



Studienarbeit

food4life - der Kalorientracker

Erstellt von:

Sven Kuczera

Robert-Frese-Straße 31

58332 Schwelm

Prüfer:

Prof. Dr. Seifert

Eingereicht am:

13.4.2018

Inhaltsverzeichnis

Abbildungsverzeichnis	II
Tabellenverzeichnis	III
Listingverzeichnis	IV
1 Das Team	1
1.1 Teamfoto	1
2 Ziel des Projekts	2
3 Benutzerhandbuch	3
4 Projektplanung	4
4.1 Pflichtenheft	4
4.2 Projektablaufplan	5
4.3 Planung der Software	5
4.3.1 Planung der Mockups	5
4.3.2 Planung der Datenstrukturen und Schnittstellen	5
4.3.3 Planung der Activities und Layouts	5
4.3.4 Planung der Navigation und des Datenaustausches zwischen den Activities	5
4.4 Geplante Aufgabenverteilung im Team	5
5 Beschreibung des Projektverlaufs	6
5.1 Tatsächliche Aufgabenverteilung im Team	6
5.2 Teammeetingprotokolle	6
5.3 Projekttagbücher aller Teammitglieder	6
5.4 Beschreibung von Problemen	6
6 Dokumentation der Software	7
6.1 Dokumentation der Paketstruktur des Android-Projektes	7
6.2 Dokumentation der Activities	7
6.2.1 Activity 1	7
6.2.2 Activity 2	7
6.2.3 Activity 3	7

6.2.4	Activity 4	7
6.2.5	Activity 5	7
6.2.6	Activity 6	7
6.2.7	Activity 7	7
6.2.8	Activity etc.	7
6.3	Dokumentation der Navigation und des Datenaustauschs zwischen Activities	7
6.4	Dokumentation der Activity-übergreifenden, persistenten Datenhaltung	7
6.5	Dokumentation der Activity-übergreifenden Klassen	7
7	Fazits aller Teammitglieder	8
7.1	Jannik Maes	8
7.2	Sven Kuczera	8
7.3	Henryk Schaffrath	8
7.4	Benedikt Burczek	8
	Anhang	9
	Quellenverzeichnis	52
	Ehrenwörtliche Erklärung	53

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

code/LipkeAntsClass.cs	13
agi_mfws414ali.tex	26
config/Config.tex	28
chapter/Titelseite.tex	32
chapter/Einleitung.tex	33
chapter/Elementares_Konzept.tex	33
chapter/Dokumentation.tex	35
chapter/Zusammenfassung.tex	49
chapter/Quellenverzeichnis.tex	50
chapter/Ehrenwoertliche_Erklaerung.tex	51

1 Das Team

1.1 Teamfoto

2 Ziel des Projekts

Das Ziel des Projektes Food4Life ist es ein neuartiges Kalorientagebuch als Android Applikation zu entwickeln. Die App hilft dem Benutzer die konsumierten Lebensmittel und deren enthaltende Kalorien in einem Tagebuch zu erfassen. Außerdem wird dem Benutzer ein auf ihn persönlich zugeschnittenes Tagespensum angezeigt. Mit der einfacheren und übersichtlichen Oberfläche und dem äußerst angenehmen Layout und dem intuitiven Interaktionsverhalten bietet die App von Food4Life eine überraschend andere Erfahrung bei der Verwendung eines „Calorie Trackers“. Zur Vereinfachung der Benutzung ist es außerdem möglich verschiedene Lebensmittel zu einem Menü hinzuzufügen und diese dann mit einem Klick in das Tagebuch einzutragen. Die App bietet zudem die Möglichkeit der statistischen Auswertung des durchschnittlichen Kalorienverbrauchs der letzten 7, 14 und 30 Tage.

3 Benutzerhandbuch

Die App food4life bietet eine neuartige und einzigartige Möglichkeit die tägliche Kalorien Einnahme zu verfolgen.

4 Projektplanung

4.1 Pflichtenheft

4.2 Projektablaufplan

4.3 Planung der Software

4.3.1 Planung der Mockups

4.3.2 Planung der Datenstrukturen und Schnittstellen

4.3.3 Planung der Activities und Layouts

4.3.4 Planung der Navigation und des Datenaustausches zwischen den Activities

4.4 Geplante Aufgabenverteilung im Team

5 Beschreibung des Projektverlaufs

5.1 Tatsächliche Aufgabenverteilung im Team

5.2 Teammeetingprotokolle

5.3 Projekttagbücher aller Teammitglieder

5.4 Beschreibung von Problemen

6 Dokumentation der Software

6.1 Dokumentation der Paketstruktur des Android-Projektes

6.2 Dokumentation der Activities

6.2.1 Activity 1

6.2.2 Activity 2

6.2.3 Activity 3

6.2.4 Activity 4

6.2.5 Activity 5

6.2.6 Activity 6

6.2.7 Activity 7

6.2.8 Activity etc.

6.3 Dokumentation der Navigation und des Datenaustauschs zwischen Activities

6.4 Dokumentation der Activity-übergreifenden, persistenten Datenhaltung

6.5 Dokumentation der Activity-übergreifenden Klassen

7 Fazits aller Teammitglieder

7.1 Jannik Maes

7.2 Sven Kuczera

7.3 Henryk Schaffrath

7.4 Benedikt Burczek

Anhang




Anhangsverzeichnis

Anhang 1:	Messreihen - Einzelspieler	11
Anhang 1.1:	Einzelspieler: LipkeAnts	11
Anhang 1.2:	Einzelspieler: aTomApfelmeisen	11
Anhang 1.3:	Einzelspieler: aTomGruppenmeisen	11
Anhang 1.4:	Einzelspieler: aTomKampfmeisen	11
Anhang 1.5:	Einzelspieler: aTomZuckermeisen	11
Anhang 2:	Messreihen - Mehrspieler	11
Anhang 2.1:	Mehrspieler: LipkeAnts - aTomApfelmeisen	12
Anhang 2.2:	Mehrspieler: LipkeAnts - aTomGruppenmeisen	12
Anhang 2.3:	Mehrspieler: LipkeAnts - aTomKampfmeisen	12
Anhang 2.4:	Mehrspieler: LipkeAnts - aTomZuckermeisen	12
Anhang 3:	Quellcode des AntMe!-Volkes <i>LipkeAnts</i>	13
Anhang 4:	Latex Quellcode	26
Anhang 4.1:	Masterdatei	26
Anhang 4.2:	Config	28
Anhang 4.3:	Titelseite	32
Anhang 4.4:	Kapitel 1 - Einleitung	33
Anhang 4.5:	Kapitel 2 - Elementares Konzept und Randbedingungen	33
Anhang 4.6:	Kapitel 3 - Dokumentation	35
Anhang 4.7:	Kapitel 4 - Zusammenfassung	49
Anhang 4.8:	Quellenverzeichnis	50









Anhang 4.9: Ehrenwörtliche Erklärung	50
--	----

Anhang 1 Messreihen - Einzelspieler









Anhang 1.1 Einzelspieler: LipkeAnts

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungerte...	 Geschlagene...	 Gefressene...	 Gesamt...
Team 1							
 LipkeAnts	4246,90	0,00	38,07	0,00	0,00	171,28	9957,40

Anhang 1.2 Einzelspieler: aTomApfelmeisen

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungerte...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 aTom Apfelmeisen	5207,07	0,00	0,00	31,59	0,00	134,70	5207,07









Anhang 1.3 Einzelspieler: aTomGruppenmeisen

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungerte...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 aTom Gruppenmeisen	6850,65	0,00	6,10	35,36	0,00	85,12	7765,80

Anhang 1.4 Einzelspieler: aTomKampfmeisen






Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungerte...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 aTom Kampfmeisen	0,00	0,00	33,90	0,00	0,00	146,02	5084,85

Anhang 1.5 Einzelspieler: aTomZuckermeisen



Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungerte...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 aTom Zuckermeisen	6752,83	0,00	0,00	14,12	0,00	95,41	6752,83

Anhang 2 Messreihen - Mehrspieler

Anhang 2.1 Mehrspieler: LipkeAnts - aTomApfelmeisen

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungert...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 LipkeAnts	2375,66	180,31	31,56	0,07	0,00	159,12	8011,21
Team 2							
 aTom Apfelmeisen	2652,50	0,00	0,00	12,81	180,29	77,95	1751,05

Anhang 2.2 Mehrspieler: LipkeAnts - aTomGruppenmeisen

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungert...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 LipkeAnts	2890,38	174,67	31,74	0,02	0,00	155,28	8524,32
Team 2							
 aTom Gruppenmeisen	2552,57	0,00	4,17	14,51	174,66	72,31	2305,04

Anhang 2.3 Mehrspieler: LipkeAnts - aTomKampfmeisen

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungert...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 LipkeAnts	3000,85	135,56	26,48	0,00	119,94	129,53	7051,66
Team 2							
 aTom Kampfmeisen	0,00	119,98	22,18	0,00	135,56	94,23	3249,39

Anhang 2.4 Mehrspieler: LipkeAnts - aTomZuckermeisen

Spieler	 Nahrung	 Getötete...	 Getötete Wanzen	 Verhungert...	 Geschlagene...	 Gefresse...	 Gesamt...
Team 1							
 LipkeAnts	3794,53	198,05	31,57	0,00	0,00	159,53	9519,63
Team 2							
 aTom Zuckermeisen	1098,08	0,00	0,00	8,84	198,05	66,16	107,83

Anhang 3 Quellcode des AntMe!-Volkes *LipkeAnts*

```

1  using AntMe.English;
2  using System;
3  using System.Collections.Generic;
4
5  /// <summary>
6  /// AntMe! project as part of the module "AGI".
7  /// </summary>
8  namespace AntMe.Player.LipkeAnts
9  {
10     [Player(
11         ColonyName = "LipkeAnts",
12         FirstName = "Felix",
13         LastName = "Lipke"
14     )]
15
16     /// <summary>
17     /// Specialized in fighting (with focus on bugs).
18     /// But also gathers apples. Ignores sugar.
19     /// </summary>
20     [Caste(
21         Name = "NormalAnt",
22         AttackModifier = 2,
23         EnergyModifier = 2,
24         LoadModifier = -1,
25         RangeModifier = -1,
26         RotationSpeedModifier = -1,
27         SpeedModifier = 0,
28         ViewRangeModifier = -1
29     )]
30
31     /// <summary>
32     /// Same as normal ants, but different behaviour.
33     /// Gathers also sugar.
34     /// </summary>
35     [Caste(
36         Name = "SugarAnt",
37         AttackModifier = 2,
38         EnergyModifier = 2,
39         LoadModifier = -1,
40         RangeModifier = -1,
41         RotationSpeedModifier = -1,
42         SpeedModifier = 0,
43         ViewRangeModifier = -1
44     )]
45     public class LipkeAntsClass : BaseAnt
46     {
47         #region Marker Info Type Members
48
49
50         private int infoTypeApple = 9;
51         private int infoTypeSugar = 8;
52         private int infoTypeBug = 7;
53         private int infoTypeForeignAnt = 6;
54
55         #endregion
56

```

```

57     #region Caste specific Members/Properties
58
59     private string normalAnt = "NormalAnt";
60     private string sugarAnt = "SugarAnt";
61
62     private bool IsGameProgressed { get; set; }
63
64     private bool IsSugarAnt
65     {
66         get { return Caste == sugarAnt; }
67     }
68
69     private bool IsNormalAnt
70     {
71         get { return Caste == normalAnt; }
72     }
73
74     private bool? HasSeenForeignAnt { get; set; }
75
76     /// <summary>
77     /// When a sugar ant has seen a foreign ant
78     /// it should behave like a normal ant.
79     /// </summary>
80     private bool ShouldSugarAntActLikeNormal
81     {
82         get { return IsSugarAnt && HasSeenForeignAnt == true; }
83     }
84
85     #endregion
86
87     #region Destination Properties
88
89     private bool HasNoDestination
90     {
91         get { return (Destination == null); }
92     }
93
94     private bool IsGoingToAntHill
95     {
96         get { return (Destination is Anthill); }
97     }
98
99     private bool IsGoingToApple
100    {
101        get { return (Destination is Fruit); }
102    }
103
104    private bool IsFollowingBug
105    {
106        get { return Destination is Bug; }
107    }
108
109    private bool IsCarringSugarToAntHill
110    {
111        get { return (IsGoingToAntHill && IsCarryingSugar); }
112    }
113
114    #endregion

```

```

115
116     #region Food Carrying Properties
117
118     private bool HasNoLoad
119     {
120         get { return (CurrentLoad == 0); }
121     }
122
123     private bool HasLoad
124     {
125         get { return (CurrentLoad > 0); }
126     }
127
128     private bool IsCarryingApple
129     {
130         get { return (CarryingFruit != null); }
131     }
132
133     private bool IsCarryingSugar
134     {
135         get { return (!IsCarryingApple && HasLoad); }
136     }
137
138     #endregion
139
140     #region Health Status Properties
141
142     private bool HasLowRangeLeft
143     {
144         get { return DistanceToAnthill > (Range - WalkedRange - 50); }
145     }
146
147     private bool HasLowEnergy
148     {
149         get { return CurrentEnergy < MaximumEnergy * 2 / 3; }
150     }
151
152     private bool MustRefresh
153     {
154         get { return HasLowRangeLeft || HasLowEnergy; }
155     }
156
157     #endregion
158
159     #region Caste
160
161     /// <summary>
162     /// Every time that a new ant is born, its job group must be set.
163     /// You can do so with the help of the value returned by
164     /// this method.
165     /// Read more: "http://wiki.antme.net/en/API1:ChooseCaste"
166     /// </summary>
167     /// <param name="typeCount">Number of ants for every caste</param>
168     /// <returns>Caste-Name for the next ant</returns>
169     public override string ChooseCaste(Dictionary<string, int>
        typeCount)
170     {
171         int countOfNormalAnts = typeCount[normalAnt];

```

```

172         int countOfSugarAnts = typeCount[sugarAnt];
173         int totalCount = countOfNormalAnts + countOfSugarAnts;
174         double sugarAntsShare = countOfSugarAnts / (double)totalCount;
175         string caste = string.Empty;
176
177         if(totalCount > 0 && (sugarAntsShare < 0.25))
178         {
179             caste = sugarAnt;
180         }
181         else
182         {
183             caste = normalAnt;
184         }
185         IsGameProgressed = totalCount > 60; // Maximum is 100
186
187         return caste;
188     }
189
190     #endregion
191
192     #region Movement
193
194     /// <summary>
195     /// If the ant has no assigned tasks, it waits for new tasks.
196     /// This method is called to inform you that it is waiting.
197     /// Read more: "http://wiki.antme.net/en/API1:Waiting"
198     /// </summary>
199     public override void Waiting()
200     {
201         if (IsNormalAnt || ShouldSugarAntActLikeNormal)
202         {
203             // Should not go straight forward to extend the search
204             // areas.
205             // Increases the possibility to spot an apple or enemy.
206             GoForward(40);
207             TurnByDegrees(RandomNumber.Number(-10, 10));
208         }
209         else if (IsSugarAnt)
210         {
211             // Should find sugar trace.
212             GoForward(150);
213             TurnByDegrees(RandomNumber.Number(-15, 15));
214         }
215     }
216
217     /// <summary>
218     /// This method is called when an ant has travelled
219     /// one third of its movement range.
220     /// Read more: "http://wiki.antme.net/en/API1:GettingTired"
221     /// </summary>
222     public override void GettingTired()
223     {
224         // Here could be called "GoToAnthill()"
225         // to ensure ants don't die of starvation
226         // or low energy (disadvantage in fight).
227
228         // BUT "GettingTired()" gets only called
229         // when there is 1/3 of range left. This

```

```

229         // is very inflexible. It doesn't covers
230         // the refresh when the ant has low energy.
231         // Also in many situations one thrid might be
232         // too late, because the ant hill is too far
233         // away (e.g. ant hill is at the very right side).
234
235         // Therefore "MustRefresh" (on tick) is an own mechanism
236         // to control when an ant should go back to ant hill.
237     }
238
239     /// <summary>
240     /// This method is called if an ant dies. It informs you that
241     /// the ant has died. The ant cannot undertake any more
242     /// actions from that point forward.
243     /// Read more: "http://wiki.antme.net/en/API1:HasDied"
244     /// </summary>
245     /// <param name="kindOfDeath">Kind of Death</param>
246     public override void HasDied(KindOfDeath kindOfDeath)
247     {
248         // As there are no more actions possible,
249         // it don't make any sense to use this event
250         // with non-static ants.
251         // However as an !static! ant it could be used to react
252         // on certain circumstances and adjust the spawn castes.
253     }
254
255     /// <summary>
256     /// This method is called in every simulation round, regardless of
257     /// additional conditions. It is ideal for actions that must be
258     /// executed but that are not addressed by other methods.
259     /// Read more: "http://wiki.antme.net/en/API1:Tick"
260     /// </summary>
261     public override void Tick()
262     {
263         // There are most likely no foreign ants.
264         if (WalkedRange > 200 && IsGameProgressed && HasSeenForeignAnt
265             == null)
266         {
267             HasSeenForeignAnt = false;
268         }
269
270         // Decision making.
271         // React on currenct circumstances.
272         if (MustRefresh) // else the ant might die.
273         {
274             GoToAnthill();
275         }
276         else if (IsCarringSugarToAntHill)
277         {
278             MakeTraceToSugar();
279         }
280         else if (IsCarryingApple)
281         {
282             // Ensure destination is not lost
283             // and spread if apple needs more carriers.
284             HandleApple();
285         }
286         else if (IsGoingToApple && !NeedsCarrier((Fruit)Destination))

```

```

286         {
287             Stop(); // Don't follow - do something different instead.
288         }
289     }
290
291     #endregion
292
293     #region Food
294
295     /// <summary>
296     /// This method is called as soon as an ant sees an apple
297     /// within its 360 degree visual range. The parameter is
298     /// the piece of fruit that the ant has spotted.
299     /// Read more: "http://wiki.antme.net/en/API1:Spots(Fruit)"
300     /// </summary>
301     /// <param name="fruit">spotted fruit</param>
302     public override void Spots(Fruit fruit)
303     {
304         if (HasNoLoad && HasNoDestination && NeedsCarrier(fruit))
305         {
306             GoToDestination(fruit);
307             MakeOtherAntsAwareOfApple();
308         }
309     }
310
311     /// <summary>
312     /// This method is called as soon as an ant sees a mound of
313     /// sugar in its 360 degree visual range. The parameter is
314     /// the mound of sugar that the ant has spotted.
315     /// Read more: "http://wiki.antme.net/en/API1:Spots(Sugar)"
316     /// </summary>
317     /// <param name="sugar">spotted sugar</param>
318     public override void Spots(Sugar sugar)
319     {
320         if (DistanceToAnthill < 600)
321         {
322             // If we are alone sugar has higher prio
323             if (HasSeenForeignAnt != true)
324             {
325                 int direction = Coordinate.GetDegreesBetween(this, sugar
326                 );
327                 int distance = Coordinate.GetDistanceBetween(this, sugar
328                 );
329
330                 int information = CreateMarkerInformation(direction,
331                     infoTypeSugar);
332                 MakeMark(information, distance);
333             }
334
335             // If ant has load, it does not need more sugar.
336             // If it has a destination: ignore the sugar,
337             // the destination might be a bug, ant hill, ...
338             if (HasNoLoad && HasNoDestination)
339             {
340                 GoToDestination(sugar);
341             }
342         }
343     }

```

```

341
342    /// <summary>
343    /// If the ant's destination is a piece of fruit, this method
344    /// is called as soon as the ant reaches its destination.
345    /// It means that the ant is now near enough to its
346    /// destination/target to interact with it.
347    /// Read more: "http://wiki.antme.net/en/API1:DestinationReached(
    Fruit)"
348    /// </summary>
349    /// <param name="fruit">reached fruit</param>
350    public override void DestinationReached(Fruit fruit)
351    {
352        if (NeedsCarrier(fruit)) // Ensure apple still needs carrier.
353        {
354            TakeFoodToAntHill(fruit);
355        }
356        if (NeedsCarrier(fruit)) // Still?
357        {
358            MakeOtherAntsAwareOfApple();
359        }
360    }
361
362    /// <summary>
363    /// If the ant's destination is a mound of sugar, this method
364    /// is called as soon as the ant has reached its destination.
365    /// It means that the ant is now near enough to its
366    /// destination/target to interact with it.
367    /// Read more: "http://wiki.antme.net/en/API1:DestinationReached(
    Sugar)"
368    /// </summary>
369    /// <param name="sugar">reached sugar</param>
370    public override void DestinationReached(Sugar sugar)
371    {
372        int direction = Coordinate.GetDegreesBetween(this, sugar);
373        int information = CreateMarkerInformation(direction,
            infoTypeSugar);
374        MakeMark(information, 80);
375        TakeFoodToAntHill(sugar);
376    }
377
378    #endregion
379
380    #region Communication
381
382    /// <summary>
383    /// Friendly ants can detect markers left by other ants.
384    /// This method is called when an ant smells a friendly
385    /// marker for the first time.
386    /// Read more: "http://wiki.antme.net/en/API1:DetectedScentFriend(
    Marker)"
387    /// </summary>
388    /// <param name="marker">marker</param>
389    public override void DetectedScentFriend(Marker marker)
390    {
391        // Rehash information.
392        var markerInfo = new MarkerInformation(marker.Information);
393
394        // Decision making, based on information.

```

```

395
396         if (markerInfo.InfoType == infoTypeBug)
397         {
398             OnDetectedBugMarker(marker);
399         }
400         else if (markerInfo.InfoType == infoTypeApple)
401         {
402             OnDetectedAppleMarker(marker, markerInfo.Data);
403         }
404         else if (markerInfo.InfoType == infoTypeSugar)
405         {
406             OnDetectedSugarMarker(markerInfo.Data);
407         }
408         else if (markerInfo.InfoType == infoTypeForeignAnt)
409         { // There is a foreign ant!
410             HasSeenForeignAnt = true;
411         }
412     }
413
414     /// <summary>
415     /// Just as ants can see various types of food, they can
416     /// also visually detect other game elements. This method
417     /// is called if the ant sees an ant from the same colony.
418     /// Read more: "http://wiki.antme.net/en/API1:SpotsFriend(Ant)"
419     /// </summary>
420     /// <param name="ant">spotted ant</param>
421     public override void SpotsFriend(Ant ant)
422     {
423         // Could be used for group building or information transfer.
424         // But there are no groups in this strategy and for all
425         // different
426         // information (apple, bug, sugar, foreign ants) markers are
427         // already made and in addition with the possibility to spread
428         // advanced information (see "CreateMarkerInformation()")
429         // there is no need for further information spreading.
430     }
431
432     /// <summary>
433     /// Just as ants can see various types of food, they can
434     /// also visually detect other game elements. This method
435     /// is called if the ant detects an ant from a friendly
436     /// colony (an ant on the same team).
437     /// Read more: "http://wiki.antme.net/en/API1:SpotsTeammate(Ant)"
438     /// </summary>
439     /// <param name="ant">spotted ant</param>
440     public override void SpotsTeammate(Ant ant)
441     {
442         // Reason why not used is same as "SpotsFriend()".
443     }
444
445     #endregion
446
447     #region Fight
448
449     /// <summary>
450     /// Just as ants can see various types of food, they can
451     /// also visually detect other game elements. This method
452     /// is called if the ant detects an ant from an enemy colony.

```



```

452     /// Read more: "http://wiki.antme.net/en/API1:SpotsEnemy(Ant)"
453     /// </summary>
454     /// <param name="ant">spotted ant</param>
455     public override void SpotsEnemy(Ant ant)
456     {
457         // It is sufficient to only spread the info once per ant.
458         if (HasSeenForeignAnt != true)
459         {
460             // Range of 500 to get many friends informed.
461             SpreadInfoForeignAntsExist(500);
462         }
463
464         // Apples are besides bugs the main source for gathering points
465         // Therefore only attack ants if not currently carrying an
466         // apple.
467         if (!IsCarryingApple)
468         {
469             // One-on-One (Offence is the beste Defense).
470             Drop(); // Sugar is less important
471             Attack(ant);
472         }
473     }
474     /// <summary>
475     /// Just as ants can see various types of food, they can
476     /// also visually detect other game elements. This method
477     /// is called if the ant sees a bug.
478     /// Read more: "http://wiki.antme.net/en/API1:SpotsEnemy(Bug)"
479     /// </summary>
480     /// <param name="bug">spotted bug</param>
481     public override void SpotsEnemy(Bug bug)
482     {
483         // Radius of 150 is optimal:
484         // a) not to small (we need friends to help us)
485         // b) not to big (if too big our friend might be too late for
486         // the fight)
487         MakeOtherAntsAwareOfBug(150);
488
489         // Apples are besides bugs the main source for gathering points
490         // Therefore only attack bugs if not currently carrying an
491         // apple.
492         if (!IsCarryingApple)
493         {
494             Drop(); // Sugar is less important
495             if (FriendlyAntsInViewrange >= 3) // Not alone!
496             {
497                 Attack(bug);
498             }
499             else // Follow as long there aren't enough friendly ants.
500             {
501                 GoToDestination(bug);
502             }
503         }
504     }
505     /// <summary>

```

```

505     /// Enemy creatures may actively attack the ant. This method
506     /// is called if an enemy ant attacks; the ant can then
507     /// decide how to react.
508     /// Read more: "http://wiki.antme.net/en/API1:UnderAttack(Ant)"
509     /// </summary>
510     /// <param name="ant">attacking ant</param>
511     public override void UnderAttack(Ant ant)
512     {
513         // Fight is inescapable.
514         Drop();
515         Attack(ant);
516     }
517
518     /// <summary>
519     /// Enemy creatures may actively attack the ant. This method
520     /// is called if a bug attacks; the ant can decide how to react.
521     /// Read more: "http://wiki.antme.net/en/API1:UnderAttack(Bug)"
522     /// </summary>
523     /// <param name="bug">attacking bug</param>
524     public override void UnderAttack(Bug bug)
525     {
526         // One ant is not enough for a bug.
527         MakeOtherAntsAwareOfBug(150);
528
529         // Fight is inescapable.
530         Drop();
531         Attack(bug);
532     }
533
534     #endregion
535
536     #region Methods
537
538     /// <summary>
539     /// Advanced marker information.
540     /// Enables ants to provide more information via the marker.
541     /// </summary>
542     /// <param name="information">Info for other ants e.g. direction.</
543     param>
544     /// <param name="infoType">Kind of spotted item (apple, bug or
545     sugar, foreign ant).</param>
546     /// <returns>The advanced marker information.</returns>
547     private int CreateMarkerInformation(int information, int infoType)
548     {
549         // Build a string which contains extended information.
550         // String must be build in such a way it can
551         // be converted to an integer without information loss.
552         string _infoType = infoType.ToString();
553         string _information = information.ToString().PadLeft(4, '0');
554         string advancedInformation = _infoType + _information;
555
556         return Convert.ToInt32(advancedInformation);
557     }
558
559     private void MakeTraceToSugar()
560     {
561         if (IsSugarAnt && HasSeenForeignAnt != true)
562         {

```

```
561         int directionToSugar = Direction + 180; // sugar is
562             opposite from ant hill.
563         int information = CreateMarkerInformation(directionToSugar,
564             infoTypeSugar);
565         int range = DistanceToAnthill > 50 ? 40 : 80;
566         MakeMark(information, range);
567     }
568 }
569 private void MakeOtherAntsAwareOfApple()
570 {
571     // If near to ant hill more carriers are not necessary needed.
572     if (DistanceToAnthill > 100)
573     {
574         // Tell other ants how many carriers are really needed.
575         int antsNeeded = Math.Max(0, 8 - FriendlyAntsInViewrange);
576         int information = CreateMarkerInformation(antsNeeded,
577             infoTypeApple);
578         MakeMark(information, 60);
579     }
580 }
581 private void MakeOtherAntsAwareOfBug(int range)
582 {
583     int information = CreateMarkerInformation(0, infoTypeBug);
584     MakeMark(information, range);
585 }
586 private void SpreadInfoForeignAntsExist(int range)
587 {
588     // Spread the information that I have seen an foreign ant.
589     int information = CreateMarkerInformation(0, infoTypeForeignAnt
590 );
591     MakeMark(information, range);
592     HasSeenForeignAnt = true;
593 }
594 private void TakeFoodToAntHill(Food food)
595 {
596     Take(food);
597     GoToAnthill();
598 }
599 private void HandleApple()
600 {
601     GoToAnthill();
602     MakeOtherAntsAwareOfApple();
603 }
604 private void OnDetectedSugarMarker(int sugarDirection)
605 {
606     if (IsSugarAnt // Normal ants should concentrate on bugs and
607         apples.
608         && HasNoDestination
609         && HasNoLoad
610         && HasSeenForeignAnt != true
611         && Direction != sugarDirection)
612     {
613     }
```

```

614         TurnToDirection(sugarDirection);
615         GoForward(150);
616     }
617 }
618
619 private void OnDetectedAppleMarker(Marker marker, int antsNeeded)
620 {
621     if (HasNoLoad
622         && HasNoDestination
623         && FriendlyAntsInViewrange < antsNeeded)
624     {
625         GoToDestination(marker); // Go to middle of the mark.
626     }
627     else
628     {
629         Stop(); // Apple does not need more carriers
630     }
631 }
632
633
634 private void OnDetectedBugMarker(Marker marker)
635 {
636     // Bugs have Prio! Therefore the ant should ignore
637     // whether it is carrying sugar or not.
638     if (FriendlyAntsInViewrange < 8 // else would too many ants go
639         to the bug.
640         && (HasNoDestination
641             || (!IsFollowingBug && !IsCarryingApple) // apples also
642             important.
643             || (IsNormalAnt && IsCarryingSugar)
644             || (ShouldSugarAntActLikeNormal && IsCarryingSugar)))
645     {
646         Drop(); // Sugar
647         GoToDestination(marker);
648     }
649 }
650
651 #endregion
652
653 #region Helper Class
654
655 /// <summary>
656 /// Helper Class for advanced marker information.
657 /// </summary>
658 public class MarkerInformation
659 {
660     /// <summary>
661     /// Extract information data and type.
662     /// </summary>
663     /// <param name="information">The advanced information from the
664     marker.</param>
665     public MarkerInformation(int information)
666     {
667         string info = information.ToString();
668         this.Data = Convert.ToInt32(info.Substring(1));
669         // First char is info type.
670         this.InfoType = Convert.ToInt32(info.Substring(0, 1));

```

```
669     }
670
671     /// <summary>
672     /// Apple: ants needed,
673     /// Sugar: direction,
674     /// Bug: zero (none),
675     /// ForeignAnt: zero (none)
676     /// </summary>
677     public int Data { get; set; }
678
679     /// <summary>
680     /// 9: Apple,
681     /// 8: Sugar,
682     /// 7: Bug,
683     /// 6: ForeignAnt
684     /// </summary>
685     public int InfoType { get; set; }
686 }
687
688 #endregion
689 }
```

Anhang 4 Latex Quellcode

Anhang 4.1 Masterdatei

```

1 \input{config/Config}
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 %% Parameter - Hier auf die eigene Arbeit anpassen
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 \newcommand{\dokumententyp}{Studienarbeit}
8 \newcommand{\abgabedatum}{13.4.2018}
9 \newcommand{\ort}{Mettmann}
10 \newcommand{\kooperationsunternehmen}{}
11 \newcommand{\dokumententitel}{food4life - der Kalorientracker}
12 \newcommand{\dokumentenautor}{Sven Kuczera}
13 \newcommand{\dokumentenautoradress}{Robert-Frese-Straße 31\\58332 Schwelm}
14 \newcommand{\dokumentenpruefer}{Prof. Dr. Seifert}
15
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17 %% Helper Commands
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19
20 \newcommand{\subheading}[1]{\bigskip\textbf{\#1}}
21
22 \newcommand{\gq}[1]{\glq\#1\grq}
23 \newcommand{\gqq}[1]{\glqq\#1\grqq}
24 \newcommand{\eq}[1]{\#1}
25 \newcommand{\eqq}[1]{\#1}
26
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 \hypersetup{
29   colorlinks=false,
30   pdfborder={0 0 0},
31   pdftitle=\dokumententitel,
32   pdfauthor=\dokumentenautor
33 }
34
35 \begin{document}
36
37 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38 %% Titelseite
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41 \input{chapter/Titelseite}
42
43 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
44 %% Draft-Einstellungen
45 %%
46 %% Für die finale Version auskommentieren!
47 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48 \%fancyhead[L]{\color{red} Stand: \today--\currenttime}
49
50 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
51 %% Verzeichnisse
52 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53

```

```

54 % Römische Seitennummerierung
55 \pagenumbering{Roman}
56
57 % Inhaltsverzeichnis
58 \tableofcontents
59
60 % Abbildungsverzeichnis
61 \fancyhead[R]{\listfigurename}
62 \listoffigures\newpage
63
64 % Tabellenverzeichnis
65 \fancyhead[R]{\listtablename}
66 \listoftables\newpage
67
68 % Quelltextverzeichnis
69 \fancyhead[R]{\lstlistlistingname}
70 \lstlistoflistings\newpage
71
72 % Kapitelüberschriften für den Arbeitstext
73 \fancyhead[R]{\leftmark}
74
75 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76 % Inhalt
77 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78
79 % Arabische Seitennummerierung
80 \pagenumbering{arabic}
81
82 \include{chapter/Das_Team}
83
84 \include{chapter/Ziel_des_Projekts}
85
86 \include{chapter/Benutzerhandbuch}
87
88 \include{chapter/Projektplanung}
89
90 \include{chapter/Beschreibung_des_Projektverlaufs}
91
92 \include{chapter/Dokumentation_der_Software}
93
94 \include{chapter/Fazits_aller_Teammitglieder}
95
96 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
97
98 \include{chapter/Anhang}
99
100 \include{chapter/Quellenverzeichnis}
101
102 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
103
104 \include{chapter/Ehrenwoertliche_Erklaerung}
105
106 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
107
108 \end{document}

```

Anhang 4.2 Config

```

1  %!TEX root = ../agi - mfs414ali.tex
2  %%
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5  % Wählen Sie die Optionen aus, indem Sie % vor der Option entfernen
6  % Dokumentation des KOMA-Script-Packets: scrguide
7
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  %% Optionen zum Layout des Artikels
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 \documentclass[%
12 paper=A4,           % alle weiteren Papierformat einstellbar
13 fontsize=12pt,      % Schriftgröße (12pt, 11pt (Standard))
14 BCOR12mm,          % Bindekorrektur, bspw. 1 cm
15 DIV14,              % breiter Satzspiegel
16 parskip=half*,      % Absatzformatierung s. scrguide 3.1
17 headsepline,        % Trennline zum Seitenkopf
18 %footsepline,       % Trennline zum Seitenfuß
19 %normalheadings,    % Überschriften etwas kleiner (smallheadings)
20 listof=totoc,       % Tabellen & Abbildungsverzeichnis ins Inhaltsverzeichnis
21 %bibtotoc,          % Literaturverzeichnis im Inhalt
22 %draft              % Überlangen Zeilen in Ausgabe gekennzeichnet
23 footinclude=false,  % Fußzeile in die Satzspiegelberechnung einbeziehen
24 headinclude=true,   % Kopfzeile in die Satzspiegelberechnung einbeziehen
25 final              % draft beschleunigt die Kompilierung
26 ]
27 {scrartcl}
28
29 %\setuptoc{toc}{totoc} % Inhaltsverzeichnis ins Inhaltsverzeichnis
30
31 % Neue Deutsche Rechtschreibung und Deutsche Standardtexte
32 \usepackage[ngerman]{babel}
33
34 % Umlaute können verwendet werden
35 \usepackage[utf8]{inputenc}
36
37 % Echte Umlaute
38 \usepackage[T1]{fontenc}
39
40 % Latin Modern Font, Type1-Schriftart für nicht-englische Texte
41 \usepackage{lmodern}
42
43 % 1/2-zeiliger Zeilenabstand
44 \usepackage[onehalfspacing]{setspace}
45
46 % Für die Defenition eigener Kopf- und Fußzeilen
47 \usepackage{fancyhdr}
48
49 % Für die Verwendung von Grafiken
50 \usepackage[pdftex]{graphicx}
51
52 % Bessere Tabellen
53 \usepackage{tabularx}
54
55 % Ausrichtung von Dezimalzahlen
56 \usepackage{siunitx}

```



```

57
58 % Für die Befehle \toprule, \midrule und \bottomrule, z.B. in Tabellen
59 \usepackage{booktabs}
60
61 % Erlaubt die Benutzung von Farben
62 \usepackage{color}
63
64 % Links im PDF
65 \usepackage{hyperref}
66
67 % Verbessertes URL-Handling mit \url{http://...}
68 \usepackage{url}
69
70 % Listen ohne Abstände \begin{compactlist}...\end{compactlist}
71 \usepackage{paralist}
72
73 % Ausgabe der aktuellen Uhrzeit für die Draft-Versionen
74 \usepackage{datetime}
75
76 % Deutsche Anführungszeichen
77 \usepackage[babel,german=quotes]{csquotes}
78
79 % Verbessert das Referenzieren von Kapiteln, Abbildungen etc.
80 \usepackage[german,capitalise]{cleveref}
81
82 % Konfiguration der Abbildungs- und Tabellenbezeichnungen
83 \usepackage[format=hang, font={footnotesize, sf}, labelfont=bf,
84   justification=raggedright,singlelinecheck=false]{caption}
85
86 % Verbessert die Lesbarkeit durch Mikrotypografie
87 \usepackage[activate={true,nocompatibility},final,tracking=true,kerning=
88   true,spacing=true,factor=1100,stretch=10,shrink=10]{microtype}
89
90 % Zitate und Quellenverzeichnis
91 \usepackage[
92   style=authoryear,          % Zitierstil
93   firstinits=false,         % false = Vornamen werden ausgeschrieben
94   natbib=true,
95   urldate=long,             % "besucht am" - Datum
96   %url=false,
97   date=long,
98   dashed=false,
99   maxcitenames=3,           % max. Anzahl Autorennamen in Zitaten
100  maxbibnames=99,           % max. Anzahl Autorennamen im
101    Quellenverzeichnis
102  backend=bibtex             % Ggf. für ältere Distributionen bibtex
103    verwenden
104    %backend=biber
105  ]{biblatex}
106
107 % Bibliograph
108 \bibliography{library/library}
109
110 % Keine Einrückung bei einem neuen Absatz
111 \parindent 0pt
112
113 % Ebenentiefe der Nummerierung
114 \setcounter{secnumdepth}{3}

```

```

111
112 % Gliederungstiefe im Inhaltsverzeichnis
113 \setcounter{tocdepth}{3}
114
115 % Tabellen- und Abbildungsverzeichnis mit Bezeichnung:
116 \usepackage[titles]{tocloft}
117
118 % Sourcecode-Listings
119 \usepackage{listings}
120
121 % Bestimmte Warnungen unterdrücken
122 % siehe http://tex.stackexchange.com/questions/51867/koma-warning-about-
123 \usepackage{scrhack}
124
125 %% http://tex.stackexchange.com/questions/126839/how-to-add-a-colon-after-
126 \makeatletter
127 \begingroup\let\newcounter\@gobble\let\setcounter\@gobbletwo
128 \globaldefs\@one \let\c@loldepth\@ne
129 \newlistof{listings}{lol}{\lstlistlistingname}
130 \endgroup
131 \let\l@lstlisting\l@listings
132 \makeatother
133
134 \renewcommand*\cftfigpresnum{Abbildung~}
135 \renewcommand*\cfttabpresnum{Tabelle~}
136 \renewcommand*\cftlistingspresnum{Listing~}
137 \renewcommand{\cftfigaftersnum}{:}
138 \renewcommand{\cfttabaftersnum}{:}
139 \renewcommand{\cftlistingsaftersnum}{:}
140 \settowidth{\cftfignumwidth}{\cftfigpresnum 99~\cftfigaftersnum}
141 \settowidth{\cfttabnumwidth}{\cfttabpresnum 99~\cftfigaftersnum}
142 \settowidth{\cftlistingsnumwidth}{\cftlistingspresnum 99~\cftfigaftersnum}
143 \setlength{\cfttabindent}{1.5em}
144 \setlength{\cftfigindent}{1.5em}
145 \setlength{\cftlistingsindent}{1.5em}
146
147 \renewcommand\lstlistlistingname{Listingverzeichnis}
148
149 % Style für Kopf- und Fußzeilenfelder
150 \pagestyle{fancy}
151 \fancyhf{}
152 \fancyhead[R]{\leftmark}
153 \fancyfoot[R]{\thepage}
154 \renewcommand{\sectionmark}[1]{\markboth{#1}{#1}}
155 \fancypagestyle{plain}{}
156
157 % Macro für Quellenangaben unter Abbildungen und Tabellen
158 \newcommand{\source}[1]{\vspace{-1mm}\footnotesize\textsf{\textbf{
  Quelle:}} \textsf{#1}\par}}
159
160 % Anpassungen der Formatierung an Eclipse-Aussehen
161 % http://jevopi.blogspot.de/2010/03/nicely-formatted-listings-in-latex-
162 \definecolor{sh_comment}{rgb}{0.12, 0.38, 0.18 } %adjusted, in Eclipse:
163 \definecolor{sh_keyword}{rgb}{0.37, 0.08, 0.25} % #5F1441

```

```

164 %\definecolor{sh_string}{rgb}{0.06, 0.10, 0.98} % #101AF9
165 % Für Druckausgabe sollte alles schwarz sein
166 \definecolor{sh_comment}{rgb}{0.0, 0.0, 0.0 }
167 \definecolor{sh_keyword}{rgb}{0.0, 0.0, 0.0 }
168 \definecolor{sh_string}{rgb}{0.0, 0.0, 0.0 }
169
170 \lstset{ %
171     language=[Sharp]C,
172     basicstyle=\small\ttfamily,
173     fontadjust,
174     xrightmargin=1mm,
175     xleftmargin=5mm,
176     tabsize=2,
177     columns=flexible,
178     showstringspaces=false,
179     rulesepcolor=\color{black},
180     showspaces=false,
181     showtabs=false,
182     stringstyle=\color{sh_string},
183     keywordstyle=\color{sh_keyword}\bfseries,
184     commentstyle=\color{sh_comment},
185     captionpos=t,
186     lineskip=-0.3em,
187     stepnumber=1,
188     breaklines=true,
189     breakatwhitespace=false,
190     float=bht,
191     literate=%
192     {Ö}{\{"O}}1
193     {Ä}{\{"A}}1
194     {Ü}{\{"U}}1
195     {ß}{\{ss}}2
196     {ü}{\{"u}}1
197     {ä}{\{"a}}1
198     {ö}{\{"o}}1
199 }
200
201 %\makeatletter
202 %\def\l@lstlisting#1#2{\@dottedtocline{1}{0em}{1.5em}{\lstlistingname\
    space{#1}}{#2}}
203 %\makeatother
204
205 % Anhangsverzeichnis
206 \usepackage[nohints]{minitoc} %Anhangsverzeichnis
207
208 \makeatletter
209 \newcounter{fktnr}\setcounter{fktnr}{0}
210 \newcounter{subfktnr}[fktnr]\setcounter{subfktnr}{0}
211
212 \renewcommand\thesubfktnr{\arabic{fktnr}.\arabic{subfktnr}}
213 \newcounter{anhangcounter}
214 \newcommand{\blatt}{\stepcounter{anhangcounter}}
215
216 \newcommand{\anhang}[1]{\setcounter{anhangcounter}{0}\refstepcounter{fktnr}
    }
217 \addcontentsline{fk}{subsection}{Anhang~\thefktnr: \hspace*{1em}#1}
218 \subsection*{\Anhang~\thefktnr \hspace*{1em} #1 \hspace*{-1em}}
219 }

```

```

220
221 \newcommand{\subanhang}[1]{\setcounter{anhangcounter}{0}\refstepcounter{
    subfktnr}
222 \addcontentsline{fk}{subsubsection}{Anhang~\thesubfktnr: \hspace*{1em}#1}
223 \subsubsection*{{Anhang~\thesubfktnr \hspace*{1em} #1 \hspace*{-1em}}}
224 }
225
226 \newcommand{\anhangsverzeichnis}{\mtcaddsection{\subsection*{
    Anhangsverzeichnis \@mkboth{FKT}{FKT}}}\@starttoc{fk}\newpage}

```

Anhang 4.3 Titelseite

```

1 %!TEX root = ../agi_mfws414ali.tex
2
3 \begin{titlepage}
4
5 \begin{center}
6
7
8 \includegraphics[scale=.5]{img/fhdw.jpg}\\
9
10 \vspace{.7cm}
11
12 \Huge{\bfseries\dokumententyp}
13
14 ~\vspace{.5cm}\\
15
16 \LARGE{\dokumententitel}
17
18 ~\vspace{1.2cm}\\
19
20
21 \large{
22
23 Erstellt von:\\ \vspace{1mm}
24
25 \dokumentenautor\\
26
27 \dokumentenautoradress
28
29
30 \vspace{1.3cm}
31
32 Prüfer:\vspace{1mm}\\
33
34 \dokumentenpruefer
35
36
37 \vspace{1.3cm}
38
39 Eingereicht am:\vspace{1mm}\\
40
41 \abgabedatum
42
43 }

```

```

44 |
45 | \end{center}
46 |
47 |
48 | \end{titlepage}

```

Anhang 4.4 Kapitel 1 - Einleitung

```

1 | %!TEX root = ../agi_mfws414ali.tex
2 | \section{Einleitung}
3 |
4 | Diese wissenschaftliche Ausarbeitung dokumentiert die programmatische
   | Umsetzung eines Ameisenvolkes auf Grundlage des AntMe!-Frameworks.
5 |
6 | Das Ziel bestand in der Entwicklung einer Strategie, welche sich nicht nur
   | im Einzelspiel sondern auch im Wettbewerb gegen andere Ameisenvölker
   | als geeignet erweist. Dabei bestand die Herausforderung in der
   | Begrenzung auf ein nicht-statisches Volk.
7 |
8 | Dargelegt werden sollen sowohl die Arbeitsweise des Ameisenvolkes als auch
   | die Strategie hinter dieser. Zu diesem Zweck werden unter anderem
   | Auszüge aus dem Quelltext, welcher dem Ameisenvolk zugrunde liegt,
   | herangezogen. Weiter wird das grundlegende Konzept von AntMe! erlä
   | utert.

```

Anhang 4.5 Kapitel 2 - Elementares Konzept und Randbedingungen

```

1 | %!TEX root = ../agi_mfws414ali.tex
2 | \section{Elementares Konzept und Randbedingungen}
3 | \label{concept}
4 |
5 | \subsection{Zielsetzung}
6 | Das Ziel des AntMe!-Projektes war es, ein Ameisenvolk zu entwickeln,
   | welches nicht nur im Einzelspieler, sondern auch im Mehrspieler-Modus
   | möglichst hohe Punktzahlen erreicht.
7 |
8 | \subsection{Vorgehensweise}
9 | Diese Ausarbeitung ist in vier Kapitel untergliedert. Im zweiten Kapitel
   | liegt der Fokus auf der Schaffung von Grundlagen für ein allgemeines
   | Verständnis der Thematik. Bei dem dritten Kapitel handelt es sich um
   | den Hauptteil, der Dokumentation des Ameisenvolkes. Abschließend wird
   | in Kapitel vier das erarbeitete Ergebnis zusammenfassend betrachtet
   | und ein Ausblick zu möglichen Verbesserungen gegeben.
10 |
11 | \subsection{AntMe!}
12 | AntMe! lässt sich in die Kategorie der Programmierspiele einordnen.
   | Hierbei handelt es sich um eine spezielle Ausprägung von
   | Computerspielen, welche während der Spielpartie keine Interaktion von
   | dem User zulassen. Der Spieler muss hingegen das Verhalten der
   | Spielfigur programmieren.\footcite[Vgl.][]{}{Wikipedia}
13 |

```

- 14 Bei AntMe! sind die Spielfiguren Ameisen, jedoch hat der Spieler nicht nur Einfluss auf das Verhalten einer einzigen Ameise sondern ist für die Steuerung eines gesamten Ameisenvolkes zuständig. Dem Spieler ist es möglich das Verhalten einer Ameise des Ameisenvolkes innerhalb eines vorgegebenen Rahmens durch Programmieren exemplarisch zu bestimmen. Jede Ameise des Ameisenvolkes verhält sich exakt nach dem gleichen Programmablauf, daher ist es wichtig, dass die Ameisen möglichst flexibel programmiert werden. Der Programmablauf von AntMe! ist ereignisgesteuert, sodass sich die Ameise immer entsprechend der aktuellen Situation verhalten kann. Weiterhin gibt es einen Mechanismus, der eine Individualisierung der Handlungsweise einer Ameise erlaubt. Hierbei handelt es sich um sogenannte Kasten, denen eine Ameise zugehören kann. Eine Kaste zeichnet sich durch eine Spezialisierung in bestimmten Bereichen aus. Zudem ist es möglich, für jede Kaste eine spezifische Verhaltensweise zu programmieren.
- 15
- 16 Das Spielkonzept von AntMe! ist der Realität nachempfunden. Als Ausgangssituation bei AntMe! startet jedes Ameisenvolk an seinem Ameisenbau. Ein Ameisenvolk kann (standardmäßig) maximal aus 100 Ameisen bestehen. Die Ameisen werden in einem festgelegten Abstand geboren. Weitere Spielelemente sind bspw. Zucker und Äpfel, die von den Ameisen gesammelt und zurück in den Bau gebracht werden können, um Punkte zu erhalten. Die erhaltenen Punkte sind beim Zucker abhängig von der Maximallast einer Ameise, diese ist durch Spezialisierung\footnote{Die Ameisen können bspw. auf das Sammeln von Nahrung spezialisiert werden. Diese Methodik wird in Zusammenhang mit der Strategie in Kapitel 3 näher betrachtet.} änderbar. Wie in der Natur haben auch die Ameisen in AntMe! einen natürlichen Gegenspieler, dieser wird in Form von Wanzen dargestellt. Wanzen sind dauerhaft auf der Suche nach Ameisen, um diese zu töten. Ameisen können diese ebenfalls, wie auch gegnerische Ameisen, angreifen und töten, wodurch Punkte generiert werden können. Stirbt eine Ameise durch eine andere, feindliche Ameise, so bekommt der Angreifer zusätzliche Punkte und dem Verlierer werden Punkte abgezogen.
- 17
- 18 \begin{table}[hbt]
- 19 \centering
- 20 \begin{minipage}[t]{.6\textwidth} % Breite der Tabelle
- 21 \caption{Punktevergabe in AntMe! im Überblick} % Überschrift
- 22 \begin{tabularx}{\columnwidth}{rX}
- 23 \toprule
- 24 Art & Punkte\\
- 25 \midrule
- 26 Apfel & +250\\
- 27 Zucker & +4 bis +10\\
- 28 Wanze & +150\\
- 29 Feindl. Ameise & +5\\
- 30 Tod durch feindl. Ameise & -5\\
- 31 \bottomrule
- 32 \end{tabularx}
- 33 \source{Eigene Darstellung in Anlehnung an \cite{AntMeWiki3}} % Quelle
- 34 \label{tab:points}
- 35 \end{minipage}
- 36 \end{table}
- 37
- 38 Des Weiteren ist zwischen dem Programm AntMe! und dem AntMe!-Framework zu unterscheiden. Bei der Anwendung von AntMe! handelt es sich um das eigentliche Spiel, also die Umgebung in der die programmierten

Ameisenvölker in Einzel- oder Mehrspieler-Partien antreten können. Das AntMe!-Framework hingegen stellt die grundlegenden Funktionalitäten für die Programmierung der Ameisen dar. AntMe! ist in C\# programmiert und kann daher in den von dem .Net-Framework unterstützten Sprachen programmiert werden.

Anhang 4.6 Kapitel 3 - Dokumentation

```

1 %!TEX root = ../agi_mfws414ali.tex
2 \section{Dokumentation des AntMe!-Projektes}
3 \label{doku}
4
5 Im folgenden Kapitel wird die im Rahmen der des AntMe!-Projektes
  umgesetzte Strategie zunächst grundlegend beschrieben, bevor die
  Strategieentscheidungen im Einzelnen erläutert und begründet werden.
  Schließlich werden diverse Messreihen herangezogen, um die erreichten
  Punktzahlen hinsichtlich der Anforderung an die Höhe bewerten zu kö
  nnen.
6
7 \subsection{Strategie}
8 Dieser Abschnitt erläutert die Strategie zunächst im Überblick, bevor die
  Umsetzung der Strategie im Detail behandelt wird. Die verschiedenen
  Domänen von AntMe!, wie die Fortbewegung oder der Kampf werden dabei
  einzeln betrachtet und die Stellen, wo diese in einander übergreifen,
  aufgezeigt. Als erstes wird erklärt, wie die Kasten realisiert wurden,
  gefolgt von den Grundfunktionalitäten der Fortbewegung, der
  Kommunikation und des Ticks. Sind die elementaren Konzepte der
  Strategie erklärt, wird das Verhalten der Ameisen im Hinblick auf die
  Nahrung und den Kampf dargelegt.
9
10 \subsubsection{Überblick} \label{ssec:overview}
11 Der grundlegende Fokus der vorliegenden Strategie liegt auf dem Kampf
  gegen Wanzen und feindliche Ameisen, die Spezialisierung der Ameisen
  ist entsprechend daraufhin ausgerichtet. Dennoch ist das Sammeln von
  Nahrung ein essentieller Bestandteil der Strategie. Aus taktischen Grü
  nden hinsichtlich der Nahrungssuche ist das Ameisenvolk in zwei Kasten
  aufgeteilt. Die Kasten unterscheiden sich einzig durch das Verhalten,
  die Spezialisierungen sind hingegen identisch. Dabei ist die eine
  Kaste stärker auf das Sammeln von Zucker ausgerichtet als die andere.
12
13 Die Spezialisierung auf den Kampf ist ein Kompromiss, da das Ameisenvolk
  sowohl im Einspieler-Modus als auch im Mehrspieler-Modus erfolgreich
  sein soll. Im Einzelspiel sind durch die Fokussierung auf die
  Nahrungssammlung i.d.R. höhere Punktzahlen erzielbar. Eine solche
  Strategie ist im Mehrspieler jedoch nur begrenzt tauglich. Einerseits
  kommt es zu erheblichen Problemen, wenn das gegnerische Ameisenvolk
  auf den Kampf gegen Ameisen ausgelegt ist. Andererseits kann dies in
  bestimmten Fällen ebenfalls nachteilig sein, wenn der Gegenspieler
  gleichermaßen ausschließlich Nahrung sammelt. Da das Spielfeld jede
  Runde erneut zufällig aufgebaut wird, kann es vorkommen, dass der
  Gegner näher an der Nahrung gelegen ist. Dadurch, dass die eigenen
  Ameisen weitere Wege zurücklegen müssen, um die Nahrung einzusammeln,
  ist eine Niederlage in diesen Situationen unausweichlich. Die
  Entscheidung über Sieg oder Niederlage ist in der Konstellation also
  relativ willkürlich, vorausgesetzt die feindlichen Ameisen sind etwa

```

gleich stark im Sammeln. Im Umkehrschluss bedeutet dies, dass die Kampf-Spezialisierung entscheidend im Mehrspieler ist. Zum einen sind die Ameisen so in der Lage sich gegen feindliche Ameisen zu verteidigen. Des Weiteren sind sie klar im Vorteil gegenüber Ameisen, die nicht im Kampf spezialisiert sind.

14
15 \subsubsection{Kasten}

16 Das Ameisenvolk \textit{LipkeAnts} wurde in zwei Kasten unterteilt. Die Kaste \textit{NormalAnt} ist die Standardkaste, also die mit dem größten Bevölkerungsanteil. Zusätzlich gibt es die Kaste \textit{SugarAnt}, welche einen stärkeren Fokus auf dem Sammeln von Zucker hat.

17
18 \subheading{Spezialisierungen}

19
20 Eine Ameise in AntMe! besitzt bestimmte Fähigkeiten bzw. Eigenschaften\footnote{In AntMe! wird von Fähigkeiten gesprochen, da es sich bei bereits Eigenschaften um ein Sprachkonstrukt in den Programmiersprachen C\# und Visual Basic handelt.}, wie bspw. die Bewegungsgeschwindigkeit, die Lebenspunkte und weitere. Jede Fähigkeit kann verbessert oder verschlechtert werden, jedoch müssen die Fähigkeiten insgesamt ausgeglichen sein, dies wird Spezialisierung genannt\footcite{AntMeWiki1}.

21
22 Den Fähigkeiten können Punkte im Wertebereich von -1 bis +2 vergeben werden. Dabei muss die Summe aller Fähigkeitswerte ≤ 0 sein, sodass keine ungerechte Verteilung möglich ist und die Spezialisierung gut durchdacht sein muss. Standardmäßig sind alle Fähigkeiten gleich null gesetzt (siehe Tabelle ~\ref{tab:standardProps}).

23
24 \begin{table}[hbt]

25 \centering

26 \begin{minipage}[t]{.7\textwidth} % Breite der Tabelle

27 \caption{Fähigkeiten ohne Spezialisierung (Standard)} % Überschrift

28 \begin{tabularx}{\columnwidth}{rXr}

29 \toprule

30 Fähigkeit & Beschreibung & Wert\\

31 \midrule

32 Geschwindigkeit & Schritte pro Runde & 4\\

33 Drehgeschwindigkeit & Grad pro Runde & 8\\

34 Last & Einheiten Nahrung & 5\\

35 Sichtweite & Schritte & 60\\

36 Reichweite & Schritte & 2250\\

37 Energie & Lebenspunkte & 100\\

38 Angriff & Lebenspunkte pro Runde & 10\\

39 \bottomrule

40 \end{tabularx}

41 \source{Eigene Darstellung in Anlehnung an \cite{AntMeWiki3}} % Quelle

42 \label{tab:standardProps}

43 \end{minipage}

44 \end{table}

45
46 Die Kasten \textit{NormalAnt} und \textit{SugarAnt} des Ameisenvolkes \textit{LipkeAnts} haben beide die gleiche Spezialisierung (siehe Tabelle \ref{tab:lipkeAntsProps}). Dies hat den Grund, dass die Ausrichtung auf den Kampf ein wichtiger Bestandteil der Strategie darstellt, um im Mehrspieler-Modus zu bestehen. Zwar wäre es eine Möglichkeit auf die Gegebenheiten des aktuellen Spieles zu reagieren, indem nur noch Ameisen geboren werden die gegen die feindlichen

Ameisen effektiv sind. Diese dynamische Anpassung während der Laufzeit ist jedoch nicht möglich, da es sich bei den \textit{LipkeAnts} um nicht-statische Ameisen handelt. Statische Ameisen verfügen über die Möglichkeit ein globales Gehirn, in Form von statischen Variablen, auf die jede Ameise zugreifen kann, nachzubilden. Bei nicht-statischen Ameisen ist zwar auch eine Kommunikation über Duftmarken möglich (siehe \ref{sssec:communication}), auf diese kann aber erst reagiert werden, wenn die Ameise bereits geboren ist.

```

47
48 \begin{table}[hbt]
49 \centering
50 \begin{minipage}[t]{.6\textwidth} % Breite der Tabelle
51 \caption{Spezialisierung \textit{LipkeAnts}} % Überschrift
52 \begin{tabularx}{\columnwidth}{rXr}
53 \toprule
54 Fähigkeit & Punkte & Wert\\
55 \midrule
56 Geschwindigkeit & 0 & 4\\
57 Drehgeschwindigkeit & -1 & 6\\
58 Last & -1 & 4\\
59 Sichtweite & -1 & 45\\
60 Reichweite & -1 & 1800\\
61 Energie & 2 & 250\\
62 Angriff & 2 & 30\\
63 \bottomrule
64 \end{tabularx}
65 \source{Eigene Darstellung in Anlehnung an \cite{AntMeWiki3}} % Quelle
66 \label{tab:lipkeAntsProps}
67 \end{minipage}
68 \end{table}
69
70 Bei der Spezialisierung der \textit{LipkeAnts} wurde für die Fähigkeiten
    Angriff und Energie die maximale Punktzahl vergeben. Die Ameisen sind
    somit im Vorteil gegen im Kampf gegen die meisten Spezialisierungen
    anderer Völker, einzig Ameisenvölker, die ebenfalls jeweils zwei
    Punkte auf Angriff und Energie gesetzt haben sind in der Lage gegen
    die \textit{LipkeAnts} zu bestehen. Im Falle von gleich starken
    Ameisen gehen die Kämpfe meistens unentschieden aus (beide Ameisen
    sterben bei dem Kampf), höchstens wenn eine der Ameisen bereits geschwächt
    in den Kampf geht verliert diese. Oder aber wenn eine der Ameisen
    zuerst angreift, da die andere vorher z.B. mit dem Tragen eines
    Apfels beschäftigt war. Weiterhin wurden die Fähigkeiten
    Drehgeschwindigkeit, Last, Sichtweite und Reichweite auf den
    Minimalwert gesetzt. Diese Fähigkeiten sind bei einer Ausrichtung auf
    den Kampf vernachlässigbar. Zudem ist aufgrund der Regeln für die
    Spezialisierung nur ein Punkt übrig, wenn zwei andere Werte bereits
    auf zwei gesetzt sind. Von den fünf restlichen Fähigkeiten ist die
    Geschwindigkeit neben der Energie und dem Angriff die wichtigste für
    die hier verfolgte Strategie und wurde daher auf null, also den
    Standardwert gesetzt. Die Geschwindigkeit ist wichtig, weil die
    Ameisen so schneller Nahrung befördern können. Auch im Kampf ist die
    Geschwindigkeit nicht zu vernachlässigen, da Ameisen, die die
    Geschwindigkeit auf minus eins gesetzt haben zu langsam sind, um eine
    Wanze einzuholen oder gegnerische Ameisen zu verfolgen.
71
72 \subheading{Bestimmung der Kaste}
73

```

```

74 | Jedes Mal, wenn eine Ameise geboren wird, muss dieser Ameise eine Kaste
    | zugewiesen werden. Zu diesem Zweck wird die Methode \textit{
    | ChooseCaste} verwendet. Innerhalb dieser Methode kann auf die aktuelle
    | Anzahl an Ameisen in den jeweiligen Kasten zugegriffen werden. Im
    | Falle der \textit{LipkeAnts} wird mit \textit{ChooseCaste} der Anteil
    | der Kaste \textit{NormalAnt} auf 75 Prozent bzw. bei der Kaste \textit{
    | SugarAnt} auf 25 Prozent reguliert.
75 |
76 | \subheading{Verhalten}
77 |
78 | Wie bereits erwähnt, unterscheidet sich das Verhalten der Kasten \textit{
    | NormalAnt} und \textit{SugarAnt}. Im Wesentlichen handelt es sich
    | dabei um die Fortbewegung, den Umgang mit Zucker sowie das Verhalten
    | bei Wanzen. Ameisen der Kaste \textit{SugarAnt} passen zudem ihr
    | Verhalten dynamisch an die gegebene Situation an. Wird eine Ameise aus
    | dieser Kaste neu geboren, so setzt diese einen stärkeren Fokus auf
    | den Zucker als die Ameisen der Kaste \textit{NormalAnt}. Sobald jedoch
    | erkannt wird, dass feindliche Ameisen im Spiel sind, ändert die
    | Ameise der Kaste \textit{SugarAnt} ihr Verhalten, um sich von diesem
    | Zeitpunkt an wie eine Ameise der Kaste \textit{NormalAnt} zu verhalten.
    | Die Unterschiede in diesen Verhaltenspunkten werden in den folgenden
    | Abschnitten genauer betrachtet.
79 |
80 | \subsubsection{Fortbewegung}
81 | Die Fortbewegung der Ameisen in AntMe! wird mithilfe dem Konzept der
    | Zustandsmaschine realisiert. Eine Ameise kann sich in den Zuständen \
    | \textit{Warten}, \textit{Drehen}, \textit{Gehen} und \textit{Ziel}
    | verfolgen} befinden (siehe Abbildung \ref{img:states}). Der
    | Ausgangszustand einer Ameise ist \textit{Warten}, der Wechsel eines
    | Zustandes in den nächsten kann explizit durch Befehle wie \textit{
    | GoForward} oder implizit durch bestimmte Ereignisse hervorgerufen
    | werden.\footcite[Vgl.] []{AntMeWiki2}
82 |
83 | \begin{figure}[hbt]
84 | \centering
85 | \begin{minipage}[t]{0.5\textwidth} % Breite, z.B. 1\textwidth
86 | \caption{Statusablauf einer Ameise} % Überschrift
87 | \includegraphics[width=1\textwidth]{img/AntStates.png}\\ % Pfad
88 | \source{\cite{AntMeWiki2}} % Quelle
89 | \label{img:states}
90 | \end{minipage}
91 | \end{figure}
92 |
93 | \subheading{Wartet}
94 |
95 | Befindet sich eine Ameise im Zustand \textit{Warten}, hat diese weder ein
    | Ziel noch den Befehl eine bestimmte Strecke zu gehen oder sich zu
    | drehen, sodass die Ameise still steht und auf weitere Befehle wartet.
    | Jede Runde wird das Ereignis \textit{Waiting} aufgerufen, dieses
    | Ereignis stellt die zentrale Steuerung der Fortbewegung dar. Aus
    | strategischen Gründen sollen sich die Ameisen der Kasten \textit{
    | NormalAnt} und \textit{SugarAnt} unterschiedlich bewegen. \textit{
    | NormalAnt}-Ameisen sollen zunächst 40 Schritte gerade aus laufen und
    | im Anschluss eine Drehung um einen zufälligen Wert zwischen -10 und
    | +10 ausführen. Dies hat zur Folge, dass die Ameisen relativ verstreut
    | über das Spielfeld laufen und somit in der Lage sind eine größere Flä
    | che abzusuchen. Die erweiterten Suchbereiche der Ameisen führen zu
    | einer höheren Wahrscheinlichkeit, dass sie auf einen Apfel, eine Wanze

```

oder weitere Spielelemente trifft. Die \textit{SugarAnt}-Ameisen zwar ebenfalls einen Richtungswechsel vornehmen, jedoch erst nach 150 Schritten, da diese eine Spur zum Zucker suchen sollen. Wenn in der aktuellen Richtung nach 150 Schritten keine Spur zu finden ist, sollen die Ameisen sich um einen zufälligen Winkel zwischen -15 Grad bis +15 Grad drehen. Der Winkel ist größer gewählt, da die Zuckerhaufen teils recht weiträumig über das Spielfeld verteilt sind. Ist eine \textit{SugarAnt}-Ameise allerdings in Kenntnis darüber, dass feindliche Ameisen existieren, soll diese sich wie eine \textit{NormalAnt}-Ameise verhalten.

96
97 \subheading{Tick}
98
99 Neben dem Wartet-Ereignis ist das Tick-Ereignis eines der wichtigsten Ereignisse für die Bewegungssteuerung der Ameisen. Da im Tick-Ereignis jedoch noch weitere Aktionen abgearbeitet werden, wird dieses in einem gesonderten Abschnitt behandelt (siehe \ref{sssec:tick}).

100
101 \subheading{Wird müde}
102
103 Sobald die restliche Strecke einer Ameise ein Drittel der verfügbaren Schritte erreicht hat wird das Ereignis \textit{GettingTired} aufgerufen. Zu diesem Zeitpunkt könnte man der Ameise befahlen zum Ameisenbau zurückzukehren, damit diese nicht verhungert. Denn wenn eine Ameise die maximale Reichweite erreicht hat, stirbt diese am Hungertod. Dies wirkt sich i.d.R. nachteilig auf die Endpunktzahl aus, da die Ameise nicht optimal eingesetzt wurde. Wenn eine Ameise bspw. zum Todeszeitpunkt ein Zucker zum Bau transportiert, lässt es diesen fallen. Zucker der fallen gelassen wird ist verloren und kann nicht mehr von anderen Ameisen aufgesammelt werden. Diese Ameise hat in einer Zeitspanne, in der eine Ameise mit genügend Reststrecke Punkte erzielt hätte keine Punkte erzielt und somit Zeit verschwendet.

104
105 Nach einigen Testläufen hat sich jedoch herausgestellt, dass die ein Drittel Marke oftmals nicht ausreichend war, um den Hungertod zu verhindern. Weiterhin kann es sein, dass der Bau relativ weit mittig gelegen ist, sodass die Ameise bei einem Drittel Reststrecke zum Bau zurückkehrt, obwohl die Distanz zum Ameisenhügel weniger als diese Reststrecke beträgt. Aus diesem Grund wurde auf den Gebrauch dieses Ereignisses verzichtet und ein eigener Mechanismus innerhalb des Tick-Ereignisses realisiert, der bezweckt, dass die Ameisen rechtzeitig zum Bau zurückkehren (siehe \ref{sssec:tick}).

106
107 \subheading{Ist gestorben}
108
109 Ein weiteres Ereignis, welches zu den Ereignissen der Fortbewegung zählt, ist der Todeszeitpunkt, denn der Tod wird als letzte Bewegung der Ameise gezählt. Ist eine Ameise gestorben, kann diese keine weiteren Aktionen ausführen. Da es sich bei dem Volk \textit{LipkeAnts} jedoch um nicht-statische Ameisen handelt, findet das Ereignis \textit{HasDied} keine Anwendungsfälle. Demgegenüber sind statische Ameisen in der Lage die Todesart im globalen Gedächtnis (statische Variablen) zu hinterlegen, sodass die anderen Ameisen darauf reagieren können. Mögliche Anwendungsfälle wäre bspw. das Zählen der aufgetretenen Todesarten, sodass bei der Bestimmung der Kaste nur noch Kasten ausgewählt werden, die eine Antwort auf eine übermäßige Todeszahl einer bestimmten Todesart darstellen.

110

```

111 \subsubsection{Kommunikation} \label{sssec:communication}
112 Die Kommunikation zwischen den Ameisen ist ein zentraler Mechanismus, ohne
    den ein erfolgreiches Ameisenvolk nicht realisierbar ist. In dem Groß-
    teil der den Ameisen zur Verfügung stehenden Ereignisse wird die
    Kommunikation verwendet.
113
114 Ameisen sind in der Lage Duftmarken abzusetzen und diese Markierungen mit
    Informationen anzureichern. Andere Ameisen aus dem eigenen Volk können
    diese Duftmarken riechen und die Informationen daraus auslesen. Es
    besteht zudem die Möglichkeit, die Größe der Markierung festzulegen.
    Die Größe hat Einfluss auf die Dauer, für die eine Duftmarke bestehen
    bleibt. Diese beiden Faktoren sind wichtig, um die Anzahl an Ameisen,
    die auf die Markierung aufmerksam werden, zu regulieren. Je größer die
    Duftmarke, desto mehr Ameisen können diese wahrnehmen, allerdings lö-
    st sie sich auch schneller wieder auf. Daher ist die richtige Größe
    oft entscheidend und ermöglicht unterschiedlichste Taktiken. Besonders
    nicht-statische Ameisen, wie die \textit{LipkeAnts} sind stark von
    der Kommunikation über Duftmarken abhängig. Es können nur die
    Duftmarken der Ameisen aus dem eigenen Volk erkannt werden.
115
116 Bei den Informationen, die über eine Duftmarke übermittelt werden können,
    handelt es sich lediglich um einen 32 Bit Integer-Wert. Die hieraus
    resultierenden Möglichkeiten sind somit standardmäßig sehr begrenzt. F-
    ür simple Strategien reicht das i.d.R. aus, da dort oft nur die
    Richtung in der sich ein Spielelement (Nahrung, Gegner) befindet ü-
    bergeben wird. Dabei handelt es sich um gleichartige Informationen.
    Die in dieser Strategie verfolgten Ansätze erfordern jedoch neben der
    eigentlichen Information zusätzlich die Möglichkeit die Art der
    Information zu unterscheiden, um divergente Informationen zu ü-
    bertragen. Zu diesem Zweck wurde die Methode \textit{
    CreateMarkerInformation} (siehe Listing \ref{list:
    createMarkerInformation}) entworfen. Dieser Methode werden die
    Information sowie der Typ der Information übergeben. Der
    Informationstyp ist wie die Information auch ein Integer-Wert, jedoch
    kann die Information verschieden lang sein, wohingegen der
    Informationstyp auf eine einstellige Zahl festgelegt wurde (als
    Klassen-Member definiert). Es wird zwischen folgenden
    Informationstypen unterschieden:
117 \begin{compactitem}
118   \item Es wurde ein Apfel gesehen
119   \item Es wurde ein Zuckerhaufen gesehen
120   \item Es wurde eine Wanze gesehen
121   \item Es wurde eine feindliche Ameise gesehen
122 \end{compactitem}
123
124 \begin{figure}[bht]
125 \begin{lstlisting}[caption=Methode zum Erstellen verbesserter Marker-
    Informationen, label=list:createMarkerInformation]
126 private int CreateMarkerInformation(int information, int infoType)
127 {
128     string _infoType = infoType.ToString();
129     string _information = information.ToString().PadLeft(4, '0');
130     string advancedInformation = _infoType + _information;
131
132     return Convert.ToInt32(advancedInformation);
133 }
134 \end{lstlisting}
135 \end{figure}

```

```

136
137 \textit{CreateMarkerInformation} wandelt die beiden Integer-Werte zunächst
    in einen String, damit diese aneinandergefügt werden können, bevor
    sie wieder in einen Integer konvertiert werden. Dies ist notwendig, um
    einerseits die Information mit führenden Nullen aufzufüllen, da die
    Informationen unterschiedlich lang sind. Des Weiteren kann so die spä-
    tere Extraktion erleichtert werden, denn die zuvor zusammengeführten
    Strings müssen später lediglich wieder getrennt werden. Für die
    Extraktion wurde die Hilfsklasse \textit{MarkerInformation} entwickelt,
    welche eine zielführende Datenhaltung realisiert (siehe Listing \ref{
    list:markerInformation}).
138
139 \begin{figure}[bht]
140 \begin{lstlisting}[caption=Hilfsklasse zum Extrahieren der verbesserten
    Marker-Informationen, label=list:markerInformation]
141 public class MarkerInformation
142 {
143     public MarkerInformation(int information)
144     {
145         string info = information.ToString();
146         this.Data = Convert.ToInt32(info.Substring(1));
147         this.InfoType = Convert.ToInt32(info.Substring(0, 1));
148     }
149
150     public int Data { get; set; }
151
152     public int InfoType { get; set; }
153 }
154 \end{lstlisting}
155 \end{figure}
156
157 Wenn eine Ameise die Duftmarke einer anderen Ameise riecht, kann sie auf
    die über die Markierung übermittelte Information zugreifen. Im Falle
    der \textit{LipkeAnts} muss diese Information zuerst noch mithilfe der
    Klasse \textit{MarkerInformation} in Hauptinformation und
    Informationstyp zerlegt werden. Anhand des Informationstyps wird dann
    entschieden, wie mit der Information verfahren werden soll. Wurde über
    die Duftmarke mitgeteilt, dass sich andere Ameisenvölker im Spiel
    befinden, merkt sich die Ameise diese Information, damit sie das
    Verhalten dementsprechend anpassen kann. Handelt es sich um eine Apfel-
    -, Zucker- oder Wanzen-Markierung, werden der Ameise jeweils zielfü-
    hrende Befehle erteilt.
158
159 \subheading{Apfel-Markierung}
160
161 Vorausgesetzt die Ameise trägt zur Zeit keinen Apfel oder Zucker, hat kein
    Ziel und die Anzahl an Ameisen aus dem eigenen Volk im 360 Grad
    Sichtfeld ist geringer als die Anzahl an benötigten Träger für den
    Apfel, soll die Ameise zum Mittelpunkt der Duftmarke gehen. Es wird
    darauf gesetzt, dass die Ameise entweder den Apfel entdeckt oder eine
    weitere Markierung auffindet, sobald sie in der Mitte der Duftmarke
    angekommen ist. Die Anzahl der Träger, die für den Apfel notwendig
    sind wird als Information über die Duftmarke übermittelt. Andernfalls
    soll die Ameise stoppen, wodurch sie das Ziel verliert und sich auf
    andere Aufgaben konzentrieren kann.
162
163 \subheading{Zucker-Markierung}
164

```

165 Bei einer Zucker-Markierung soll wie auch bei der Apfel-Markierung überprüft werden, dass die Ameise weder Last noch Ziel hat. Zusätzlich sollen nur \textit{SugarAnt}-Ameisen auf die Markierung reagieren. Zuletzt wird abgefragt, ob die über die Duftmarke übertragene Richtung des Zuckers eine andere ist, als die aktuelle Richtung der Ameise. Denn wenn alle Abfragen zutreffend sind, soll die Ameise sich in die Richtung des Zuckers drehen und 150 Schritte geradeaus laufen, da die Spur zum Zucker eventuell nicht präzise genug ist, um die Ameise zielsicher zum Zucker zu führen. Deswegen soll die Ameise eine längere Strecke gerade aus laufen um dem vorzubeugen. Für eine stabile Zuckerstraße ist der Anteil der \textit{SugarAnt}-Ameisen mit 25 Prozent zu gering.

166 \subheading{Wanzen-Markierung}

167

168

169 Wird eine Wanzen-Markierung entdeckt wird als erstes geprüft, ob sich weniger als acht Ameisen aus dem eigenen Volk in der Nähe befinden. Sind es mehr wird die Markierung ignoriert, weil nur eine begrenzte Anzahl an Ameisen für das Besiegen einer Wanze notwendig sind. Zusätzlich muss eine von folgenden vier Bedingungen zutreffen, damit die Ameise auf die Duftmarke reagiert:

170 \begin{compactitem}

171 \item Die Ameise hat kein Ziel,

172 \item \textbf{oder} die Ameise hat keine Wanze zum Ziel \textbf{und} trägt keinen Apfel,

173 \item \textbf{oder} es handelt sich um eine \textit{NormalAnt}-Ameise \textbf{und} sie befördert Zucker,

174 \item \textbf{oder} es handelt sich um eine \textit{SugarAnt}-Ameise, die sich wie eine normale Ameise verhalten soll \textbf{und} sie befördert Zucker.

175 \end{compactitem}

176

177 Die Zusatzbedingungen bilden also zahlreiche Fälle ab, in denen die Ameise auf die Duftmarke reagieren soll. So soll sie die Markierung ignorieren, wenn sie einen Apfel trägt, aber wahrnehmen, wenn sie Zucker trägt (verallgemeinert), denn Wanzen und auch Äpfel bringen aufgrund der Spezialisierung auf Kampf mehr Punkte ein, als Zucker. Auch wenn die Ameise auf dem Weg zu einer feindlichen Ameise oder einem Marker ist, soll sie die Wanzen-Markierung wahrnehmen, da Wanzen Priorität haben. Zudem soll die Ameise nicht abgelenkt werden, wenn sie bereits einer Wanze folgt. Ist eine Kombination zutreffend, soll die Ameise den Zucker fallen lassen, falls sie welchen trägt und in die Mitte der Markierung laufen.

178

179 Über die Möglichkeit Duftmarken zu riechen hinaus, existieren zudem Ereignisse, die ausgelöst werden, wenn eine Ameise aus dem eigenen Volk bzw. aus der eigenen Kaste gesehen wurden (\textit{SpotsFriend}, \textit{SpotsTeammate}). Diese können dazu genutzt werden, um gezielt mit den Ameisen im eigenen Umkreis zu kommunizieren. In der hier vorgestellten Strategie werden diese jedoch nicht verwendet. Dies hat den Grund, dass bereits für alle wichtigen Informationen entsprechende Duftmarken erstellt werden, wie z.B. \gqq{In Richtung x befindet sich Zucker!}, \gqq{Es gibt einen Apfel in der Nähe der noch Träger braucht!}. Mögliche Anwendungsfälle wären beispielsweise die Übermittlung von Informationen zwischen den Ameisen, wie \gqq{Ich trage einen Apfel, hilf mir!}. Darüber hinaus könnten diese Ereignisse dazu genutzt werden die Bildung von Gruppen zu koordinieren, allerdings sieht diese Strategie keine Gruppenbildung vor. Und Informationen über

die Spielelemente (Äpfel, Zucker, ...) werden bereits auf eine effektivere Art kommuniziert. Weil eine Ameise nahezu dauerhaft von befreundeten Ameisen umgeben ist, werden die Ereignisse viel zu oft aufgerufen, wodurch die Markierungen sich überlagern und nicht mehr präzise sind, die Ameisen würden durch den vielen Input zu sehr abgelenkt.

180
181 \subsubsection{Tick} \label{sssec:tick}
182 Das \textit{Tick}-Ereignis wird ohne Ausnahme in jeder Runde aufgerufen, dies ermöglicht eine präzise Steuerung der Ameisen sowie die Realisierung zahlreicher taktischer Konzepte. Bei den \textit{LipkeAnts} wird im \textit{Tick} zwischen dem Setzen von Parametern und dem Ausführen von Aktionen unterschieden.

183
184 Innerhalb des Ereignisses \textit{Tick} wird der Parameter \textit{HasSeenForeignAnt} auf \textit{false} gesetzt, wenn basierend auf verschiedenen Anhaltspunkten davon ausgegangen werden kann, dass keine feindlichen Ameisen existieren. Dies trifft zu, wenn die Ameise bereits mehr als 200 Schritte gelaufen ist, zu dem Geburtszeitpunkt bereits über 60 Ameisen existierten und das Attribut noch \textit{null} ist. Das Attribut \textit{HasSeenForeignAnt} nimmt Einfluss auf das Verhalten der \textit{SugarAnt}-Ameisen bzw. den Umgang mit Zucker.

185
186 Des Weiteren wird auf bestimmte Gegebenheiten abgefragt, trifft eine Abfrage zu soll eine Aktion ausgeführt werden, die auf die aktuelle Situation reagiert. Die Abfragen sind über einen \textit{if-else-if}-Block umgesetzt, sodass in jeder Runde nur eine Aktion durchgeführt werden kann. Zuerst wird überprüft, ob die Ameise zum Bau zurückkehren soll, um sich aufzuladen\footnote{Bei der Rückkehr zum Ameisenbau werden alle Werte, wie Leben oder Reststrecke zurückgesetzt.}. Dies trifft zu, wenn die Ameise nur noch eine geringe Reststrecke zur Verfügung hat oder die Lebenspunkte niedrig sind. Die Reststrecke muss inklusive einem Puffer von 50 Schritten größer sein, als die Distanz zum Ameisenhügel. Im Gegensatz zu dem Ereignis \textit{GettingTired} gewährleistet dieser Mechanismus, dass die Ameise nicht verhungert. Denn der Ameisenbau kann bei jedem Spiel unterschiedlich positioniert sein, sodass ein Drittel (siehe \textit{GettingTired}) Reststrecke teils nicht ausreichend ist. Weiter müssen die Lebenspunkte mindestens zwei Drittel der Gesamtmenge betragen, dies hat den Grund, dass die Ameise sonst im Kampf gegen Gegner chancenlos stirbt.

187
188 Wenn eine Ameise noch ausreichend Reststrecke und Lebenspunkte besitzt, werden dieser Befehle erteilt, abhängig davon, ob die Ameise aktuell einen Apfel bzw. Zucker transportiert oder auf dem Weg zu einem Apfel ist. Wie die einzelnen Mechanismen für den Umgang mit Nahrungsmitteln umgesetzt sind wird in Kapitel \ref{sssec:food} im Detail erläutert.

189
190 \subsubsection{Kampf}
191 Der Hauptfokus der \textit{LipkeAnts} liegt auf dem Töten von Wanzen, da diese wie auch Äpfel 150 Punkte einbringen, im Gegensatz zum Apfel müssen die Ameisen jedoch nicht erst zum Ameisenbau zurückkehren für den Punkterhalt, sondern können sich direkt zur nächstgelegenen Wanze oder Nahrungsquelle begeben. Gegnerische Ameisen werden vordergründig aus strategischen Gründen bekämpft, so gesehen als vorbeugende Maßnahme, falls die Gegner auf den Kampf gegen andere Ameisen ausgerichtet sind (\gqq{Angriff ist die beste Verteidigung}). Jedoch bringt dieses Vorgehen indirekt noch einen weiteren Vorteil mit sich. Handelt es sich bei den gegnerischen Ameisen um neutrale, also Ameisen,

	<p>die keine feindlichen Ameisen angreifen, sind diese i.d.R. chancenlos unterlegen. Denn diese werden auf der einen Seite in ihrer Strategie (z.B. Nahrung sammeln) stark beeinträchtigt. Eine Zuckerstraße wird schwieriger aufzubauen und ein Apfel kommt oft nicht am Bau an, da die Ameisen vorher getötet wurden. Weiterhin verlieren die Gegner für jede getötete Ameise Punkte, wohingegen die \textit{LipkeAnts} Punkte gewinnen.</p>
<p>192 193</p>	<p>Wird eine feindliche Ameise entdeckt, sollen die befreundeten Ameisen darüber benachrichtigt werden. Diese Information ist wichtig für das bereits erwähnte dynamische Verhalten der Ameisen mit Hinblick auf den Umgang mit Zucker, welches von der Existenz feindlicher Ameisen abhängt. Über dies hinaus werden alle Ameisen, die keinen Apfel befördern dazu angewiesen die gegnerische Ameise anzugreifen. Zuvor soll jedoch der Zucker fallen gelassen werden, falls die Ameise welchen trägt, weil das Angreifen nur möglich ist, wenn die Ameise keine Last hat. Wird eine Ameise hingegen von einer gegnerischen Ameise zuerst angegriffen, soll diese sofort alle Last abwerfen, falls vorhanden, und den Gegner ebenfalls angreifen, ein Kampf ist in dieser Situation unausweichlich. Im Gegensatz zu den meisten anderen Situationen sollen keine anderen Ameisen per Duftmarke über den Kampf mit einer feindlichen Ameise informiert werden. Beide Kasten der \textit{LipkeAnts} haben die Maximalpunktzahl der für den Kampf wichtigen Fähigkeiten Angriff und Energie, sodass ein Einzelkampf im Durchschnitt siegreich bis maximal unentschieden ausgeht. Höchstens gegen mehrere Ameisen gleichzeitig würden Probleme aufkommen, dies ist aber zu vernachlässigen.</p>
<p>194 195</p>	<p>Anders als bei dem Kampf gegen andere Ameisen, werden für das Besiegen einer Wanze mehrere Ameisen benötigt, aufgrund der hohen Lebenspunkte einer Wanze (1000). Daher werden im Augenblick der Kenntnissnahme einer Wanze die befreundeten Ameisen benachrichtigt. Der Umkreis der Markierung darf nicht zu groß sein, damit nur die Ameisen informiert werden, die auch noch rechtzeitig zum Kampf kommen können. Zu klein darf die Duftmarke jedoch auch nicht sein, da sonst zu wenige Ameisen mit der Nachricht erreicht. Nach der Benachrichtigung wird entschieden wie die Ameise fortfährt, trägt sie einen Apfel, soll die Ameise die Wanze ignorieren. Der Hauptfokus gilt zwar den Wanzen, allerdings gibt es auch nur eine begrenzte Anzahl und daher ist es wichtig, dass die Ameisen zielführend auf die verschiedenen Tätigkeiten, wie Äpfel sammeln und Kämpfen aufgeteilt werden. Hat die Ameise hingegen Zucker aufgeladen, soll dieser fallen gelassen werden. Zucker wird die geringste Priorität zuteil, selbst den \textit{SugarAnt}-Ameisen wird befohlen sich auf die Wanze zu konzentrieren. Sind in der Sichtweite noch mindestens drei weitere befreundete Ameisen aufzufinden, soll die Wanze angegriffen werden. Bei einer geringeren Anzahl wird nur der Befehl zum Folgen der Wanze gegeben. Eine Ameise alleine würde nämlich nichts gegen die Wanze ausrichten können. Einzig wenn die Ameise bereits unter Angriff der Wanze steht soll sie die Wanze alleine angreifen, weil zu diesem Zeitpunkt die Flucht meist schon zu spät ist. Selbstverständlich muss auch hier zuerst das Nahrungsmittel fallen gelassen werden, sofern die Ameise eines transportiert. Finden andere Ameisen die Wanze zeitnah war das Opfer der Ameise nicht umsonst, dauert es länger hat die Wanze sich leider schon wieder regeneriert. Aus diesem Grund werden in solch einer Situation ebenfalls befreundete Ameisen über die Wanze informiert.</p>
<p>196 197</p>	<p>\subsubsection{Nahrung} \label{sssec:food}</p>

- 198 Auch wenn die Ameisen in der Spezialisierung auf Kampf ausgerichtet sind, ist das Sammeln von Nahrung unerlässlich, um auf eine hohe Punktzahl zu kommen, das Töten von Wanzen und feindlichen Ameisen alleine generiert nicht genügend Punkte. Dabei sind vor allem Äpfel sehr wichtig, da diese 150 Punkte pro Stück einbringen und die \textit{LipkeAnts} Ameisen auf der anderen Seite nur Zucker in der Höhe von vier Punkten tragen können. Die meisten Punkte werden zwar in dieser Strategie durch Äpfel und Wanzen gewonnen, allerdings existiert immer nur eine begrenzte Anzahl dieser Spielelemente, sodass einige Ameisen oft im \gqq{Leerlauf} sind (nichts zu tun haben). Um dem entgegenzuwirken soll zum einen jede Ameise, die an einem Zuckerhaufen vorbei kommt, ein Stück Zucker sammeln. Viel wichtiger aber wurden 25 Prozent der Ameisen speziell darauf ausgerichtet mehr Zucker zu sammeln.
- 199 \subheading{Apfel}
- 200
- 201
- 202 Wenn eine Ameise ein Spielelement innerhalb des 360 Grad Sichtfeldes sieht, wird ein entsprechendes Ereignis ausgelöst, so auch im Falle eines Apfels. Sieht die Ameise einen Apfel, soll sie zuvor prüfen, ob die Aktuelle Last gleich null ist, kein Ziel gesetzt ist und ob der Apfel überhaupt noch Träger braucht. Denn wenn die Ameise bspw. bereits Zucker trägt oder auf dem Weg zu einer Wanze ist, soll diese den Apfel ignorieren. Zudem erhöhen zwar mehr Träger die Geschwindigkeit, mit der der Apfel zum Ameisenbau getragen wird, aber die Anzahl, wo die Geschwindigkeit zunimmt ist nach oben hin begrenzt. Treffen die genannten Bedingungen alle zu, wird der Ameise befohlen, zu dem Apfel zu gehen und gleichzeitig andere Ameisen über eine Duftmarke auf diesen Apfel aufmerksam machen, um den Apfel schneller zum Bau tragen zu können. Mithilfe einer Duftmarke wird den anderen Ameisen die Information übermittelt, wie viele Träger noch für den Apfel gebraucht werden. In mehreren Testläufen hat sich herausgestellt, dass ab etwa fünf Ameisen der Apfel ausreichend schnell transportiert werden kann. Würden die Ameisen stärker auf das Sammeln spezialisiert sein, also eine höhere Last und eine schnellere Geschwindigkeit besitzen, würden wahrscheinlich auch weniger Ameisen ausreichen. Damit auch sichergestellt werden kann, dass immer genügend Ameisen zu Hilfe kommen, wird zunächst von acht Ameisen, die gerufen werden sollen ausgegangen. Die Zahl wurde absichtlich höher als fünf gewählt, da es vorkommen kann, dass die Ameisen, die auf die Duftmarken aufmerksam werden bereits ein Ziel oder eine Last haben. Zusätzlich werden von den acht noch die Anzahl Ameisen, die sich bereits im Sichtfeld befinden abgezogen, da davon ausgegangen wird, dass diese den Apfel ohnehin sehen. Der gewählte Radius ist mittelgroß, damit zwar genügend Helfer gefunden werden, aber nicht zu viele Ameisen abgelenkt werden, da die benötigte Trägerzahl begrenzt ist.
- 203
- 204 Während eine Ameise unterwegs zu einem Apfel ist, soll diese überprüfen, ob der Apfel wirklich noch Träger braucht, da sich dies jede Runde ändern kann. Geprüft wird dies im \textit{Tick}. Wenn der Apfel keine Träger mehr braucht, soll die Ameise stehen bleiben, wodurch sie das Ziel verliert und wieder in den Wartet-Modus übergeht.
- 205
- 206 Ist die Ameise am Apfel angekommen, wird erst überprüft, ob der Apfel noch Träger braucht, ist dies der Fall soll die Ameise den Apfel zum Ameisenhügel tragen. Nachdem die Ameise den Apfel genommen hat, wird ein weiteres Mal geprüft, ob der Apfel nun auch noch Träger braucht. Trifft dies immer noch zu, sollen andere Ameisen darüber

207	benachrichtigt werden.
208	Des Weiteren wird jede Runde im \textit{Tick} sichergestellt, dass die Ameise das Ziel (den Bau) nicht verloren hat, da es bei dem Tragen eines Apfels dazu kommen kann, dass die Ameise das Ziel verliert und die Ameisen mit dem Apfel umherirren. Weiterhin sollen andere Ameisen auf den Apfel aufmerksam gemacht werden, indem die Ameisen eine Duftspur hinter sich herziehen, der andere Ameisen zum Apfel folgen können.
209	
210	\subheading{Zucker}
211	
212	Die \textit{SugarAnt}-Ameisen legen mittels Duftmarken eine Spur zu den Zuckerhaufen, sodass eine \gqq{Zuckerstraße} entsteht, sobald mehrere Ameisen diese Spur aufgenommen haben. Eine Zuckerstraße ermöglicht ein effektives Sammeln von Zucker, da andere Zucker-Ameisen schnell auf diese aufmerksam werden. \textit{SugarAnt}-Ameisen, die während ihrer Suche keinen Zucker gefunden haben, werden aber spätestens nach der Rückkehr zum Ameisenhügel (Ameise muss sich aufladen) auf die Zuckerstraße aufmerksam.
213	
214	Sieht eine Ameise einen Zuckerhaufen, der weniger als 600 Schritte vom Ameisenbau entfernt ist, soll die Ameise zum Zucker gehen, vorausgesetzt, die Ameise hat weder eine aktuelle Last noch ein Ziel. Ist der Zucker jedoch 600 oder mehr Schritte entfernt, wird dieser ignoriert, da sich das Einsammeln des Zuckers aufgrund der Entfernung nicht lohnen würde. Hat die Ameise zudem noch keine feindlichen Ameisen gesehen, wird ihr befohlen eine Duftmarke abzusetzen, mit der Information, in welche Richtung der Zucker liegt. Die Größe der Markierung ist abhängig von der Entfernung der Ameise zum Zuckerhaufen.
215	
216	Sobald die Ameise am Zucker angekommen ist, soll diese erneut eine Markierung setzen, diesmal aber größer, als bei der Sichtung des Zuckers, da auch Ameisen die weiter weg sind benachrichtigt werden sollen, dazu muss der Radius der Markierung größer als die Sichtweite der Ameisen sein. Ist dies erledigt, erhält die Ameise den Befehl den Zucker zurück zum Ameisenhügel zu befördern.
217	
218	Trägt die Ameise ein Stück Zucker, ist unterwegs zum Ameisenbau, gehört der \textit{SugarAnt}-Kaste an und hat bisher keine feindlichen Ameisen gesehen, so soll die Ameise eine Spur zum Zucker legen. Sind bereits gegnerische Ameisen gesichtet worden, wird dem Zucker eine geringere Priorität entgegengebracht. Dies geschieht, indem sie jedes Mal im \textit{Tick} eine Duftmarke absetzt mit der Information in welcher Richtung sich der Zucker befindet. Die Richtung ist die aktuelle +180 Grad, da der Zucker sich in die entgegengesetzte Richtung vom Ameisenbau (aktuelle Richtung) befinden muss. Dadurch, dass diese Duftmarke jede Runde bis zum Erreichen des Hügel gesprüht wird, entsteht eine längere Strecke von Duftmarken, sodass die Wahrscheinlichkeit steigt, dass mehrere andere \textit{SugarAnt}-Ameisen auf die Spur zum Zuckerhaufen aufmerksam werden. Der Sprühradius der Duftmarke ist abhängig von der Entfernung zum Ameisenhügel, denn wenn die Ameise noch weiter von dem Bau entfernt ist, soll die Marke kleiner sein, damit eine möglichst präzise Spur zum Zuckerhaufen ermöglicht wird. Denn einerseits bleiben Duftmarken mit einem kleineren Radius länger erhalten, andererseits würde eine zu breite Spur ungenauer werden. Ist die Entfernung zum Bau unter 50 Schritten,

soll der Sprühradus deutlich größer ausfallen, damit Ameisen, die gerade vom Ameisenbau kommen direkt die Spur aufnehmen können.

219
220 \subsection{Messwerte}
221 In diesem Abschnitt sollen die \textit{LipkeAnts} hinsichtlich ihrer Gebrauchstauglichkeit getestet werden, Kriterium ist dabei das Erreichen von hohen Punktzahlen. Zu diesem Zweck werden sowohl in der Kategorie Einzelspiel als auch im Mehrspieler Messreihen durchgeführt. Diese Messreihen sind zielführend, um eine Einordnung zu erhalten, was als \textit{gute} Punktzahl gilt, denn gut ist in diesem Falle relativ.

222
223 \subsubsection{Rahmenbedingungen der Messreihe}
224 Die Messgruppe besteht aus einigen der im AntMe!-Spiel enthaltenen Demo-Ameisen. Hier wurden ausschließlich nicht-statische Ameisenvölker ausgewählt\footnote{In der Anwendung von AntMe! werden diese zwar als statische Ameisenvölker ausgewiesen, laut dem Quelltext auf sind diese jedoch nicht-statisch, siehe dazu \cite{GitHub2016}.}, da das im Rahmen dieser Ausarbeitung entwickelte Volk \textit{LipkeAnts} ebenfalls ein nicht-statisches ist. Statische und nicht-statische Völker können nicht aussagekräftig verglichen werden, da diese sich grundlegend von der Spielweise unterscheiden.

225
226 \pagebreak
227 In der Messgruppe enthaltene Ameisenvölker:
228 \begin{compactitem}
229 \item \textit{LipkeAnts}
230 \item aTomApfelmeisen
231 \item aTomGruppenmeisen
232 \item aTomKampfmeisen
233 \item aTomZuckermeisen
234 \end{compactitem}

235
236 Die Messgruppe deckt die Spezialisierung auf das Sammeln von Nahrung mit den Ausprägungen nur Apfel oder nur Zucker ab. Das Volk \textit{aTomGruppenmeisen} ist in zwei Kasten aufgeteilt. Die eine Kaste konzentriert sich auf das Sammeln von Äpfeln und auch Zucker, die andere verteidigt die Sammler gegen Wanzen. Das letzte der Demo-Ameisenvölker \textit{aTomKampfmeisen} ist ausschließlich auf den Kampf mit Wanzen und feindlichen Ameisen spezialisiert. Mit dieser breiten Testabdeckung ist ein aussagekräftiges Ergebnis über das Verhalten der \textit{LipkeAnts} möglich.

237
238 Eine hohe Anzahl an Durchläufen ermöglicht es Ungenauigkeiten auszuschließen, sodass eine repräsentative Messreihe erstellt werden kann. Dabei ist zu beachten, dass die Durchläufe jeweils eine ausreichende Länge aufweisen (Anzahl Runden). Aus diesem Grund werden für jedes Testszenario 100 Durchläufe mit je 5000 Runden\footnote{5000 Runden für ein Spiel sind die Standardeinstellungen von AntMe!} durchgeführt. Streuungen können durch den zufälligen Aufbau des Spielfeldes auftreten. Dabei kann es bspw. dazu kommen, dass der Ameisenbau eine besonders kurze oder weite Entfernung zum Zucker aufweist.

239
240 \subsubsection{Einzelspieler}
241 Die \textit{LipkeAnts} schneiden mit durchschnittlich annähernd 10000 Punkten sehr gut ab und sind das beste Volk in der Testreihe. Und das, obwohl es auf den Kampf spezialisiert ist. Das die Ausrichtung auf den Kampf im Einzelspieler-Modus nachteilig ist, lässt sich an dem

Volk \textit{aTomKampfmeisen} erkennen, welches mit durchschnittlich etwa 5000 Punkten den letzten Platz belegt. Dies wurde in der vorliegenden Strategie berücksichtigt und eine Balance zwischen dem Bekämpfen von Wanzen und dem Sammeln von Nahrung hergestellt. Auch wenn das Sammeln von Äpfeln als zweite Hauptquelle für Punkte etabliert wurde, zeigt sich an den \textit{aTomApfelmeisen}, dass eine reine Ausrichtung auf die Äpfel ebenfalls nicht zielführend ist, die \textit{aTomZuckermeisen} sind dagegen ertragreicher. Allerdings können die \textit{LipkeAnts} aufgrund der Kampf-Spezialisierung nicht effizient Zucker sammeln. Die \textit{aTomGruppenmeisen} sind ähnlich wie die \textit{LipkeAnts} eine Kombination aus Apfel, Zucker und Kampfameisen, dies zeigt sich auch in den Punkten. Jedoch haben diese einen stärkeren Fokus auf der Nahrung. Im Vergleich ist also bei der Kombination der verschiedenen Ansätze der Fokus auf dem Kampf gegen Wanzen die beste Wahl.

242

243 Die Höchstpunktzahl, die die \textit{LipkeAnts} in 100 Durchläufen erreichen konnten waren 11882 und die niedrigste Punktzahl 7124. Die Unterschiede lassen sich auf einen unterschiedlichen Aufbau des Spielfeldes zurückführen. Ist der Ameisenbau sehr weit am Rand und die Äpfel, der Zucker und die Wanzen weiter weg ist es schwieriger für die Ameisen Punkte zu erzielen. Da die durchschnittliche Punktzahl aber mit knapp 10000 Punkten näher an der Höchstpunktzahl liegt, sind solche Ausreißer seltener als Punktzahlen mit zehn bis elftausend Zahlen. Dies ist besonders bei den \textit{aTomZuckermeisen} zu erkennen, wo in einigen wenigen Durchläufen null Punkte erzielt wurden, da der Zucker zu weit weg war.

244

```
245 \begin{table}[hbt]
246 \centering
247 \begin{minipage}[t]{.65\textwidth} % Breite der Tabelle
248 \caption{Messreihe Einzelspieler, 100 Durchläufe je 5000 Runden} % Überschrift
249 \begin{tabularx}{\columnwidth}{llrr}
250 \toprule
251 Ameisenvolk & Punkte & Höchste & Niedrigste \\
252 \midrule
253 \textit{LipkeAnts} & 9957,40 & 11882 & 7124 \\
254 aTomGruppenmeisen & 7765,80 & 9654 & 5496 \\
255 aTomZuckermeisen & 6752,83 & 10740 & 0 \\
256 aTomApfelmeisen & 5207,07 & 5500 & 4250 \\
257 aTomKampfmeisen & 5084,85 & 6900 & 2700 \\
258 \bottomrule
259 \end{tabularx}
260 \source{Eigene Darstellung (siehe Anhang \ref{appendix:testSingle})} % Quelle
261 \label{tab:singlePlayer}
262 \end{minipage}
263 \end{table}
```

264

265 \subsubsection{Mehrspieler}

266 Die Stärke der \textit{LipkeAnts} zeigt sich besonders im Mehrspieler. Ameisenvölker, die nicht auf den Kampf spezialisiert sind, werden in keinem Fall gegen Kampf-Völker gewinnen können, zumindest ist dies äußerst unwahrscheinlich. Alle Demo-Völker, bis auf die \textit{aTomKampfmeisen}, kämpfen nicht gegen andere Ameisen, die \textit{aTomGruppenmeisen} nur gegen Wanzen. Am extremsten ist das bei den \textit{aTomZuckermeisen} zu beobachten, weil diese durch die

```

267 Zuckerstraßen fast alle auf einem Punkt aufhalten und somit den \
268 textit{LipkeAnts} hilflos ausgeliefert sind. Die \textit{
269 aTomApfelmeisen} sind zwar nicht so stark auf einem Fleck, aber da die
270 \textit{LipkeAnts} ebenfalls Äpfel sammeln, haben die \textit{
271 aTomApfelmeisen} Schwierigkeiten damit, den Apfel zum eignen
272 Ameisenbau zu bringen, bevor sie von den \textit{LipkeAnts} getötet
273 werden. Einzig die \textit{aTomKampfmeisen} können bessere Punktzahlen
274 erreichen, da sie ebenfalls kämpfen. Da diese aber nur kämpfen und
275 nicht auch noch sammeln, ist auch dieses Volk unterlegen.
267 \begin{table}[hbt]
268 \centering
269 \begin{minipage}[t]{.7\textwidth} % Breite der Tabelle
270 \caption{Messreihe Mehrspieler, 100 Durchläufe je 5000 Runden} % Ü
271 berschrift
272 \begin{tabularx}{\columnwidth}{\lrcrll}
273 \toprule
274 & Volk \#1 & : & Volk \#2 & \\
275 \midrule
276 LipkeAnts & 9519,63 & & 107,83 & aTomZuckermeisen\\
277 LipkeAnts & 8011,21 & & 1751,05 & aTomApfelmeisen\\
278 LipkeAnts & 8524,32 & & 2305,04 & aTomGruppenmeisen\\
279 LipkeAnts & 7051,66 & & 3249,39 & aTomKampfmeisen\\
280 \bottomrule
281 \end{tabularx}
282 \source{Eigene Darstellung (siehe Anhang \ref{appendix:testMulti})} %
283 Quelle
284 \label{tab:multiPlayer}
285 \end{minipage}
286 \end{table}

```

Anhang 4.7 Kapitel 4 - Zusammenfassung

```

1 %!TEX root = ../agi_mfws414ali.tex
2 \section{Zusammenfassung}
3
4 Das zu Anfang gesetzte Ziel, eine Strategie zu entwerfen und umzusetzen,
5 mit der hohe Punktzahlen sowohl im Einzelspieler als auch im
6 Mehrspieler-Modus möglich sind, konnte erfolgreich erreicht werden. Für
7 die Überprüfung, dass dieses Ziel erreicht werden konnte, wurden
8 diverse repräsentative Messreihen durchgeführt.
9
10 Bei der Entwicklung einer zielführenden Strategie sind verschiedene
11 Probleme aufgetreten, die gelöst werden mussten, wie bspw. die
12 Notwendigkeit, dass über eine Duftmarke verschiedenartige
13 Informationen geteilt werden können. Zudem sind weitere Probleme
14 aufgetreten, die einem nicht-statischen Ameisenvolk vorbehalten sind,
15 da diese im Vergleich zu statischen relativ stark eingeschränkt sind
16 in den Umsetzungsmöglichkeiten. Das geht so weit, dass bspw. das
17 Ereignis, dass eine Ameise gestorben ist, nur von statischen Ameisen
18 zu gebrauchen ist. Die größten Probleme traten bei dem effektiven
19 Einsatz von verschiedenen Kästen auf. Im Rahmen dieses Projektes hat
20 sich herausgestellt, dass statische Ameisenvölker das größte Potenzial
21 hinsichtlich der Kasten-Mechanik aufweisen. Statische Völker sind
22 dazu fähig die eingesetzten Kästen der vorliegenden Spielsituation

```

anzupassen. Weiterhin erwies sich die Realisierung von Gruppierungen als problematisch. Mit Gruppierungen sind nicht die unterschiedlichen Kasten gemeint (siehe aTomGruppenAmeisen), sondern das Gruppieren von Ameisen zu Fünfer- oder Zehner-Gruppen. Diese Strategie wurde zeitweise verfolgt, jedoch wieder verworfen, da sich dies nicht erfolgreich nutzen lies. Auch hier stößt ein nicht-statisches Ameisenvolk an die Grenzen. Als Fazit zu einem nicht-statischen Ameisenvolk lässt sich sagen, dass die größte Schwierigkeit, die es zu meistern gilt, die Beschränkungen sind. Generell lassen sich mit statischen Ameisen komplexere Strategien einfacher umsetzen.

7
8 Des Weiteren gab es auch einige Herausforderungen bei der Programmierung eines Ameisenvolkes. Das Entwickeln einer eigenen Basis-Klasse, welche die Basis-Klasse von AntMe! erweitert ist nicht möglich. Auch die Verwendung von Konstanten und Enumerationen war nicht möglich, da das Ameisenvolk sonst als statisch gekennzeichnet wurde. Allerdings spiegelt dies auch den Praxisalltag wieder, wo man des Öfteren ebenfalls mit Einschränkungen umgehen muss, wenn man bspw. Frameworks oder API's einsetzt.

Anhang 4.8 Quellenverzeichnis

```

1 %!TEX root = ../agi_mfws414ali.tex
2 \section*{Quellenverzeichnis}
3 \addcontentsline{toc}{section}{Quellenverzeichnis}
4 \fancyhead[R]{Quellenverzeichnis}
5
6 \defbibheading{mono}{\subsection*{Monographien}}
7 \defbibheading{mag}{\subsection*{Aufsätze in Sammelbänden und
8   Zeitschriften}}
9 \defbibheading{art}{\subsection*{Zeitungsartikel}}
10 \defbibheading{web}{\subsection*{Internetquellen}}
11 \defbibheading{leg}{\subsection*{Rechtsprechung}}
12 \defbibheading{comp}{\subsection*{Unternehmensunterlagen/Gesprächsnotizen
13   }}
14
15 \setlength{\bibitemsep}{1.5\itemsep}
16 \setlength{\bibhang}{2em}
17
18 \renewcommand{\baselinestretch}{1.50}\normalsize
19
20 \begingroup
21 \sloppy
22
23 \printbibliography[heading=mono,keyword=mono]
24 \printbibliography[heading=mag,keyword=mag]
25 \printbibliography[heading=web,keyword=web]
26
27 % Bei Bedarf einkommentieren: (erzeugt sonst Warnungen)
28 % \printbibliography[heading=art,keyword=art]
29 % \printbibliography[heading=leg,keyword=leg]
30 % \printbibliography[heading=comp,keyword=comp]
31 \endgroup

```

Anhang 4.9 Ehrenwörtliche Erklärung

```
1 %!TEX root = ../agi_mfws414ali.tex
2
3 \section*{Ehrenwörtliche Erklärung}
4 \addcontentsline{toc}{section}{Ehrenwörtliche Erklärung}
5 \fancyhead[R]{Ehrenwörtliche Erklärung}
6
7 Hiermit erkläre ich, dass ich die vorliegende \dokumententyp{} selbständig
   angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich
   benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß ü
   bernommenes Gedankengut habe ich als solches kenntlich gemacht. Diese
   Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde
   vorgelegen.
8 \vspace{20mm}
9
10 \ort, \abgabedatum
11
12 \includegraphics[scale=1]{img/Unterschrift.jpg}
13
14 \vspace{-10mm}
15
16 \underline{\hspace{8cm}}\\\dokumentenautor
```

Quellenverzeichnis

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Studienarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mettmann, 13.4.2018



Sven Kuczera