Dokumentation zur Lösung der Aufgaben des 22. Bundeswettbewerb Informatik

Moritz Beller
Marcel Schmittfull

16. November 2003

Inhaltsverzeichnis

1 Vorwort

Während der Arbeiten an den gestellten Aufgaben stand für die Autoren vor allem das geschickte Lösen dieser im Vordergrund. Dabei mussten sie oft einsehen, dass Kompromisse zwischen der eigentlichen Aufgabenstellung und der praktischen Durchführbarkeit unabdingbar waren, um die Aufgaben in einem sinnvollen Zeitrahmen lösen zu können. Bei gegebener Stelle weisen die Autoren darauf bei der Programmbesprechung detaillierter hin.

Leider war es ihnen auch nicht möglich, alle Aufgaben im gegebenen Zeitraum zu lösen. Sie entschlossen sich deshalb, die Aufgabe „Grab“ nicht zu realisieren.

Um eine maximale Dynamik zu erhalten, wurden verschiedene Programmiersprachen verwendet. Für Aufgabe „Raff“ wurde Perl, für „Keuch“ und „Flitz“ Java eingesetzt. Dies hatte zumeist praktische Gründe und liegt daran, dass sich einige Probleme mit bereits bestehenden Programmstücken ergänzten oder die eine Programmiersprache offensichtlich besser geeignet war als die andere. Es ist also keinesfalls Willkür, sondern hat nachvollziehbare Gründe.

Es sei der Stadtbücherei Schweinfurt gedankt für ihre freundliche Erlaubnis, die hier vorliegende Dokumentation mit ihrem Laserdrucker aufs Papier bringen zu dürfen. Ebenfalls herzlichen Dank aussprechen möchten die Autoren den vielen Entwicklern von LaTeX, ohne die die Dokumentation ebenfalls nicht – zumindest in dieser Form – möglich gewesen wäre.

Marcel Schmittfull,
Moritz Beller

2 Raff

2.1 Programmbesprechung

Das Programm kann über zwei verschiedene Möglichkeiten aufgerufen werden: Entweder mit oder ohne einen optionalen Parameter für die Kursdatei:
`perl raff.pl -kursdatei.txt`
Wird kein Parameter angegeben, so sucht das Programm automatisch im aktuellen Verzeichnis nach einer Kursdatei kurse.txt.
`perl raff.pl`
Diese muss für ein erfolgreiches Einlesen in der folgenden Art aufgebaut sein:

Aufbau der Kursdatei

```
1  Waehrung1 Umtauschkurs Waehrung2
2  Waehrung3 Umtauschkurs Waehrung4
3  ...
```

Beispielkursdatei

```
1 Euro 1.95583 DM
2 US-Dollar 0.411 Euro
3 DM 0.8 US-Dollar
```

Die Suche nach dem besten Tauschzyklus besteht darin, dass alle möglichen Tauschreihenfolgen der Reihe nach durchprobiert und hierauf miteinander verglichen werden. Um alle möglichen Tauschreihenfolgen zu erhalten, werden alle möglichen Permutationen der Währungsindizes generiert und in ein Array `$alleZyklen` gespeichert. Bei zum Beispiel drei bzw. vier Währungen wird also folgende Liste von Permutationen erzeugt:

3

2	1	0
0	1	2
0	2	1
1	0	2
1	2	0
2	0	1
2	1	0

4

3	2	1	0
0	1	2	3
0	1	3	2
0	2	1	3
0	2	3	1
0	3	1	2
0	3	2	1
1	0	3	2
1	0	2	3
1	2	3	0
1	2	0	3
1	3	2	0
1	3	0	2

...

Aus dem Schulunterricht ist bekannt, dass es bei n Indizes bzw. Währungen genau $n! := 1 \cdot 2 \cdot 3 \cdots n$ verschiedene Permutationen gibt. Betrachtet man also eine Ziffer in der s-ten Spalte, wenn die Spalten wie in den Tabellen gezeigt gezählt werden, so gibt es genau $s!$ Permutationen mit dieser Ziffer an der s-ten Stelle. Somit kann die Liste der Permutationen dadurch generiert werden, dass die Spalten s in einer Schleife durchlaufen werden, in der wiederum jeweils alle Zeilen durchlaufen werden. Dann wird in einer Spalte s jeder Indexwert genau $s!$ Zeilen zugewiesen. Beispielsweise wird in der obigen Tabelle mit der Währungsanzahl 4 in der 3. Spalte jeder Indexwert $1, 2, \ldots$ in genau $3! = 6$ Zeilen geschrieben. In der 2. Spalte wird jeder Indexwert nur noch $2! = 2$ Mal wiederholt und schließlich in der 1. Spalte nur noch $1! = 1$ Mal. Fügt man dieser Vorgehensweise noch die Bedingung hinzu, dass innerhalb einer Zeile jeder Index nur ein Mal vorkommen darf, erhält man folgenden Algorithmus.

```
83  # Aktueller Index
84  $a = 0;
85  # Hilfsvariable, die verhindert, dass eine Ziffer doppelt in einer Zeile vorkommt.
86  $zahlKamSchonVor = 0;
87  # Durchlaufe alle Spalten...
88  for ($spalte = 0; $spalte < $anzahlWaehrungen; $spalte++) {
89      # Setze $a am Anfang jeder Spalte auf 0
90      $a = 0;
91      # Durchlaufe alle Zeilen...
92      for ($zeile = 0; $zeile < fakultaet($anzahlWaehrungen); $zeile++) {
93      # falls unerlaubtes $a
94      if ($zahlKamSchonVor == 1 || ($zeile != 0 && $zeile % fakultaet(
            $anzahlWaehrungen-$spalte-1) == 0)) {
95          # gehe zu naechstem $a
96          $a = naechst($a);
97      }
98      # setze Hilfsvariable wieder zurueck
99      $zahlKamSchonVor = 0;
100     # ueberpruefe alle vorhergehenden Spalten danach, ob $a bereits vorkam
101     for ($k = 0; $k < $spalte; $k++) {
102         if ($a == $alleZyklen[$zeile][$k]) {
103         $zahlKamSchonVor = 1;
104         # durchlaufe selbe Zeile nochmals
105         $zeile--;
106         }
107     }
108     # speichere $a in das $alleZyklen Array
109     $alleZyklen[$zeile][$spalte] = $a unless $zahlKamSchonVor == 1;
110     }
111 }
```

Hierbei lauten die Subfunktionen `naechst` und `fakultaet` wie folgt.

```
168 sub naechst {
169     my $p = shift;
170     return undef if $p >= $anzahlWaehrungen;
171     return 0 if $p == $anzahlWaehrungen-1;
172     return $p+1;
173 }
174
175 sub fakultaet {
176     my $n = shift;
177     return undef if $n < 0;
178     return 1 if $n == 0;
179     return $n * fakultaet($n - 1);
180 }
```

Nun müssen nur noch die einzelnen Permutationen bzw. Reihenfolgen der Reihe nach durchprobiert werden.

```
114 # Durchlaufe alle Zyklenmoeglichkeiten, suche dabei den Zyklus mit
115 # groesstem Gewinn pro Tausch und speichere diesen Zyklus.
116 $maximalerGewinnProTausch = 0;
117 for ($i=0; $i<fakultaet($anzahlWaehrungen); $i++) {
118     # Produkt der Kurse von Tausch zu Tausch
119     $aktuellesKurseProdukt = 1;
120     $bereitsErfassteRuecktauschStelleJ = 0;
121     # Durchlaufe den aktuellen Zyklus Tausch fuer Tausch
122     for ($j=0; $j<$anzahlWaehrungen; $j++) {
123     # Falls bereits mindestens ein Tausch gemacht wurde
124     if ($j>1) {
125         # und falls es moeglich ist, zu erster Waehrung zurueck zu tauschen
126         if ($kurs[$alleZyklen[$i][$j]][$alleZyklen[$i][0]]) {
127         # und diese Ruecktauschmoeglichkeit vorher noch nicht erfasst wurde
128         if ($j > $bereitsErfassteRuecktauschStelleJ) {
129             $bereitsErfassteRuecktauschStelleJ = $j;
130             # berechne Kurseprodukt
131             $aktuellesKurseProdukt *= $kurs[$alleZyklen[$i][$j]][$alleZyklen[$i
                    ][0]];
132             # wenn aktuellesKurseProdukt maximal
133             if($aktuellesKurseProdukt/($j+1) >= $maximalerGewinnProTausch) {
134             # es wurde ein Zyklus gefunden
135             $zyklusGefunden = 1;
136             # speichere diesen Zyklus
137             $besterZyklus = $i;
138             # und die Stelle, an der wieder zur Ausgangswaehrung gewaechselt wird
139             $ruecktauschStelleDesBestenZyklus = $j;
140             # und den neuen maximalen Gewinn pro Tausch
141             $maximalerGewinnProTausch = $aktuellesKurseProdukt/($j+1);
142             }
143         }
144         }
145     }
146     # Abbruch, falls Kurs j -> j+1 nicht vorhanden
147     last if (!$alleZyklen[$i][$j+1] || !$kurs[$alleZyklen[$i][$j]][$alleZyklen[$i][
            $j+1]]);
148     # sonst: berechne aktuellesKurseProdukt
149     $aktuellesKurseProdukt *= $kurs[$alleZyklen[$i][$j]][$alleZyklen[$i][$j+1]];
150     }
151 }
```

Dabei ist ein Zyklus ist so definiert, dass ausschließlich die erste und die letzte Währung äquivalent sind. Befindet sich also innerhalb eines Zykluses eine Wiederholung zweier Währungen, so handelt es sich nicht um einen korrekten Zyklus. Folglich kann man das arithmetische Mittel verwenden:

```
141 $maximalerGewinnProTausch = $aktuellesKurseProdukt/($j+1);
```

Als die Autoren beinahe die Erledigung der Aufgabe abgeschlossen hatten und testweise die Kurstabelle eingaben, fiel ihnen auf, dass die Kurstabelle Gebrauch von verschiedenen Plural-Formen macht. (Entenpeseta wird zu Entepeseten. Froschkrone wird zu Froschkronen.) Nun wäre es möglich gewesen, die Ähnlichkeit zweier Währungen mit umständlichen Berechnungen, wie sie beispielsweise von Suchmaschinen angewendet werden („Soundex-Verfahren“) zu erfahren und aufgrund dieser Ergebnisse zu entscheiden, ob es sich um den Plural eines bereits in der eingelesenen Kurstabelle befindlichen Singulars handelt. Dies wäre jedoch sicherlich nicht im Sinne der Aufgabenstellung gewe-

sen und hätte zudem zu einer unnötigen Verkomplizierung der Aufgabe geführt. Deshalb fügten wir, um die originale Kurstabelle unverändert einlesen zu können, die folgende Zeile Code ein, die die ersten vier Buchstaben einer Währung als „ID“ benutzt, um so dem oben angesprochenen Singluar/Plural-Problem auf adäquate Weise beizukommen. Sie ist standardmäßig im Code *deaktiviert,* kann jedoch durch Entfernen des #-Kommentars (wie in der folgenden Zeile geschehen) aktiviert werden.

```
25 $_ =~ s/(.{1,4}).*? (.*?) (.{1,4}).*?/$1 $2 $3/; # Ermoegliche Singluar/Plural
```

2.2 Ergebnisse für versch. Tageskurs-Tabellen

Die Ausgaben von raff.pl wurden in eine Datei umgeleitet und deren Inhalt in die Listing-Kästen kopiert. Als letztes ist die in der Aufgabenstellung geforderte Tagekurstabelle realisiert.

Tageskurstabelle I

```
1 NY-City-Dollars 5.5 Yen
2 NY-City-Dollars 0.4 Pfund
3 NY-City-Dollars 0.15 Cola-Mark
4 Yen 0.04 NY-City-Dollars
5 Cola-Mark 5.2 Yen
6 Yen 0.8 Cola-Mark
7 Cola-Mark 7.8 Pfund
8 Pfund 4.5 Yen
9 Pfund 1.5 NY-City-Dollars
```

```
  Bester Tauschzyklus: NY-City-Dollars -> Yen -> Cola-Mark -> Pfund ->
NY-City-Dollars
Durchschnittlicher Gewinn pro Tausch: 12.87
```

Tageskurstabelle II

```
1 Marcel-Mark 9.8 Moritz-Mark
2 Marcel-Mark 0.7894 Marc-Mark
3 Moritz-Mark 0.56 Marcel-Mark
4 Marc-Mark 5.5 Moritz-Mark
5 Marc-Mark 0.9 Marcel-Mark
```

```
  Bester Tauschzyklus: Marcel-Mark -> Marc-Mark -> Moritz-Mark ->
Marcel-Mark
Durchschnittlicher Gewinn pro Tausch: 0.810450666666667
```

Tageskurstabelle III

```
1 Baerentaler 2.70 Mausmark
2 Entenpeseten 0.75 Kroetendollar
3 Entenpeseten 0.70 Froschkronen
4 Entenpeseten 1.80 Wolfspfunde
5 Froschkronen 1.10 Kroetendollar
6 Froschkronen 1.10 Entenpeseten
```

```
Kroetendollar 2.00 Wolfspfunde
Kroetendollar 1.90 Wolfspfunde
Kroetendollar 1.05 Froschkronen
Mausmark 1.30 Entenpeseten
Mausmark 0.90 Kroetendollar
Wolfspfunde 0.70 Entenpeseten
Wolfspfunde 0.20 Baerentaler
Wolfspfunde 0.50 Mausmark
```

```
Bester Tauschzyklus: Wolfspfunde -> Mausmark -> Entenpeseten ->
Krötendollar -> Wolfspfunde
Durchschnittlicher Gewinn pro Tausch: 0.4168125
```

2.3 Quellcode

Obwohl auszugsweise bereits weiter oben zu Dokumentationszwecken verwendet, entschloss man sich der Lesbarkeit und Einfachheit halber den Quelltext voll wiederzugeben.

```
#! /usr/bin/perl -w

### Aufgabe 1 "Raff", 22. Bundeswettbewerb Informatik
### Copyright (C) 2003 by Moritz Beller (NOSPAMmomo@NOSPAM4momo.de)
### and Marcel Schmittfull (marcel-sl@NOSPAMgmx.de)

# Bestimmt Kurstabellendatei
# Aufruf ueber Argument: raff.pl -KURSDATEI
$file = $ARGV[0];
# Ansonsten als Standard kurse.txt
if(!$ARGV[0]) { $file = "kurse.txt"; }

# Deklaration von Variablen
$kurs = [];
$alleZyklen = [];
$anzahlWaehrungen = 0;
$anzahlWaehrungen = 0;

# Liest $file ein
do_readin($file);

# Ermitteln der Datei-Struktur
foreach(@fileZeilen) {
    if($_ =~ m/(.*?) (.*?) (.*?)/) { # Pruefe, ob Format eingehalten
    # $_ =~ s/(.{1,4}).*? (.*?) (.{1,4}).*?/$1 $2 $3/; # Ermoegliche Singular/Plural
    my $name = $1;

    my $exists = 0;
    foreach(@bezeichnung) {
        if($_ eq $name) {
        $exists = 1;
        }
    }

    if(!$exists) {
        push(@bezeichnung,$name);
    }
    }
}

# Vereinfacht das Auslesen der Kurstabelle
foreach(@bezeichnung) {
    foreach(@fileZeilen) {
    $_ =~ s/$bezeichnung[$anzahlWaehrungen]/$anzahlWaehrungen/;
```

```
45      }
46      $anzahlWaehrungen ++;
47  }
48
49  # Weisst Werte aus der Kurstabelle dem 2d-Array
50  # $kurs zu.
51  foreach(@fileZeilen) {
52      # Pruefe, ob Format eingehalten
53      if ($_ =~ m/(.*?) (.*?) (.*)/) {
54      # Pruefe, ob ein Kurs derselben Waehrungen bereits eingelesen
55      # ist und schreibe nur neuen kurs, wenn er besser (groesser) als
56      # der bereits eingelesene kurs ist.
57      if ($kurs[$1][$3] && $2>$kurs[$1][$3]) {
58          $kurs[$1][$3] = $2;
59      }
60      else {
61          $kurs[$1][$3] = $2;
62      }
63      }
64  }
65
66  # Speichere alle moeglichen Zyklen fuer Waehrungstausche nach
67  # folgendem Muster in das zweidimensionale $alleZyklen Array:
68  #
69  # 0 1 2 3
70  # 0 1 3 2
71  # 0 2 1 3
72  # 0 2 3 1
73  # 0 3 1 2
74  # 0 3 2 1
75  # 1 0 3 2
76  # 1 0 2 3
77  # 1 2 3 0
78  # 1 2 0 3
79  # 1 3 2 0
80  # 1 3 0 2
81  # ...
82
83  # Aktueller Index
84  $a = 0;
85  # Hilfsvariable, die verhindert, dass eine Ziffer doppelt in einer Zeile vorkommt.
86  $zahlKamSchonVor = 0;
87  # Durchlaufe alle Spalten...
88  for ($spalte = 0; $spalte < $anzahlWaehrungen; $spalte++) {
89      # Setze $a am Anfang jeder Spalte auf 0
90      $a = 0;
91      # Durchlaufe alle Zeilen...
92      for ($zeile = 0; $zeile < fakultaet($anzahlWaehrungen); $zeile++) {
93      # falls unerlaubtes $a
94      if ($zahlKamSchonVor == 1 || ($zeile != 0 && $zeile % fakultaet(
            $anzahlWaehrungen-$spalte-1) == 0)) {
95          # gehe zu naechstem $a
96          $a = naechst($a);
97      }
98      # setze Hilfsvariable wieder zurueck
99      $zahlKamSchonVor = 0;
100     # ueberpruefe alle vorhergehenden Spalten danach, ob $a bereits vorkam
101     for ($k = 0; $k < $spalte; $k++) {
102         if ($a == $alleZyklen[$zeile][$k]) {
103         $zahlKamSchonVor = 1;
104         # durchlaufe selbe Zeile nochmals
105         $zeile--;
106         }
107     }
108     # speichere $a in das $alleZyklen Array
109     $alleZyklen[$zeile][$spalte] = $a unless $zahlKamSchonVor == 1;
110     }
111 }
```

```
112
113
114 # Durchlaufe alle Zyklenmoeglichkeiten, suche dabei den Zyklus mit
115 # groesstem Gewinn pro Tausch und speichere diesen Zyklus.
116 $maximalerGewinnProTausch = 0;
117 for ($i=0; $i<fakultaet($anzahlWaehrungen); $i++) {
118     # Produkt der Kurse von Tausch zu Tausch
119     $aktuellesKurseProdukt = 1;
120     $bereitsErfassteRuecktauschStelleJ = 0;
121     # Durchlaufe den aktuellen Zyklus Tausch fuer Tausch
122     for ($j=0; $j<$anzahlWaehrungen; $j++) {
123     # Falls bereits mindestens ein Tausch gemacht wurde
124     if ($j>1) {
125         # und falls es moeglich ist, zu erster Waehrung zurueck zu tauschen
126         if ($kurs[$alleZyklen[$i][$j]][$alleZyklen[$i][0]]) {
127         # und diese Ruecktauschmoeglichkeit vorher noch nicht erfasst wurde
128         if ($j > $bereitsErfassteRuecktauschStelleJ) {
129             $bereitsErfassteRuecktauschStelleJ = $j;
130             # berechne Kurseprodukt
131             $aktuellesKurseProdukt *= $kurs[$alleZyklen[$i][$j]][$alleZyklen[$i
                    ][0]];
132             # wenn aktuellesKurseProdukt maximal
133             if($aktuellesKurseProdukt/($j+1) >= $maximalerGewinnProTausch) {
134             # es wurde ein Zyklus gefunden
135             $zyklusGefunden = 1;
136             # speichere diesen Zyklus
137             $besterZyklus = $i;
138             # und die Stelle, an der wieder zur Ausgangswaehrung gewaechselt wird
139             $ruecktauschStelleDesBestenZyklus = $j;
140             # und den neuen maximalen Gewinn pro Tausch
141             $maximalerGewinnProTausch = $aktuellesKurseProdukt/($j+1);
142             }
143         }
144         }
145     }
146     # Abbruch, falls Kurs j -> j+1 nicht vorhanden
147     last if (!$alleZyklen[$i][$j+1] || !$kurs[$alleZyklen[$i][$j]][$alleZyklen[$i][
            $j+1]]);
148     # sonst: berechne aktuellesKurseProdukt
149     $aktuellesKurseProdukt *= $kurs[$alleZyklen[$i][$j]][$alleZyklen[$i][$j+1]];
150     }
151 }
152
153 # Printe Ergebnis
154
155 if ($zyklusGefunden) {
156     print "\nBester Tauschzyklus: ";
157     for ($i=0; $i<$ruecktauschStelleDesBestenZyklus+1; $i++) {
158     print $bezeichnung[$alleZyklen[$besterZyklus][$i]]." -> ";
159     }
160     print $bezeichnung[$alleZyklen[$besterZyklus][0]];
161     print "\n";
162     print "Durchschnittlicher Gewinn pro Tausch: ".$maximalerGewinnProTausch;
163     print "\n";
164 }
165 else { print "\nKein bester Zyklus gefunden!\n"; }
166
167
168 sub naechst {
169     my $p = shift;
170     return undef if $p >= $anzahlWaehrungen;
171     return 0 if $p == $anzahlWaehrungen-1;
172     return $p+1;
173 }
174
175 sub fakultaet {
176     my $n = shift;
177     return undef if $n < 0;
```

```
178     return 1 if $n == 0;
179     return $n * fakultaet($n - 1);
180 }
181
182 sub do_readin {
183     open(DATEI,"<$_[0]") || die "Error 404 - File not found";
184     while(<DATEI>)
185     { push(@fileZeilen,"$_"); }
186     close(DATEI);
187 }
```

3 Keuch

Durch seine Programmierung in Java ist ein Einsatz der Alarmstrategie auf allen verfügbaren Betriebssystemen gewährleistet – selbst auf Palms und Handys sollte nach geringem Portierungsaufwand eine lauffähige Verision ohne weiteres möglich sein.

Keuch ist komplett objektorientiert, es existieren eine Schadstoff-, eine Server- und eine Handyklasse. Über eine Messfunktion greift ein Handy-Objekt auf die Umwelt zu und holt sich die aktuellen Messwerte. Die Serverklasse nimmt dabei eine passive Rolle an und wird vom Handy-Objekt bei Bedarf angesprochen.

Es gibt zwei Möglichkeiten, Keuch aufzurufen: Zum einen mit allen Parametern

```
java Keuch [anzahlHandys] [anzahlServer] [handyKapazitaet] [etliche]
[einige] [groessereAnzahl] [kurzeZeit] [schlafDauer] [zeigeAlleHandys]
[angezeigtesHandy] [zeigeServerlog] [soundAn]
```

und zum anderen mit der vorgegebenen Standard-Konfiguration *default.*

```
java Keuch default
```

3.1 Bewusstes Alarmauslösen

Ein Problem der Alarmstrategie des Servers besteht darin, dass durch eine relativ geringe Zahl an „Störern“ das komplette System durcheinander geraten kann.

Die mutwilligen Angreifer müssten nur durch Herabsetzen der eigenen Grenzwerte Gelb-Nachrichten an den Server verschicken. Dieser leitet dann jede einzelne einkommende Gelb-Nachricht an zehn weitere zufällige Handy-Besitzer. Es kann also leicht eine „Denial-of-Server-Attack“ ausgeführt werden, indem der zuständige Zellserver mit einer Vielzahl an Gelb-Nachrichten von verschiedenen „manipulierten“, das heißt mit extrem niedrigen Grenzwerten versehenen, Handys „überflutet“ wird und somit an alle Handys in seiner Zelle eine Rot-Meldung verschickt. Dazu braucht man nur genügend manipulierte Handys, die eine so große Zahl Gelb-Nachrichten in kurzer Zeit an den Server weiterleiten können, dass dieser eine Rot-Mitteilung erzeugt, die an alle Handy-Besitzer in der Zelle gelangt.

Um solchen mutwilligen Angriffen nicht weiter nichts entgegensetzen zu können, müssen wir die Alarmstrategie des Servers dahingehend verändern, dass nur jeweils eine Gelb-Nachricht pro Schadstoff und definierbarem Zeitraum von einem Handy versandt werden darf. Das heißt dann ganz konkret, dass Handy1 innerhalb von einer Stunde nur eine Gelb-Warnmeldung für den Stoff Ozon, eine Warnmeldung für Stickstoff usw. verschicken darf. Da es allerdings aus Sicht der Sicherheit bedenklich wäre, diese Sperre in einem Client – sprich Handy – zu realisieren, da findige Hacker sicher bald eine Umgehung der Software eingerichtet hätten, müsste der Server für alle einkommenden Gelbnachrichten prüfen, ob innerhalb der letzten 60 Minuten bereits eine Warnmeldung von diesem Handy für diesen Schadstoff abgeschickt worden ist und entsprechend darauf reagieren: Entweder als Reaktion das Zulassen der Gelbnachricht, womit die Speicherung der Zeit und ein Gelbnachrichten-Versand an zehn zufällige weitere Handys einhergeht oder die Ablehnung der Gelbnachricht.

Zusätzlich zu dieser ersten Sicherheitsvorkehrung muss der Server über jede Grenzwertänderung auf einem Client informiert werden und soll bei einer zweiten Grenz-

wertänderung innerhalb von 24 Stunden den Änderungswunsch nicht akzeptieren. Dies bedeutet, dass Handy1 bei einem Änderungsversuch von den Grenzwerten von Ozon erst beim Server anfragt, ob ihm diese Aktion überhaupt gestattet ist. Dafür benötigen wir für jedes Handy auf dem Server ein 2d-Array `lastChange`, das die letzte Aktualisierung für die jeweiligen Schadstoffe abspeichert. Wenn dies der Fall ist, setzt der Server die Variable `lastChange` für die letzte Änderung des Grenzwertes von Ozon auf Handy1. Will Handy1 beispielsweise nach fünf Minuten wieder eine Grenzwertveränderung erreichen, so muss es erneut eine Anfrage an den Server richten. Dieser liest den Inhalt von `lastChange` für Ozon ein und erkennt, dass die letzte Änderung erst fünf Minuten zurückliegt. Der Änderungwunsch wird zurückgewiesen. So kann vermieden werden, dass eine wiederholte Grenzwertmanipulation stattfindet.

3.2 Zellservererweiterungen

Eine sinnvolle Erweiterung wäre sicherlich, dass ein Zellserver bei Empfang einiger Rot-Nachricht für denselben Schadstoff innerhalb kurzer Zeit, Rot-Meldungen an alle Benutzer in seinem System und an vier weitere, benachbarte Zellenserver verschickt. So wäre nicht nur eine Zellserver-interne sondern auch eine Kommunikation der Server untereinander denkbar. Ernsthafte Schadstoffprobleme – repräsentiert durch die Rot-Meldungen – gelangen in kurzer Zeit an alle für die Nachricht relevanten Stellen.

Ebenfalls einen großen Nutzen hätte eine interaktive Veränderbarkeit der Strategievariablen – bei Nacht zum Beispiel sollten zwei Gelb-Warnungen viel mehr zählen als um die Mittagszeit, wenn fast alle Handys am System angeschlossen sind. Das heiß, dass eine Art Punkte-System, abhängig von der derzeit auf dem Server eingeloggten Zahl an Handys, eingeführt werden müsste.

$$\text{Einige} = \frac{\text{Handyzahl}}{20} \tag{1}$$

$$\text{Größere Zahl} = \frac{\text{Handyzahl}}{10} \tag{2}$$

Die Gleichungen (1) und (2) regulieren einfach anhand der am Server angemeldeten Handyzahl die Strategievariablen „einige“ und „größere“. Bei nur einer geringen Zahl an Handys kann natürlich auch nur eine geringe Zahl an Gelbnachrichten verschickt werden. Die Gleichungen reagieren auf diesen Sachverhalt, indem sie die benötigten Meldungen der Zahl der gemeldeten Handys anpassen. Für den Idealfall der Vollauslastung des Servers ($\triangleq$ 100 angemeldeten Handys) hat „einige“ den Wert fünf, bei Wenigbeanspruchung (z.B. zwölf Handys) den Wert 1,2 inne. „Größere“ verhält sich analog dazu.

$$\text{Kurze Zeit} = \frac{1}{\text{Handyzahl}} \cdot 1000 \text{ Zeitschritte} \tag{3}$$

In Gleichung (3) definieren wir, dass die reziproke Handyzahl multipliziert mit 1000 Zeiteinheiten (1 Zeiteinheit $\triangleq$ beispielsweise einer Sekunde) die Strategievariable „kurze Zeit“ liefert.

3.3 Quellcode

Handy.java

```
1  import java.util.Random;
2  import java.awt.*;
3
4  public class Handy {
5
6      // Index des Handys
7      int index;
8      // Array mit allen Schadstoffen
9      Schadstoff[] schadstoffe;
10     // Server bei dem das Handy im Moment eingeloggt ist
11     Server meinServer;
12     // Alle Server
13     Server[] alleServer;
14     // Strategievariablen
15     int kurzeZeit, etliche;
16     // Hilfsvariablen
17     int i, j, summeGelbs, angezeigtesHandy;
18     double hilf1, hilf2;
19     boolean zeigeAlleHandys, soundAn;
20     // Zufallsobjekt
21     Random random;
22
23     // Konstruktor
24     public Handy (int index, Server[] alleServer, Schadstoff[] schadstoffe, int
           kurzeZeit, int etliche, Random random) {
25     // Speichere Index
26     this.index = index;
27     // Speichere Schadstoffliste
28     this.schadstoffe = schadstoffe;
29     // Speichere Strategievariablen
30     this.kurzeZeit = kurzeZeit;
31     this.etliche = etliche;
32     // Erzeuge Zufallsobjekt
33     this.random = random;
34     this.alleServer = alleServer;
35     // Handy loggt sich in zufaelligen Server ein (waere in der Realitaet
36     // der naeheste Server, wird hier aber per Zufall ermittelt.)
37     if (alleServer.length > 0)
38         login(alleServer[random.nextInt(alleServer.length-1)]);
39     }
40
41     // Login bei Server
42     void login (Server a) {
43     meinServer = a;
44     meinServer.login(this);
45     }
46
47     // Aendere Grenzwerte
48     void aendereGrenzwert (Schadstoff s, double grenzwert) {
49     s.grenzwert = grenzwert;
50     }
51
52
53     /* Naechster Zeitschritt */
54     public void naechsterZeitschritt (boolean zeigeAlleHandys, int angezeigtesHandy
           , Server[] alleServer, boolean soundAn) {
55     this.zeigeAlleHandys = zeigeAlleHandys;
56     this.angezeigtesHandy = angezeigtesHandy;
57     this.soundAn = soundAn;
58     messeAktuellenLuftgehalt();
59     pruefeGrenzwerte();
60     pruefeGelbs();
```

```
61      if (random.nextInt(10) == 1) {
62          meinServer.logout(this);
63          login(alleServer[random.nextInt(alleServer.length-1)]);
64      }
65      // Bearbeite Zeit Array
66      for (i=0; i<schadstoffe.length; i++) {
67          for (j=0; j<schadstoffe[i].gelbNachrichten.length-1; j++) {
68          schadstoffe[i].gelbNachrichten[j] = schadstoffe[i].gelbNachrichten[j+1];
69          }
70      }
71      }
72
73      // Messe aktuellen Luftgehalt (gehoert nicht zur Alarmstrategie, ist
74      // aber fuer eine Simulation der Strategie noetig).
75      void messeAktuellenLuftgehalt () {
76      for (i=0; i<schadstoffe.length; i++) {
77          // Zufallswert fuer Simulation..
78          hilf1 = random.nextDouble()/10;
79          if (random.nextInt(100) < 55)
80          hilf2 = 1;
81          else
82          hilf2 = -1;
83          if (schadstoffe[i].aktuellerLuftgehalt == 0)
84          hilf1 = schadstoffe[i].aktuellerLuftgehalt + hilf1*hilf2*schadstoffe[i].
                grenzwert;
85          else
86          hilf1 = schadstoffe[i].aktuellerLuftgehalt + hilf1*hilf2*schadstoffe[i].
                aktuellerLuftgehalt;
87          if (hilf1 < 0)
88          hilf1 = 0.;
89          else
90          hilf1 = Math.rint(hilf1*100.)/100.;
91          schadstoffe[i].aktuellerLuftgehalt = hilf1;
92      }
93      }
94
95      // Ueberpruefe, ob irgendwelche Grenzwerte ueberschritten sind und reagiere
96      // demenstprechend.
97      void pruefeGrenzwerte () {
98      for (i=0; i<schadstoffe.length; i++) {
99          if (schadstoffe[i].aktuellerLuftgehalt > schadstoffe[i].grenzwert) {
100         grenzwertUeberschritten(schadstoffe[i]);
101         }
102     }
103     }
104
105     // Ueberpruefe, ob "Etliche" Gelb-Nachrichten in "kurzer Zeit" entstanden
106     void pruefeGelbs () {
107     for (i=0; i<schadstoffe.length; i++) {
108         summeGelbs = 0;
109         for (j=0; j<kurzeZeit; j++) {
110         summeGelbs += schadstoffe[i].gelbNachrichten[j];
111         }
112         if (summeGelbs > etliche) {
113         etlicheGelbInKurzerZeit(schadstoffe[i]);
114         }
115     }
116     }
117
118     /* Taetigkeiten des Handys */
119
120     void grenzwertUeberschritten (Schadstoff s) {
121     if (zeigeAlleHandys == true || angezeigtesHandy == this.index) {
122         // akustische Warnung
123         beep();
124         // optische Warnung
125         System.out.println(this.index+": Grenzwertueberschreitung von "+s.name+" (
                aktuell "+s.aktuellerLuftgehalt+", Grenzwert "+s.grenzwert+")");
```

```
126     }
127     // Gelb-Nachricht an Server
128     meinServer.gelbVonHandy(s, this);
129     }
130 
131     // Eine Gelb-Nachricht vom Server
132     public void einGelbVonServer (Schadstoff s) {
133     if (zeigeAlleHandys == true || angezeigtesHandy == this.index) {
134         // akustische Warnung
135         beep();
136         // optische Warnung
137         s.gelbNachrichten[s.gelbNachrichten.length-1] += 1;
138     }
139     }
140 
141     // "Etliche" Gelb-Nachrichten in "kurzer Zeit"
142     void etlicheGelbInKurzerZeit (Schadstoff s) {
143     if (zeigeAlleHandys == true || angezeigtesHandy == this.index) {
144         // akustische Warnung
145         beep();
146         // optische Warnung
147         //messeAktuellenLuftgehalt();
148         System.out.println(this.index+": Gelb-Nachricht fuer "+s.name+" (aktuell "+s
                .aktuellerLuftgehalt+", Grenzwert "+s.grenzwert+")");
149     }
150     }
151 
152     // Rot-Nachricht von Server
153     public void rotVonServer (Schadstoff s) {
154     if (zeigeAlleHandys == true || angezeigtesHandy == this.index) {
155         // akustische Warnung
156         beep();
157         // optische Warnung
158         //messeAktuellenLuftgehalt();
159         System.out.println(this.index+": Rot-Nachricht fuer "+s.name+" (aktuell "+
                this.schadstoffe[s.index].aktuellerLuftgehalt+", Grenzwert "+this.
                schadstoffe[s.index].grenzwert+")");
160     }
161     }
162 
163     void beep () {
164     if (soundAn) {
165         Toolkit.getDefaultToolkit().beep();
166         Thread me = Thread.currentThread();
167         me.setPriority(Thread.MIN_PRIORITY);
168         //me.setPriority(Thread.MAX_PRIORITY);
169         try {
170         me.sleep(80);
171         }
172         catch (InterruptedException e) {
173         return;
174         }
175     }
176     }
177 }
```

Server.java

```
1  import java.util.Vector;
2  import java.util.Random;
3 
4  public class Server {
5 
6      // Liste aller im Moment eingeloggten Handys
7      public Handy[] eingeloggteHandys;
8      // Liste der Schadstoffe
```

```
9       Schadstoff[] schadstoffe;
10      // Alle Server
11      Server[] alleServer;
12      // Liste der benachbarten Server
13      public Server[] nachbarServer;
14
15      // Server Index
16      public int index;
17
18      // Zufallsobjekt
19      Random random = new Random();
20
21      // Strategievariablen
22      int einige;
23      int kurzeZeit;
24      int groessereAnzahl;
25
26      // Hilfsvariablen
27      int i, j, k, a, summeGelbs;
28      int momentanEingeloggteHandys;
29      boolean zeigeServerLog;
30
31      // Konstruktor
32      public Server (Schadstoff[] schadstoffe, int handyKapazitaet, int einige, int
            kurzeZeit, int groessereAnzahl, int index, Random random) {
33      this.schadstoffe = schadstoffe;
34      this.einige = einige;
35      this.kurzeZeit = kurzeZeit;
36      this.groessereAnzahl = groessereAnzahl;
37      // handyKapazitaet ist maximale Anzahl eingeloggter Handys
38      eingeloggteHandys = new Handy[handyKapazitaet];
39      this.random = random;
40      this.index = index;
41      }
42
43      // Login eines Handys
44      public void login (Handy h) {
45      if (momentanEingeloggteHandys < eingeloggteHandys.length-1) {
46          eingeloggteHandys[momentanEingeloggteHandys] = h;
47          momentanEingeloggteHandys++;
48          if (zeigeServerLog)
49          System.out.println("Login Handy "+h.index+" auf Server "+this.index);
50      }
51      }
52
53      // Logout eines Handys
54      public void logout (Handy h) {
55      for (i=0; i<momentanEingeloggteHandys; i++) {
56          if (eingeloggteHandys[i].equals(h)) {
57          for (j=i; j<momentanEingeloggteHandys-i; j++) {
58              eingeloggteHandys[j] = eingeloggteHandys[j+1];
59          }
60          momentanEingeloggteHandys--;
61          if (zeigeServerLog)
62              System.out.println("Logout Handy "+eingeloggteHandys[i].index+" von
                    Server "+this.index);
63          break;
64          }
65      }
66      }
67
68      /** Naechster Zeitschritt **/
69      public void naechsterZeitschritt (boolean zeigeServerLog) {
70      this.zeigeServerLog = zeigeServerLog;
71      pruefeGelbs();
72      // Im Gelb-Nachrichten-Array jedes Schadstoffs wird der aelteste Werte
73      // an Stelle 0 geloescht und alle anderen Werte (von Stelle 1 bis
74      // length-2) um eins nach links verschoben. An Stelle length-1 steht 0
```

```
75      // (Anfangswert fuer diesen Zeitschritt, der dann durch Handys
76      // veraendert werden kann.
77      for (i=0; i<schadstoffe.length; i++) {
78          for (j=0; j<schadstoffe[i].gelbNachrichten.length-1; j++)
79          schadstoffe[i].gelbNachrichten[j] = schadstoffe[i].gelbNachrichten[j+1];
80          schadstoffe[i].gelbNachrichten[schadstoffe[i].gelbNachrichten.length-1] = 0;
81      }
82      }
83
84      // Ueberpruefe, ob "groessere Anzahl" Gelb-Nachrichten in "kurzer Zeit"
            entstanden
85      void pruefeGelbs () {
86      for (i=0; i<this.schadstoffe.length; i++) {
87          summeGelbs = 0;
88          for (j=0; j<kurzeZeit; j++) {
89          summeGelbs += this.schadstoffe[i].gelbNachrichten[j];
90          }
91          if (summeGelbs > groessereAnzahl) {
92          // Rot an alle Handys
93          if (momentanEingeloggteHandys > 0)
94              for (k=0; k<momentanEingeloggteHandys; k++)
95              eingeloggteHandys[k].rotVonServer(schadstoffe[i]);
96          // Rot an alle Nachbarserver
97          if (nachbarServer.length > 0)
98              for (k=0; k<nachbarServer.length; k++)
99              if (nachbarServer[k] != null)
100                 nachbarServer[k].rotVonServerAnNachbar(schadstoffe[i]);
101         }
102     }
103     }
104
105
106     /* Server muss arbeiten */
107
108     // Gelb-Nachricht von einem Handy
109     public void gelbVonHandy (Schadstoff s, Handy absender) {
110     // merke Gelb-Nachricht
111     this.schadstoffe[s.index].gelbNachrichten[this.schadstoffe[s.index].
            gelbNachrichten.length-1] += 1;
112     // Sende Gelb-Nachricht an "einige" Handys
113     for (i=0; i<einige; i++) {
114         if (momentanEingeloggteHandys > 1) {
115         a = random.nextInt(momentanEingeloggteHandys-1);
116         // Gelb darf nur an andere Handys als Absender geschickt
117         // werden!
118         if (a != absender.index) {
119             eingeloggteHandys[a].einGelbVonServer(s);
120         }
121         }
122         else i--;
123     }
124     }
125
126     // Rot-Nachricht von anderem Server
127     void rotVonServerAnNachbar (Schadstoff s) {
128     // keine Angaben in Aufgabenstellung
129     ;
130     }
131 }
```

Schadstoff.java

```
1  public class Schadstoff {
2
3      // Name des Schadstoff
4      public String name;
```

```
5       // Index
6       public int index;
7       // Grenzwert
8       public double grenzwert;
9       // Aktueller Luftgehalt
10      public double aktuellerLuftgehalt;
11      // Speicher-Array der letzten Nachrichten
12      public int[] gelbNachrichten;
13      // Strategievariablen
14      int kurzeZeit;
15
16      // Konstruktor
17      public Schadstoff (String name, int index, double defaultGrenzwert, int
            kurzeZeit) {
18      this.name = name;
19      this.index = index;
20      this.grenzwert = defaultGrenzwert;
21      this.kurzeZeit = kurzeZeit;
22      gelbNachrichten = new int[kurzeZeit];
23      }
24  }
```

Keuch.java

```
1   import java.util.Random;
2   import java.awt.*;
3
4   public class Keuch {
5
6       public static void main (String[] args) {
7
8       // Hilfsvariablen
9       int i, j;
10
11      // Zufallsobjekt
12      Random random = new Random();
13
14      // Anzahl der Handys und Server
15      int anzahlHandys = 990;
16      int anzahlServer = 10;
17      // Maximale Anzahl gleichzeitig auf einem Server eingeloggter Handys.
18      int handyKapazitaet = 100;
19
20      /* Strategievariablen */
21      // Handy
22      int etliche = 300; // Warnung ab "etliche" Gelb-Nachrichten
23      // Server
24      int einige = 2; // Gelb an "einige" Handys weitersenden
25      int groessereAnzahl = 30; // Warnung ab "groessere Anzahl" Gelb-Nachrichten
26      // Handy und Server
27      int kurzeZeit = 20;
28
29      // Zeit zwischen Messungen der Schadstoffbelastung in ms
30      int schlafDauer = 10;
31
32      // Schadstoffnamen und default-Grenzwerte
33      String[] schadstoffNamen = new String[] {"Stickstoffoxide", "Ozon", "
            Kohlenmonoxid", "Schwefelmonoxid", "Blei","Schwebeteilchen", "Russ", "Benzol
            ", "Schwefeldioxid", "Stickstoffdioxid"};
34      double[] defaultGrenzwerte = new double
            [] {0.2,0.05,55.,13.,0.1,0.2,8.,10.,5.,9.};
35      int anzahlSchadstoffe = schadstoffNamen.length;
36
37      // Sonstiger Input
38      boolean zeigeAlleHandys = true;
39      int angezeigtesHandy = 1;
```

```
40      boolean zeigeServerlog = false;
41      boolean soundAn = false;
42
43      // Einlesen der Argumente aus der HTML Datei
44      if (args.length == 1 && args[0].equals("default")) {
45          // default, mache nichts
46          ;
47      }
48      else if (args.length != 12) {
49          System.out.println("\nKeuch Parametereingabe: java Keuch [anzahlHandys] [
                anzahlServer] [handyKapazitaet] [etliche] [einige] [groessereAnzahl] [
                kurzeZeit] [schlafDauer] [zeigeAlleHandys] [angezeigtesHandy] [
                zeigeServerlog] [soundAn]\n\nBeispiel: java Keuch
                 990 10 100 300 2 30 20 10 true 1 false false\n\nAlternativ: java Keuch
                default");
50          System.exit(0);
51      }
52      else {
53          System.out.println("anzahlHandys "+(anzahlHandys = Integer.parseInt(args[0])
                ));
54              System.out.println("anzahlServer "+(anzahlServer = Integer.parseInt(args
                    [1])));
55          System.out.println("handyKapazitaet "+(handyKapazitaet = Integer.parseInt(
                args[2])));
56          System.out.println("etliche "+(etliche = Integer.parseInt(args[3])));
57          System.out.println("einige "+(einige = Integer.parseInt(args[4])));
58          System.out.println("groessereAnzahl "+(groessereAnzahl = Integer.parseInt(
                args[5])));
59          System.out.println("kurzeZeit "+(kurzeZeit = Integer.parseInt(args[6])));
60          System.out.println("schlafDauer "+(schlafDauer = Integer.parseInt(args[7])))
                ;
61          if (args[8].equals("true"))
62          zeigeAlleHandys = true;
63          else
64          zeigeAlleHandys = false;
65          System.out.println("zeigeAlleHandys "+zeigeAlleHandys);
66          System.out.println("angezeigtesHandy "+(angezeigtesHandy = Integer.parseInt(
                args[9])));
67          if (args[10].equals("true"))
68          zeigeServerlog = true;
69          else
70          zeigeServerlog = false;
71          System.out.println("zeigeServerlog "+zeigeServerlog);
72          if (args[11].equals("true"))
73          soundAn = true;
74          else
75          soundAn = false;
76          System.out.println("soundAn "+soundAn);
77      }
78      System.out.println("\n************\nSTART\n************\n");
79
80
81      // Erzeuge Schadstoff Listen fuer Handys und Server
82      Schadstoff[][] handySchadstoffe = new Schadstoff[anzahlHandys][anzahlSchadstoffe
            ];
83      Schadstoff[][] serverSchadstoffe = new Schadstoff[anzahlServer][
            anzahlSchadstoffe];
84      for (i=0; i<anzahlSchadstoffe; i++) {
85          for (j=0; j<anzahlHandys; j++) {
86          handySchadstoffe[j][i] = new Schadstoff(schadstoffNamen[i],i,
                defaultGrenzwerte[i], kurzeZeit);
87          }
88          for (j=0; j<anzahlServer; j++) {
89          serverSchadstoffe[j][i] = new Schadstoff(schadstoffNamen[i],i,
                defaultGrenzwerte[i], kurzeZeit);
90          }
91      }
92
```

```
93      // Erzeuge Server Objekte
94      Server[] alleServer = new Server[anzahlServer];
95      for (i=0; i<anzahlServer; i++)
96          alleServer[i] = new Server(serverSchadstoffe[i], handyKapazitaet, einige,
                kurzeZeit, groessereAnzahl, i, random);
97
98      // Sage jedem Server seine Nachbarn
99      for (i=0; i<anzahlServer; i++) {
100         alleServer[i].nachbarServer = new Server[anzahlServer-1];
101         for (j=0; j<anzahlServer; j++) {
102         if (j<i)
103             alleServer[i].nachbarServer[j] = alleServer[j];
104         if (j>i)
105             alleServer[i].nachbarServer[j-1] = alleServer[j];
106         }
107     }
108
109     // Erzeuge Handy Objekte
110     Handy[] handys = new Handy[anzahlHandys];
111     for (i=0; i<anzahlHandys; i++)
112         handys[i] = new Handy(i, alleServer, handySchadstoffe[i], kurzeZeit, etliche
                , random);
113
114     // Timer
115     Thread me = Thread.currentThread();
116     me.setPriority(Thread.MIN_PRIORITY);
117     try {
118         while (true) {
119         System.out.println("\n********************\nNaechster Zeitschritt\n
                ********************\n");
120         me.sleep(schlafDauer);
121         // naechster Zeitschritt auf Server
122         for (i=0; i<alleServer.length; i++) {
123             alleServer[i].naechsterZeitschritt(zeigeServerlog);
124         }
125         // und im Handy
126         for (i=0; i<handys.length; i++) {
127             handys[i].naechsterZeitschritt(zeigeAlleHandys, angezeigtesHandy,
                    alleServer, soundAn);
128         }
129         }
130     }
131     catch (InterruptedException e) {
132         return;
133     }
134     }
135 }
```

4 Flitz

Wie in der Aufgabe gefordert, wurde ein Fisch mittels eines im Programm einstellbaren Kreises realisiert. Da diese lieblosen Körper leider nur recht funktionell die Bewegungen der eigentlich hochentwickelten, putzigen Fische widerspiegeln, soll durch die folgende Abbildung ein wenig kaschiert werden.[1] Es kann nicht gesagt werden, dass der verwendete Algorithmus den schnellsten Weg zur optimalen Endsituation liefert; er soll vielmehr das soziale Verhalten der Fische möglichst realitätsgetreu abbilden und den angestrebten Demonstrationseffekt abbilden. Als Mindestvoraussetzung wird eine aktuelle Javaversion benötigt. Der Rechner sollte jedoch zudem mindestens über eine 1,8 GHz-CPU (getestet wurden sowohl Intel- als auch AMD-Prozessoren) und 256 MB Ram, besser 512 MB, verfügen, damit die Animation ansprechend schnell läuft.

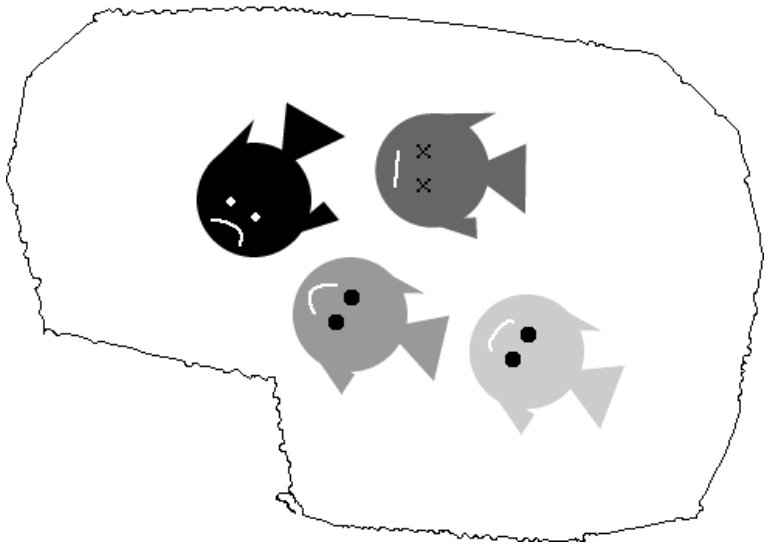

Abbildung 1: Im virtuellen Fischteich geht es munter zu ...

4.1 Das soziale Verhalten

Das soziale Verhalten der Fische soll durch die Bewegung dieser modelliert werden. Hierzu soll ein Fisch-Objekt folgende Membervariablen und -funktionen besitzen.

- `int x, y` Koordinaten des Fisches
- `int[] freunde, feinde` Liste der Feinde, die der Fisch mag bzw. nicht mag.
- `Color farbe` Farbe des Fischs.
- `void schwimme (int xZiel, int yZiel) {`

[1] Man beachte den enormen Wusel-Faktor des Programms, der sich anschickt Spiele wie „Die Siedler“ zu überragen!

```
            x = xZiel;
            y = yZiel;
    }
```

Nun zur zeitlichen Bewegung der Fische. Es soll jeder Fisch gleich schnell sein, d.h. jeder Fisch kann innerhalb eines Zeitschrittes nur in eines seiner Nachbarkästchen schwimmen: Ein Fisch kann also $(0, 0)$ oder $(1, 0)$ oder $(1, 1)$ oder $(0, 1)$ oder $(-1, 1)$ oder $(-1, 0)$

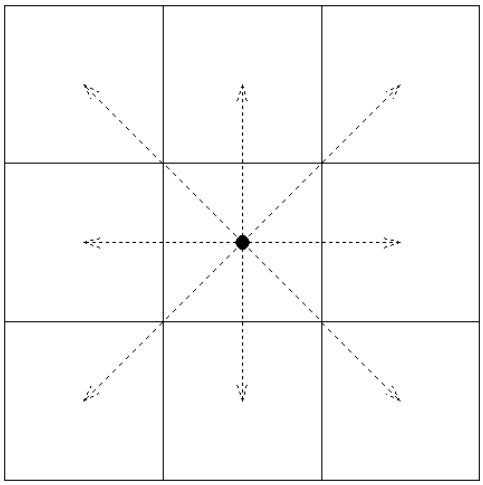

Abbildung 2: Bewegungsmöglichkeiten eines Fisches.

oder $(-1, -1)$ oder $(0, -1)$ oder $(1, -1)$ schwimmen. Welche dieser neun Möglichkeiten ein Fisch wählt, ist abhängig von den Positionen der Fische, die er meidet, und den Positionen der Fische, zu denen er sich angezogen fühlt. Diese Abhängigkeit soll darin bestehen, dass eine Art *durchschnittliche Schwimmrichtung* bestimmt wird, d.h. wenn ein Fisch zwei Freunde in der Richtung $(1, 0)$ und $(0, 1)$ besitzt, wird er im nächsten Zeitschritt nach $(1, 1)$ schwimmen. Somit bevorzugt es der Fisch in einen Bereich mit möglichst vielen Freunden zu schwimmen, als sich lediglich zu einem einzelnen Freund zu begeben. Diese Modellierung hat folgende Vorteile. Auch wenn die Goldfische „ihre Eigenheiten“ haben, kann angenommen werden, dass sie sich am wohlsten fühlen, wenn sie sich in der Nähe möglichst vieler Freunde aufhalten. Schwimmt jeder Fisch immer in seine *durchschnittliche Schwimmrichtung*, so finden sich recht schnell Freundepaare und -gruppen, d.h. Fische, die sich alle gegenseitig zueinander hingezogen fühlen. Auf der anderen Seite stoßen sich Feinde ab, wenn bei der *durchschnittlichen Schwimmrichtung* ein Feind in Richtung (x, y) wie ein Freund in Richtung $(-x, -y)$ gezählt wird. Wenn also beispielsweise ein Fisch einen Freund in der Richtung $(0, 1)$ und einen Feind in der Richtung $(1, 0)$ hat, schwimmt er nach $(-1, 1)$.

Die Formel für die *durchschnittliche Schwimmrichtung* erhält man wie folgt. Die x-Komponente $\Delta x_{\rightarrow}$ der neuen Schwimmrichtung ist das arithmetische Mittel der x-Komponenten der Abstände zu den anderen Fischen, wobei die x-Komponenten der Feinde mit -1 multipliziert werden. Also

$$\Delta x_{\rightarrow} = \frac{(\Delta x_{A_1} + \Delta x_{A_2} + \cdots + \Delta x_{A_{|A|}}) + (\Delta x_{B_1} + \Delta x_{B_2} + \cdots + \Delta x_{B_{|B|}}) \cdot (-1)}{|A| + |B|}, \quad (4)$$

wenn A Freunde und B Feinde sind und $|a|$ bzw. $|b|$ für die Anzahl der Freunde bzw. Feinde steht. Selbiges gilt natürlich auch für die y-Koordinate

$$\Delta y_{\rightarrow} = \frac{(\Delta y_{A_1} + \Delta y_{A_2} + \cdots + \Delta y_{A_{|A|}}) + (\Delta y_{B_1} + \Delta y_{B_2} + \cdots + \Delta y_{B_{|B|}}) \cdot (-1)}{|A| + |B|}. \quad (5)$$

Weil ein Fisch laut Abbildung 5 nur in benachbarte Felder schwimmen soll, wird die endgültige Richtung $(\Delta x^*_{\rightarrow}, \Delta y^*_{\rightarrow})$ durch den Quotienten $\Delta x_{\rightarrow}/\Delta y_{\rightarrow}$ und die Vorzeichen von $\Delta x_{\rightarrow}$ und $\Delta y_{\rightarrow}$ bestimmt. (Vgl. die Funktion `welchesKaestchen` im Quelltext.)

Hiermit wollen wir uns aber noch nicht zufrieden geben, denn es ist für einen Fisch sicherlich sinnvoll die Abstände der anderen Fische mit in seine Richtungsentscheidung einfließen zu lassen. Diese Abhängigkeit vom Abstand $d = \sqrt{\Delta x^2 + \Delta y^2}$ zu den anderen Fischen wird durch den Faktor $\frac{1}{d}$ vor jeder Richtungskomponente umgesetzt, d.h. je näher ein fremder Fisch ist, desto größer ist dessen Einwirkung auf die Richtungsentscheidung eines Fisches. Den einzelnen Richtungskomponenten werden also die Prioritäten $\frac{1}{d}$ zugewiesen. Setzt man dies in die Gleichung (4) und (5) ein, so erhält man für die x-Koordinate

$$\Delta x_{\rightarrow} = \frac{\left(\frac{\Delta x_{A_1}}{d_{A_1}} + \frac{\Delta x_{A_2}}{d_{A_2}} + \cdots + \frac{\Delta x_{A_{|A|}}}{d_{A_{|A|}}}\right) - \left(\frac{\Delta x_{B_1}}{d_{B_1}} + \frac{\Delta x_{B_2}}{d_{B_2}} + \cdots + \frac{\Delta x_{B_{|B|}}}{d_{B_{|B|}}}\right)}{\left(\frac{1}{d_{A_1}} + \frac{1}{d_{A_2}} + \cdots \frac{1}{d_{A_{|A|}}}\right) + \left(\frac{1}{d_{B_1}} + \frac{1}{d_{B_2}} + \cdots \frac{1}{d_{B_{|B|}}}\right)} \quad (6)$$

und für die y-Koordinate

$$\Delta y_{\rightarrow} = \frac{\left(\frac{\Delta y_{A_1}}{d_{A_1}} + \frac{\Delta y_{A_2}}{d_{A_2}} + \cdots + \frac{\Delta y_{A_{|A|}}}{d_{A_{|A|}}}\right) - \left(\frac{\Delta y_{B_1}}{d_{B_1}} + \frac{\Delta y_{B_2}}{d_{B_2}} + \cdots + \frac{\Delta y_{B_{|B|}}}{d_{B_{|B|}}}\right)}{\left(\frac{1}{d_{A_1}} + \frac{1}{d_{A_2}} + \cdots \frac{1}{d_{A_{|A|}}}\right) + \left(\frac{1}{d_{B_1}} + \frac{1}{d_{B_2}} + \cdots \frac{1}{d_{B_{|B|}}}\right)}. \quad (7)$$

Durch Zuschaltung der Option „Privatsphäre“ kann ein Übereinnanderschwimmen der Fische verhindert werden. Jeder Fisch hat eine Art Box um sich, an die er keine anderen Fische heranlässt.

Wenn sehr viele Fische die Privatsphaehren anderer Fische verletzen, schwimmen ein paar zufällig ausgewählte Fische paranoid ($\hat{=}$ völlig planlosem, vom Algorithmus befreitem Umherschwimmen) durch die Gegend, so dass weiterhin Bewegung im Teich herrscht und das Wasser zirkuliert.

Zuletzt soll noch kurz auf die Randbedingung des Teiches eingegangen werden. Weil ein Fisch seinen Teich nicht verlassen kann, wird er, wenn er versucht den Teich zu verlassen, wieder in den Teich „zurückgeschmissen“, d.h. seine Δx- und Δy-Schwimmkoordinaten werden so korrigiert, dass die Koordinaten (x, y) eines Fisches die Ränder nicht überschreiten.

4.2 Die drei Aufgaben

Bei Aufgabe Eins lässt sich bei geringer eingestellter Fischmenge oft sehr schön beobachten, wie alle Fische in eine wilde Hetzjagd auf den „traurigen Goldfisch“ verfallen,

der sich daraufhin am Rand entlang vor ihnen zu flüchten versucht. Es ist jedoch nicht garantiert, dass dies geschieht, da die Fische untereinander trotz ihrer gemeinsamen Zuneigung zu dem als Viereck dargestellten Einzelgänger immer noch gewisse Zu- und Abneigungen hegen. Bei jedem Durchlauf aber ist der Wunsch des „traurigen Goldfisches“ zu sehen, sich abzusondern. Er befindet sich deshalb meist in Randregionen des Teiches.

Abbildung 3: Drei Fische rücken unaufhaltsam näher ...

In Aufgabe zwei treffen sich alle Fische – bei ausgeschalteter Privatsphäre – unweigerlich irgendwann in einem Punkt. Da das Verhältnis der Zuneigung das der Abneigung übertrifft, werden die Fische zu dem „Treffpunkt“ auch dann scheinbar magisch angezogen, wenn sich Goldfische darauf befinden, zu denen sie ein feindschaftliches Verhältnis haben. Wird die Privatsphäre aktiviert, herrscht ein aufgeregtes Treiben um den Mittelpunkt: Die Tierchen positionieren sich, bedingt durch die Paranoiden ständig neu.

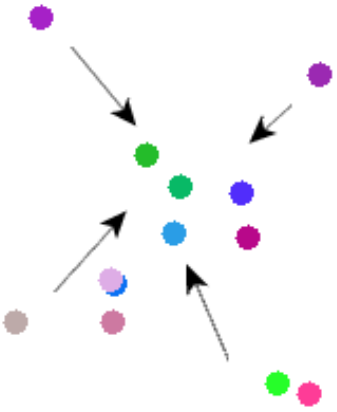

Abbildung 4: Die Fisch treffen sich

Ein dazu gegenteiliger Effekt ist für die letzte Aufgabe zu beobachten gewesen: Hier finden sich kleiner Gruppe von Fischen am Rand zusammen, und das Zentrum wird generell gemieden. Die Betrachtung ist besonders bei großer Fischmenge und aktivierter Privatsphäre empfehlenswert, da sich sonst die Fische zu Gruppen am Rand zusammenschließen und möglicherweise nichts geschieht.

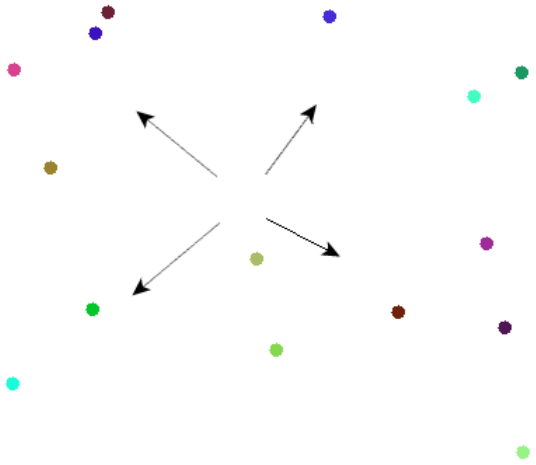

Abbildung 5: Positionierungsbewegungen zum Rand hin

4.3 Quellcode und Programmhinweis

Die Visualisierung der Fischbewegung wird mittels Java implementiert. Hierbei wird auf das Applet JavaPsi[2] zurückgegriffen, das bereits einige nützliche Voraussetzungen für die grafische Umsetzung enthält. Die endgültige Implementierung der Visualisierung wird im Folgenden aufgelistet.

Fisch.java

```
1  public class Fisch {
2
3      // Eigene Koordinaten x und y
4      public int x;
5      public int y;
6
7      // Eigene Koordinaten x und y zur Zwischenspeicherung
8      public int xAlt;
9      public int yAlt;
10
11     // Liste aller Freunde und Feinde
12     public int[] freunde;
13     public int[] feinde;
14
15     // Eigene Farbe
16     public java.awt.Color farbe;
17
18     // Schwimme zum Punkt (xZiel, yZiel)
19     public void schwimme (int xZiel, int yZiel) {
20     x = xZiel;
21     y = yZiel;
22     }
23
24 }
```

[2]JavaPsi - Zur Quantenmechanik am Computer © 2003 Marcel Schmittfull.
http://javapsi.sf.net/

Flitz.java

```
/** Flitz.java (C) 2003 by Marcel Schmittfull and Moritz Beller **/

// Ben{\"o}tigte externe Klassen
import java.applet.Applet;
import java.awt.*;
import java.awt.image.MemoryImageSource;
import java.awt.event.*;
import java.lang.Math;
import java.util.Random;


/************** GRAFIK **************/

// Die grafische Oberflaeche wird {\"u}ber ein selbstgestaltetes Canvas Objekt
// realisiert.
class GCanvas extends Canvas {
    GFrame frame;
    GCanvas (GFrame f) {
    frame = f;
    }
    public void update (Graphics g) {
    frame.updateFlitz(g);
    }
    public void paint (Graphics g) {
    update(g);
    frame.updateFlitz(g);
    }
};

// Entwurf der grafischen Oberflaeche
class GLayout implements LayoutManager {
    public GLayout () {}
    public void addLayoutComponent (String name, Component comp) {}
    public void removeLayoutComponent (Component c) {}
    public Dimension preferredLayoutSize (Container target) {
    return new Dimension(500,500);
    }
    public Dimension minimumLayoutSize (Container target) {
    return new Dimension(400,400);
    }
    public void layoutContainer (Container target) {
    int barWidth = 0;
    int i;
    for (i = 1; i < target.getComponentCount(); i++) {
        Component m = target.getComponent(i);
        if (m.isVisible()) {
        Dimension d = m.getPreferredSize();
        if (d.width > barWidth)
            barWidth = d.width;
        }
    }
    Insets insets = target.getInsets();
    int targetw = target.getSize().width - insets.left - insets.right;
    int cw = targetw-barWidth;
    int targeth = target.getSize().height - (insets.top+insets.bottom);
    target.getComponent(0).setLocation(insets.left, insets.top);
    target.getComponent(0).setSize(cw, targeth);
    cw += insets.left;
    int h = insets.top;
    for (i=1; i<target.getComponentCount(); i++) {
        Component m = target.getComponent(i);
        if (m.isVisible()) {
        Dimension d = m.getPreferredSize();
        if (m instanceof Scrollbar || m instanceof DecentScrollbar)
            d.width = barWidth;
        if (m instanceof Label) {
```

```
67                  h += d.height/5;
68                  d.width = barWidth;
69              }
70              m.setLocation(cw, h);
71              m.setSize(d.width, d.height);
72              h += d.height;
73              }
74      }
75      }
76  };
77
78  // Applet
79  public class Flitz extends Applet {
80      GFrame frame;
81      void destroyFrame() {
82      if (frame != null)
83          frame.dispose();
84      frame = null;
85      }
86      public void init() {
87      System.out.println("\"Flitz\": Aufgabe 3, 22. Bundeswettbewerb Informatik\n\
            nCopyright (C) 2003 by\n\tMarcel Schmittfull (marcel-sl@gmx.de)\n\tMoritz
            Beller (momo@4momo.de)\n");
88      frame = new GFrame(this);
89      frame.init();
90      }
91      public void destroy() {
92      if (frame != null)
93          frame.dispose();
94      frame = null;
95      }
96  };
97
98  class GFrame
99      extends Frame
100     implements ComponentListener, ActionListener, MouseMotionListener, MouseListener
            , ItemListener, DecentScrollbarListener {
101
102     public GFrame (Flitz a) {}
103
104     public String getAppletInfo() {
105     return "Flitz (C) 2003 by Marcel Schmittfull  -- http://javapsi.sourceforge.net
            /";
106     }
107
108     /* Grafikvariablen */
109     Thread engine = null;
110     Dimension winSize;
111     Image dbimage;
112     Image animImage;
113
114     Checkbox nextCheck;
115     Checkbox privatsphaereCheck;
116     Choice teilaufgabeChoice;
117     Button neuButton;
118     DecentScrollbar speedBar, anzahlFischeBar, fischRadiusBar, anteilFreundeBar;
119     Label speedLabel, anzahlFischeLabel, fischRadiusLabel, anteilFreundeLabel;
120     int pixels[];
121     MemoryImageSource source;
122     int pause;
123
124     GCanvas canvas;
125     Flitz applet;
126     Dimension d;
127     Graphics g2;
128     int toth;
129
130     // Variabeln f{\"u}r den Algorithmus
```

```
131     Random random;
132     int i, j, k, l, m, a, b, c, rand, ausnahmeFisch;
133     boolean nextFrame, zuNah, privatsphaere;
134     int summeAllerZuNahs;
135     int anzahlFische;
136     int fischRadius;
137     int xSchwimm, ySchwimm;
138     double xZaehler, yZaehler, nenner;
139     Fisch[] fische;
140     int[] restlicheIndizes;
141     double anteilFreunde, anteilFeinde, anzahlFreunde;
142     boolean inited;
143 
144     static final double pi = 3.14159265358979323846;
145 
146     public void paint (Graphics g) {
147     canvas.repaint();
148     }
149 
150     // Etwas Benutzerinteraktion
151     public void componentHidden (ComponentEvent e){}
152     public void componentMoved (ComponentEvent e){}
153     public void componentShown (ComponentEvent e) {}
154     public void componentResized (ComponentEvent e) {}
155     public void actionPerformed (ActionEvent e) {
156     if (e.getSource() == neuButton) {
157         initWerte();
158         canvas.repaint();
159     }
160 
161     }
162     public void itemStateChanged (ItemEvent e) {
163     if (e.getSource() == nextCheck)
164         canvas.repaint();
165     if (e.getSource() == privatsphaereCheck)
166         canvas.repaint();
167     if (e.getItemSelectable() == teilaufgabeChoice) {
168         initWerte();
169         canvas.repaint();
170     }
171     }
172     public void scrollbarValueChanged (DecentScrollbar ds) {
173     if (ds == speedBar)
174         canvas.repaint();
175     if (ds == anzahlFischeBar) {
176         initWerte();
177         canvas.repaint();
178     }
179     if (ds == fischRadiusBar) {
180         fischRadius = (int) (fischRadiusBar.getValue()/4);
181         canvas.repaint();
182     }
183     if (ds == anteilFreundeBar) {
184         if (teilaufgabeChoice.getSelectedIndex() == 3) {
185         initWerte();
186         canvas.repaint();
187         }
188     }
189     }
190     public void scrollbarFinished (DecentScrollbar ds) {
191     if (ds == speedBar)
192         canvas.repaint();
193     }
194     public boolean processEvent (Event ev) {
195     if (ev.id == Event.WINDOW_DESTROY) {
196         applet.destroyFrame();
197         return true;
198     }
```

```
199 //  return super.processEvent(ev);
200     return processEvent(ev);
201     }
202     MenuItem getMenuItem(String s) {
203     MenuItem mi = new MenuItem(s);
204     mi.addActionListener(this);
205     return mi;
206     }
207     CheckboxMenuItem getCheckItem(String s) {
208     CheckboxMenuItem mi = new CheckboxMenuItem(s);
209     mi.addItemListener(this);
210     return mi;
211     }
212     public void mouseClicked (MouseEvent e) { }
213     public void mouseEntered (MouseEvent e) { }
214     public void mouseExited (MouseEvent e) { }
215     public void mousePressed (MouseEvent e) { }
216     public void mouseReleased (MouseEvent e) { }
217     public void mouseDragged (MouseEvent e) { }
218     public void mouseMoved (MouseEvent e) { }
219
220     /* Initialisiere Grafik */
221     public void init () {
222     setLayout(new GLayout());
223     canvas = new GCanvas(this);
224     canvas.addComponentListener(this);
225     canvas.addMouseMotionListener(this);
226     canvas.addMouseListener(this);
227     add(canvas);
228
229     teilaufgabeChoice = new Choice();
230     teilaufgabeChoice.add("Teilaufgabe 1");
231     teilaufgabeChoice.add("Tailaufgabe 2");
232     teilaufgabeChoice.add("Teilaufgabe 3");
233     teilaufgabeChoice.add("Anteil Freunde per Schieberegler");
234     teilaufgabeChoice.addItemListener(this);
235     add(teilaufgabeChoice);
236
237     nextCheck = new Checkbox("Animation an");
238     nextCheck.addItemListener(this);
239     nextCheck.setState(true);
240     add(nextCheck);
241     privatsphaereCheck = new Checkbox("Privatsphaehre an");
242     privatsphaereCheck.addItemListener(this);
243     privatsphaereCheck.setState(true);
244     add(privatsphaereCheck);
245
246     add(neuButton = new Button("Neu"));
247     neuButton.addActionListener(this);
248
249     add(speedLabel = new Label("Simulationsgeschwindigkeit", Label.CENTER));
250     add(speedBar = new DecentScrollbar(this, 15, 1, 100));
251     add(anzahlFischeLabel = new Label("Anzahl Fische", Label.CENTER));
252     add(anzahlFischeBar = new DecentScrollbar(this, 30, 1, 100));
253     add(fischRadiusLabel = new Label("Fisch Radius", Label.CENTER));
254     add(fischRadiusBar = new DecentScrollbar(this, 40, 1, 100));
255     add(anteilFreundeLabel = new Label("Anteil Freunde", Label.CENTER));
256     add(anteilFreundeBar = new DecentScrollbar(this, 40, 1, 100));
257
258     setSize(600, 400);
259     canvas.setBackground(Color.black);
260     canvas.setForeground(Color.lightGray);
261     setTitle("Flitz - 22. Bundeswettbewerb Informatik");
262
263     show();
264
265     try {
266         String param = applet.getParameter("PAUSE");
```

```
267         if (param != null)
268         pause = Integer.parseInt(param);
269     } catch (Exception e) { }
270
271     Color gray1 = new Color(76,  76,  76);
272     Color gray2 = new Color(127, 127, 127);
273
274     random = new Random();
275     initWerte();
276     }
277
278     /*************** Initialisiere Anfangswerte fuer Algorithmus **************/
279     public void initWerte () {
280
281     // Fischgr{\"o}{\ss}e
282     fischRadius = (int) (fischRadiusBar.getValue()/4);
283
284     // Anzahl der Fische
285     anzahlFische = (int) (anzahlFischeBar.getValue()/2);
286
287     // Erzeuge Fischobjekte
288     fische = new Fisch[anzahlFische];
289     for (int i = 0; i < anzahlFische; i++) {
290         fische[i] = new Fisch();
291     }
292     restlicheIndizes = new int[anzahlFische];
293
294     // Freunde und Feinde
295     if (teilaufgabeChoice.getSelectedIndex() == 0) {
296         // Verhaeltis Freunde zu Feinde
297         anteilFreunde = random.nextDouble();
298         anteilFeinde = 1-anteilFreunde;
299     }
300
301     if (teilaufgabeChoice.getSelectedIndex() == 1) {
302         // Verhaeltis Freunde zu Feinde
303         anteilFreunde = 0.7;
304         anteilFeinde = 1-anteilFreunde;
305     }
306     if (teilaufgabeChoice.getSelectedIndex() == 2) {
307         // Verhaeltis Freunde zu Feinde
308         anteilFreunde = 0.3;
309         anteilFeinde = 1-anteilFreunde;
310     }
311
312     if (teilaufgabeChoice.getSelectedIndex() == 3) {
313         anteilFreunde = (int) (anteilFreundeBar.getValue()/100);
314         anteilFeinde = 1-anteilFreunde;
315     }
316
317     // Erstelle Freunde und Feinde Arrays
318     for (i=0; i<anzahlFische; i++) {
319         // Mit 50%iger Wahrscheinlickeit wird abgerundet, sonst aufgerundet
320         if (random.nextInt(1) == 0)
321         anzahlFreunde = Math.floor(anteilFreunde*anzahlFische);
322         else
323         anzahlFreunde = Math.ceil(anteilFreunde*anzahlFische);
324         fische[i].freunde = new int[(int) (anzahlFreunde)];
325         fische[i].feinde = new int[anzahlFische-(int) (anzahlFreunde)];
326     }
327     // Zufaellige Reihenfolge
328     for (i=0; i<anzahlFische; i++) {
329         // restliche Indizes
330         for (j=0; j<anzahlFische; j++)
331         restlicheIndizes[j] = j;
332         for (j=0; j<anzahlFische; j++) {
333         rand = random.nextInt(anzahlFische-j);
334         if (j<fische[i].freunde.length)
```

```
335             fische[i].freunde[j] = restlicheIndizes[rand];
336         else
337             fische[i].feinde[j-fische[i].freunde.length] = restlicheIndizes[rand];
338         // loesche Index j aus restlicheIndizes
339         for (k=rand; k<restlicheIndizes.length-1; k++) {
340             restlicheIndizes[k] = restlicheIndizes[k+1];
341         }
342         }
343     }
344 
345     if (teilaufgabeChoice.getSelectedIndex() == 0) {
346         // Fisch 0, der alle hasst und von allen geliebt wird
347         // hasst alle
348         fische[0].freunde = new int[] { };
349         fische[0].feinde = new int[anzahlFische];
350         for (i=1; i<anzahlFische; i++)
351         fische[0].feinde[i] = i;
352         // wird von allen geliebt
353         // Zufaellige Reihenfolge
354         for (i=1; i<anzahlFische; i++) {
355         // restliche Indizes
356         fische[0].feinde[0] = 0;
357         for (j=1; j<anzahlFische; j++)
358             restlicheIndizes[j] = j;
359         for (j=1; j<anzahlFische; j++) {
360             rand = random.nextInt(anzahlFische-j+1);
361             if (j<fische[i].freunde.length)
362             fische[i].freunde[j] = restlicheIndizes[rand];
363             else
364             fische[i].feinde[j-fische[i].freunde.length] = restlicheIndizes[rand];
365             // loesche Index j aus restlicheIndizes
366             for (k=rand; k<restlicheIndizes.length-1; k++) {
367             restlicheIndizes[k] = restlicheIndizes[k+1];
368             }
369         }
370         }
371     }
372 
373     // Fische schwimmen zu zufaelligen Koordinaten
374     for (i=0; i<anzahlFische; i++) {
375         if (canvas.getSize().width>0 && canvas.getSize().height>0)
376         fische[i].schwimme(random.nextInt(canvas.getSize().width), random.nextInt(
                canvas.getSize().height));
377         else
378         fische[i].schwimme(random.nextInt(100), random.nextInt(100));
379     }
380 
381     // Farben fuer Fische (per Zufall zugewiesen)
382     for (i=0; i<anzahlFische; i++) {
383         fische[i].farbe = new Color(random.nextInt(255), random.nextInt(255), random
                .nextInt(255));
384     }
385     inited = true;
386     }
387 
388     /**** Berechne naechsten Zeitschritt, update ****/
389     public void updateFlitz (Graphics realg) {
390 
391     if (! inited) {
392         Thread me1 = Thread.currentThread();
393         me1.setPriority(Thread.MIN_PRIORITY);
394         try {
395         me1.sleep(500);
396         }
397         catch (InterruptedException e) {
398         return;
399         }
400     }
```

```
401
402     privatsphaere = privatsphaereCheck.getState();
403
404     /* Zunaechst etwas Grafik... */
405         d = winSize = canvas.getSize();
406     if (winSize.width == 0) {
407         System.out.println("win width is 0 !!");
408         return;
409     }
410     toth = winSize.height;
411     dbimage = createImage(d.width, d.height);
412     g2 = dbimage.getGraphics();
413     if (winSize == null || winSize.width == 0) {
414         System.out.println("win width is 0 !!");
415         return;
416     }
417     g2.setColor(canvas.getBackground());
418     g2.fillRect(0, 0, winSize.width, winSize.height);
419     g2.setColor(canvas.getForeground());
420
421
422     /************** Beginn Algorithmus **************/
423     // Speichere alte Positionen
424     for (i=0; i<anzahlFische; i++) {
425         fische[i].xAlt = fische[i].x;
426         fische[i].yAlt = fische[i].y;
427     }
428     summeAllerZuNahs = 0;
429
430     // Berechne naechsten Schwimmschritt f{\"u}r alle Fische nach Formel (3)
431     // und (4) in der Dokumentation
432     for (i=0; i<anzahlFische; i++) {
433         xSchwimm = ySchwimm = 0;
434         xZaehler = yZaehler = nenner = 0.;
435         for (j=0; j<anzahlFische; j++) {
436         if (i != j && abstand(fische[i], fische[j]) != 0) {
437             for (k=0; k<fische[i].freunde.length; k++) {
438             if (fische[i].freunde[k] == j) {
439                 xZaehler -= abstandX(fische[i], fische[j]) / abstand(fische[i],
                        fische[j]);
440                 yZaehler -= abstandY(fische[i], fische[j]) / abstand(fische[i],
                        fische[j]);
441                 break;
442             }
443             }
444             for (k=0; k<fische[i].feinde.length; k++) {
445             if (fische[i].feinde[k] == j) {
446                 xZaehler += abstandX(fische[i], fische[j]) / abstand(fische[i],
                        fische[j]);
447                 yZaehler += abstandY(fische[i], fische[j]) / abstand(fische[i],
                        fische[j]);
448                 break;
449             }
450             }
451             nenner += 1./abstand(fische[i], fische[j]);
452         }
453         }
454         xSchwimm = welchesKaestchen(xZaehler/nenner, yZaehler/nenner, true);
455         ySchwimm = welchesKaestchen(xZaehler/nenner, yZaehler/nenner, false);
456         // schwimme zu neuer Position
457         fische[i].schwimme(fische[i].xAlt + xSchwimm, fische[i].yAlt + ySchwimm);
458
459         // Privatsphaehre der Fische
460         if (privatsphaere) {
461         zuNah = false;
462         for (j=0; j<anzahlFische; j++) {
463             if (i!=j && Math.sqrt((fische[i].x-fische[j].xAlt)*(fische[i].x-fische[j
                    ].xAlt)+(fische[i].y-fische[j].yAlt)*(fische[i].y-fische[j].yAlt))
```

```
                < fischRadius) {
464             zuNah = true;
465             break;
466             }
467         }
468         }
469
470         // Falls Privatsphaehre verletzt
471         if (privatsphaere && zuNah) {
472         // schwimme wieder zurueck
473         fische[i].schwimme(fische[i].x - xSchwimm, fische[i].y - ySchwimm);
474         // addiere 1 zur summeAllerZuNahs
475         summeAllerZuNahs += 1;
476         }
477
478         // Raender
479         if (random.nextInt(1) == 0)
480         rand = -1;
481         else
482         rand = 1;
483         if (fische[i].x < 0+fischRadius)
484         fische[i].schwimme(fische[i].x+2,fische[i].y+rand);
485         if (fische[i].y < 0+fischRadius)
486         fische[i].schwimme(fische[i].x+rand,fische[i].y+2);
487         if (fische[i].x > winSize.width-fischRadius)
488         fische[i].schwimme(fische[i].x-2,fische[i].y+rand);
489         if (fische[i].y > winSize.height-fischRadius)
490         fische[i].schwimme(fische[i].x+rand,fische[i].y-2);
491     }
492
493     // Wenn sehr viele Fische die Privatsphaehren anderer Fische verletzen,
494     // schwimmen ein paar Fische paranoid durch die Gegend, sodass
495     // weiterhin Bewegung im Teich herrscht und das Wasser zirkuliert.
496     if (summeAllerZuNahs > anzahlFische*0.9) {
497         if (false) {
498         // bestimme paranoiden Fisch
499         a = random.nextInt(anzahlFische-1);
500         // Plus oder Minus Bewegung
501         b = (int) (Math.pow(-1,random.nextInt(100))*fischRadius);
502         // c=1 -> x Bewegung     c=2 -> y Bewegung
503         c = random.nextInt(1)+1;
504         for (i=0; i<10; i++) {
505             if (! privatsphaereWirdVerletzt(fische[a], (c%2)*b, ((c%2)-1)*(-b))) {
506             fische[a].x += (c%2)*b;
507             fische[a].y += ((c%2)-1)*(-b);
508             }
509         }
510         }
511         else {
512         fische[random.nextInt(anzahlFische-1)].x += Math.pow(-1,random.nextInt(100))
                *4;
513         fische[random.nextInt(anzahlFische-1)].y += Math.pow(-1,random.nextInt(100))
                *4;
514         fische[random.nextInt(anzahlFische-1)].x += Math.pow(-1,random.nextInt(100))
                *4;
515         fische[random.nextInt(anzahlFische-1)].y += Math.pow(-1,random.nextInt(100))
                *4;
516         }
517     }
518
519     // Male Fische
520     for (i=0; i<anzahlFische; i++) {
521         g2.setColor(fische[i].farbe);
522         if (teilaufgabeChoice.getSelectedIndex()==0 && i==0)
523         g2.fillRect(fische[i].x-fischRadius/2,fische[i].y-fischRadius/2,fischRadius,
                fischRadius);
524         else
```

```
525         g2.fillOval(fische[i].x-fischRadius/2,fische[i].y-fischRadius/2,fischRadius,
                fischRadius);
526     }
527
528     // Animation
529     realg.drawImage(dbimage, 0, 0, this);
530     if (nextCheck.getState()) {
531         Thread me = Thread.currentThread();
532         me.setPriority(Thread.MIN_PRIORITY);
533         try {
534         if (true) {
535             if (speedBar.getValue() < 4) {
536             me.sleep(15000/(4*4*4));
537             }
538             else if (speedBar.getValue() > 90) {
539             me.sleep(0);
540             }
541             else {
542             me.sleep(300000/(speedBar.getValue()*speedBar.getValue()*speedBar.
                    getValue()));
543             // me.sleep((int) (java.lang.Math.exp(20./speedBar.getValue())*(.1/5)));
544             }
545         }
546         else
547             me.sleep(0);
548         } catch( InterruptedException e ) {
549         return;
550         }
551         canvas.repaint();
552     }
553
554     }
555
556     private double abstand (Fisch a, Fisch b) {
557     return Math.sqrt((a.xAlt-b.xAlt)*(a.xAlt-b.xAlt)+(a.yAlt-b.yAlt)*(a.yAlt-b.yAlt)
            );
558     }
559
560     private int abstandX (Fisch a, Fisch b) {
561     return a.xAlt-b.xAlt;
562     }
563
564     private int abstandY (Fisch a, Fisch b) {
565     return a.yAlt-b.yAlt;
566     }
567
568     private int welchesKaestchen (double xin, double yin, boolean returnX) {
569     // Achtung: In Java ist Origo links oben, d.h. x nach rechts positiv
570     // und y nach unten positiv!
571     int xout, yout;
572     if (Math.abs(xin/yin) >= Math.atan(pi/8.) && Math.abs(xin/yin) <= Math.atan(3.*
            pi/8.)) {
573         xout = signum(xin);
574         yout = signum(yin);
575     }
576     else if (Math.abs(xin) >= Math.abs(yin)) {
577         xout = signum(xin);
578         yout = 0;
579     }
580     else if (Math.abs(xin) <= Math.abs(yin)) {
581         xout = 0;
582         yout = signum(yin);
583     }
584     else {
585         System.out.println("Kein K{\"a}stchen gefunden! x"+xin+" y"+yin);
586         xout = 0;
587         yout = 0;
588     }
```

```
589     if (returnX)
590         return xout;
591     else
592         return yout;
593     }
594 
595     private int signum (double a) {
596     if (a >= 0) return 1;
597     else return -1;
598     }
599 
600     private boolean privatsphaereWirdVerletzt (Fisch a, int xSchwimm, int ySchwimm)
            {
601     for (j=0; j<anzahlFische; j++) {
602         if (i!=j && Math.sqrt((a.xAlt+xSchwimm-fische[j].xAlt)*(a.xAlt-fische[j].
                xAlt)+(a.yAlt+ySchwimm-fische[j].yAlt)*(a.yAlt+ySchwimm-fische[j].yAlt))
                 < fischRadius) {
603         return true;
604         }
605     }
606     return false;
607     }
608 };
```

5 Babbel

Diese Aufgabe setzt eine sehr gute Beherrschung der Muttersprache und vor allem des Englischen voraus. Um sie wirklich angemessen beantworten zu können, ist de facto ein Sprachwissenschaftler zu Rate zu ziehen. (Bei ausreichend tiefer Betrachtung würde der Stoff sicher auch für eine Doktorarbeit auf dem Gebiet reichen.) Obwohl der Autor diese Voraussetzung (noch) nicht erfüllt, hat er versucht, die Fragen, sofern ihm dies möglich war, hintergründig und detailiert zu beantworten.

Dazu musste der Autor auch einige Fachbegriffe verwenden. Die weniger geläufigen sind in Fußnoten am jeweiligen Seitenende mit kurzer Erklärung angegeben.

Während der Arbeiten an der Aufgabe erkannte er das Potenzial, das halb-automatische Übersetzungen bei weiterer Verbesserung des Computerteils in sich haben: Eine sehr grobe Vorübersetzung kann dem menschlichen Übersetzer aus Fleisch und Blut bereits eine Vorstellung vom Text liefern, und der sich so gegebenenfalls bei entsprechender Fachliteratur mit weniger gebräuchlichem Vokabular auf ihm Unbekanntes einstellen. Menschen, die der Fremdsprache überhaupt nicht mächtig sind, erfahren durch die verfügbaren Übersetzungsmöglichkeiten wenigstens das Thema und Teile des Inhaltes. Für mehr sind Computer-Übersetzungen momentan noch nicht zu gebrauchen. Aber, wie schon der amerikanische Dramatiker Maxwell Anderson schrieb:

> „Feldherren und Könige treten ab. Reichtümer zerrinnen, ohne eine Spur zu hinterlassen.“

5.1 Teilaufgabe Eins

Einige Beispielsätze in Englisch[3] und deren Übersetzung nach der in der Aufgabe vorgestellten Methode: (Die Übersetzung wurde dadurch erschwert, dass in der Aufgabenstellung nicht definiert wird, welche Bedeutung bei mehreren Übersetzungsmöglichkeiten pro Wort Vorrang hat.)

The cats eat the small mouse.[4] Das/Die Katzen fressen das/die klein Maus.

The big cat runs after the small mice. Die groß Katze rennt nach die klein Mäuse.

Later, the big mice run into the small mouse's hole. Später, die groß Mäuse rennen in das schmal Maus's Loch.

Aus diesen Beispielen wird ersichtlich, dass ein Übersetzungswerkzeug dieser Art keine Chance hat, logische Strukturen im Satz, wie Satzbau und Grammatik, zu erkennen. Schwächen treten deshalb bei Adjektiven und bei in verschiedenen Kasus[5] stehenden

[3] Die Qualität der Beispielsätze lässt freilich keine Rückschlüsse auf die Englischkenntnisse des Autors zu.

[4] Bei diesem und den weiteren Sätzen geht nicht eindeutig aus der Fragestellung hervor, wie man ihn übersetzen müsste.

[5] lat. casus = der Fall

Substantiven auf. Desweiteren werden die Wörter einfach der Reihe nach stupide abgearbeitet, ohne je einen Sinn zu ergeben. Der Genitiv im letzten Beispiel findet keine Beachtung und verfälscht somit gar den Sinn des Satzes. Das Übersetzungsergebnis ist insgesamt deshalb so schlecht, weil sich der Satz anhört, als sei er von einem der Sprache nicht mächtigen Fremden gesprochen, der nicht im Stande ist, die Worte ihrer jeweiligen Funktion im Satz anzupassen. Verben, die mit Präpoistion[6] stehen, wie im zweiten Beispiel „(to) run after“ hinterherlaufen heißt, und dann eine andere Bedeutung als in der Infinitv-Form haben, können ebenfalls nicht adäquat übersetzt werden.

Ohne Original ist ein Textverständnis erheblich gefährdet oder gar – nicht möglich. Einzige sinnvolle Verwendung kann diese Methode bei interaktiven Wörterbüchern, wie LEO-Online[7], oder Rechtschreibprüfungen finden. Für Übersetzungen einzelner Sätze oder gar längerer Texte dagegen ist sie völlig ungeeignet.

5.2 Teilaufgabe Zwei

Während des Testens der verschiedenen Übersetzungswerkzeuge stellte der Autor fest, dass eine nahezu vollständige Konvergenz zwischen den beliebtesten – und wohl auch besten – Tools bestand. Trotzdem wurden die folgenden Übersetzungen mit drei verschiedenen Programmen[8], nämlich den Google Language-Tools[9], FreeTranslation.com[10] und Systran[11], durchgeführt.

Mr. Miller died: The late Mr. Miller was of course not alive anymore.
Google: „Herr Miller starb: Späte Herr Miller war selbstverständlich mehr nicht lebendig.“
FreeTranslation.com: „Herr Miller ist gestorben: Der späte Herr Miller war selbstverständlich nicht lebend mehr.“
Systran: „_ Herr Miller sterben: _ d spät Herr Miller sein selbstverständlich nicht lebendig mehr. _“ (Die korrekte Übersetzung müsste „Herr Müller starb: Der verstorbene Herr Müller war natürlich nicht mehr lebendig“ lauten.)
Dieses erste Beispiel zeigt eine der großen Schwächen der Computerübersetzungen: Ihnen ist eine Texterschließung unmöglich. Während „the late Mr. Miller“ in einem beliebigen anderen Text sehr wohl „der späte Herr Müller“ heißen kann, ist dies aufgrund des vorhergehenden Satzes im Beispiel nicht möglich. Hierbei handelt es sich um eine reine Sinnfrage, die von einem Computer in absehbarer Zeit wohl nicht beherrscht werden kann. Daraus wird bereits ersichtlich, dass zurzeit noch keine vollautomatische Übersetzung eines (längeren) Textes denkbar ist. Abgesehen von dieser ersten Schwäche der Übersetzung treten zwei weitere Fehler nach dem Doppelpunkt auf: Der Artikel wird von Google aus nicht näher nachvollziehbaren Gründen nicht mitübersetzt. Außerdem

[6] Präpositionen bezeichnen Verhältnisse zwischen Personen und Sachen. Sie drücken kausale, lokale, modale und temporale Aspekte aus. Beispiele für Präp. sind „ab, unter, jenseits, von, auf“.
[7] LEO – Link Everything Online, http://www.dict.leo.org
[8] Dies war wohl mit dem Begriff „Funktion“ im Aufgabentext gemeint
[9] http://google.de/language_tools?hl=de
[10] http://freetranslation.com/
[11] http://www.systranbox.com/systran/box

ist die Syntax[12] des Satzes falsch. (Es müsste „nicht mehr“ an Stelle von „mehr nicht“ heißen.) Während FreeTranslation.com am besten abschnitt, ist Systrans Übersetzung völlig unbrauchbar.

Because of the hot weather the children were not allowed to play in the garden.
Google: „Wegen des heißen Wetters wurden den Kindern nicht erlaubt, im Garten zu spielen.“
FreeTranslation.com: „Wegen des heißen Wetters wurden die Kinder nicht erlaubt, im Garten zu spielen.“
Systran: „_ wegen d heiß verwittern d Kind sein nicht lassen zu Spiel in d Garten. _“ (Die korrekte Übersetzung müsste „wegen des heißen Wetters durften die Kinder nicht im Garten spielen“ heißen.)
Bis auf einen Numerusfehler[13] (wurde) schafft Google es immerhin, mit einer guten Übersetzung aufzuwarten. Systran scheint in einer anderen Liga zu spielen, was den Fortschritt der computergestützten Übersetzung in den letzten Jahren deutlich zeigt. Unerklärlich bleibt weiterhin, warum Systran Zeilenumbrüche (aus ästhetischen Gründen entfernt) und Unterstriche in den Satz einfügt und den Artikel „the“ als „d“ deutet.

I enjoy relaxing in the sun.
Google: „Ich genieße, in der Sonne mich zu entspannen.“
FreeTranslation.com: „Ich genieße Entspannung in der Sonne.“
Systran: „_ ich genießen relaxing in d Sonne. _“ (Richtig wäre „ich genieße das Entspannen in der Sonne“ gewesen.)
Googles Übertragung ins Deutsche weist einen falschen Satzbau auf. Bei der Lösung von FreeTranslation.com ist zu monieren, dass „Entspannung“ durch „das Entspannen“ ersetzt werden sollte.

Did you get the same results as Paul?
Google: „Erhielten Sie die gleichen Resultate wie Paul?“
FreeTranslation.com: „Haben Sie die gleichen Ergebnisse als Paul erhalten?“
Systran: „_ Sie erhalten d gleich Resultat wie Paul? _“ (Richtig ist Googles Übersetzung.)
Anhand dieses Beispiels zeigt sich, dass computergestützte Übersetzungssysteme in der Lage sind, auch einfache Fragen korrekt ins Deutsche zu übertragen. Sie lesen zumindest nicht einfach über das Fragezeichen (und andere Satzzeichen) hinweg, sondern beziehen dies und die veränderte Wortstellung im Satz folgerichtig in Ihre Übersetzung mit ein. FreeTranslation.com kann offensichtlich nicht zwischen „als“ und „wie“ unterscheiden. Obwohl Google als einziger den Satz richtig übersetzt, fragt sich, ob hier wirklich die Höflichkeitsform im Singular verwendet werden sollte. Das englische „you“ dürfte eher dem deutschen „du“ oder „ihr“ entsprechen. In Anbetracht des Kunstwerkes von Systran, wiederholt nicht einmal auch nur annähernd den Sinn zu treffen, mag es aber vielleicht als abwegig erscheinen, über solche Feinheiten zu diskutieren.

[12] griech. syntaxis = Zusammenstellung, Anordnung, Aneinanderreihung
Lehre von der Kombination von Wörtern zu Sätzen
[13] lat. numerus = Zahl, Anzahl
Hier: Verwechslung von Singular/Plural

5.3 Teilaufgabe Drei

- Viele Fehler entstehen durch eine falsche Sprachsyntax. (Vergleiche dazu Abschnitt 5.2 auf Seite 38.)
  Sprachen sind komplizierte, aber nicht streng logische dynamische Gebilde. Computer sind komplizierte, aber streng logische Maschinen. Sie arbeiten immer wieder nach denselben (einfältigen?) Strukturen und sind auch auf diesen Horizont der „Denkweise“ beschränkt. Es ist also schon von Grund auf ein schwieriges Unterfangen, Übersetzungen mit einem „Rechner“[14] anfertigen zu lassen. (Man beachte nur die Programmiersprachen: Verglichen mit echten Sprachen kann man sie im besten Fall als einfach und abstrakt bezeichnen.)

- Dem Computer ist ein Textverständnis unmöglich. Er kann deshalb bei Wörtern mit mehreren Bedeutungen, deren Übersetzungen sich nur aus dem Zusammenhang des Textes ergeben, nicht die in diesem Fall passende auswählen und benutzen. (Siehe hierzu auch Abschnitt 5.2 auf Seite 38.)

- Eine weitere fundamentale Schwierigkeit ist das Übersetzen von Phrasen und Redewendungen. Man denke nur an das in England oft gebrauchte „home sweet home“ (möglicherweise etwas in der Richtung wie „trautes Heim, Glück allein“), das die Google Language-Tools für ein „süsses Haupthaus“ halten.

[14] engl. (to) compute = rechnen, berechnen