Berner Fachhochschule
Technik und Informatik

Projekt 2 Modul 7302

Fachbereich Technik und Infromatik
Frühlingssemester 2012

Ants - AI Challenge

Studierende: Lukas Kuster
Stefan Käser

Professoren: Dr. Jürgen Eckerle

Datum: 13. Juni 2012

Management Summary

Ants AI Challenge ist ein Programmierwettbewerb, bei welchem ein Bot programmiert wird der ein Ameisenvolk steuert. Das Ameisenvolk soll auf einer Map Futtersuchen sowie gegnerische Völker angreifen und vernichten. Dabei müssen Problem wie die Pfadsuche, das Verteilen von Aufgaben sowie das Schwarmverhalten gelöst werden. In unserer Arbeit wollten wir herausfinden was es alles braucht um einen solchen intelligenten Bot zu schreiben und gegen andere Mitspieler anzutreten. Wir konzetrierten uns auf die Aufgabenverteilung sowie die Pfadsuche. Diese Erfahrungen wollen wir für die Bachelorarbeit mitnehmen, wo wir an einer aktiven Challenge teilnehmen möchten oder uns in der für dieses Projekt verwendete Challenge vertiefen.

Datum	13. Juni 2012
Name Vorname	Lukas Kuster
Unterschrift	.....
Name Vorname	Stefan Käser
Unterschrift	.....

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

TODO

Dieses Dokument dient einerseites zur Illustration der LaTeX Vorlage anhand des Corporate Designs der Berner Fachhochschule und andererseits als Anleitung für deren Verwendung. Dabei wird vorausgesetzt, dass der Benutzer bereits Erfahrungen mit LaTeX besitzt oder gewillt ist, sich während der Benutzung in das Thema einzuarbeiten. Im Quellenverzeichnis sind einige nützliche Einträge zu diversen Büchern und Dokumenten im Internet über LaTeX zu finden.

2. Spielbeschrieb

2.1. Der Wettbewerb

Die AI Challenge[1] ist ein internationaler Wettbwerb des University of Waterloo Computer Science Club der im Zeitraum Herbst 2011 bis Januar 2012 stattgefunden hat. Das Spiel ist ein zugbasiertes Multiplayerspiel in welchem Sich Ameisenvölker gegenseitig bekämpfen. Ziel einer AI-Challenge ist es, einen möglichst Bot zu schreiben, der die gegebenen Aufgaben mit möglichst intelligent Algrithmen löst. Die zu lösenden Aufgaben der Ants AI Challenge sind die Futtersuche, das Explorieren der Karten, das Angreifen von gegnerischen Völkern und deren Ameisenhaufen sowie dem Schützen des eigenen Ameisenhaufen.

2.2. Spielregeln

Nachfolgend sind die wichtigsten Regeln, die während dem Spiel berücksichtigt, werden müssen aufgelistet.

- Pro Zug können alle Ameisen um ein Feld (vertikal oder horizontal) verschoben werden.
- Pro Zug steht insgesamt eine Rechenzeit von einer Sekunde zur Verfügung.
- Bewegt sich eine Ameise in die 4er Nachbarschaft eines Futterpixel, wird dieses "gefressenünd beim nächsten Zug entsteht beim Ameisenhügel eine neu Ameise
- Die Landkarte besteht aus passierbaren Landpixel sowie unpassierbaren Wasserstellen
- Ein Gegener wird geschlagen, wenn im Kampfradius der eigenen Ameise mehr eigene Ameise stehen als gegnerische Ameisen im Kampfradius der Ameise die angegriffen wird.
- Ein Gegner ist ausgeschieden wenn alle seine eigenen Ameisenhügel vom Gegner vernichtet wurden. Pro verlorener Hügel wir minus ein Punkt berechnet, pro zerstörter Hügel plus 2 Punkte.
- Steht nach einer definierbaren Zeit (Anzahl Züge) kein Sieger fest, werden die Punkte gezählt.

Die ausführlichen Regeln können auf der Webseite nachgelesen werden: http://aichallenge.org/specification.php

2.3. Schnittstelle

Die Spielschnittstelle ist simpel gehalten. Nach jeder Spielrunde erhält der Bot das neue Spielfeld mittels String-InputStream, die Spielzüge gibt der Bot dem Spielcontroller mittels String-OutputStream bekannt. Unser MyBot leitet vom Interface Bot[2] ab. Ein Spielzug wird im folgendem Format in den Output-Stream gelegt:

o <Zeile> <Spalte> <Richtung>

Beispiel:

```
o 4 7 W
```

Die Ameise wird von der Positon Zeile 4 und Spalte 7 nach Westen bewegt.
Der Spielcontroller ist in Python realisiert, der Bot kann aber in allen gänglichen Programmiersprachen wie Java, Python, C#, C++ etc. geschrieben werden.

[1] http://www.aichallenge.org
[2] das Interface ist im Code unter ants.bot.Bot.Java auffindbar

3. Implementation

3.1. Tasks

Die Tasks bzw. Aufgaben des Bots wurden in eigenen Klassen implementiert. Das Interface Task [1] definiert eine setup()-Methode welche den Task initiert, sowie eine perform()-Methode welche den Task ausführt. Im Program werden die Tasks nach deren Wichtigkeit ausgeführt, was auch der nachfolgenden Reihenfolge entspricht. Jeder Task kann nur auf die unbeschäftigten Ameisen zur Verfügung, d.h. jene welchen noch keine Aufgabe zugeteilt wurde.

3.1.1. MissionTasks

Dieser Task prüft alle akutellen Missionen auf deren Gültigkeit wie zum Beispiel, ob die Ameise der Mission noch am Leben ist. Falls gültig, wird der nächste Schritt der Mission ausgeführt.

3.1.2. GatherFoodTask

Für jedes Food-Tile wird in einem definierbaren Radius r die nächsten Ameisen bestimmt. Danach wird aufsteigend der Luftliniendistanz versucht mit dem Pfadsuchalgorithmus SiMPLE oder falls dieser kein Pfad gefunden hat mit A* eine passierbare Route gesucht. Falls diese existiert wird mit der Ameise und dem Food-Tile eine GatherFoodMission erstellt, welche die Ameise zum Food-Tile führt. Zu jedem Food-Tile wir immer nur eine Ameise geschickt.

3.1.3. AttackHillsTask

Sobald gegnerische Ameisenhaufen sichtbar sind, sollen diese angegriffen werden, da dies +2 Punkte gibt. Die Kriterien, dass eine Pfad zum gegnerischen Haufen gesucht wird, sind die selben wie beim GatherFoodTask, ausser dass mehrere Ameisen das Ziel angreifen können. Es wird ein AttackHillMission erstellt.

3.1.4. CombatTask

Beim Angriffstask wird berechnet ob wir in einem Kampfgebiet (viewRadius2) die Überhand, d.h mehr Ameisen platziert haben. Falls ja wird die gegnerische Ameise angegriffen.

3.1.5. DefendAreaTask

Dieser Task wäre vogesehen um eine Region wie zum Beispiel der eingene Ameisenhügel zu schützen. Dieser Task ist aber noch nicht implementiert.

3.1.6. ExploreTask

Für alle noch unbeschäftigten Ameisen wird mittels ManhattenDistance der nächste Ort gesucht, der noch nicht sichtbar, also unerforscht ist. Falls ein Pfad mittels Pfadsuchalgorithmus gefunden wird, wird eine ExplorerMission/refsec:implementation.Missionen erstellt, das heisst die Ameise wird den gefundenen Pfad in den nächsten Spielzügen ablaufen.

[1] das Interface ist im Code unter ants.tasks.Bot.Java auffindbar

3.1.7. FollowTask

Der FollowTask ist für Ameisen angedacht welche aktuell keine Aufgabe haben. Diese Ameisen sollen einfach einer beschäftigten Ameise folgen, damit diese nicht alleine unterwegs ist.

3.1.8. ClearHillTask

Dieser Task bewegt alle Ameisen, welche neu aus unserem Hügel ßchlüpfenünd noch keinen Befehl haben, davon weg. So werden nachfolgende Ameisen nicht durch diese blockiert.

3.1.9. ClusteringTask

Der ClusteringTask wird als Vorbereitung für den HPA* Algorithmus verwendet. Hier wird alle sichtbaren Kartenregionen ein Clustering vorgenommen. Das Clustering wird im Kapitel 3.3.3 im Detail beschreiben.

3.2. Missionen

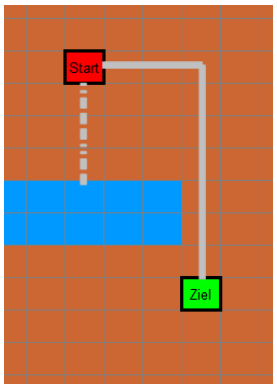
Eine Mission dauert über meherer Spielzüge. Die meisten Missionen (GatherFoodMission,ExploreMission,AttackHillMission,AttackAntM sind Pfadmissionen[2] bei welchen die Ameise einem vorgegebenen Pfad, der bereits beim Erstellen der Mission berechnet wurde, folgt. Je nach spezifischer Mission sind aber die Abbruchbedingungen anders. Zum Beispiel die GatherFoodMission ist nur solange gültig wie das Futter noch nicht von einer anderen Ameise eingesammelt wurde.

3.3. Pfadsuche

Wir haben drei mögliche Pfadalgorithmen in unserem Code eingebaut. Via Klasse PathFinder kann für die Pfadsuche der Alogrithmus ausgewählt werden.

3.3.1. Simple Algorthmus

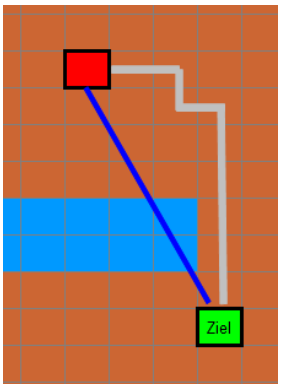
Der Simple Algorithmus versucht das Ziel zu erreichen indem er zuerst die eine, dann die andere Achse abläuft. Sobald ein Hindernis in den Weg kommt bricht der Algorithmus ab. Im folgenden Beispiel sucht der Alogrithmus den Vertikal-Horizontal Pfad. Da dieser Pfad wegen dem Wasserhindernis (blau) nicht ans Ziel führt, wird via Horizontal-Vertikal Pfad gesucht. Hier wird der Pfad gefunden. Dieser Algorithmus ist wie der Name sagt sehr einfach aufgebaut und kostet wenig Rechenzeit. Dafür kann er keinen Hindernissen ausweichen.



[2]Die abstrakte Klasse PathMission ist im Code unter ants.missions.PathMission.java auffindbar

3.3.2. A* Algorthmus

Beim A* Algorithmus wird für jeden expandierten Knoten einen heuristischen Wert f(x) für gesamte Pfadlänge berechnet. Dass heisst f(x) besteht aus einem Teil g(x) welches die effektiven Kosten vom Startknoten zum aktuellen Knoten berechnet. Der andere Teil ist ein heuristischer Wert der Pfadkosten welche bis zum Zielknoten noch anfallen werden. Dieser Wert muss die effektiven Kosten zum Ziel immer unterschätzen. Dies ist in unserem Spiel dadurch gegeben, dass sich die Ameisen nicht diagonal bewegen können, wir aber für den heuristischen Wert die Luftlinie zum Ziel nehmen. Die Pfadsuche wird immer bei dem Knoten fortgesetzt welcher die kleinsten Kosten f(x) hat.



Das Bild zeigt, dass der effektive Pfad (grau) vom expandierenden roten Knoten minimal 10 Pixel lang sein kann. Die Luftlinie (blau) als heurstischer Pfad hat aber nur die Länge 7.6 Pixel. Damit erfüllt unsere Implementation die Anforderungen des Algorithmus.

Dieser Algorithmus wird in unserem Code für eine Pfadsuche über alle Pixel (jedes Pixel ist ein Node) verwendet aber auch für die berechneten Kanten welche im HPA* verwendet werden.

3.3.3. HPA* Algorthmus

Eine Pfadsuche A* überalle Pixel ist sehr teuer, da es viel Pfade gibt, die zum Teil nur ein Pixel nebeneinander liegen. Es werden bis zum Schluss verschiedenen Pfaden nachgegangen. Abhilde zu dieser sehr feinmaschigen Pfadsuche bietet der HPA* bei welchem im sogenanten Clustering übermehrere Pixel verlaufende Kanten und Knoten berechnet werden.

3.4. JavaScript Addon für HMTL-Gameviewer

TODO

4. Rückblick

4.1. Resultate

TODO

4.2. Herausforderungen

TODO

4.3. Ziele für Bachelorarbeit

TODO

A. Beliebiger Anhang

Phasellus eget velit massa, sed faucibus nisi. Etiam tincidunt libero viverra lorem bibendum ut rutrum nisi volutpat. Donec non quam vitae lacus egestas suscipit at eu nisi. Maecenas non orci risus, at egestas tellus. Vivamus quis est pretium mauris fermentum consectetur. Cras non dolor vitae nulla molestie facilisis. Aliquam euismod nisl eget risus pretium non suscipit nulla feugiat. Nam in tortor sapien. Nam lectus nibh, laoreet eu ultrices nec, consequat nec sem. Nulla leo turpis, suscipit in vulputate a, dapibus molestie quam. Vestibulum pretium, purus sed suscipit tempus, turpis purus fermentum diam, id cursus enim mi a tortor. Proin imperdiet varius pellentesque. Nam congue, enim sit amet iaculis venenatis, dui neque ornare purus, laoreet porttitor nunc justo vel velit. Suspendisse potenti. Nulla facilisi.

B. Weiterer Anhang

B.1. Test 1

Phasellus eget velit massa, sed faucibus nisi. Etiam tincidunt libero viverra lorem bibendum ut rutrum nisi volutpat. Donec non quam vitae lacus egestas suscipit at eu nisi. Maecenas non orci risus, at egestas tellus. Vivamus quis est pretium mauris fermentum consectetur. Cras non dolor vitae nulla molestie facilisis. Aliquam euismod nisl eget risus pretium non suscipit nulla feugiat. Nam in tortor sapien.

B.1.1. Umfeld

Nam lectus nibh, laoreet eu ultrices nec, consequat nec sem. Nulla leo turpis, suscipit in vulputate a, dapibus molestie quam. Vestibulum pretium, purus sed suscipit tempus, turpis purus fermentum diam, id cursus enim mi a tortor. Proin imperdiet varius pellentesque. Nam congue, enim sit amet iaculis venenatis, dui neque ornare purus, laoreet porttitor nunc justo vel velit. Suspendisse potenti. Nulla facilisi.

Literaturverzeichnis

Stichwortverzeichnis