Georg-August-Universität Göttingen
Institut für Informatik
Proseminar: Artifical Life
Leitung: Prof. Dr. Winfried Kurth, Dipl.-Inf. Reinhard Hemmerling
Sommersemester 2011

Sticky Feet: Evolution in einer physikalischen Simulation vieler Kreaturen

Alwin Maier
Wilhelm-Weber-Straße 1a
37073 Göttingen
Studiengang: Angewandte Informatik
E-Mail: alwin.maier@stud.uni-goettingen.de
Eingereicht am: 30. September 2011

Inhaltsverzeichnis

1 Einleitung

Viele komplexe Vorgänge in der Natur können wir bis heute noch nicht verstehen. Einer der Gründe dafür ist die Tatsache, dass diese Vorgänge seit vielen Milionen Jahren andauern.

Vor allem in der Evolutionsbiologie hat es sich als nützlich erwiesen biologische Prozesse in Computersimulationen zu beobachten, die in der Natur zu langsam ablaufen um sie mitzuerleben. Deswegen beschäftigt sich ein Teil der Artificial Life Forschung mit der Simulation biologischer Prozesse.

Diese Hausarbeit beruht auf der Ausarbeitung der Arbeit von Greg Turk: *Sticky Feet: Evolution in a Multi–Creatur Physical Simulation* von 2010[1], in der mehrere, sogenannte Kreaturen, simulatan in einer modellierten Welt leben, sich entwickeln und bekämpfen und sich bei Erfolg fortpflanzen. In dieser künstlich simulierten Evolution entwickelte sich nach kurzer Zeit eine beindruckende Vielfalt an Kreaturen mit unterschiedlichem Verhalten und Aussehen.

Der erste Teil dieser Arbeit beschäftigt sich mit dem physikalschen Modell, in dem die Simulationen abliefen. Hierbei wird auch kurz auf die mathematischen Grundlagen nach [3] eingegangen, die für die Implementierung der Berechnungen nötig sind. Nach diesem abstrakten Einstieg wird auf die Art der Interaktion zwischen den Kreaturen und deren Fortpflanzung eingegangen und am Schluss der Ablauf und die Ergebnisse der Simulationen geschildert.

[1] Vgl. [2]

2 Das pysikalische Modell

Die einzigen Objekte die simuliert werden sind Kreaturen. Diese bestehen aus einer Menge von Massepunkten[2], welche durch Segmente zu einem zusammenhängenden Objekt verbunden werden. Als Segmente wurden Sprungfedern gewählt, die in Schwingungen versetzt werden können. Durch die Oszillation eines Segments erfahren die daran gebundenen Massepunkte i und j die Kraft der gedämpften Schwingung

$$F_i^{spring} = -(k_s \cdot (L - L_{rest}) + k_d \cdot \dot{L}\frac{L}{|L|}) \cdot \frac{L}{|L|} \quad (2.1)$$

$$F_j^{spring} = -F_i^{spring} \quad (2.2)$$

wobei gilt:

$$L = P_i - P_j$$
$$\dot{L} = V = V_i - V_j$$

mit P_i bzw. P_j die Position und V_i bzw. V_j die Geschwindigkeit der Punktmasse i bzw. j und L_{rest} die Ruhelänge, d.h. die Länge des Segments im ausgeschwungenen Zustand. Global als Konstante festgelegt sind k_s die Federkonstante und k_d die Dämpfungskonstante [3, Teil C, Seite 7].

Aufgrund der Dämpfung kommt jede Schwingung irgendwann zur Ruhe, weswegen die Ruhelänge jedes Segments mit der Länge L_{seq}, der Amplitude a, der Frequenz f und der Phase p periodisch im Verlauf der Zeit t nach folgender Vorschrift geändert wird:

$$L_{rest} = L_{seq}(1 + asin(ft + 2\pi p)), \quad (2.3)$$

Durch diese Festlegung wird jedes Segment dauerhaft zur Schwingung angeregt und kommt somit nicht zur Ruhe. Man kann es als die Energie ansehen, welche eine Kreatur aufbringt um sich zu bewegen, nur dass jede Kreatur einen unerschöpflichen Energiespender hat.

Die einfachsten Kreaturen bestehen aus zwei Massepunkten und einem verbindenden Segment. Mit dem aktuellen Wissen über das Modell würde sich die Bewegung einer solchen Kreatur auf ein Hin- und Herschwingen der beiden Punktmassen beschränken.

[2]Ein Massepunkte oder auch Punktmasse ist ein idealisierter Körper, welcher eine Masse, aber keine räumliche Ausdehnung besitzt. Durch diese Eigenschaft ist er einfach zu simulieren. Dieser Körper kann eine Geschwindigkeit haben und reagiert wie jeder andere Körper auf Kräfte.

Der Schwerpunkt der Kreatur würde aber unverändert bleiben, selbst wenn beide Punkte unterschiedliche Massen hätten. In Abbildung 2 ist die Bewegung einer solchen Kreatur vereinfacht dargestellt.

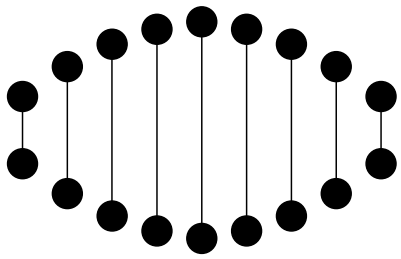

Abbildung 2.1: Die Bewegungsmöglichkeiten einer einfachen Kreatur ohne Einfluss äußerer Kräfte. Wie man sieht bewegen sich die Punktmassen an beiden Enden des Segments, die Kreatur als Ganzes hingegen bewegt sich nicht.

Damit sich aber nicht nur die Massepunkte einer Kreatur bewegen, sondern die Kreatur selbst, verändert jeder Massepunkt periodisch seinen Zustand von klebrig zu rutschig. Genauer gesagt wird für jeden Massepunkt i die Reibungskraft

$$F_i^{friction} = -k_f \cdot f_i \cdot V_i \tag{2.4}$$

berechnet. Dabei ist k_f der globale Reibungskoeffizient konstant und V_i, wie oben, der Geschwindigkeit der Punktmasse i. Zusätzlich wird für jede Punktmasse ein Wert f_i berechnet. Dieser berechnet sich wie folgt: Für jeden Massepunkt i wird f_i mit null initialisert. Anschließend wird für jedes Segment k wird ein spezifischer Reibungskoeffizient g_k berechnet:

$$g_k = \begin{cases} -m_k & \text{falls } \cos(ft + 2\pi p) \leq 0 \\ m_k & \text{falls } \cos(ft + 2\pi p) > 0 \end{cases} \tag{2.5}$$

f, p sind wie oben beschrieben von jedem Segment abhängige Eigenschaften. Verbindet nun das Segment k zwei Punktmassen i und j miteinander, so wird g_k zu f_i addiert und von f_j subtrahiert.

Nun ist jede Kreatur in der Lage sich als Ganzes durch die Simulation zu bewegen. Für eine einfache Kreatur aus zwei Massepunkten und einem Segment könnte die Bewegung wie in Abbildung 2 aussehen. Diese Technik gleicht der einer Raupe, die sich zur Fortbewegung zusammenzieht und wieder streckt. [2, Seite 497f.] *Bemerkung: Eine Kreatur aus nur zwei Massepunkten und einem Segment wird nie in der Lage sein, die*

Richtung der Fortbewegung zu ändern. Dafür ist eine komplexere Kreatur aus mindestens drei Massepunkten nötig, wie z.B. in Abbildung 3.1 dargestellt.

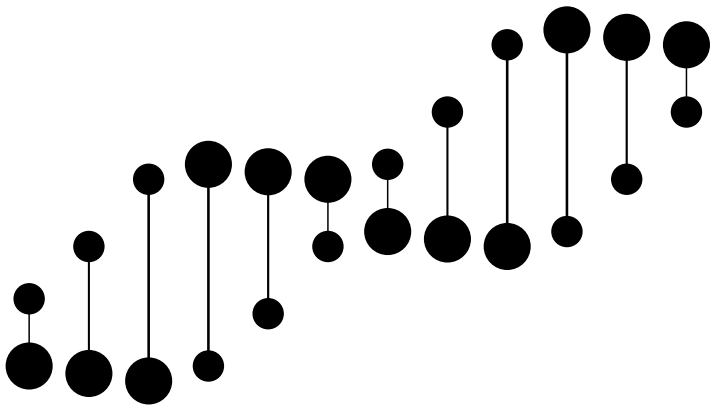

Abbildung 2.2: Eine Kreatur in Bewegung. Massepunkte mit einem großen Radius signalisieren eine hohe Reibung, wohingegen kleine Punkte rutschiger sind.

2.1 Eulersches Polygonzugverfahren

Um später die Position einer Kreatur zu jedem Zeitpunkt bestimmen zu können, muss man folgende Differentialgleichung (ODE) lösen können:

$$F(x(t)) = m \cdot \frac{\partial^2 x(t)}{\partial^2 t} \Leftrightarrow \frac{\partial^2 x(t)}{\partial^2 t} = \frac{F(x(t))}{m}, \tag{2.6}$$

oder einfacher:

$$F = ma \Leftrightarrow a = \frac{F}{m}. \tag{2.7}$$

Dabei handelt es sich um das zweite newtonsche Gesetz, welches die grundlegenden Bewegungen der Mechanik beschreibt und somit auch alle Bewegungen in der Simulation festlegt. Allerdings entspricht Gleichung 2.6 bzw. Gleichung 2.7 einer ODE zweiter Ordnung. Um eine Gleichung erster Ordnung zu erhalten führt man die Variable $v = \dot{x}$ ein und erhält damit zwei ODEs erster Ordnung:

$$\dot{v} = \frac{F}{m} \tag{2.8}$$

$$\dot{x} = v \tag{2.9}$$

Gibt man zusätzlich Anfangswerte für v ($v(t_0) = v_0$) und x ($x(t_0) = x_0$) zu einem beliebigen Startzeitpunkt t_0 vor, erhält man ein sogenanntes Anfangswertproblem (AWP),

dass in allgemeiner Form folgendermaßen geschrieben werden kann:

$$\dot{x}(t) = f(x(t), t) \tag{2.10}$$

$$x(t_0) = x_0 \tag{2.11}$$

Die einfachste Möglichkeit AWPs numerisch zu lösen, stellt das eulersche Polygonzugverfahren da. Um den Wert von x zum Zeitpunkt $t_k + h$ zu bestimmen, berechnet man folgenden Ausdruck:

$$x(t_k + h) = x(t_k) + h \cdot f(x(t_k), t_k), k = 1, 2, \cdots, \tag{2.12}$$

mit h einer zuvor festgelegten Schrittgröße. Offensichtlich ist, dass für kleine h der Fehler abnimmt, der Rechenaufwand hingegen zunimmt. [3, Seite B1ff., Seite C1]

2.2 Implementation

Mit Hilfe des eulerschen Polygonzugverfahren und dem Wissen wie man zu jedem Zeitpunkt die wirkenden Kräfte an einem beliebigen Massepunkt bestimmen kann, lässt sich der Aufenthaltsort jeder Kreatur zu jedem Zeitpunkt in der Simulation berechnen.

Da der Sourcecode der originalen Implementation unter [1] frei verfügbar ist, wird im Folgenden auf diesen eingegangen:

```
1  class creature {
2      ...
3      point[] points;      // point masses of creature
4      ...
5      segment[] segments; // segments that connect the points
6      ...
7
8      // calculate forces/accelerations for the point masses
9      void calculate_acceleration()
10     {
11         int i;
12
13         // zero out the accelerations and
14         // any artificial friction
15         for (i = 0; i < pnum; i++) {
16             points[i].ax = 0;
17             points[i].ay = 0;
18             points[i].friction = 0;
19         }
```

```
20
21          // calculate forces due to segments (springs)
22          for (i = 0; i < snum; i++) {
23             segment s = segments[i];
24             // spring constant for this segment
25             float k = s.k_spring * spring_k_global;
26             // spring damping value for segment
27             float damp = s.k_damp * spring_damp_global;
28
29             // calculate rest length of spring,
30             // which may vary sinusoidally with time
31             float rest_length = s.length;
32             float amp = s.amp +
33                   s.controllers[CONTROL_AMP].value;
34             float ph = s.phase + phase;
35             float sin_theta = sin(time * s.freq
36                   + ph * 2 * PI);
37             float cos_theta = cos(time * s.freq
38                   + ph * 2 * PI);
39             if (s.amp != 0) {
40                rest_length += amp * s.length * sin_theta;
41             }
42
43             point p1 = points[s.i1];
44             point p2 = points[s.i2];
45             float dx = p1.x - p2.x;
46             float dy = p1.y - p2.y;
47             float len = sqrt (dx*dx + dy*dy);
48
49             // equation of motion for a damped spring
50             float dvx = p1.vx - p2.vx;
51             float dvy = p1.vy - p2.vy;
52             float damping = damp
53                   * (dx * dvx + dy * dvy) / len;
54             float fx = -(k * (len - rest_length) + damping)
55                   * dx / len;
56             float fy = -(k * (len - rest_length) + damping)
57                   * dy / len;
58
59             // add appropriate forces to each point
60             p1.ax += fx;
61             p1.ay += fy;
62             p2.ax -= fx;
63             p2.ay -= fy;
64
65             // bias of spring to more one
```

```
66              // or the other mass more
67              float foot_amp = s.foot_amp
68                    + s.controllers[CONTROL_FOOT_AMP].value;
69              if (cos_theta > 0)
70                 foot_amp *= -1;
71
72              foot_amp = foot_amp
73                    + s.controllers[CONTROL_FOOT_SHIFT].value;
74
75              // incorporate bias by modifying the mass'
76              // friction coefficients
77
78              p1.friction -= foot_amp;
79              p2.friction += foot_amp;
80           }
81
82           // clamp any frictional foot_amp ("sticky feet")
83           // to proper range of [0,1]
84           for (i = 0; i < pnum; i++) {
85              if (points[i].friction < 0)
86                 points[i].friction = 0;
87              if (points[i].friction > 1)
88                 points[i].friction = 1;
89           }
90
91           // viscous damping
92           if (viscous_damping > 0)
93              for (i = 0; i < pnum; i++) {
94
95                 // regular viscous damping
96                 points[i].ax -= viscous_damping
97                          * points[i].vx;
98                 points[i].ay -= viscous_damping
99                          * points[i].vy;
100
101                // maybe extra artificial damping due to
102                // "sticky feet"
103
104                float sticky = points[i].friction;
105                float k = 10.0;
106
107                points[i].ax -= k * sticky * points[i].vx;
108                points[i].ay -= k * sticky * points[i].vy;
109             }
110
111          // convert forces to accelerations
```

```
112         for (i = 0; i < pnum; i++) {
113             points[i].ax /= points[i].mass;
114             points[i].ay /= points[i].mass;
115         }
116
117     }
118
119     ...
120
121     // take one timestep for the creature
122     void one_timestep()
123     {
124         int i;
125
126         // calculate accelerations due to
127         // forces on the point-masses
128         calculate_acceleration();
129
130         // perform modified Euler simulation step
131         for (i = 0; i < pnum; i++) {
132
133             point p = points[i];
134
135             // don't move points with fixed positions
136             if (p.fixed_position)
137                 continue;
138
139             p.xold = p.x;
140             p.yold = p.y;
141
142             float vx_old = p.vx;
143             float vy_old = p.vy;
144             p.vx = p.vx + dt * p.ax;
145             p.vy = p.vy + dt * p.ay;
146             p.x = p.x + dt * (p.vx + vx_old) * 0.5;
147             p.y = p.y + dt * (p.vy + vy_old) * 0.5;
148         }
149     }
150 }
```

Die Klasse `creatur` definiert die Struktur einer Kreatur. In den Zeile 3 und 5 werden Felder für die Punktmassen bzw. für die Segmente deklariert, aus denen die jeweilige Kreatur besteht. Die Methode `calculate_acceleration` berechnet für jeden Massepunkt i die Federkraft F_i^{spring} nach Gleichung 2.2, die jedes Segment auf den Massepunkt ausübt (Zeilen 22-63) und die Reibungskraft $F_i^{friction}$ (in abgewandelter Form) nach Glei-

chung 2.4 (Zeilen 67-109). Nach Gleichung 2.7 wird für jeden Massepunkt die Beschleunigung berechnet und in dem entsprechenden Feld des `point` Objekts abgespeichert (Zeilen 112-115). Die Methode `one_timestep` ruft die Methode `calculate_acceleration` auf (Zeile 128) und berechnet damit die aktuelle Position der Kreatur mit Hilfe des eulerschen Polygonzugverfahren (Zeilen 131-147): Zuerst wird die aktuelle Geschwindigkeit über das eulersche Polygonzugverfahren mit Hilfe der Beschleunigung bestimmt (Zeilen 144-145) und dann mit dem Mittelwert aus alter Geschwindigkeit (`vx_old, vy_old`) und neuer Geschwindigkeit (`vx, vy`) auf selbige Art die neue Position aller Massepunkte der Kreatur bestimmt.

2.3 Kritik am Modell

Wie beschrieben haben die Kreaturen einen unbeschränkten Vorrat an Energie, mit dem sie die Schwingung ihrer Segmente aufrecht erhalten. Das vereinfacht zwar das Modell und die Implementierung, ist aber unrealistisch. Man hätte den Kreaturen stattdessen einen beschränkten Vorrat an Energie zur Verfügung stellen können, der sich mit der Zeit leert. Ist der Vorrat einer Kreatur erloschen, kann sich die Ruhelänge des Segments nicht mehr ändern und die Kreatur wird durch die Dämpfung der Schwingung mit der Zeit bewegungsunfähig. Bei jedem Zweikampf könnte man den Vorrat des Siegers um einen festen Wert, oder um den Vorrat seiner Beute auffüllen. Durch Mutationen könnte man die Möglichkeit bieten die Größe des Vorrats zu verändern. Aber auch diese Erweiterung birgt einen Nachteil: Unter diesen Bedingungen ist die Entwicklung eines Fluchtverhaltens nur bedingt möglich.

3 Interaktion

Die Interaktion zwischen den Kreaturen beschränkt sich darauf, dass sich die Kreaturen gegenseitig jagen, da dieses Verhalten in der Natur weit verbreitet ist. Für die Simulation bedeutet das, dass die Evolution von den Jagdfähigkeiten der Kreaturen beeinflusst wird. Zuerst muss festgelegt werden, auf welche Art eine Kreatur eine andere erbeuten kann. Dazu wird jede Kreatur mit einem Herz und einem Mund ausgestattet. Sowie das Herz, als auch der Mund ist ein kreisförmiger Bereich, der einen festen Massepunkt umgibt. Dabei stellt das Herz die Schwachstelle der Kreaturen dar. Kommt der Mund einer fremden Kreatur in den Bereich des Herzens einer anderen Kreatur, so stirbt Letztere. Diese Festlegung bringt mehrere Konsequenzen mit sich: Jede Kreatur kann sowohl Jäger als auch Gejagter sein, auch beides gleichzeitig ist nicht auszuschließen. Die Möglichkeit, dass sich zwei Kreaturen gleichzeitig erbeuten, ist durch diese Modellierung sehr unwahrscheinlich. Weiter muss man davon ausgehen, dass die Entwicklung der Kreaturen beinflusst wird: Im Laufe der Simulation werden wohl viele Kreaturen entstehen, deren Münder nach vorne gerichtet sind, so wie wir es aus der Natur kennen.

Damit die Kreaturen eine Möglichkeit haben um andere Kreaturen wahrzunehmen und entsprechend zu reagieren, kann jedes Segment einer Kreatur mit einem Sensor ausgestattet sein. Diese Sensoren können bestimmte Eigenschaften ihres Segements verändern, wenn sie ein Herz oder einen Mund einer fremden Kreatur wahrnehmen. Dazu detektieren die Sensoren einen bestimmten Bereich in ihrer Umgebung, der von Sensor zu Sensor in der Größe variieren kann. Weiter unterschieden sich Sensoren in der Stärke des Einflusses oder der Ausrichtung relativ zu ihrem Segment, d.h. Länge und Winkel eines Sensors sind ebenfalls variabel. Welche Eigenschafften von Segmenten durch Sensoren verändern werden können, ist in Tabelle 3.1 dargestellt.

Eigenschaft	Auswirkung
Amplitude	das Segment schwingt stärker/schwächer
Reibungskoeffizient	ein Massepunkt wird klebriger/rutschiger
Reibungszahl	die Kreatur wird langsamer/schneller oder ändert die Richtung

Tabelle 3.1: Einflüsse von Sensoren und Beispiele ihrer Auswirkungen

Dazu hat jeder Sensor einen festen, vorzeichenunbeschränkten Wert, die Regulierungsstärke, welcher entsprechend des Einflusses zur Amplitude, zum Reibungskoeffizient oder zur Reibungszahl des Segments addiert. Dies kann unter anderem zu den oben genannten

Effekten führen. [2, Seite 498f.]

In Abbildung 3.1 ist eine einfache Kreatur mit Herz, Mund und zwei Sensoren dargestellt.

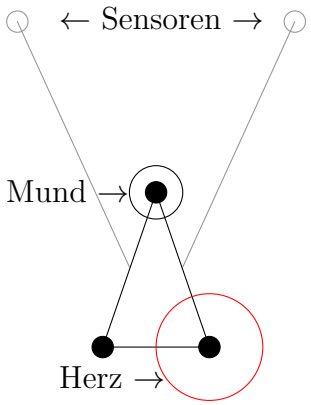

Abbildung 3.1: Beispiel einer einfachen Kreatur aus 3 Massepunkten, mit Herz, Mund und 2 Sensoren. Die Abbildung wurde aus [2, Seite 498] übernommen.

4 Fortpflanzung

Im folgenden Abschnitt wird beschrieben, was mit dem Sieger eines Zweikampfes passiert. Im Gegensatz zu seiner Beute wird dieser nicht aus der Simulation entfernt, sondern hat das Privileg sich zu reproduzieren. Die neue Kreatur wird daraufhin an einem zufälligen, aber freien Ort der Simulation hinzugefügt. Dabei kann es vorkommen das kein freier Platz gefunden wird. Deswegen ist es möglich die Anzahl der Versuche einen freien Platz zu finden nach oben begrenzt[3]. Bei einer Reproduktion gibt es folgende zwei Möglichkeiten: Entweder ist die neue entstandene Kreatur ein identisches Abbild, oder bei der Reproduktion kommt es zur Mutation. Die Wahrscheinlichkeit für eine Reproduktion mit Mutation beträgt in den Simulationen 10%. Auch mehrere Mutationen sind möglich: Für jede weiter Mutation beträgt die Wahrscheinlichkeit 40%. Also beträgt die Wahrscheinlichkeit für drei Mutationen noch über 1% ($0.1 \cdot 0.4^2 = 0.016$).

Die Art der Mutationen lässt sich in drei Kategorien unterteilen: Segmentmutationen, Verhaltensmutationen und physische Mutationen. [2, Seite 499f.]

Segmentmutationen Diese Mutationsklasse beschreibt Mutationen die genau eine Eigenschaft eines zufällig gewählten Segments verändern. Möglichen Eigenschaften sind die folgenden: Segmentlänge, Amplitude der Schwingung, Frequenz der Schwingung, Phase der Schwingung oder die Reibungszahl.[2, Seite 499]

Verhaltensmutationen In diese Mutationsklasse fallen alle Mutationen, die Sensoren betreffen. Entweder wird eine Eigenschafft des Sensors verändert, oder alle Eigenschaften eines Sensors werden auf einen anderen Sensor übertragen. Mögliche Eigenschaften die angepasst werden können sind die Länge und der Winkel des Sensors relativ zum Segment, der Radius des Bereichs den der Sensor überwacht, auf welches Körperteil (Mund oder Herz) der Sensor ausschlägt, oder welchen Effekt der Sensor auf das Segment hat. Wird letzeres verändert, so wird auch eine neue Regulierungsstärke gewählt. [2, Seite 499f.]

Physische Mutationen Mutationen die den Phänotypen einer Kreatur verändern fallen unter diese Kategorie. Genau genommen sind das folgende: Herz oder Mund werden an einem anderen Massepunkt versetzt, ein Segment wird entfernt (nur wenn die Kreatur dabei nicht in zwei Teile geteilt wird), ein neues Segment wird hinzugefügt (dieses

[3] In den Simulationen des folgenden Abschnitts war die Anzahl auf 40 Versuche begrenzt.

Segment ist ein baumelndes Segment und ist nur an einem Ende mit der Kreatur verbunden), zwei baumelnde Segmente werden durch ein drittes Segment verbunden, zwei baumelnde Segmente werden miteinander verbunden oder zwei neue Segmente werden hinzugefügt und bilden mit einem schon bestehenden Segment ein Dreieck. [2, Seite 500]

5 Simulation

Die Forschungsergebnisse wurden aus drei unterschiedlichen Klassen von Simulationsläufen gewonnen, die in den folgenden Abschnitten genauer beschrieben werden. Die erste Klasse der Simulationen umfasst die sogenannten „Lone Ancestor Runs“ [2, Seite 500f.]. Diese Simulationsklasse simulierte die eigentliche (künstliche) Evolution. Auf den Evolutionsergebnissen aufbauend wurde ein Tunier unter den besten Kreaturen aus den Evolutionsläufen ausgetragen. In diesem „Tournament Across Simulation“ [2, Seite 502f.] wurde die am besten angepasste Kreatur auf Basis der Evolutionsläufe ermittelt. Die dritte Simulationsklasse beinhaltet die „Contests Between Generations“ [2, Seite 502] Simulationen, in denen zwei Teams aus Kreaturen mit unterschiedlichen Entwicklungsalter gegeneinander antraten.

Die Umgebungen in denen diese Simulationen abliefen, waren stets die Selben. Lediglich die Start– und Endbedingungen variierten innerhalb der drei Simulationsklassen. Die Umgebung, bestehend aus einem einfachen, zweidimensionalen Rechteck, hatte keine hindernde Objekte wie Pflanzen oder Steine. In dieser fiktiven Landschaft konnten sich die Kreaturen frei bewegen, mit der Vereinfachung, dass Kreaturen, die eines der Enden erreichten nicht von diesem blockiert wurden, sondern über die Enden hinaus laufen konnten und ihre Bewegung an der gegenüber liegenden Seite fortführen konnten.

5.1 Lone ancestor runs

Alle der insgesamt 100 Evolutionsläufe hatten dieselben Startbedingungen. Nur der verwendete Zufallsgenerator wurde in jedem Lauf neu initialisiert[4]. Alle Läufe begannen mit 100 Kreaturen. Davon waren 99 der Kreaturen bewegungsunfähig. Diese bestanden aus einem nicht oszillierenden Segment, das zwei Massepunkte verband. Einer der Massepunkte war das Herz. Einen Mund gab es in diesem Fall nicht, d.h. diese Kreaturen waren so gemacht, dass sie selber keinen Zweikampf gewinnen konnten. Sie dienten der beweglichen Kreatur als Beute. Diese Kreatur bestand, wie die anderen auch, aus zwei Massepunkten, die ebenfalls durch ein Segment verbunden waren. Bei dieser Kreatur oszillierte das einzige Segment hingegen synchron zu den Reibungsverlagerungen der beiden Endpunkte. Dieses Verhalten führte dazu, dass sich diese Kreatur in geradliniger Bewegung fortbewegen konnte. Der Mund befand sich am vorderen Massepunkt

[4]Deterministische Zufallsgeneratoren erzeugen bei der gleichen Initialisierung, der sog. Saat (engl. seed), stets die selbe Folge von Pseudozufallszahlen. Damit die Simulationen nicht völlig identisch ablaufen, musste jeweils eine andere Saat gewählt werden.

und das Herz hinten. Die typische Simulationsdauer betrug dabei mindestens 2.000.000 Zeiteinheiten, war aber, aufgrund fehlender Endbedingung, nicht nach oben begrenzt.

Da zu Anfang jeder Simulation nur eine einzige Kreatur die physischen Vorraussetzugen hatte um sich fortzubewegen und somit auch als einzige in der Lage war sich fortzupflanzen, stand im Voraus fest, dass diese Kreatur der Vorfahre[5] aller im Laufe der Evolution entstehenden Kreaturen sein wird. In jeder der Simulationen kam es zwangsläufig zu einem Zusammenstoß zwischen der einzigen beweglichen Kreatur und einer der statischen Kreaturen. Die bewegliche Kreatur vermehrte sich und die andere wurde aus der Simulation entfernt. Mit der Zeit gab es immer mehr bewegliche und immer weniger statische Kreaturen. Die Artenvielfalt nahm zu und es kam zu Zweikämpfen zwischen zwei beweglichen Kreaturen. Dabei waren Zweikämpfe zwischen Kreaturen gleicher Art weniger interessant, da der Sieger in diesem Fall zufällig war[6] und nicht darauf beruht welche Kreatur besser an ihre Umgebung angepasst war. Letzteres ist der Grund warum Kämpfe zwischen unterschiedlichen Typen spannender waren. Hierbei war der Sieger nicht immer zufällig, da die Kreaturen im Laufe der Evolution unterschiedliche Verhaltensmuster entwickeln konnten und durch physische Mutationen Vor– oder Nachteile in bestimmten Kampfsituationen hatten. [2, Seite 500f.]

Ergebnisse In jedem der 100 Läufe entwickelten sich unterschiedliche Kreaturen, die sich optisch kaum ähnelten[7]. Trotzdem ließen sich einige, grobe Gemeinsamkeiten feststellen: Nach ausreichender Simulationsdauer konnte man beobachten, dass die meisten der Kreaturen den Mund an einem der vorderen und das Herz an einem der hinteren Massepunkten hatten. Dabei hatte der Mund mehr Bewegung als das Herz, welches meist nur hinterher gezogen wurde. Der Grund dafür ist denkbar einfach: Ein weit vorne positionierter Mund hat für Kreaturen den Vorteil, dass sie mit hoher Wahrscheinlichkeit zuerst eine andere Kreatur erwischt, als andersrum. Verstärkt wird dieser Vorteil durch das Hin- und Herschwingen des Mundes, was außerdem den Vorteil hat, mit dem Mund ein größeres Gebiet zu erreichen. Umgekehrt ist es mit den Herzen. Ein in Bewegungsrichtung weit hinten liegendes Herz, wird nicht so schnell erreicht. Würde das Herz genauso schwingen wie ein Mund, besteht die Gefahr, dass sich das Herz auf einen feindlichen Mund zubewegt und dadurch zur leichten Beute wird. Weiter hat sich Schnelligkeit als klarer Vorteil herrausgestellt, ganz einfach aus dem Grund, dass eine schnelle

[5] (engl. ancestor)

[6] Welche Kreatur einen solchen Zweikampf gewinnt, basiert nur auf den Bewegungsrichtungen und den Positionen beider Kreaturen und ist somit zufällig

[7] Vgl. Abbildung 5.1

Kreatur pro Zeiteinheit auf mehr feindliche Kreaturen treffen kann und somit häufiger die Möglichkeit hat sich zu reproduzieren. [2, Seite 501]

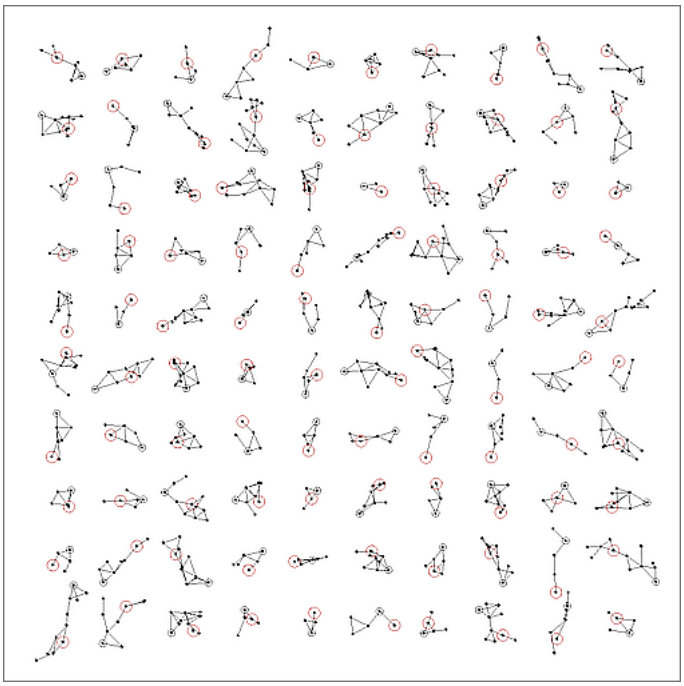

Abbildung 5.1: Diese Abbildung zeigt die erfolgreichsten Kreaturen aus den einzelnen lone ancestor runs. Die Abbildung stammt aus [2, Seite 501]

5.2 Contests Between Generations

Zu jedem der 100 Evolutions-Läufe wurde ein Wettkampf zwischen zwei Teams simuliert. Das erste Team bestand dabei aus 50 Kopien derjenigen Kreatur, welche nach 500.000 Zeiteinheiten zahlenmäßig am stärksten vertreten war. Das zweite Team war analog aufgebaut, nur dass die entsprechende Kreatur erst nach 2.000.000 Zeiteinheiten gewählt wurde. Damit bestand das zweite Team aus Kreaturen, die sich erst viel später im Laufe der selben Evolution entwickelt haben. Ziel dieser Simulationsklasse war es herauszufinden, ob spätere Generationen wirklich erfolgreicher im Kampf ums Überleben sind als Kreaturen früherer Generationen. Um diese Annahme zu festigen wurden die Fortpflanzungregeln modifiziert: Der Sieger eines Zweikampfes musste auf das Privileg der Reprodution und somit auch der Mutation verzichten. Stattdessen wurde der Verlierer an einem zufälligen Ort zurück in die Simulation gesetzt und nicht entfernt. Hierbei wurde nicht zwischen gleichen bzw. unterschiedlichen Kreaturen unterschieden. Durch

diese Einschränkung wurde sichergestellt, dass immer 50 Kopien beider Kreaturen in der Simulation vorhanden waren. Gezählt wurden nun die Anzahl der Siege beider Teams[8]. Leider geht die Dauer pro Simulation nicht aus dem Bericht hervor. Die Ergebnisse aller Wettkämpfe sind in Abbildung 5.2 graphisch dargestellt. [2, Seite 502]

Ergebnisse Aus Abbildung 5.2 ist zu entnehmen das man mit der Annahme nicht falsch lag. Die genauen Ergebnisse lassen sich aus der Graphik wie folgt ablesen: Auf der Abszissenachse sind die Siege des ersten Teams (500.000 Zeiteinheiten) und auf der Ordinatenachse die Siege des zweiten Teams (2.000.000 Zeiteinheiten) aufgetragen. Die Diagonale entspricht genau der Identität und kennzeichent somit die Grenze für den Sieg eines Teams. Demnach stehen Punkte, die auf der Diagonalen liegen, für ein Unendschieden, Punkte unterhalb der Linie für Siege des ersten Teams und Punkte überhalb der Linie für Siege des zweiten Teams. Wie man sieht, hat das zweite Team weit mehr Siege errungen. In einigen Wettkämpfen ist die Anzahl der gewonnen Zweikämpfe des zweiten Teams sogar signifikant größer, wohingegen alle Siege des ersten Teams relativ knapp waren. In Zahlen gefasst sieht das Gesamtergebniss der Generationswettkämpfe folgendermaßen aus: elf der Wettkämpfe wurden von den Kreaturen der früheren Generation gewonnen, 87 der Wettkämpfe gewannen die Teams der späteren Generation und zwei Wettkämpfe endeten unendschieden. Diese Werte wurden als Bestätigung der Annahme aufgefasst. Man muss davon ausgehen, dass nur genau gleich viele Siege beider Teams als Unendschieden gewertet wurden, da viele Wettkämpfe mit einem knappen Endstand zu Ende gingen. Hier hätte man gegebenfalls etwas toleranter sein können, aber andererseits hätte dies das endgültigen Resultat nicht beinflusst. Auffallend ist dabei auch die Spanne in der sich die Summen der Siege beider Teams bewegen. Diese reicht von nahezu null bis über 1.300[9]. [2, Seite 502]

5.3 Tournament Across Simulations

In den Evolutionsläufen hatten sich eine Vielfalt an Fortbewegungstechniken und Verhaltensmuster herausgebildet. Auch unter den jeweils dominierenden Kreaturen der einzelnen Simulationen haben sich verschiedene Techniken durchgesetzt. Aber welche dieser 100 Kreaturen setzt sich endgültig gegen die übrigen durch? Diese Frage wollte man mit Hilfe eines Tuniers klären. Dieses Tunier setzte sich aus zwei Runden zusammen. Die erste Runde bestand aus 10 Simulationen in denen zehn der ausgewählten Kreaturen

[8] Zweikämpfe zwischen Kreaturen aus dem gleichen Team wurden in der Wertung ignoriert.
[9] Genau Werte lassen sich hier nicht angeben, da nur die Graphik und keine genauen Werte vorliegen.

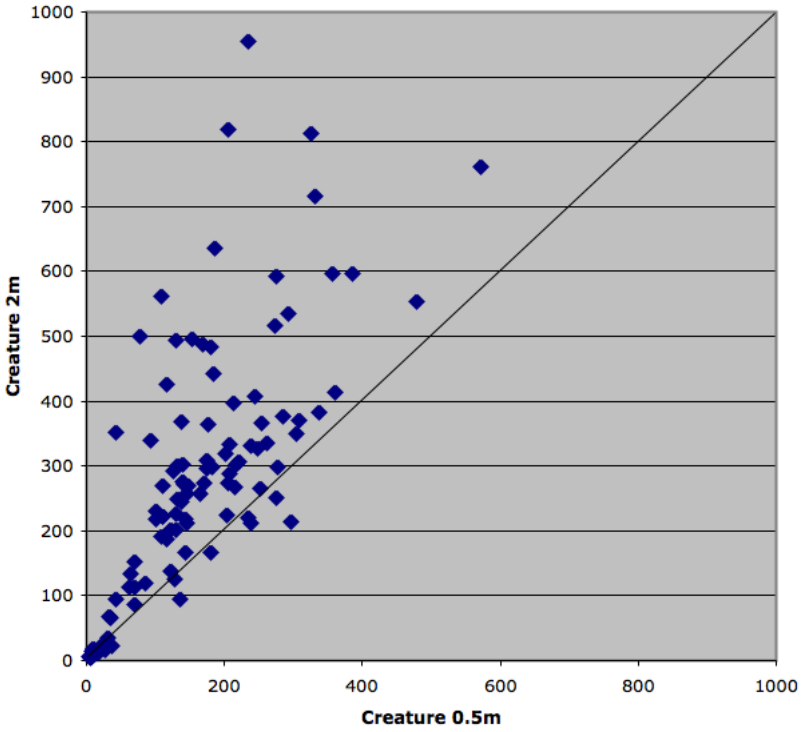

Abbildung 5.2: Die Ergebnisse der Generationswettkämpfe. Jeder der Punkte steht für das Resultat einer einzelnen Simulation. Die Abbildung stammt aus [2, Seite 502]

gegeneinander antraten. Jede Simulation begann mit jeweils zehn Kopien der teilnehmenden Kreaturen. Der Sieger eines Zweikampfs durfte sich kopieren (ohne Mutation), der Verlierer wurde entfernt. Gewinner eines Wettkampfes war diejenige Kreatur, welche sich gegen die anderen behaupten konnte und als einzige überlebte. Die zehn Gewinner der ersten Runde traten in der zweiten Runde unter den gleichen Bedingungen gegeneinander an. [2, Seite 502]

Ergebnisse Gewinner der zweiten Runde und somit Sieger des Tuniers war die hellgrüne Kreatur aus Abbildung 5.3. Mit zehn Massepunkten und 13 Segmenten war diese Kreatur komplexer als die meisten der anderen Teilnehmer. Der von Seite-zu-Seite schwingende, nach vorne gerichtete Mund, das bewegungslose Herz, welches die Kreatur hinter sich herzog, die gut aufeinander abgestimmten Schwingungen der Segmente und die Sensoren auf beiden Seiten, mit denen sich die Kreatur sowohl zur einen als auch zur anderen Seite bewegen konnte, haben wohl maßgeblich zu dem Erfolg beigetragen. Diese Kreatur war in ihrer Evolutionsumgebung gefärlicher als jede andere und setz-

te sich schließlich mit ihrem agressiven Jagdverhalten auch in dieser Tunierumgebung durch. Das Verhalten dieser Kreatur bestätigt nochmal die Beobachtungen, welche aus den Evolutionsläufen gewonnen wurden. [2, Seite 502f.]

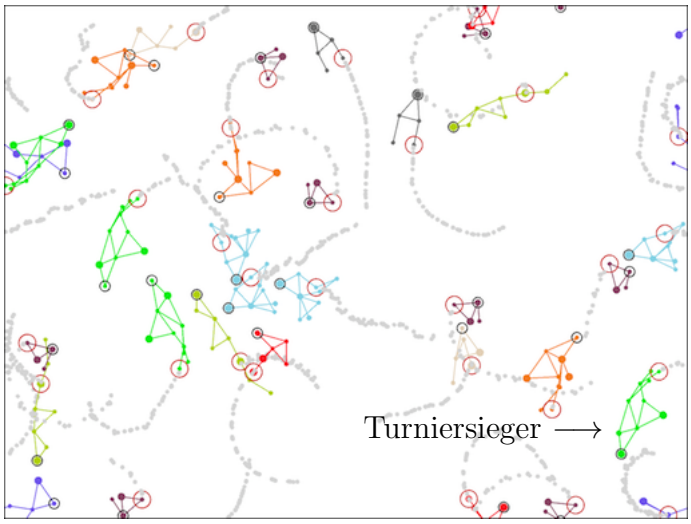

Abbildung 5.3: Die Abbildung stammt aus [2, Seite 503]

5.4 Anregungen

Interessant wäre noch zu wissen, ob die kontrahierenden Kreaturen in den Generationswettkämpfen mit einander „verwandt“ gewesen sind und ob die Art der Kreaturen aus dem ersten Teams im Laufe der Evolution ausgestorben ist. Allgemein wäre die Möglichkeit, die Veränderungen einer Spezies im Laufe der Zeit zurückzuverfolgen zu können, schön gewesen.

Literatur

[1] Greg Turk. Sticky feet: Evolution in a multi–creatur physical simulation. `http://www.cc.gatech.edu/~turk/stickyfeet/index.html`. zuletzt abgerufen am 30.09.2011.

[2] Greg Turk. Sticky feet: Evolution in a multi–creatur physical simulation. In *Proceedings of Artificial Life XII*, pages 496–503, Odense, Denmark, 2010.

[3] Andrew Witkin and David Baraff. Physically based modeling: Principles and practice. In *SIGGRAPH*, 1997. Course Notes.