange an einer Stelle aufhalten, um einen Berg zu erklimmen.

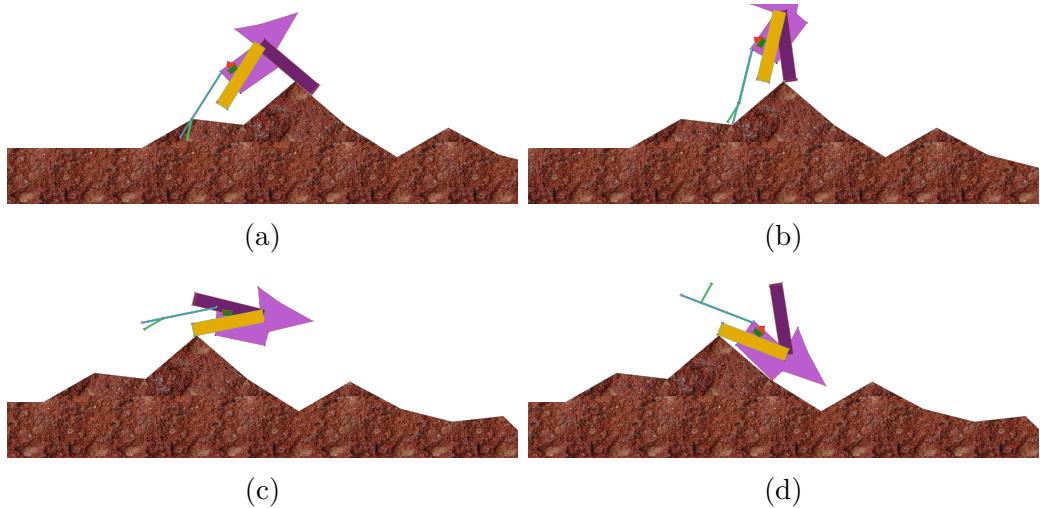


Abbildung 4.11: 11000. Generation

#### 4.4.2.5 Diversität

Der Einbruch ab ca. der 3000. Generation, lässt sich damit erklären, dass die Hüpfer gefunden worden sind. Die Diversität konnte sich steigern im Vergleich zum Anfang der Simulation und spricht für die Turnierselektion.

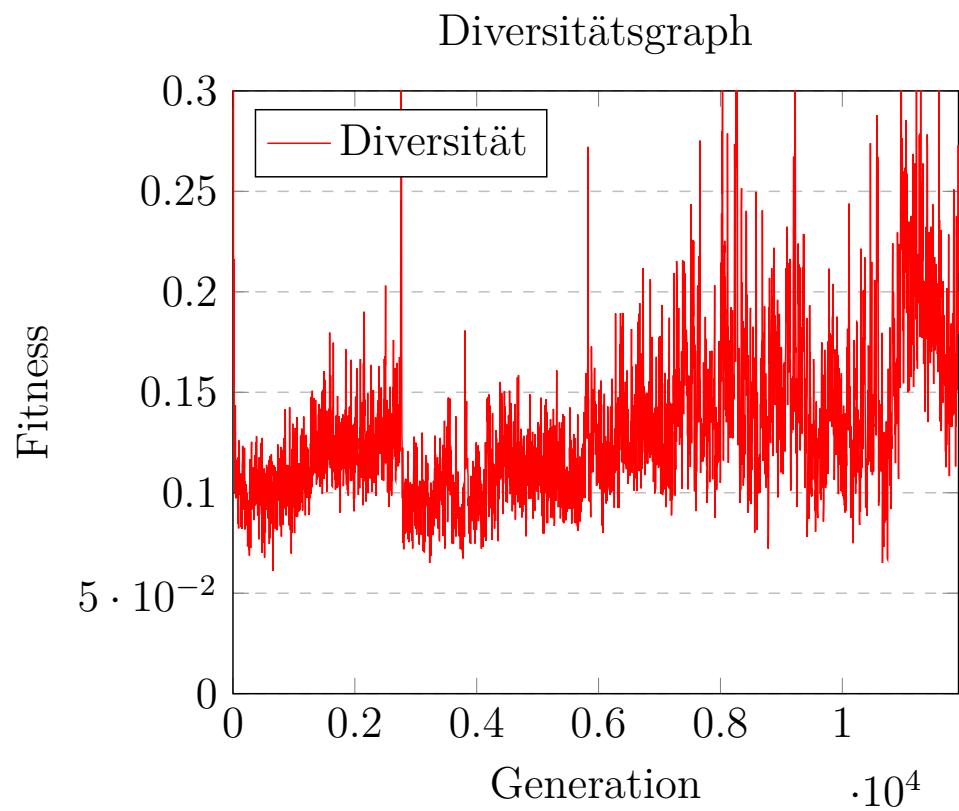


Abbildung 4.12: Diversität pro Generation

## 4.5 Evolvieren auf Evolvierbarkeit — fünfter Simulationslauf

### 4.5.1 Konfiguration

Simulationsparameter	
Populationsgrösse	120
Selektionsstrategie	Turnierselektion
Iterationen	ca. 12000
Iterationsdauer	30s
Schwierigkeit erhöhen pro	100 Generationen
Steigungszuwachs	0.01
Maximale Steigung	1.0
Höchste Y-Koordinate Zuwachs	0.02
Ziel	Ervolvieren auf Evolvierbarkeit
Wahrscheinlichkeit Mutation	
Allgemein	0.05
Engine	
add	0.005
remove	0.005
lens	0.01
id	0.001
parameter	0.05

Tabelle 4.5: Simulationsparameter

Das Ziel dieser Simulation ist das Evolvieren auf Evolvierbarkeit, um einen Vergleich zur allgemeinen Lösung zu ziehen. Darum wurde das Ziel der Konfiguration modifiziert, um das Evolvieren auf Evolvierbarkeit zu erreichen.

### 4.5.2 Auswertung

Der Graph in Abb. 4.13 auf der nächsten Seite schlägt weniger aus im Vergleich zu Abb. 4.7 auf Seite 40. Dies ist damit zu erklären, dass beim Evolvieren auf Evolvierbarkeit, der Parcours erst nach 100 Generationen ausgewechselt wird und somit die Fitnesswerte weniger schwanken pro Generation. Wenn der Parcours gewechselt wird, treten Ausreisser (lokale Minima und Maxima) auf. Die gefundenen Individuen bewegen sich mit Hüpfen fort.

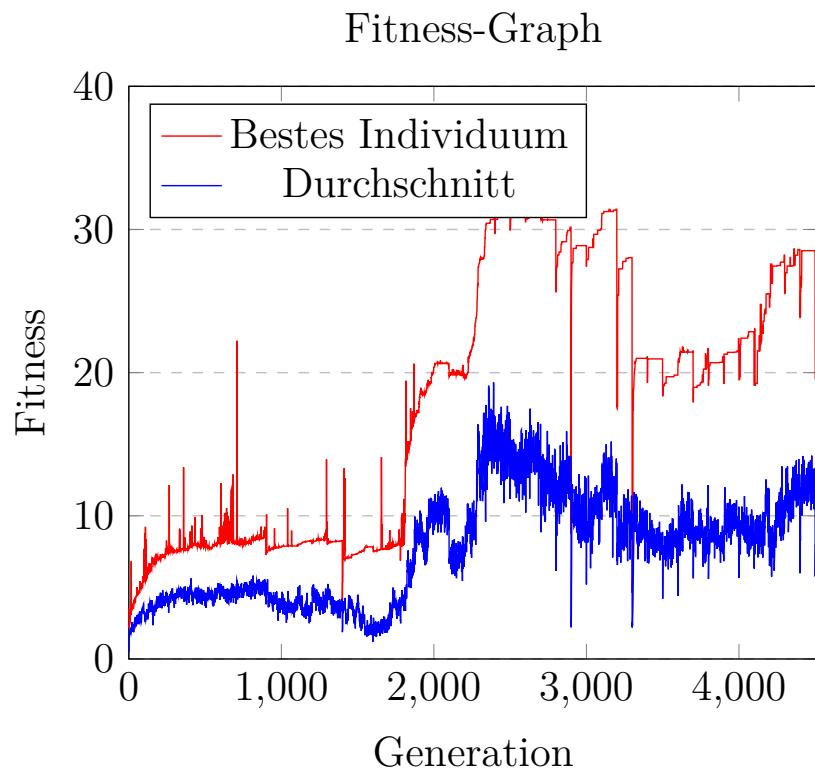


Abbildung 4.13: Bestes Individuum und durchschnittliches Individuum

#### 4.5.2.1 Diversität

Die Diversität (Abb. 4.14) ist im Vergleich zum vierten Simulationslauf (Abb. 4.12 auf Seite 43) deutlich niedriger. Die allgemeine Lösung erlaubt es diversere Individuen zu finden.

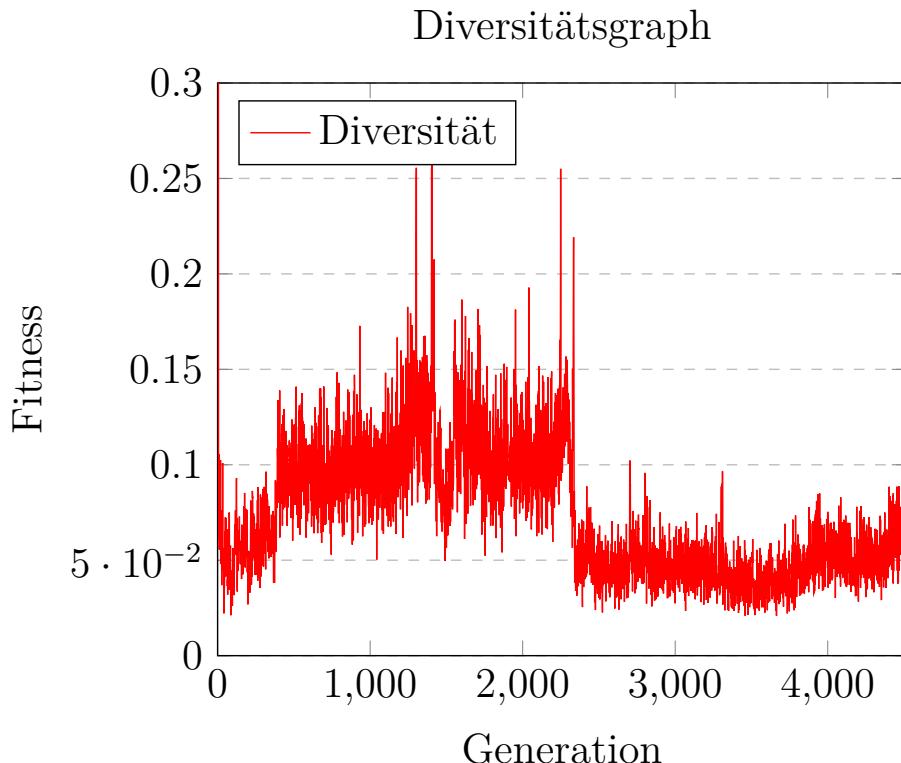


Abbildung 4.14: Diversität pro Generation

# 5 Diskussion und Ausblick

## 5.1 Diskussion der Resultate

Die gefundenen Resultate wiederspiegeln Lösungen innerhalb einer künstlichen Umgebung. In dieser Umgebung wird eine vereinfachte Physik verwendet, welche nur eine Annäherung an die reale Physik ist. Deshalb sind die Resultate nicht ohne Vorbehalte auf die reale Welt übertragbar.

Das Evolvieren von artifiziellen Tieren und ihrer Steuerung ist möglich mit dem implementierten evolutionären Algorithmus. Es entwickelte sich keine klassische Laufbewegung, sondern eine Bewegung welche es den Individuen erlaubt, sich durch den Parcours fortzubewegen.

Es stellt sich heraus, dass eine Simulation viel Zeit beansprucht und äusserst rechenintensiv ist. 12000 Generationen zu simulieren, dauert rund 10 Tage.

Es kann die Hypothese aufgestellt werden, dass die JavaScript-Engine V8 momentan noch nicht ausreichend schnell ist. Eine weiterführende Arbeit könnte untersuchen, wie die Leistungen anderer Implementationen von Javascript-Engines im Vergleich zu V8 abschneiden würde.

### 5.1.1 Wie kann eine Steuerung der Bewegung implementiert werden?

Wie in Abschnitt 3.2 auf Seite 18 beschrieben ist die Steuerung der Bewegung in zwei Teile aufgeteilt. Es existiert nur ein Bewegungsablauf pro Individuum. Damit wird erreicht, dass keine Synchronisation zwischen mehreren Motoren stattfinden muss. Was wiederum die Komplexität eines Motors reduziert.

Die Aufteilung bietet mehrere Vorteile. Die Implementation des Motors und der Mutation des Bewegungsablaufs kann getrennt werden. Die Mutation verarbeitet die Repräsentation des Bewegungsablaufs. Innerhalb der Mutation muss deshalb einzig die Struktur der Daten bekannt sein. Vorteilhaft für die Implementation des Motors ist, dass dieser nur den vorgegebenen Ablauf abarbeiten muss.

Ein Nachteil der Implementation des Motors ist, dass das Feedback-System nicht trivial implementiert werden kann. Der Motor muss anhand des Feedbacks entscheiden, was weiter geschehen soll. Für jede Kombination von Zustand des Bewegungsablaufs und Art des Feedbacks muss definiert werden, was der nächste Zustand des Motors sein muss. Zustände werden während der Evolution erstellt und gelöscht. Somit müssen diese Regeln dynamisch erstellt werden können.

Aus diesen Gründen ist das Feedback-System nicht in den Motor integriert. Weiterführende Arbeiten könnten diesen Punkt aufnehmen und untersuchen Abschnitt 5.2.1 auf Seite 52.

### 5.1.2 Wie kann diese Steuerung evolviert werden?

Damit die Steuerung der Bewegung, der Bewegungsablauf, evolviert werden kann, liegt diese in einer Form vor, die es erlaubt einzelne Werte zu manipulieren. Der Bewegungsablauf beinhaltet eine Liste von möglichen Bewegungen. In der Liste können Bewegungen beliebig gruppiert werden. Wie in Abschnitt 3.2.2.1 auf Seite 20 beschrieben, wird die Information zu einer Bewegung in mehrere Teile aufgespalten. Die gewählte Darstellungsform bietet eine fein gegliederte Beschreibung des Bewegungsablaufs. Diese Gliederung erlaubt es den Bewegungsablauf einfach zu entwickeln.

Wie erwartet ist der Bewegungsablauf hauptverantwortlich für die Fitness eines Individuums. Dies führt dazu, dass Individuen mit einem ausgereiften Bewegungsablauf mit gröserer Wahrscheinlichkeit für die Reproduktion selektiert und anschliessen mutiert werden.

Die erzielten Resultate mit der Implementation des Motors sind zufriedenstellend und liegen innerhalb der Erwartungen. In der gegebenen Form werden Bewegungsmuster entwickelt, die an den Parcours angepasst werden. Es gilt zu beachten, dass diese Resultate unter dem von Floreano [3] beschriebenen Problem der evolutionären Robotik leiden.

Die entwickelten Individuen nutzen an bestimmten Stellen Eigenschaften des Parcours aus. Eigenschaften sind hier besondere Geländeformationen innerhalb eines Parcours. Damit lässt sich herleiten, weshalb die Simulation mit einer höheren Wahrscheinlichkeit für das Hinzufügen von Bewegungen als das Löschen bessere Resultate liefert. Mit den zusätzlichen Bewegungen kann das Individuum seinen Bewegungsablauf an weitere Geländeformationen adaptieren und erreicht somit eine höhere Fitness.

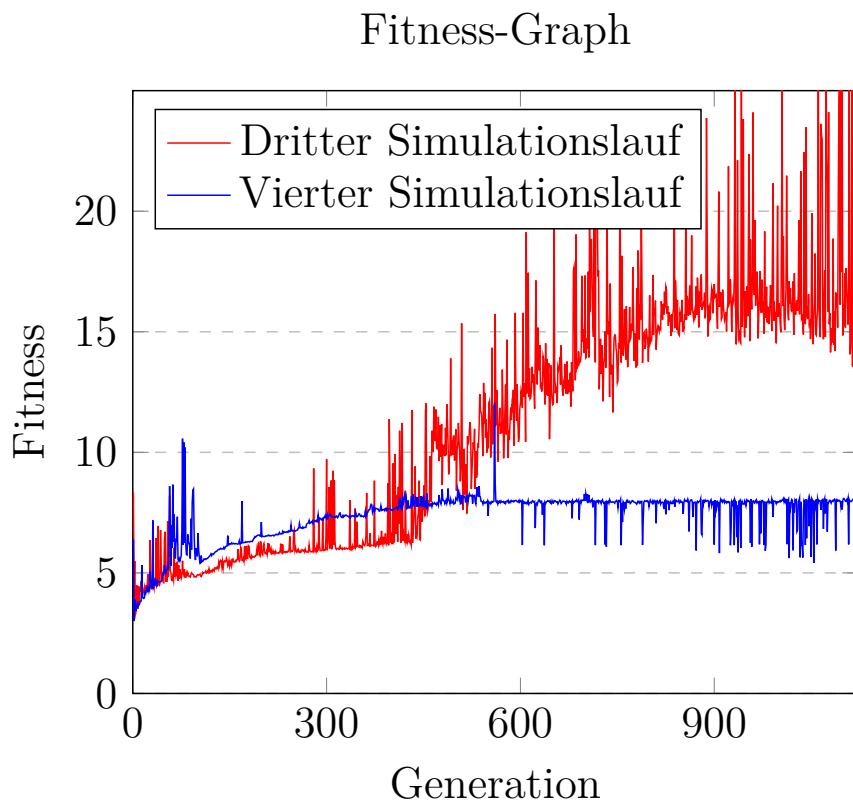


Abbildung 5.1: Fittestes Individuum, Vergleich dritter und vierter Lauf

Der Vergleich zwischen den Simulationsläufen zeigt, dass im dritten Lauf die Individuen fitter waren. Es kann jedoch kein allgemeiner Schluss gezogen werden, da der Vergleich nur 1200 Generationen beinhaltet. Die Grösse der Datenstrukturen im dritten Simulationslauf hat die Simulation stark verlangsamt, weshalb die Simulation abgebrochen werden musste.

Damit dies validiert werden kann, muss ein geeignetes Verhältnis zwischen Hinzufügen und Löschen von Bewegungen gefunden werden, so dass die Simulation länger laufen kann. Dieses Verhältnis ist jedoch abhängig von äusseren Faktoren und wird durch die verfügbaren Ressourcen begrenzt. Das Verhältnis kann durch mehrere Versuche bestimmt werden.

### 5.1.3 Wie kann die Geometrie der Tiere evolviert werden?

Die Konstruktion der geometrischen Objekte, aus denen sich ein Individuum zusammensetzt, wird unter Abschnitt 3.1.2 auf Seite 14 und Abschnitt 3.1.3.3 auf Seite 16 besprochen. Welche Geometrie die richtige ist, wird durch die Selektion und Mutation der Individuen bestimmt. Falls die Körperpunkte mit Hilfe eines viereckigen Bereiches konstruiert worden sind, wäre die Konstruktion eines Körpers schwieriger ausgefallen (Abschnitt 3.1.3.3 auf Seite 16). Der Kreis scheint nach wie vor, die einfachste Art der Körperpunktekonstruktion zu sein.

Der Ansatz der Konstruktion kann erweitert werden, so dass innerhalb des Kreises gewisse Bereiche eingeschränkt werden. Damit wird erreicht, dass der Körper eine gewisse Form annimmt. Ebenfalls kann so eine minimale oder maximale Grösse vorgegeben werden.

Es gilt zu prüfen ob die Einschränkung, dass ein Körper eines Individuums aus 4–8 Punkten besteht, geändert werden kann. Damit ein Körper geformt werden kann, müssen mindesten drei Punkte vorhanden sein. Werden mehr Punkte zugelassen, so können komplexere Formen gebildet werden. Ausserdem muss die Mutation der Anzahl Körperpunkte implementiert werden (Abschnitt 4.2.2.1 auf Seite 35). Sonst werden bereits nach kurzer Zeit nur Individuen mit derselben Anzahl Körperpunkten existieren.  $N$ -eckige Individuen haben das Potenzial besser abzuschneiden, als die Bestehenden.

### 5.1.4 Nimmt die Diversität mit zunehmenden Generationen stetig ab?

Es wird erwartet, dass die anfänglich hohe Diversität stetig sinken würde, bis sie einen Tiefpunkt zum Ende der Simulation erreicht hätte.

Analysiert man die Diversitätsgraphen aus dem vierten (Abschnitt 4.4 auf Seite 39) und fünften Simulationslauf (Abschnitt 4.5 auf Seite 44) ist ersichtlich, dass ein deutlicher Abfall der Diversität in den ersten Generationen aufgetreten ist. Bei der allgemeinen Lösung erholt sich die Diversität mit zunehmenden Generationen. Gewisse Ausreisser erreichen sogar die Diversität der ersten Generationen. Jedoch zeichnet sich ein anderes Bild bei dem Evolvieren auf Evolvierbarkeit ab. Hier fällt die Fitness ab der 2000. Generation auf ein niedriges Niveau.

Durch den sich ständigen verändernden Parcours, müssen sich die Individuen der allgemeinen Lösung mehr anpassen, als ihre Gegenspieler. Deshalb liefert die allgemeine Lösung vielfältigere Individuen.

Diese Aussagen über die Diversität sind nur gültig für den Einsatz von Turnierselektion. Um eine Aussage über andere Selektionsstrategien zu machen, müssten andere Strategien implementiert werden (Abschnitt 5.2.4 auf Seite 53).

### 5.1.5 Wie sieht der Bewegungsablauf und die Geometrie eines evolvierten Tieres aus?

Es wird erwartet, dass die Individuen eine Bewegung entwickeln, welche noch an den Gang einer Ameise erinnert. Stattdessen haben sich drei unterschiedliche Bewegungen entwickelt:

- Rollbewegung
- Hüpfbewegung
- Ruderbewegung

Bei der Rollbewegung kippen die Individuen vornüber und drehen sich einmal um sich selber. Die Geometrie des Körpers tendiert zu einer kreisartigen Form. Beine spielen bei dieser Art von Bewegung eine untergeordnete Rolle.

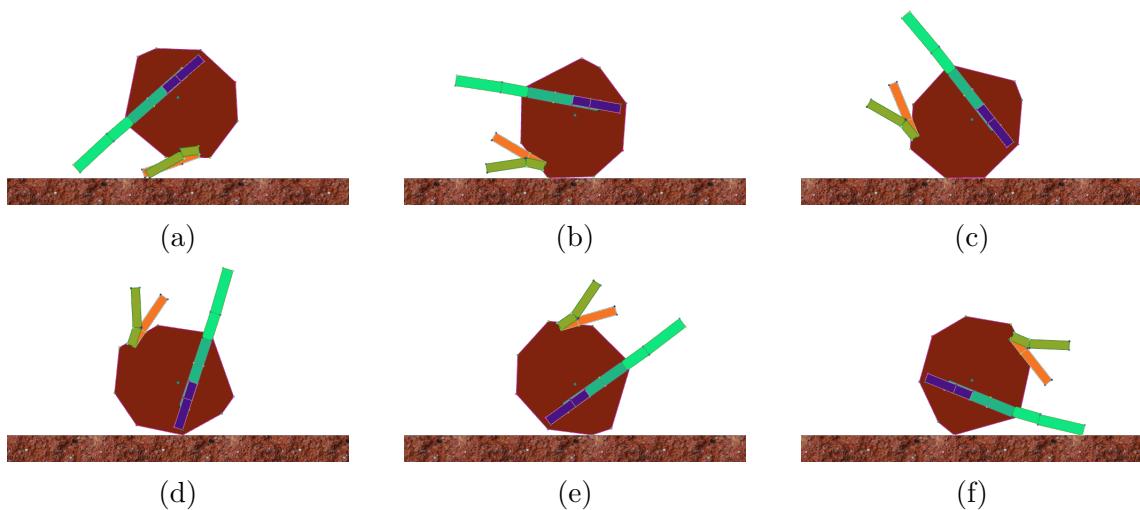


Abbildung 5.2: Rollbewegung im Zeitraffer

Die Hüpfbewegung in Abb. 5.3 auf der nächsten Seite führen die Individuen aus, in dem sie sich mit einem Bein vom Boden abstossen. Es lässt sich keine Tendenz, zu einer bestimmten Körperform erkennen.

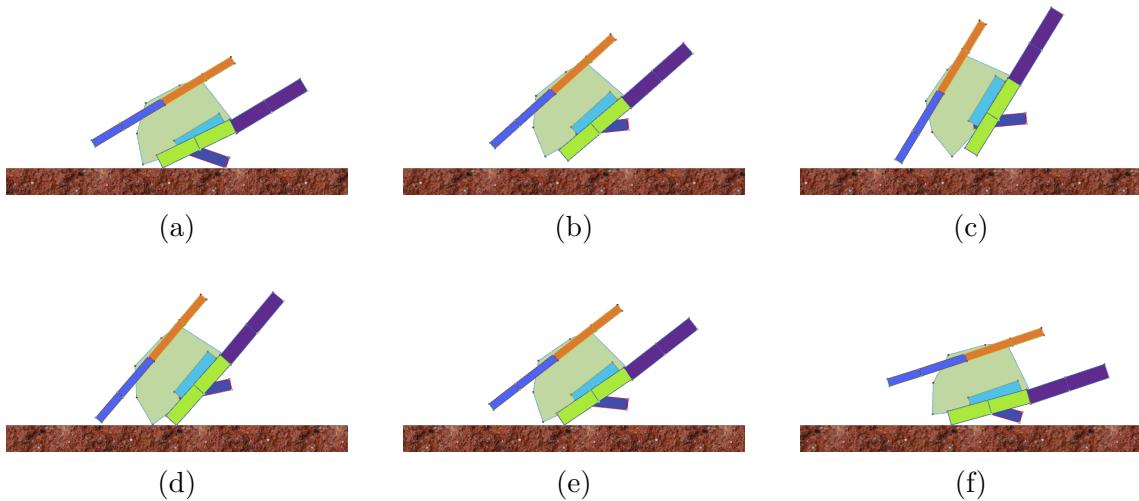


Abbildung 5.3: Hüpfbewegung im Zeitraffer

Die Ruderer in Abb. 5.4 benutzen ein Bein um die Bewegung durchzuführen, alle anderen Beine verharren starr am Körper. Die starren Beine haben sie von ihren Vorfahren (den Hüpffern) geerbt und noch nicht korrigiert. Diese Gegebenheit kann man als “frozen accident” bezeichnen. Alle Beine für die Ruderbewegung zu nutzen, würde zu besseren Fitnesswerten führen. Die Massen aller Beinpaare, ausser dass sich bewegende, haben sich verkleinert. Der Trend bei den Ruderern liegt bei flachen Körpern.

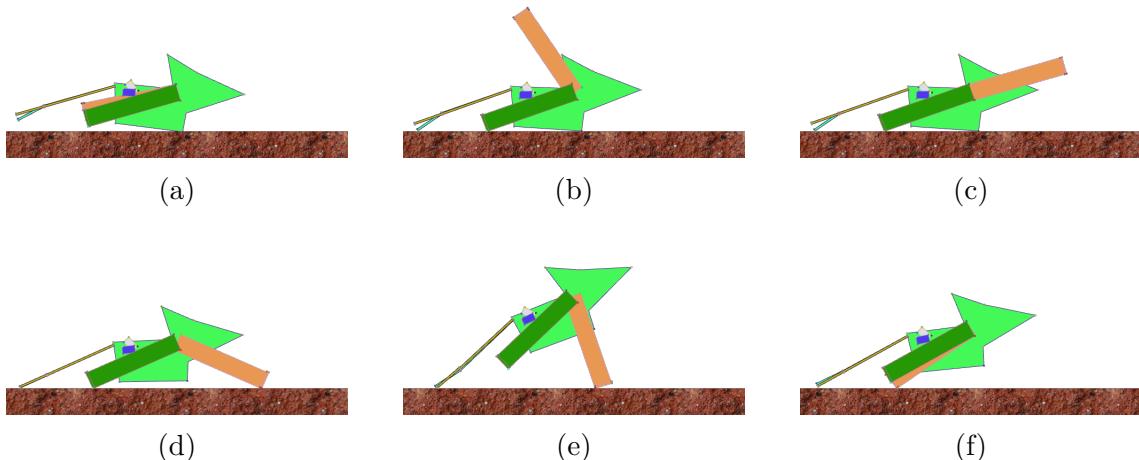


Abbildung 5.4: Ruderbewegung im Zeitraffer

Die gefundenen Resultate zeigen, dass nicht alle Beine benützen werden müssen, um sich durch den Parcours fortbewegen zu können. Eine Anpassung der Wahrscheinlichkeit für das Hinzufügen von Bewegungen (Abschnitt 5.1.2 auf Seite 47), könnte die Ruderer veranlassen, alle Beine zu bewegen. Weiterführende Simulationen mit angepassten Parameter könnten fähig sein, alle Beine auszunutzen.

### 5.1.6 Individuum: Allgemeine Lösung vs. evolvieren auf Evolvierbarkeit

Es wird das beste Individuum der 4000. Generation (Abb. 5.5 auf der nächsten Seite) aus dem vierten Simulationslauf (Abschnitt 4.4 auf Seite 39) mit dem fünften

Simulationslauf (Abschnitt 4.5 auf Seite 44) miteinander verglichen.

Die Individuen haben beide Hüpfbewegungen entwickelt. Jedoch streckt das eine Individuum (Abb. 5.5a) alle Beine, während das Andere (Abb. 5.5b) fast alle Beine angewinkelt hat. Das Strecken der Beine bringt eine bessere Balance, dies ist der Grund, warum das Individuum aus der allgemeinen Lösung einen besseren Fitnesswert aufweist. Ebenso weist der Körper des Individuum, welches auf Evolvierbarkeit evolviert wurde, eine grössere Masse auf. Die Masse aller Beine ist folglich grösser beim Individuum der allgemeinen Lösung. Während bei der Geometrie leichte Unterschiede festgestellt worden sind, ist die Steuerung beider Individuen fast identisch.

Die Resultate sprechen dafür, dass mit der allgemeinen Lösung fittere Individuen gefunden werden können. Um diese Aussage zu unterstreichen, sollten noch mehr Simulationen mit dem Evolvieren auf Evolvierbarkeit durchgeführt werden.

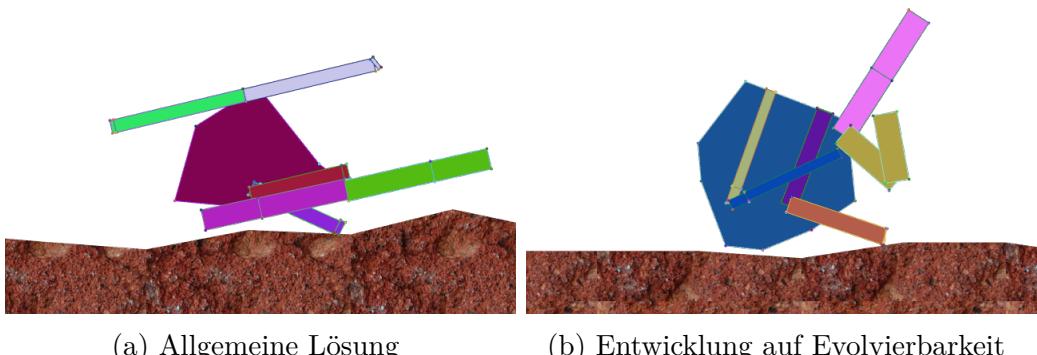


Abbildung 5.5: 4000. Generation

### 5.1.7 Hypothese Körperpunkte

Um eine Aussage über die Hypothese der Körperpunkte (Abschnitt 3.1.1.1 auf Seite 14) zu treffen, müsste wie erwähnt unter Abschnitt 4.2.2.1 auf Seite 35, die Mutation der Anzahl Körperpunkte implementiert werden. Spätestens ab der 10. Generation bei jedem Simulationslauf, existieren nur noch Individuen mit der gleichen Anzahl Körperpunkten. Die Hypothese kann erst validiert werden, wenn die Anzahl der Körperpunkte mutiert wird.

### 5.1.8 Hypothese Mutationswahrscheinlichkeiten

Wie in Abschnitt 4.4.2.2 auf Seite 41 festgestellt worden ist, hat sich die Hypothese über die Mutationswahrscheinlichkeiten (Abschnitt 3.6.6.1 auf Seite 29) bewahrheitet. Kleinere Mutationswahrscheinlichkeiten tragen dazu bei, fittere Individuen zu finden. Hohe Mutationswahrscheinlichkeiten bedeuten, dass viele Bereiche eines Individuum in einem Schritt variiert werden können. Dies führt dazu, dass vorteilhafte, sinnvolle Mutationen durch unvorteilhafte überschattet werden. Damit werden wertvolle Lösungen verspielt.

Insbesondere für die Entwicklung des Bewegungsablaufs ist dies relevant. Mit kleineren Wahrscheinlichkeiten werden feinere Änderungen vorgenommen, die in der Simulation ausgewertet werden. Fallen die Änderungen zu gross aus, ist es möglich, dass im Ablauf eine unvorteilhafte Mutation zu Beginn des Bewegungsablaufs eine vorteilhafte Änderung zunichtemacht.

Die Wahrscheinlichkeit der Mutation steuert, ob der evolutionäre Algorithmus in eine Zufallssuche abgleitet [7]. Daraus folgt, dass die Wahrscheinlichkeit der entscheidende Faktor ist für einen positiven oder negativen Effekt der Mutation.

## 5.2 Ausblick

Die erstellte Simulationsapplikation ist gut erweiterbar und ermöglicht Interessierten sie weiter auszubauen. Der Programmcode soll für Interessierte unter <http://github.com> freigegeben werden.

### 5.2.1 Feedback an den Bewegungsmotor

Eine wichtige Komponente im Bewegungsablauf stellt das Feedback dar. Mit einem Feedback-System können Ereignisse (Boden berühren) oder Beschaffenheiten des Parcours (Steigung und Gefälle) erkannt und verarbeitet werden.

Eine Möglichkeit ist es den Bewegungsablauf bzw. die Bewegung zu erweitern. So dass nicht nur die Bewegung weiterentwickelt wird, sondern auch wie die Rückmeldung an den Motor aussieht. Die Rückmeldung kann als zusätzliches Attribut einer Bewegung definiert werden. Das hat zur Folge, dass der Motor weiterhin als *EA* implementiert werden kann. Eine Rückmeldung ist dann eine weitere Möglichkeit für einen Zustandswechsel. Wobei der nächste Zustand bzw. Bewegung nicht die im Bewegungsablauf nachfolgende sein muss.

Als weitere Option kann der Motor selbst ergänzt werden. Dabei steigt die Komplexität der Implementation des Motors, da dieser ein Regelwerk (oder ähnlich) verwenden muss, um die Zustandswechsel zu organisieren.

In der Simulation ist bereits vorhanden, dass erkannt wird, wann ein Bein den Boden berührt. Diese Rückmeldung wird jedoch nicht verarbeitet. Es wird vermutet, dass die Erweiterung um ein solches System die Fitnesswerte im Verlauf der Generationen steigert. Interessant ist auch, ob das Feedback-System zu komplexeren Bewegungsabläufen führt.

### 5.2.2 *N*-beinige Tiere

Zu *n*-beinigen Tieren können folgende Forschungsfragen formuliert werden:

- Warum existieren nur symmetrische Anordnungen der Beinpaare in der Natur?
- Warum haben Spinnen acht Beine?
- Warum haben Insekten sechs Beine?

Momentan beschränkt der definierte Bewegungsablauf die Individuen auf sechs Beine. Wenn auf jedem Bein ein Bewegungsablauf definiert wird, wäre es möglich artifizielle Tiere mit *n*-Beinen zu evolvieren. Eine ungerade Anzahl und asymmetrische Anordnung der Beine wäre so realisierbar. Es muss aber mit erhöhtem Rechenaufwand gerechnet werden, da die Bewegungsabläufe miteinander synchronisiert werden müssen und mehr Beine bewegt werden. Mit Hilfe der Implementation von den genannten Vorschlägen, wäre es möglich die Forschungsfragen zu *n*-beinigen Tieren zu beantworten.

### 5.2.3 Austauschen der Physik-Engine

Ein grosses Potential birgt der Wechsel der *Physik-Engine*. Die verwendete *Physik-Engine* "p2.js" zeigt während der Simulation einige Schwächen.

Besonders die Kollisionserkennung ist unvollständig implementiert. Das führt zu den beschriebenen Fehlern (Abschnitt 3.4.2 auf Seite 22).

Ein weiterer Kritikpunkt ist die Geschwindigkeit. Damit bessere Resultate erzielt werden können, müssen mehr Individuen in einer Population simuliert werden. Zudem soll die Simulation schneller berechnet werden können.

Von einem Wechsel der *Physik-Engine* wird vor allem erhofft, dass grössere Populationen berechnet werden können und schneller simuliert werden kann.

### 5.2.4 Hypothese Selektionsstrategie

Die zeitliche Beschränkung dieser Arbeit hat es nicht erlaubt, die ursprünglich geplante Hypothese über den Vergleich von Selektionsstrategien durchzuführen. In der Hypothese geht es darum die Auswirkungen von den Selektionsstrategien Turnierselektion, Rangselektion und fitnessproportionale Selektion auf die Simulationsresultate zu untersuchen.

### 5.2.5 Hypothese Allgemeine Lösung vs. evolvieren auf Evolvierbarkeit

Es war nicht genügend Zeit vorhanden folgende Hypothese zu validieren: Individuen welche durch die allgemeine Lösung gefunden werden, sollten bei unterschiedlichen Parcours im Durchschnitt besser abschneiden, als ihre Gegenspieler (Entwicklung auf Evolvierbarkeit), da sie sich schneller an eine neue Umgebung anpassen können. Wenn immer der gleiche Parcours benutzt wird, sollten die Individuen welche auf Evolvierbarkeit evolviert worden sind, auf Dauer überlegen sein.

Um diese Hypothese zu validieren, sollte für beide Lösungsansätze je zwei Simulationsläufe mit 1000 Iterationen durchgeführt werden:

- Simulationslauf mit gleichem Parcours
- Simulationslauf mit 10 unterschiedlichen Parcours

Folgende Kriterien müssten erfüllt sein:

- Es sollen zwei bestehende Populationen verwendet werden.
- Die Parcours sollten identisch für beide Lösungsansätze sein

Abschliessend sollte ein Vergleich angestellt werden zwischen den Lösungsansätzen und ihrer Resultate. Die dafür notwendige Implementierung der Parcours-Wiederverwendung ist noch offen.

### 5.2.6 Parcours

Ein Schritt in Richtung Annäherung an die Realität ist die Integration verschiedener Terrain-Typen in den Parcours. Weitere Typen wie Eis, Gras oder Sand können hinzugefügt werden. Jeder dieser Terrain-Typen verlangt andere Anpassungen des Individuums. Eisige Untergründe bieten wenig Halt und Reibung. Sandige Terrains sind ein weicher Untergrund und benötigen deshalb andere Eigenschaften.

Eine interessante Fragestellung ist, wie die Terrain-Typen das Aussehen und den Bewegungsablauf beeinflussen. Ein Parcours der einen Mix aus verschiedenen Terrain-Typen beinhaltet ist ebenfalls interessant zu untersuchen. Darin werden alle Anforderung kombiniert und das Individuum muss diese abdecken.

Zusammen mit einem Feedback-System für die Bewegung und verschiedenen Terrain-Typen kann untersucht werden, ob komplexere Bewegungsabläufe entstehen und wie diese sich in unterschiedlichen Umgebungen verhalten.

# Literaturverzeichnis

- [1] R. Dawkins, „Universal Darwinism,” in *Evolution from molecules to man*, D. S. Bendall, Hrsg. Cambridge: Cambridge University Press, 1983, Kap. 20, S. 403–425.
- [2] G. Hornby, A. Globus, D. Linden, und J. Lohn, Serie SPACE Conferences and Exposition. American Institute of Aeronautics and Astronautics, Sep 2006, Kap. Automated Antenna Design with Evolutionary Algorithms.
- [3] D. Floreano und L. Keller, „Evolution of Adaptive Behaviour in Robots by Means of Darwinian Selection,” *PLoS Biol*, Vol. 8, Nr. 1, S. 1–8, Jan 2010.
- [4] D. Floreano und C. Mattiussi, *Bio-inspired artificial intelligence: theories, methods, and technologies*. Cambridge, Massachusetts: The MIT Press, 2008.
- [5] K. Weicker, *Evolutinäre Algorithmen*, 3. Aufl. Leipzig, Deutschland: Springer Vieweg, 2015.
- [6] R. Füchslin, „Bachelorarbeit 2016,” [https://tat.zhaw.ch/tpada/arbeit\\_vorschau.jsp?arbeitID=14414](https://tat.zhaw.ch/tpada/arbeit_vorschau.jsp?arbeitID=14414), 2016, [Online; Stand: 03.06.2016].
- [7] J. R. Sampson, „Adaptation in Natural and Artificial Systems (John H. Holland),” *SIAM Review*, Vol. 18, Nr. 3, S. 529–530, 1976.
- [8] A. E. Eiben und J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Deutschland: Springer-Verlag, 2003.
- [9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3. Aufl. Berlin, Deutschland: Springer-Verlag, 1996.
- [10] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.
- [11] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
- [12] L. J. Fogel, A. J. Owens, und M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: Wiley, 1966.
- [13] I. Rechenberg, *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Friedrich Frommann Verlag, 1973.
- [14] F. H. C. Crick, „The origin of the genetic code,” *Journal of Molecular Biology*, Vol. 38, Nr. 3, S. 367 – 379, 1968.
- [15] B. Physics, „Stepping The World,” [http://bulletphysics.org/mediawiki-1.5.8/index.php/Stepping\\_The\\_World](http://bulletphysics.org/mediawiki-1.5.8/index.php/Stepping_The_World), 2013, [Online; Stand: 03.06.2016].

## 5 Literaturverzeichnis

- [16] R. Gaul, „Modelling and Solving Physical Constraints,” <http://gamedevelopment.tutsplus.com/tutorials/modelling-and-solving-physical-constraints--gamedev-12578>, 2013, [Online; Stand: 03.06.2016].
- [17] Wikipedia, „JavaScript Object Notation,” [https://de.wikipedia.org/wiki/JavaScript\\_Object\\_Notation](https://de.wikipedia.org/wiki/JavaScript_Object_Notation), 2016, [Online; Stand: 03.06.2016].
- [18] T. H. Morgan, „Illustration of Crossing Over,” [https://en.wikipedia.org/wiki/Genetic\\_recombination](https://en.wikipedia.org/wiki/Genetic_recombination), 1916, [Online; Stand: 01.06.2016].

# Glossar

## .NET

.NET ist eine von Microsoft entwickelte Software-Plattform, die zur Entwicklung und Ausführung von Anwendungen verwendet wird. 22

## asynchrones Messaging

Beim asynchronen Messaging tauschen Prozesse Daten über Nachrichten aus, welche sie sich gegenseitig schicken. Asynchron bedeutet das Sender und Empfänger nicht auf Nachrichten warten, sondern anderen Rechenaufgaben nachgehen können. Erst wenn eine Nachricht empfangen wird, fokussiert sich der Prozess auf das Abarbeiten der Nachricht. 25

## Constraint

Ein Constraint hat die Funktion 2 Objekte in einer *Physik-Engine* einzuschränken [16]. 17

## EA

Endlicher Automat *siehe* Endlicher Automat, 18, 52,

## einfaches Polygon

Ein einfaches Polygon ist in der Geometrie eine flache Form bestehend aus geraden, sich nicht überschneidenden Linien, die einen geschlossenen Pfad formen. 14

## Endlicher Automat

Ein endlicher Automat (Zustandsmaschine, Zustandsautomat) ist ein Modell eines Verhaltens, bestehend aus Zuständen, Zustandsübergängen und Aktionen. 18, 59

## EP

Evolutionäre Programmierung 21,

## ES

Evolutionäre Strategien 21,

## F#

F# ist eine funktionale Sprache, die mit einer sauberen Syntax und einem ausgereiften Typensystem besticht. 22

## GA

Genetische Algorithmen 21,

## GP

Genetische Programmierung 21,

## Hexapod

Die Sechsfüßer (griech. Hexapoda) oder Hexapoden gehören dem Stamm der Gliederfüßer (Arthropoda) an, sie bilden einen Unterstamm von diesen. 14

## joint-driven

Ein Motor der “joint-driven” arbeitet steuert Bewegungen über ein Drehgelenk. 18

## JSON

JSON bedeutet JavaScript Object Notation. Es ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustausches zwischen Anwendungen [17]. 28

## NPM

Node Package Manager 23,

## Physik-Engine

Eine Physik-Engine ist eine eigenständige Funktionseinheit des Programms, welche zur Simulation von physikalischen Prozessen dient. 3, 16, 17, 18, 20, 21, 22, 24, 27, 35, 52, 53, 57

# Abbildungsverzeichnis

1.1	Skizze Parcours . . . . .	3
2.1	Genotyp und Phänotyp . . . . .	5
2.2	D. Floreano und C. Mattiussi, <i>Bio-inspired artificial intelligence: theories, methods, and technologies.</i> Cambridge, Massachusetts: The MIT Press, 2008, S.18 . . . . .	6
2.3	Beispiel Reale-Werte-Repräsentation . . . . .	6
2.4	Darstellung einer Rechnung als Baum . . . . .	7
2.5	D. Floreano und C. Mattiussi, <i>Bio-inspired artificial intelligence: theories, methods, and technologies.</i> Cambridge, Massachusetts: The MIT Press, 2008, S.24 . . . . .	8
2.6	Schritte einer gekürzten Rangselektion . . . . .	8
2.7	Turnierselektion . . . . .	9
2.8	T. H. Morgan, „Illustration of Crossing Over,” <a href="https://en.wikipedia.org/wiki/Genetic_recombination">https://en.wikipedia.org/wiki/Genetic_recombination</a> , 1916, [Online; Stand: 01.06.2016]	10
2.9	D. Floreano und C. Mattiussi, <i>Bio-inspired artificial intelligence: theories, methods, and technologies.</i> Cambridge, Massachusetts: The MIT Press, 2008, S.27 . . . . .	11
2.10	Invertieren der Bits . . . . .	11
2.11	Mutation von Reale-Werte-Repräsentationen . . . . .	12
2.12	Mutation von Reale-Werte-Repräsentationen . . . . .	12
2.13	D. Floreano und C. Mattiussi, <i>Bio-inspired artificial intelligence: theories, methods, and technologies.</i> Cambridge, Massachusetts: The MIT Press, 2008, S.29 . . . . .	12
3.1	Konzept Körper . . . . .	14
3.2	Konzept Bein . . . . .	15
3.3	Berechnung der Körperpunkte veranschaulicht . . . . .	17
3.4	<i>Endlicher Automat</i> veranschaulicht . . . . .	18
3.5	Bewegungsablauf im Zeitraffer . . . . .	20
3.6	Vor Korrektur . . . . .	23
3.7	Nach Korrektur . . . . .	23
3.8	Haupt- und Simulationsprozess . . . . .	25
3.9	Ablauf der Simulation . . . . .	26
3.10	flacher Start-Parcours 1. Generation . . . . .	27
3.11	Parcours 100. Generation . . . . .	27
3.12	Parcours 1000. Generation . . . . .	27
3.13	Domänenmodell Hauptprozess . . . . .	30
3.14	Domänenmodell Simulationsprozess . . . . .	31
4.1	Bestes Individuum und durchschnittliches Individuum . . . . .	33
4.2	Durchschnittliches Individuum pro Anzahl Körperpunkte . . . . .	34
4.3	Bestes Individuum und Durchschnittliches Individuum . . . . .	35
4.4	Durchschnittliches Individuum pro Anzahl Körperpunkte . . . . .	36

## *Abbildungsverzeichnis*

4.5	Bestes Individuum und durchschnittliches Individuum . . . . .	37
4.6	Diversität pro Generation . . . . .	38
4.7	Bestes Individuum und durchschnittliches Individuum . . . . .	40
4.8	200. Generation . . . . .	40
4.9	3000. Generation . . . . .	41
4.10	7000. Generation . . . . .	42
4.11	11000. Generation . . . . .	42
4.12	Diversität pro Generation . . . . .	43
4.13	Bestes Individuum und durchschnittliches Individuum . . . . .	45
4.14	Diversität pro Generation . . . . .	45
5.1	Fittestes Individuum, Vergleich dritter und vierter Lauf . . . . .	47
5.2	Rollbewegung im Zeitraffer . . . . .	49
5.3	Hüpfbewegung im Zeitraffer . . . . .	50
5.4	Ruderbewegung im Zeitraffer . . . . .	50
5.5	4000. Generation . . . . .	51

# Tabellenverzeichnis

3.1 Konfigurationstabelle Simulation . . . . .	24
4.1 Simulationsparameter . . . . .	32
4.2 Simulationsparameter . . . . .	34
4.3 Simulationsparameter . . . . .	36
4.4 Simulationsparameter . . . . .	39
4.5 Simulationsparameter . . . . .	44

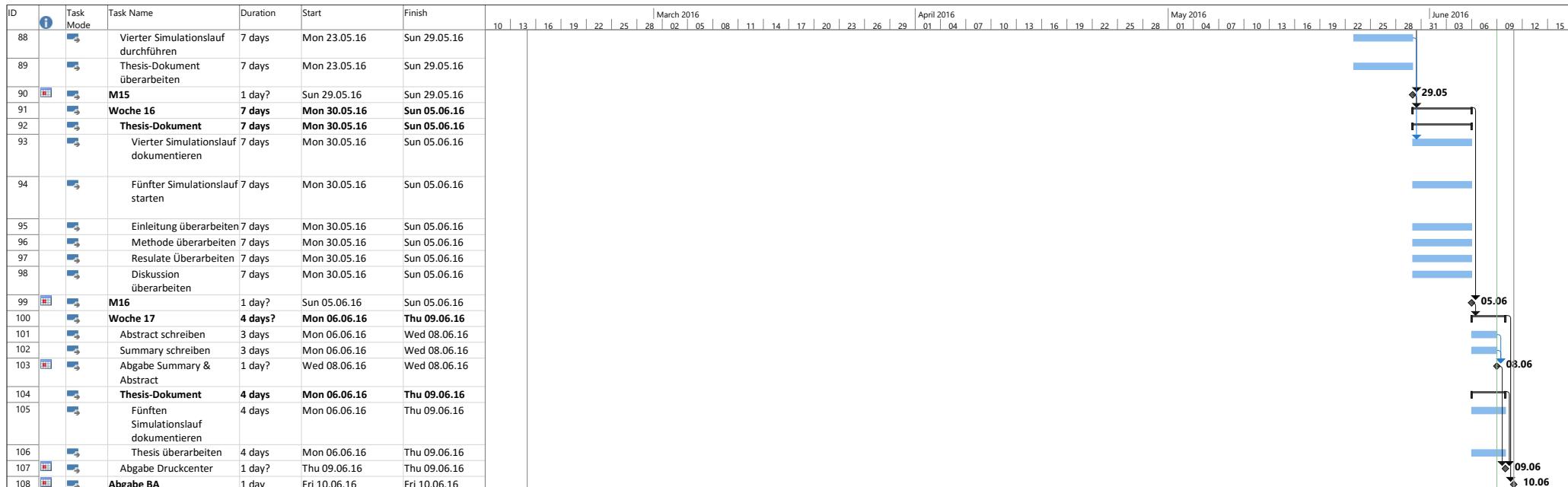
# **Anhang**

## **1 Projektmanagement**

Die verschiedenen Arbeitspakete werden eine Woche im Voraus geplant. Dieses Vorgehen wird auch als rollende Planung bezeichnet. Ein Projektvorgehensmodell wird nicht benutzt, jedoch ist eine Projektplanung mit Hilfe von MS Project 2016 erstellt worden. Alle wichtigen Meilensteine finden sich im Projektplan. Der Stand der Arbeit wurde an einem wöchentlichen Treffen mit den Betreuern besprochen. Das Beschlussprotokoll wurde vor dem nächsten Treffen verschickt.







Project: project\_schedule.mpp  
Date: Thu 09.06.16



## 1.1 Protokolle

# Beschlussprotokoll

---

## Besprechung Bachelor-Arbeit

---

Donnerstag, 25.2.16, 15:00 Uhr, TP 215, Winterthur

Anwesende:

- D. Flumini
- R. Füchslin
- F. Hediger
- O. Stern
- F. Tanner

Vorsitz: R. Füchslin, O. Stern

Protokoll: F. Tanner

## Beschlüsse

1. Wöchentliche Sitzung festgelegt auf Donnerstag jeweils um 15:00 Uhr. Während den Sitzungen soll ein Beschlussprotokoll geführt werden.
2. Für die Arbeit muss ein Projektplan ausgearbeitet werden.
3. Persönliche Arbeitsmaterialien dürfen im Labor im Gebäude TP deponiert werden. Der Zutritt ist mit einem gültigen Batch jederzeit erlaubt.
4. Evolutionstechnische Fragen in jedem Fall mit R. Füchslin besprechen.
5. Der Quellcode der Software wird marginal bewertet.
6. Biologische Grundlagen aus der Theorie auslassen, da nicht relevant für Modell.
7. Im Konzept soll ersichtlich sein, weshalb Entscheide für oder gegen Algorithmen, Technologien, etc. gefällt wurden.
8. Die Fragestellung sowie die Bedingungen für Erfolg und die Erwartungen müssen klar definiert werden. Mit der Programmierung erst beginnen, wenn das Konzept erstellt ist.
9. An Herrn R. Füchslin darf die Arbeit vor der Abgabe abgegeben werden für eine Rückmeldung.

Schluss der Sitzung: 16:05 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 3.3.16, 15:00 Uhr, TP 215, Winterthur

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Nächstes Treffen wird via Skype abends um 8 Uhr geführt.
2. Nachtrag zum Beschluss der letzten Sitzung: "Der Quellcode der Software wird marginal bewertet.": Der Quellcode wird analysiert und bewertet, wobei die Analyse aber auf einer groben Detailstufe bleibt.
3. Bewertungsraster wird zusammen mit Herr Stern ausgefüllt.

Schluss der Sitzung: 16:05 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 17.3.16, 15:00 Uhr, TP 215, Winterthur

Anwesende:

- R. Füchslin
- F. Hediger
- O. Stern
- F. Tanner

Vorsitz: R. Füchslin, O. Stern

Protokoll: F. Tanner

## **Beschlüsse**

1. Fokus auf Bewegung, Motor legen. Überlegen, ob Sensorisches Feedback-System benutzt werden kann.
2. Bewertungsraster-Vorlage wird von Herrn Stern an Herrn Füchslin gesendet. Die Vorlage wird von Herrn Füchslin ausgefüllt und in der nächsten Sitzung besprochen.
3. Mit Herrn Füchslin kann eine Probepäsentation mit Feedback durchgeführt werden. Herr Stern wird nicht anwesend sein.
4. In 2 Wochen, am 31.3.16, soll eine erste Version der Bewegung präsentiert werden.

Schluss der Sitzung: 16:00 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 24.3.16, 15:00 Uhr, TP 215, Winterthur

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Konzentration auf die absolut notwendigen Dinge. Beilagen (wie z.B. Füsse/abgerundete Stümpfe) können bei Gelegenheit nachimplementiert werden.
2. Es soll sichtbar sein, dass die Physik-Engine kontrollierbar ist.
3. Ein weiterer "stepping stone" wird das Feedback zu analysieren und in die Bewegung einfließen zu lassen.

Schluss der Sitzung: 16:00 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 31.3.16, 15:00 Uhr, TP 215, Winterthur

Anwesende:

- D. Flumini
- R. Füchslin
- F. Hediger
- O. Stern
- F. Tanner

Vorsitz: R. Füchslin, O. Stern

Protokoll: F. Tanner

## **Beschlüsse**

1. Nun den Fokus aus die Parametrisierbarkeit und das Feedback-System der Bewegung legen.
2. Nicht zu viele Parameter wählen. Philamente werden ausgedünnt, d.h. gute Regionen sind schwieriger zu erreichen. Sobald kein evolutiver Fortschritt stattfindet sind ev. zu viele Parameter im Spiel.
3. Was sind die Parameter (eingrenzen) die evolvieren werden sollen.
4. Was ist machbar in Bezug auf physikalische Bedingungen der Körperform?
5. Ziel ist zeigen, dass etwas evolviert wird.
6. In Arbeit ausführen, wie die Bewegung modelliert und realisiert wird.

Schluss der Sitzung: 15:55 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 7.4.16, 14:50 Uhr, via Skype

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Erste Ziellinie des praktischen Teils wurde erreicht. Weiterhin grosses Augenmerk auf Parametrisierbarkeit der Bewegung legen.
2. An der Dokumentation wird weitergearbeitet. Besonders zu beachten ist, dass die Resultate der Simulation interpretiert werden müssen.

Schluss der Sitzung: 15:55 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 14.4.16, 15:00 Uhr, TP 215, Winterthur

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Gedanken machen, ob und welches Ziel verfolgt wird:
  - Parcours nach n Schritten ändern: Evolvierbarkeit evolvieren
  - Parcours für jede Generation ändern: Allgemeine Lösung (Sackmesser)
  - Parcours belassen: Spezialisierung
2. INIT anfragen, ob Server zur Verfügung steht. Technische Optimierung Render-nodes anschliessend.

Schluss der Sitzung: 16:00 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 21.4.16, 15:00 Uhr, via Skype

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Fehler (Kollision Polygon mit Höhenfeld) in der Engine gefunden. Dokumentieren und eventuell umgehen.
2. Betreffend Rechenzentrum nochmals nachfragen.
3. Mutation der Bewegung soll als bald als möglich fertiggestellt werden, damit anschliessend simuliert werden kann.
4. Leistung wird nicht von der Engine direkt berechnet. Beschränkung ändern, falls der Autor der Physik-Engine keine Lösung bereitstellt. Deshalb soll die maximale Winkelgeschwindigkeit eines Gelenkes beschränkt werden.

Schluss der Sitzung: 15:20 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 28.4.16, 15:15 Uhr, via Skype

Anwesende:

- D. Flumini
- R. Füchslin
- F. Hediger
- O. Stern
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Server ist aufgesetzt. Simulation bricht nach 4-8 Stunden ab. Momentan auf Fehlersuche.
2. Erste Implementation der Parametrisierung der Bewegung fertiggestellt.
3. Relation zwischen Winkelgeschwindigkeit und Masse herstellen. Masse mal Quadrat der Winkelgeschwindigkeit ( $m * w^2$ ).
4. Beinlänge, -dicke beschränken. Masse auf gesamtes Tier verteilen. Parameterraum beschränken.
5. Masse proportional zur Fläche. Masse = 1. Falls zu kompliziert Beschränkungen auf Länge und Breite setzen für Körperteile.

Schluss der Sitzung: 15:20 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Freitag, 6.5.16, 11:30 Uhr, via Skype

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Diversitätsreport besprochen. Beschlossen, dass ein einheitlicher Vektor der alle Teile des Genotyps beinhaltet verwendet wird.

Schluss der Sitzung: 12:10 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 12.5.16, 15:05 Uhr, TP 215, Winterthur

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Diversitätsreport besprochen. Beschluss, dass ein einheitlicher Vektor der alle Teile des Genotyps beinhaltet verwendet wird.
2. Diversität einzelner Teile des Genoms bei genügen Zeit separat auswerten.

Schluss der Sitzung: 16:00 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 19.5.16, 15:05 Uhr, TP 215, Winterthur

Anwesende:

- R. Füchslin
- F. Hediger
- O. Stern
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Entwurf ebenfalls an Herr Stern schicken.
2. Performance-Problem mit Mutation lösen.
3. Ein hartes "closing date" definierten und entsprechende Richtlinien beachten.
4. Nächstes Treffen via Telefon/Skype.

Schluss der Sitzung: 16:00 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 23.5.16, 10:50 Uhr, via Skype

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Optimierung der Mutation vollzogen.
2. Fokus aufs Schreiben legen.

Schluss der Sitzung: 11:05 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 30.5.16, 14:35 Uhr, via Skype

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Gesamt Struktur muss überlegt werden. Argumentation und Fragestellungen klar definieren.
2. Schluss welche Fragestellung wurde wie gut beantwortet.
3. Gut-zum-Druck von Herrn Füchslin (siehe Anleitung zu Abgabe). Termin mit Herrn Füchslin am Mittwoch 8.6. Anruf auf Mobiltelefon.

Schluss der Sitzung: 15:00 Uhr

# **Beschlussprotokoll**

---

## **Besprechung Bachelor-Arbeit**

---

Donnerstag, 3.6.16, 15:00 Uhr, via Skype

Anwesende:

- R. Füchslin
- F. Hediger
- F. Tanner

Vorsitz: R. Füchslin

Protokoll: F. Tanner

## **Beschlüsse**

1. Quellen zu Abbildungen aus Literaturverzeichnis ins Abbildungsverzeichnis

Schluss der Sitzung: 15:20 Uhr