# API1:Befehlsliste

Aus AntMe! Wiki

Ameisen verstehen leider nicht jedes Wort, nur eine kleine Liste von Möglichkeiten.

## Inhaltsverzeichnis



## Befehle

### Gehen

Die Einheit von Längenangaben bei den folgenden Befehlen ist Schritte. Zum Vergleich: eine Ameise ist vier Schritte lang, das Spielfeld misst bei einem Spiel mit einem Spieler 1200 x 900 Schritte.

| Befehl | Kurzbeschreibung |
|---|---|
| GeheGeradeaus | Die Ameise geht geradeaus. Das Ziel der Ameise bleibt unangetastet. Wenn ein Wert angegeben wird, wird die Ameise wieder ihr Ziel anvisieren, nachdem sie die angegebene Entfernung zurückgelegt hat. |
| GeheZuZiel | Die Ameise speichert das angegebene Ziel und geht dort hin. |
| GeheWegVon | Die Ameise dreht sich in die Richtung die vom angegebenen Ziel weg zeigt und geht dann geradeaus. Das Ziel der Ameise bleibt unangetastet und es kann eine Entfernung angegeben werden. |

| | |
|---|---|
| GeheZuBau | Die Ameise speichert den nächstgelegenen Bau als Ziel und geht dort hin. |
| BleibStehen | Die Ameise bleibt stehen und vergisst ihr aktuelles Ziel. In der nächsten Runde wird das Ereignis Wartet() aufgerufen werden. |

## Drehen

Alle Angaben werden als ganze Winkel im Uhrzeigersinn in der Einheit Grad gemacht. 0 entspricht also rechts, 90 entspricht unten, 180 entspricht links, 270 entspricht oben und 360 entspricht wieder rechts. Die übergebenen Werte müssen sich nicht zwischen 0 und 359 befinden. Größere oder kleinere Werte werden automatisch umgerechnet.

Das Drehen einer Ameise hat immer Vorrang vor dem Gehen. D.h. wenn eine Ameise einen Dreh-Befehl bekommt, während sie einen Geh-Befehl noch nicht vollständig abgearbeitet hat, wird der Geh-Befehl unterbrochen und nach der Drehung fortgesetzt. Sich gleichzeitig drehen und gehen kann die Ameise nicht.

| Befehl | Kurzbeschreibung |
|---|---|
| DreheInRichtung | Die Ameise dreht sich in die angegebene Richtung. Die Drehrichtung wird dabei automatisch bestimmt. |
| DreheUmWinkel | Die Ameise dreht sich um den angegebenen Winkel. Positive Werte drehen die Ameise nach rechts, negative nach links. |
| DreheUm | Die Ameise dreht sich um 180 Grad in die entgegengesetzte Richtung. Hat die selbe Wirkung wie DreheUmWinkel(180). |
| DreheZuZiel | Die Ameise dreht sich in die Richtung des angegebenen Ziels. |

## Nahrung

AntMe! verfügt über zwei unterschiedliche Nahrungsmittel: Zucker und Apfel (bzw. Obst). Beide lassen sich mit dem Befehlen Nimm aufnehmen, haben aber unterschiedliche Auswirkungen auf die Eigenschaften der Ameise. Zucker wird in Würfelchen aufgeteilt und sorgt für eine Erhöhung von AktuelleLast. Äpfel hingegen müssen von mehreren Ameisen getragen werden und lassen sich anhand der Eigenschaft AktuelleLast erkennen.

| Befehl | Kurzbeschreibung |
|---|---|
| Nimm | Die Ameise nimmt die angegebene Nahrung auf. Bei einem Zuckerhaufen nimmt sie so viel wie möglich weg, bis sie ihre maximale Last erreicht hat (siehe AktuelleLast und MaximaleLast). Im Falle eines Obststückes beginnt die Ameise das Obst zu tragen (siehe GetragenesObst). |
| LasseNahrungFallen | Die Ameise lässt die gerade getragene Nahrung fallen. Zucker geht dabei verloren, Äpfel bleiben liegen und können wieder aufgenommen werden. Der Befehl muss nicht ausgeführt werden um Nahrung im Bau abzuliefern. Das passiert dort automatisch. |

## Markierungen

Markierungen sind für Ameisen die primäre Kommunikationsmethode. Ameisen, die anderen Ameisen etwas mitteilen wollen, nutzen Markierungen um Informationen weiterzugeben.

| Befehl | Kurzbeschreibung |
|---|---|
| SprüheMarkierung | Die Ameise sprüht an der aktuellen Stelle eine Duftmarkierung. Mögliche Parameter sind eine in der Markierung hinterlegte Information (diese kann im |

| | |
|---|---|
| | Ereignis Sieht(Markierung) über markierung.Information ausgelesen werden) und die Ausbreitung der Markierung. Je größer die Ausbreitung, desto schneller verschwindet die Markierung wieder. |

### Kampf

Zwar verfolgt AntMe! einen eher friedlichen Absatz, dennoch haben Ameisen auch in diesem Spiel natürliche Feinde - die Wanzen. Wanzen und gegnerische Ameisen können aktiv angegriffen werden. Mehr Infos zum Ablauf eines Kampfes gibt es im Kapitel Ameisenentwicklung.

| Befehl | Kurzbeschreibung |
|---|---|
| GreifeAn | Die Ameise speichert die angegebene Wanze oder die angegebene feindliche Ameise als Ziel und geht dort hin. Wenn die Ameise angekommen ist, beginnt der Kampf. |

### Fehlersuche

Wenn Ameisen mal nicht das tun, was sie sollen, braucht es ein paar Hilfsmittel zur Fehlersuche. Neben den Werkzeugen aus Visual Studio gibt es auch von AntMe! ein bisschen Hilfe.

| Befehl | Kurzbeschreibung |
|---|---|
| Denke | Mit Hilfe dieses Befehls gibt die Ameise Denkblasen aus, die zur Fehlersuche eingesetzt werden können. |

## Eigenschaften

Eigenschaften erlauben das Abfragen von Zuständen und Werten aus Ameisen und anderen Spielelementen.

### Kastenabhängig

Die folgenden Werte einer Ameise werden von ihrer Kaste bestimmt.

| Eigenschaft | Kurzbeschreibung |
|---|---|
| MaximaleEnergie | Gibt die maximale Energie der Ameise an. Die Einheit ist Lebenspunkte. |
| MaximaleGeschwindigkeit | Gibt die maximale Geschwindigkeit der Ameise an. Die Einheit ist Schritte pro Runde. |
| MaximaleLast | Gibt die maximal tragbare Last der Ameise an. Die Einheit ist Nahrungspunkte. Dieser Wert bestimmt, wie viel Zucker die Ameise auf einmal tragen kann und wie schnell sie ohne die Hilfe anderer Ameisen einen Apfel tragen kann. |
| Reichweite | Gibt die Reichweite in Schritten an die die Ameise zurücklegen kann, bevor sie vor Hunger stirbt. Nachdem die Ameise ein Drittel dieser Strecke zurückgelegt hat, wird das Ereignis WirdMüde() aufgerufen und der Wert von IstMüde auf wahr gesetzt. Siehe ZurückgelegteStrecke. |
| Angriff | Gibt den Angriffswert der Ameise an. Der Angriffswert bestimmt wie viele Lebenspunkte die Ameise einem Gegner in jeder Runde abzieht. Die Einheit ist Lebenspunkte. |
| Sichtweite | Gibt den Wahrnehmungsradius der Ameise in Schritten an. Dieser Radius bestimmt wie weit die Ameise von Spielelementen wie z.B. Zucker entfernt sein muss damit die Ameise sie sieht. Die |

| | |
|---|---|
| | Blickrichtung der Ameise spielt dabei keine Rolle. |
| Drehgeschwindigkeit | Gibt die Geschwindigkeit an mit der sich eine Ameise drehen kann. Die Einheit ist Grad pro Runde. |

## Sonstiges

Die folgenden Werte repräsentieren den aktuellen Zustand einer Ameise.

| Eigenschaft | Kurzbeschreibung |
|---|---|
| AktuelleEnergie | Gibt die aktuelle Energie der Ameise an. Die Einheit ist Lebenspunkte. Hat die Ameise 0 Lebenspunkte oder weniger, dann stirbt sie. Dieser Wert ist immer kleiner oder gleich MaximaleEnergie. |
| AktuelleGeschwindigkeit | Gibt die aktuell mögliche Geschwindigkeit der Ameise an. Die Einheit ist Schritte pro Runde. Der Wert wird von der aktuellen Last der Ameise beeinflusst. Ameisen die unter voller Last bewegt werden, können nur die Hälfte ihrer Maximalgeschwindigkeit erreichen. Diese Eigenschaft liefert immer einen Wert größer 0 zurück, auch wenn die Ameise still steht. Dieser Wert ist immer kleiner oder gleich MaximaleGeschwindigkeit. |
| AktuelleLast | Gibt die aktuelle Last an, die die Ameise gerade trägt. Die Einheit ist Nahrungspunkte. Dieser Wert ist immer kleiner oder gleich MaximaleLast. |
| WanzenInSichtweite | Gibt die Anzahl der Wanzen im Wahrnehmungsradius der Ameise zurück. Das Ergebnis dieser Berechnung ist von der Sichtweite der Ameise abhängig. |
| AnzahlFremderAmeisenInSichtweite | Gibt die Anzahl der feindlichen Ameisen im Wahrnehmungsradius der Ameise zurück. Das Ergebnis dieser Berechnung ist von der Sichtweite der Ameise abhängig. |
| AnzahlAmeisenInSichtweite | Gibt die Anzahl der Ameisen desselben Volkes des im Wahrnehmungsradius der Ameise zurück. Das Ergebnis dieser Berechnung ist von der Sichtweite der Ameise abhängig. |
| AnzahlAmeisenDerSelbenKasteInSichtweite | Gibt die Anzahl der befreundeten Ameisen desselben Volkes und derselben Kaste im Wahrnehmungsradius der Ameise zurück. Das Ergebnis dieser Berechnung ist von der Sichtweite der Ameise abhängig. |
| AnzahlAmeisenDesTeamsInSichtweite | Gibt die Anzahl der befreundeten Ameisen desselben Teams im Wahrnehmungsradius der Ameise zurück. Das Ergebnis dieser Berechnung ist von der Sichtweite der Ameise abhängig. |
| EntfernungZuBau | Gibt die Entfernung in Schritten zum nächstgelegenen befreundeten Ameisenbau an. |
| GetragenesObst | Gibt das aktuell getragene Obststück zurück. Wenn die Ameise gerade kein Obst trägt, zeigt dieser Verweis ins Leere. |

| | |
|---|---|
| Kaste | Gibt des Namen der Kaste der Ameise zurück. |
| Ziel | Gibt das aktuelle Ziel der Ameise zurück. Wenn die Ameise gerade kein Ziel hat, zeigt dieser Verweis ins Leere. |
| IstMüde | Gibt an ob die Ameise müde ist. Die Ameise wird müde, sobald sie ein Drittel ihrer maximalen Reichweite zurückgelegt hat. Nach dem Übergang des Wertes dieser Eigenschaft von falsch auf wahr wird das Ereignis WirdMüde() aufgerufen. |
| RestStrecke | Gibt an wie viele Schritte die Ameise noch geradeaus gehen wird, bevor sie wieder ihr Ziel anvisiert. Dieser Wert wird in jeder Runde um AktuelleGeschwindigkeit verringert. |
| RestWinkel | Gibt an wie viele Grad die Ameise sich noch drehen wird, bevor sie wieder geradeaus gehen wird. Dieser Wert wird in jeder Runde um DrehGeschwindigkeit verringert. |
| Richtung | Gibt die Blickrichtung der Ameise auf dem Spielfeld an. |
| Angekommen | Gibt an ob die Ameise an ihrem Ziel angekommen ist. |
| ZurückgelegteStrecke | Diese Eigenschaft gibt die Gesamtanzahl an Schritten zurück die die Ameise seit ihrem letzten Besuch in einem Ameisenbau zurückgelegt hat. Siehe Reichweite |

# Hilfsmethoden

Dieser Bereich behandelt Methoden, die neben den Befehlen der Ameise weitere Hilfe anbieten.

## Nahrung

Die folgenden Methoden erlauben die Untersuchung von Nahrungsmitteln.

| Ereignis | Kurzbeschreibung |
|---|---|
| BrauchtNochTräger | Ermittelt ob das angegebene Obst noch mehr Ameisen zum Tragen benötigt. |

## Navigation

Um in den folgenden Befehlen Bezug auf die aktuelle Ameise zu nehmen, wird in C# das Schlüsselwort this verwendet, in Visual Basic das Schlüsselwort Me.

| Ereignis | Kurzbeschreibung |
|---|---|
| Koordinate.BestimmeEntfernung | Bestimmt die Entfernung in Schritten zwischen zwei angegebenen Spielelementen. |
| Koordinate.BestimmeRichtung | Bestimmt vom ersten angegebenen Spielelement aus die Blickrichtung zum zweiten angegebenen Spielelement. |

### Zufallszahlen

| Ereignis | Kurzbeschreibung |
|---|---|
| Zufall.Zahl | Erzeugt eine zufällige Zahl zwischen den angegebenen Werten. Wenn nur ein Parameter angegeben wird, wird eine Zahl zwischen 0 und dem angegebenen Wert - 1 bestimmt, wenn zwei Parameter angegeben werden, wird eine Zahl zwischen dem unteren Wert und dem oberen Wert - 1 bestimmt. |





# API1:Ereignisse

Aus AntMe! Wiki

Ereignisse stellen verschiedene Auslöser im Leben einer einzelnen Ameise dar. Sie erlauben es Code zu hinterlegen, mit dem auf diese Ereignisse reagiert werden kann. Ein Ereignis könnte zum Beispiel "Sieht Zucker" sein. Darauf könnte die Ameise dann mit dem Befehl "Gehe zu Zucker" reagieren.

## Inhaltsverzeichnis



## Fortbewegung

Um die Ameise grundsätzlich in Bewegung zu setzen, braucht es Ereignisse, die ohne Zusammenhang zur Umgebung auftreten.

| Ereignis | Kurzbeschreibung |
|---|---|
| Wartet | Wenn die Ameise keinerlei Aufträge hat, wartet sie auf neue Aufgaben. Um dir das mitzuteilen, wird diese Methode hier aufgerufen. |
| WirdMüde | Erreicht eine Ameise ein drittel ihrer Laufreichweite, wird diese Methode aufgerufen. |
| IstGestorben | Wenn eine Ameise stirbt, wird diese Methode aufgerufen. Man erfährt dadurch, wie die Ameise gestorben ist. Die Ameise kann zu diesem Zeitpunkt aber keinerlei Aktion mehr ausführen. |
| Tick | Diese Methode wird in jeder Simulationsrunde aufgerufen - ungeachtet von |

zusätzlichen Bedingungen. Dies eignet sich für Aktionen, die unter Bedingungen ausgeführt werden sollen, die von den anderen Methoden nicht behandelt werden.

# Nahrung

Ereignisse, die im Zusammenhang mit Nahrungsmitteln eintreffen. Einerseits beim Erspähen von Dingen, andererseits beim Erreichen dieser.

| Ereignis | Kurzbeschreibung |
|---|---|
| Sieht(Obst) | Sobald eine Ameise innerhalb ihres Sichtradius einen Apfel erspäht wird diese Methode aufgerufen. Als Parameter kommt das betroffene Stück Obst. |
| Sieht(Zucker) | Sobald eine Ameise innerhalb ihres Sichtradius einen Zuckerhügel erspäht wird diese Methode aufgerufen. Als Parameter kommt der betroffene Zuckerghügel. |
| ZielErreicht(Obst) | Hat die Ameise ein Stück Obst als Ziel festgelegt, wird diese Methode aufgerufen, sobald die Ameise ihr Ziel erreicht hat. Ab jetzt ist die Ameise nahe genug um mit dem Ziel zu interagieren. |
| ZielErreicht(Zucker) | Hat die Ameise eine Zuckerhügel als Ziel festgelegt, wird diese Methode aufgerufen, sobald die Ameise ihr Ziel erreicht hat. Ab jetzt ist die Ameise nahe genug um mit dem Ziel zu interagieren. |

# Kommunikation

Diese Ereignisse behandeln Situationen, die mit der Interaktion mit anderen Ameisen zu tun haben.

| Ereignis | Kurzbeschreibung |
|---|---|
| RiechtFreund(Markierung) | Markierungen, die von anderen Ameisen platziert werden, können von befreundeten Ameisen gewittert werden. Diese Methode wird aufgerufen, wenn eine Ameise zum ersten Mal eine befreundete Markierung riecht. |
| SiehtFreund(Ameise) | So wie Ameisen unterschiedliche Nahrungsmittel erspähen können, entdecken Sie auch andere Spielelemente. Entdeckt die Ameise eine Ameise aus dem eigenen Volk, so wird diese Methode aufgerufen. |
| SiehtVerbündeten(Ameise) | So wie Ameisen unterschiedliche Nahrungsmittel erspähen können, entdecken Sie auch andere Spielelemente. Entdeckt die Ameise eine Ameise aus einem befreundeten Volk (Völker im selben Team), so wird diese Methode aufgerufen. |

# Kampf

Leider muss manchmal auch gekämpft werden. Diese Ereignisse treffen ein, wenn Feinde erspäht werden oder die Ameise in einen Kampf verwickelt wird.

| Ereignis | Kurzbeschreibung |
|---|---|
| SiehtFeind(Ameise) | So wie Ameisen unterschiedliche Nahrungsmittel erspähen können, entdecken Sie auch andere Spielelemente. Entdeckt die Ameise eine Ameise aus einem feindlichen Volk, so wird diese Methode aufgerufen. |
| SiehtFeind(Wanze) | So wie Ameisen unterschiedliche Nahrungsmittel erspähen können, entdecken Sie auch andere Spielelemente. Entdeckt die Ameise eine Wanze, so wird diese Methode aufgerufen. |
| WirdAngegriffen(Ameise) | Es kann vorkommen, dass feindliche Lebewesen eine Ameise aktiv angreifen. Sollte eine feindliche Ameise angreifen, wird diese Methode hier aufgerufen und die Ameise kann entscheiden, wie sie darauf reagieren möchte. |

| | |
|---|---|
| WirdAngegriffen(Wanze) | Es kann vorkommen, dass feindliche Lebewesen eine Ameise aktiv angreifen. Sollte eine Wanze angreifen, wird diese Methode hier aufgerufen und die Ameise kann entscheiden, wie sie darauf reagieren möchte. |

## Kasten

Der Bereich der Kasten betrifft Methoden, die zur Auswahl und Steuerung der Kasten dienen.

| Ereignis | Kurzbeschreibung |
|---|---|
| BestimmeKaste | Jedes mal, wenn eine neue Ameise geboren wird, muss ihre Berufsgruppe bestimmt werden. Das kannst du mit Hilfe dieses Rückgabewertes dieser Methode steuern. |



# Ameisenentwicklung

Aus AntMe! Wiki

Das Kernkonzept von AntMe! ist die Entwicklung von Ameisenvölkern. Das geschieht mit Hilfe von C# und den üblichen Standard-Komponenten von Microsofts .NET Framework. Dieser Artikel soll dir etwas genauer beschreiben was im Hintergrund passiert, worauf es ankommt und wie man damit umgehen kann.

## Inhaltsverzeichnis

# Entwicklung

Neue Ameisen sind Klassen, die bestimmte Bedingungen erfüllen. Üblicherweise werden die Klassen in einer eigenen Klassenbibliothek einer beliebigen .NET Sprache erstellt und als kompilierte Datei im Simulator geladen.

## Projekt Vorbereitungen

Das Projekt aus dem die Ameisen-Bibliothek erstellt wird muss ein paar Grundbedingungen erfüllen. Je nach Entwicklungsumgebung werden folgende Punkte unterschiedlich gehandhabt, weshalb hier nur allgemeine Punkte aufgelistet sind.

- Projekttyp Klassenbibliothek (Class Library) mit dem Format .dll als Ausgabeformat
- Das Ziel-Framework muss .NET Framework 4.0 (oder älter) sein.
- Ziel-Architektur muss 32 Bit sein. (Dank einer Abhängigkeit zur 32 Bit basierten XNA Bibliothek)
- Referenzen auf folgende Assemblies (zu finden im Installationsverzeichnis von AntMe!)
    - AntMe.SharedComponents.dll
    - AntMe.Simulation.dll

Für eine komfortable Nutzung des Projektes und der Möglichkeit die Ameise richtig zu debuggen gibt es weitere Dinge zu erledigen.

- Der Standard-Namespace für Ameisen sollte AntMe.Player.[Username] sein.
- Passend dazu sollte der Name der ausgegebenen Datei AntMe.Player.[Username].dll sein.
- Build-Parameter für eine bequemeres Debugging
    - Start-Applikation auf die AntMe-Anwendung legen (AntMe.exe, zu finden im Installationsverzeichnis)
    - Start-Parameter anpassen. Der Aufruf muss den Parameter "/file=[Dateinamen]" übergeben

Im Normalfall lässt sich ein passendes Projekt mit Hilfe des Code Generators (Beschrieben in Lektion 2: Die erste Ameise) automatisch generieren. Die manuelle Erstellung (am Beispiel Visual Studio) wird in Projekt erstellen detailliert beschrieben.

## Die Ameisen-Klasse

Das erstellte Projekt kann beliebige Dateien enthalten. Es ist aber wichtig, dass es für jedes enthaltene Ameisenvolk auch eine Klasse gibt, die die folgenden Bedingungen erfüllt:

- Die Klasse muss public sein.
- Sie muss von einer der lokalisierten Basisklassen erben
    - AntMe.Deutsch.Basisameise (für eine deutsche Ameise)
    - AntMe.English.BaseAnt (für eine englische Ameise)
- Es muss ein passendes Spieler-Attribut auf der Klasse sein
    - AntMe.Deutsch.Spieler (für eine deutsche Ameise)
    - AntMe.English.Player (für eine englische Ameise)

- Mindestens ein Kasten-Attribut das die Ameisen Kasten beschreibt
    - AntMe.Deutsch.Kaste (für deutsche Ameisen)
    - AntMe.English.Caste (für englische Ameisen)

Nach der Implementierung der Klassenstruktur lässt sich die Funktionalität mit Hilfe überschriebener Methoden hinzufügen. Hier eine vollständige Liste der überschreibbaren Methoden und verfügaren Methoden und Eigenschaften.

Im Normalfall lässt sich ein passendes Projekt mit Hilfe des Code Generators (Beschrieben in Lektion 2: Die erste Ameise) automatisch generieren. Die manuelle Erstellung (am Beispiel Visual Studio) wird in Projekt erstellen detailliert beschrieben.

## Laufzeit

Ameisen werden beim Start (oder zum Zeitpunkt des Ladens) vom Spiel in einen gesicherten Bereich (AppDomain) geladen und untersucht. Das Spiel speichert dann verschiedene Meta-Informationen (Hauptsächlich die Daten aus den Spieler- und Kasten-Attributen) die im Spiel zur KI-Auswahl angezeigt werden. Dazu werden die KI-Dateien auf öffentliche Klassen untersucht, die einerseits von einer der lokalisierten Basis-Klassen erben und außerdem über ein Spieler-Attribut verfügt.

Wird eine KI dann tatsächlich in eine Simulation geschickt, wird wieder eine gesicherte AppDomain als Raum für die Simulation erstellt und die Ameisen-Klasse in vollem Umfang geladen. Danach beginnt die Simulation nach folgenden Regeln.

**Phase 1 (Initialisierung)**

- Rahmenparameter für die Simulation wird ermittelt (Siehe Einstellungen)
- Spielfeldgröße wird ermittelt (Abhängig von Spieleranzahl)
- Für jeden Spieler wird die Position des Ameisenhügels per Zufall bestimmt.

**Phase 2 (Rundenberechnung)**

Die AntMe! Simulation ist rundenbasiert. Das bedeutet, das die Bewegungen und Interaktionen der Elemente einmal pro Runde berechnet werden. Eine normale Spielrunde dauert 5000 Runden, wobei eine Simulationsrunde bei normaler Geschwidigkeit 15 mal pro Sekunde durchläuft und eine Simulation somit etwa 4,5 Minuten dauert.

Für jede Simulationsrunde gibt es eine feste Reihenfolge wie Dinge berechnet werden.

- Verarbeitung der Nahrungsmittel
    - Leere Zuckerhügel wegräumen: Bedingung ist hier, dass die Menge des Zuckers 0 ist.
    - Neue Zuckerhügel erzeugen: Es wird geprüft ob die Gesamtanzahl der Zuckerberge groß genug ist und die Respawn-Wartezeit abgewartet wurde. Wird ein neuer Berg erstellt, wird die Position per Zufall bestimmt und der und seine Menge durch den Standardwert in den Einstellungen bestimmt.

        - Neue Äpfel erzeugen: Ähnlich wie beim Zucker muss geprüft werden, ob zu wenig Äpfel auf dem Spielfeld sind und ob die Respawn-Zeit vorbei ist. Dann wird ein Apfel zufällig auf dem Spielfeld platziert.
- Wanzenbewegung
    - Falls Wanze im Kampf verwickelt: Angriffspunkte auf die beteiligten Gegner verteilen. Betroffene Ameisen erhalten einen Aufruf auf WirdAngegriffen(Wanze)
    - Kein Kampf: Wanzen bewegen sich zufällig auf dem Spielfeld
- Ameisen (Berechnungen werden Teamweise durchgeführt. Erst Team 1, Team 2,..., Team 8)
    - Ermittlung der ganzen Umgebungsinformationen (Variablen wie AnzahlAmeisenInSichtweite)
    - Ameisenbewegung (Verarbeitung der Fortbewegung wie Reststrecke und Restdrehung)
    - Prüfung der zurückgelegten Strecke
        - Ameisen verhungern, wenn die maximale Reichweite überschritten wurde. Es wird IstGestorben aufgerufen.
        - WirdMüde wird aufgerufen, wenn ein drittel der Reichweite erreicht wurde. Es wird WirdMüde aufgerufen.
    - Sichtungen (Ameisen und Wanzen)
        - Ist eine Wanze in der Nähe die nicht bereits Ziel der Ameise ist, dann wird SiehtFeind(Wanze) aufgerufen.
        - Wird eine gegnerische Ameise gesichtet die noch nicht das Ziel der Ameise ist, wird SiehtFeind(Ameise) aufgerufen.
        - Wird eine befreundete Ameise gesichtet die noch nicht das Ziel der Ameise ist, wird SiehtFreund(Ameise) aufgerufen.
        - Wird eine Ameise aus einem befreundeten Volk gesichtet, wird SiehtVerbündeten(Ameise) aufgerufen.
    - Kampfabwicklung
        - Im Falle eines Kampfes mit der Wanze werden die Angriffspunkte der Ameise auf die Wanzen-Lebenspunkte verrechnet.
        - Im Kampf mit anderen Ameisen werden die Angriffspunkte angewendet. Außerdem erhält die Ameise einen Aufruf auf WirdAngegriffen(Ameise).
    - Zielprüfung (Ameise prüft, ob sie das angestrebte Ziel erreicht hat)
        - Prüfen, ob das erreichte Ziel der Ameisenhügel ist. Dadurch werden ein paar wichtige Dinge zurück gesetzt:
            - Ziel wird zurück gesetzt.
            - Zurückgelegte Strecke wird auf 0 gesetzt.
            - Damit verbunden wird auch der IstMüde-Flag zurückgesetzt.
            - Getragene Nahrungsmittel (Obst und Zucker) werden entfernt und dem Punktestand hinzu addiert (Punktewerte gemäß Einstellungen).
            - Liste der bekannten Markierungen wird geleert.
            - Energie der Ameise wird regeneriert.

            - Prüfen, ob das erreichte Ziel Zucker ist. Dadurch wird das Ziel zurück gesetzt und ZielErreicht(Zucker) aufgerufen.
            - Prüfen, ob es sich um einen Apfel handelt. Dadurch wird das Ziel zurück gesetzt und ZielErreicht(Obst) aufgerufen.
            - Prüfen, ob es sich um einen Gegner handelt. Hier wird das Ziel explizit NICHT zurück gesetzt, damit das Ziel weiterhin angegriffen wird.
        - Sichtungen (Nahrung und Markierungen)
            - Prüfen, ob die Ameise einen Zuckerberg sehen kann. Aufruf auf Sieht(Zucker).
            - Prüfen, ob die Ameise einen Apfel sehen kann. Aufruf auf Sieht(Obst).
            - Prüfen, ob die Ameise eine neue Markierung erschnüffelt hat. Aufruf auf RiechtFreund(Markierung)
        - Untätigkeit. Sollte die Ameise weder Ziel noch Reststrecke haben, wird Wartet aufgerufen.
        - In jeder Runde wird zum Schluss bedingungslos Tick aufgerufen.
        - Tote Ameisen werden vom Spielfeld entfernt.
        - Neue Ameisen werden erstellt, sollte die Maximalanzahl noch nicht erreicht worden sein und der Respawn-Zeit vorbei ist. Es wird eine neue Instanz der Ameise erzeugt und anschließend BestimmeKaste um die Kaste für die neue Ameise zu bestimmen.
        - Markierungen werden aktualisiert (Ausdehnungsgröße berechnen und abgelaufene Markierungen entfernen).
        - Früchte vom Spielfeld entfernen die nahe genug am Ameisenhügel liegen.
    - Wanzen Lebenszyklus
        - Tote Wanzen entfernen
        - Wanzen, die nicht angegriffen werden, erholen sich nach einiger Zeit (Siehe Einstellungen).
        - Neue Wanzen erstellen, sollte die Maximalanzahl Wanzen nicht erreicht sein.
    - Obst/Ameisen-Cluster bewegen.

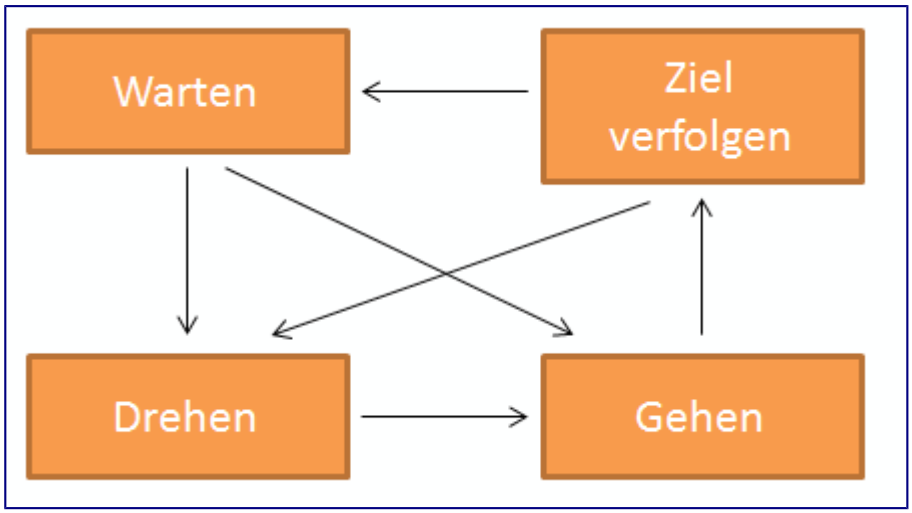
# Funktionsweise

In den folgenden Kapiteln wird die logische Funktionsweise des Spiels beschrieben. Dabei werden unterschiedliche Aspekte des Spiels beleuchtet.

## Fortbewegung

Einer der wichtigen Bereiche ist sicher die Fortbewegung und dessen Funktionsweise.

**State Maschine**




Ameisenstates

Bei Ameisen handelt es sich um klassische State Machines. Das bedeutet, dass sich die Ameise in unterschiedlichen Stati befinden kann und durch Spieler-Befehle oder äußere Ereignisse wechseln kann. Das ist eines der wichtigsten Konzepte beim Steuern der Ameise und führt oft zu Verwirrung. Zwar gibt es noch einige andere Einflüsse, aber grundsätzlich wird die Bewegung der Ameise durch 3 Eigenschaften beeinflusst:

- Das aktuelle Ziel
- RestStrecke
- RestWinkel

**Wartet**: Die Ameise beginnt mit einem leeren Ziel und 0 Reststrecke und Restwinkel und befindet sich damit im State "Wartet". Solange die Ameise sich in diesem State befindet, wird in jeder Runde das Wartet-Ereignis aufgerufen. Der Aufruf einer beliebigen Dreh-Methode sorgt für eine Veränderung des Restwinkels. Aufrufe auf Geh-Methoden verändert den Wert der Reststrecke. In diesem State wird in jeder Runde die folgende Bedingungsliste abgearbeitet und dadurch State-Wechsel verursacht:

1. Ist Restwinkel != 0? Dann wechselt die Ameise in den Drehen-State
2. Ist Reststrecke > 0? Dann wechselt die Ameise in den Gehen-State
3. Ist das Ziel != null? Dann wird ein neuer Kurs ermittelt und die ermittelten Werte in die Rest-Felder gefüttert. Dadurch wechselt die Ameise promt in einen der anderen States.

**Drehen**:

**Gehen**:

**Ziel verfolgen**:

**Apfel Cluster**

### Wahrnehmung

### Kampfsystem

Von „http://wiki.antme.net/de_root/index.php?title=Ameisenentwicklung&oldid=10“

- Diese Seite wurde zuletzt am 15. Mai 2015 um 17:26 Uhr geändert.
- Diese Seite wurde bisher 3.220 mal abgerufen.

# Einstellungen

Aus AntMe! Wiki

Die Einstellungen (Settings) des Spiels erlauben das Festlegen diverser Werte der Simulation.

## Inhaltsverzeichnis



## Spielfeld

Diese Einstellungen beziehen sich auf globale Dinge wie die Erstellung des Spielfeldes und die Standard-Radien der unterschiedlichen Spiel-Elemente. Einige der Parameter fließen in die Berechnung der Spielfeld-Fläche ein. Die Formel dafür sieht folgendermaßen aus. Auf Basis der Anzahl Spieler wird die Gesamtfläche in Ameisenquadratschritte berechnet und dann in eine Fläche mit einem Seitenverhältnis von 4:3 aufgeteilt.
Seit Version 1.7.4 wird ein neuer Verteilungsalgorithmus verwendet um Obst, Zucker und den/die Ameisenhügel fair auf dem Spielfeld zu platzieren.

```
float area = (1 + (playerCount * PlayGroundSizePlayerMultiplier)) *
PlayGroundBaseSize;
int playgroundWidth = (int)Math.Round(Math.Sqrt(area * 4f/3f));
int playgroundHeight = (int)Math.Round(Math.Sqrt(area * 3f/4f));
```

| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| PlayGroundBaseSize | 550000 | Basis-Flächenmaß (Maßeinheit Ameisenquadratschritte) für die Berechnung der Grundfläche. |
| PlayGroundSizePlayerMultiplier | 1 | Multiplikator, mit dem die Anzahl Spieler multipliziert wird, um die Spielfeldfläche zu berechnen. |
| AntHillRadius | 32 | Gibt den Radius von Ameisenhügeln an. |
| BattleRange | 5 | Gibt den Angriffsradius von allen Spielelementen an. Dieser Radius wird zum eigentlichen Radius des Spiel-Elementes addiert. |
| AntHillRandomDisplacement(1.7.4) | 0.5f | Prozentuale Verschiebung des Ameisenhügels vom Kreispunkt zu einem Zufälligem Punkt auf der Map (0 => Kreispunkt - 1 => Zufallspunkt). |
| SpawnCellSize(1.7.4) | 100 | Größe einer Spawnzelle in Ameisenschritten. |
| RestrictedZoneRadius(1.7.4) | 300 | Gesperrter Bereich um den Ameisenhügel für den Spawnalgorithmus. |
| FarZoneRadius(1.7.4) | 1500 | Gibt die Entfernung an, nach der die Spawnzellen gesperrt werden, um zu große Entfernungen zu verhindern. |
| DecreasValue(1.7.4) | 2f | Basiswert mit dem die umliegenden Spawnwerte vermindert werden in Abhängigkeit zu ihrer Entfernung zur Startzelle. |
| RegenerationValue(1.7.4) | 0.1f | Wert mit dem die Spawnwerte aller Zellen regeneriert werden (vor jedem neuem Nahrungsspawn). |

# Lifetime and Respawn

Dieser Bereich beschäftigt sich mit der Lebenszeit, Erstellung und gleichzeitiger Existenz von Spielelementen.

## Anzahl gleichzeitiger Einheiten

Die Anzahl gleichzeitiger Einheiten bestimmt die Anzahl von Spiel-Elementen, die zur selben Zeit auf dem Spielfeld existieren dürfen. So gibt beispielsweise der Wert "AntSimultaneousCount" an, wie viele Ameisen ein Spieler zur selben Zeit haben kann. Das entspricht auch der maximalen Anzahl Ameisen (pro Volk), die im Idealfall gleichzeitig leben können. Dieser Wert wird aber noch mit dem unten folgenden PlayerMultiplier verrechnet.

| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| AntSimultaneousCount | 100 | Maximale Anzahl an gleichzeitig existierenden Ameisen (pro Volk) |
| BugSimultaneousCount | 5 | Maximale Anzahl Wanzen |
| SugarSimultaneousCount | 1 | Maximale Anzahl Zuckerhügel |
| FruitSimultaneousCount | 2 | Maximale Anzahl Äpfel |

Die maximale Anzahl gleichzeitiger Einheiten wird noch mit dem folgenden Multiplier verrechnet, um eine Abhängigkeit zur Anzahl Spieler einzubringen. Die folgende Formel beschreibt die Vorgehensweise.

```
int limit = (int) (BugSimultaneousCount * (1 + (BugCountPlayerMultiplier * playerCount));
```

| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| BugCountPlayerMultiplier | 1 | Multiplikator für die Anzahl gleichzeitiger Wanzen |
| SugarCountPlayerMultiplier | 1 | Multiplikator für die Anzahl gleichzeitiger Zuckerhügel |
| FruitCountPlayerMultiplier | 1 | Multiplikator für die Anzahl gleichzeitiger Äpfel |
| AntCountPlayerMultiplier | 0 | Multiplikator für die Anzahl gleichzeitiger Ameisen |

### Gesamtanzahl Einheiten

Mit Hilfe dieser Werte lässt sich die Gesamtanzahl von Einheiten limitieren. Dies betrifft die Anzahl jemals existierender Einheiten.

| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| AntTotalCount | 999999 | Maximale Anzahl von Ameisen insgesamt |
| BugTotalCount | 999999 | Maximale Anzahl von Wanzen insgesamt |
| SugarTotalCount | 999999 | Maximale Anzahl von Zuckerhügeln insgesamt |
| FruitTotalCount | 999999 | Maximale Anzahl von Äpfel insgesamt |

Parallel zu den Fixwerten der letzten Tabelle spielt ein Multipier die Abhängigkeit zur aktuellen Spieleranzahl mit ins Endergebnis.

```
int totalCount = (int) (AntTotalCount * (1 + (AntTotalCountPlayerMultiplier * playerCount))
```

| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| AntTotalCountPlayerMultiplier | 0 | xxx |
| BugTotalCountPlayerMultiplier | 0 | xxx |
| SugarTotalCountPlayerMultiplier | 0 | xxx |
| FruitTotalCountPlayerMultiplier | 0 | xxx |

| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| xxx | xxx | xxx |
| xxx | xxx | xxx |
| xxx | xxx | xxx |
| xxx | xxx | xxx |

```
AntRespawnDelay = 15;
BugRespawnDelay = 75;
SugarRespawnDelay = 150;
FruitRespawnDelay = 225;
```

# Wanzen

```
BugAttack = 50;
BugRotationSpeed = 3;
```

```
BugEnergy = 1000;
BugSpeed = 3;
BugRadius = 4;
BugRegenerationValue = 1;
BugRegenerationDelay = 5;
```

# Nahrungsmittel

```
SugarAmountMinimum = 1000;
SugarAmountMaximum = 1000;
FruitAmountMinimum = 250;
FruitAmountMaximum = 250;
FruitLoadMultiplier = 5;
FruitRadiusMultiplier = 1;
```

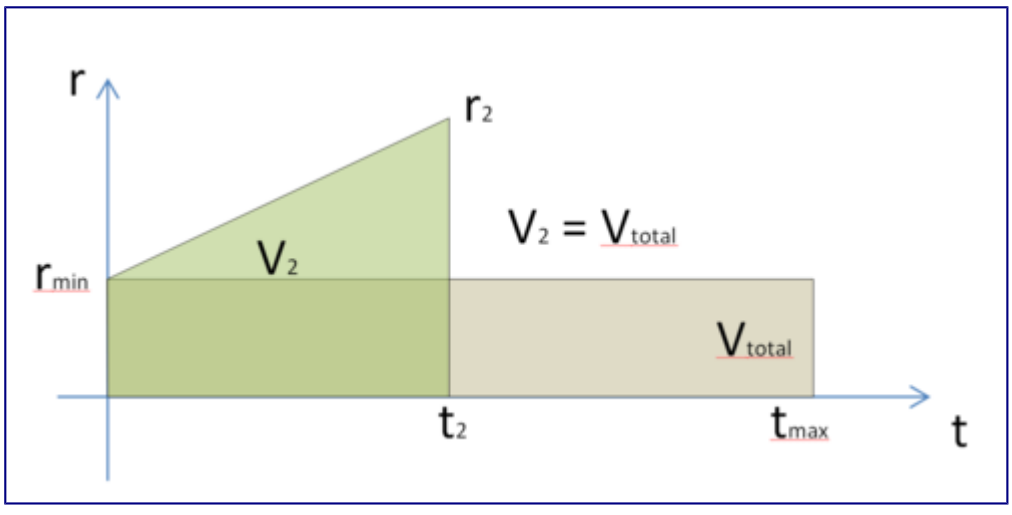
# Markierungen

Mit den Settings im Bereich der Marker lassen sich die Vorgänge für die Ausbreitung und Größe von Markierungen beeinflussen. Die maximale Lebensdauer einer Markierung ergibt sich aus dem imaginäre Sprühvolumen einer Ameise, das sich aus den Werten der Minimal-Größe und des Maximal-Alters ergibt. Die Markierung wird dabei als Halbkugel behandelt.

```
int Vmax = (r³ * PI * t) / 2;
```

Sollte die Markierung aber größer als der Minimal-Radius sein, wird berechnet wie lange eine Markierung mit dem verfügbaren Sprühvolumen und einem maximalen Radius erhalten bleibt. Dadurch ergibt sich die neue Verweildauer der Markierung. WICHTIG: Markierungen werden abgerundet (seit Version 1.7.4). Unterschreitet die Lebenszeit einer Markierung ein einzelnes Frame, wird keine Markierung mehr gesprüht.

```
int t = ((rmin³ * PI/2) + ((r-rmin)³ * PI/4)) / V;
```




| Eigenschaft | Standardwert | Beschreibung |
|---|---|---|
| MarkerDistance | 13 | Mindestabstand zwischen zwei Markierungen. Befindet sich eine spühende Markierung zu nah an anderen Markierungen, wird keine Markierung erzeugt. |
| MarkerSizeMinimum | 20 | Mindestgröße für eine neue Markierung. Dies ist auch der |

| | | |
|---|---|---|
| | | Standard-Wert bei fehlendem Größen-Parameter. |
| MarkerMaximumAge | 150 | Maximales Lebensalter einer Markierung in Runden. |

Mit den Standard-Parametern ergibt sich eine maximale Markierungsgröße von 287 Ameisenschritte bei einem einzigen Sichtbarkeits-Frame.

# Punkte

```
PointsForFoodMultiplier = 1;
PointsForFruits = 0;
PointsForBug = 150;
PointsForForeignAnt = 5;
PointsForEatenAnts = 0;
PointsForBeatenAnts = -5;
PointsForStarvedAnts = 0;
```

# Kasten

```
CasteSettings = new SimulationCasteSettings();
CasteSettings.Offset = -1;
CasteSettings.Columns = new SimulationCasteSettingsColumn[4];
```

| Eigenschaft | -1 | 0 | 1 | 2 | Beschreibung |
|---|---|---|---|---|---|
| Attack | 0 | 10 | 20 | 30 | Bla |
| Energy | 50 | 100 | 175 | 250 | Bla |
| Load | 4 | 5 | 7 | 10 | Bla |
| Range | 1800 | 2250 | 3400 | 4500 | Bla |
| RotationSpeed | 6 | 8 | 12 | 16 | Bla |
| Speed | 3 | 4 | 5 | 6 | Bla |
| ViewRange | 45 | 60 | 75 | 90 | Bla |