34. Bundeswettbewerb Informatik – Aufgabe 2

Team „ByteSector“

Lösungsidee

Als grundlegendes Konzept zur Entwicklung der Simulation wurde eine leicht abgewandelte Version des MVC-Musters verwendet. Der Unterschied besteht lediglich darin, dass die Daten des Models erst an den Controller übergeben werden und von dort aus weitergeleitet werden, statt direkt von der View abgerufen zu werden.

Die Simulation selbst wird in Züge eingeteilt, in denen alle Ameisen nacheinander agieren. Dies macht es möglich, die Simulation in zeitliche Einheiten zu unterteilen und verhindert, dass Ameisen gleichzeitig mit einer Ressource interagieren.

Der Benutzer soll die Möglichkeit haben, die Simulation manuell einen Zug weiter zu schalten, aber auch einen automatischen Ablauf starten und stoppen zu können.

Jedes Feld der 500x500-Felder großen Welt enthält eine bestimmte Menge an Futter und Duftpunkten und kann Teil des Nestes sein. Die am Anfang der Simulation als Futterquellen gewählten Felder werden mit 50 Futtereinheiten bestückt, der Rest bleibt auf null. Nimmt eine Ameise sich ein Stück Futter von einem Feld, sinkt die Anzahl an Einheiten um eine, legt eine Ameise Futter im Nest ab, wird die Futtereinheit aus der Simulation entfernt und zwecks Auswertung wird der Wert eines Zählers erhöht.

Wurde durch eine Ameise ein Duftpunkt auf ein Feld gelegt, beginnt dieser, sich mit jedem Zug abzubauen, bis er so schwach ist, dass er von den Ameisen nicht mehr wahrgenommen wird. In der Simulation wird eine exponentielle Abnahme der Stärke des Duftes verwendet. Dadurch halten oft verwendete Duftwege zwar länger, verschwinden aber auch wieder rasch, sobald keine Ameisen mehr durch ihre Verwendung zu Futter gelangen und den Duft verstärken. Der in der Abnahme verwendete Faktor ist für alle Felder gleich und wird am Anfang der Simulation aus der durch den Benutzer angegebenen Verdunstungszeit errechnet. Diese wird als die Anzahl von Zügen definiert, die ein Feld mit einem Duftpunkt von den Ameisen als duftend wahrgenommen wird.

Bei der Interpretation der Aufgabenstellung ist es zu einigen Fragen gekommen, die nicht unbedingt eindeutig geklärt werden konnten, weshalb sich nach einiger Überlegung selbst für eine Möglichkeit entschieden wurde:

- Bewegungsmöglichkeiten
  - Es ist mehrdeutig, ob "direkt angrenzende" bzw. "angrenzende" Felder diagonal erreichbare Felder einschließt, was die Anzahl an möglichen Bewegungen von vier auf acht erhöhen würde.
  - Es wird angenommen, dass nur vertikal und horizontal angrenzende Felder erreichbar sind.
  - Die Formulierung "direkt angrenzend" scheint eine Einschränkung bei der Auswahl der angrenzenden Felder zu sein. Diagonal erreichbare Felder nicht einzuschließen, scheint die einzig mögliche Einschränkung zu sein, weshalb angenommen wird, dass diese gemeint ist. Die weiter unten in der Aufgabenstellung verwendete Formulierung "angrenzende" wird als verkürzte, aber dennoch bedeutungsgleiche Umformulierung von "direkt angrenzend" interpretiert.
- Verdunstungszeit im Standard-Szenario
  - Aus der Aufgabenstellung geht hervor, dass das Programm für alternative Werte für die Anzahl an Ameisen, Futterquellen, die Position des Nestes und die Verdunstungszeit erweitert werden soll. Unklar ist, was die Verdunstungszeit im Standard-Szenario sein soll.

- Die Verdunstungszeit für das Standard-Szenario wird selbst frei gewählt.
- Da eine Verdunstungszeit festgelegt werden muss, allerdings keine vorgeschrieben wird, wird sie zwangsweise selbst gewählt.

- Ameisen pro Feld
  - Es wird nicht angesprochen, ob ein Feld durch eine Ameise blockiert werden kann.
  - Es wird angenommen, dass Ameisen sich nicht gegenseitig blockieren.
  - Das gleichzeitige Starten von 100 Ameisen in einem 2x2 Nest im Standard-Szenario würde zu einer Verletzung von Regeln führen, wenn Ameisen sich gegenseitig blockieren würden.
- Verhalten bei Duftentdeckung
  - Es geht aus der Formulierung "läuft in Richtung der stärksten Konzentration weiter weg vom Nest" nicht eindeutig hervor, ob eine Ameise sich mit höherer Priorität zur höheren Konzentration, aber dafür näher zum Nest (Fall F-A), zwingend weiter weg vom Nest (Fall F-B) bewegen soll, oder die Priorität abhängig von einem Verhältnis zwischen Entfernungs- und Konzentrationsänderung ist (Fall F-C).
  - Es wird F-B angenommen: Duftpunkte in Feldern, die sich näher am Nest befinden als die Ameise selbst, werden ignoriert. Zur Not wird weiterhin zufällig weitergelaufen.
  - F-A könnte dazu führen, dass unbeladene Ameisen beladenen zum Nest hinterherlaufen, ohne Futter aufzunehmen, da die Duftspur in Richtung Nest frischer bzw. intensiver ist. F-C würde weitere Kalkulationen mit weiteren selbst gewählten Koeffizienten erfordern und damit über die in der Aufgabenstellung beschriebenen Regeln zu weit hinausgehen. F-B kann zwar dazu führen, dass Ameisen Duftspuren ignorieren, wenn sie sich von außen, sprich weiter vom Nest entfernt, an sie annähern, allerdings ist es wahrscheinlich, dass die Ameisen in diesem Falle die Duftspur so durchlaufen, dass sie Felder mit Duftspur hinter sich bringen und damit wieder wahrnehmen. Es ist möglich, falls die Ameise lediglich Felder mit Duftpunkten sieht, die sich näher am Nest befinden, dass sie sich daraufhin dazu entscheidet, zufällig zu laufen, und letztendlich doch eines dieser Felder betritt. Ebenfalls für Fall F-B spricht, dass man die Formulierung so auffassen könnte, dass "in Richtung der stärksten Konzentration" nur eine Erweiterung der Anweisung "läuft [...] weiter weg vom Nest" ist und somit das Entfernen vom Nest höchste Priorität hat.

Umsetzung

Die Umsetzung erfolgt als ein in Java geschriebenes Programm. Exportiert wird diese Version als Jar-Datei gepaart mit einer Batch-Datei zur Ausführung. Getestet wurde das Programm unter Windows 7 Home Premium SP1 64-Bit mit JDK 1.8.0_65 64-Bit.

Die Simulationsparameter (Anzahl der Ameisen, Anzahl der Futterquellen, Verdunstungszeit und Nestposition) werden in der Batch-Datei angegeben.

In der Umsetzung werden insgesamt acht verschiedene Klassen erstellt:

Position	Hält zwei Integer *x* und *y*. Wird in den meisten Funktionen als Standardformat bei der Übertragung von Koordinaten verwendet.
Main	Nimmt die in der Batch-Datei angegebenen Simulationsparameter entgegen und erstellt Instanzen der Klassen View, Model und Controller.
Controller	Beobachtet das Model (implementiert Observer) und aktualisiert die View nach Vollendung eines Zuges. Nimmt die Benutzereingaben durch die View entgegen (implementiert ActionListener) und gibt dementsprechend Befehle zum Ausführen von Zügen an das Model weiter.
View	Erbt von JFrame. Beinhaltet alle sichtbaren Objekte, sprich die Anzeige der Simulation und alle Knöpfe.

PanelSim Teil der View, erbt von JPanel. Zeigt die Welt in der Simulation an. Ameisen werden schwarz, das Nest braun, Futterquellen weiß, duftende Felder violett und der Rest grün markiert.

Model Wird vom Controller beobachtet, erbt demnach von Observable. Beinhaltet alle Daten der Simulation selbst, inklusive eines Arrays aller MapTiles (*world*) und Ameisen (*ameisen*).

MapTile Stellt eines der 500x500 Felder in der Simulation dar. Beinhaltet die Anzahl an Futtereinheiten auf dem Feld als Integer und die Anzahl an Duftpunkten als Double. Berechnet im Konstruktor den Faktor fVerdunstung aus der Verdunstungszeit für die exponentielle Abnahme der Duftstärke.

Ameise Stellt eine Ameise in der Simulation dar. Beinhaltet Position der Ameise als Integer, einen Boolean, um die Ameise als beladen oder unbeladen zu markieren, und Funktionen zur Simulation des Verhaltens der Ameise.

Jeder Zug läuft über die Funktion zugAusführen() des Model-Objektes ab. Die Verdunstung findet hierbei vor dem Agieren der Ameisen statt:

```
public void zugAusführen(){
	// Verdunstung simulieren
	for (MapTile[] i : world)
		for (MapTile j : i)
			j.decayDuft();

	// Ameisen agieren lassen
	for (Ameise a : ameisen)
		a.zug();

	// Anzahl an gemachten Zügen erhöhen
	zgeInsg++;

	// Controller informieren, dass Zug beendet wurde
	setChanged();
	notifyObservers();
}
```

Die Verdunstung des Duftes wird durch die MapTiles selbst simuliert:

```
/** Senkt die Intensität des Duftstoffes exponentiell.
 *     Sinkt die Intensität auf oder unter 0.1, gilt der Duftstoff als verdunstet.
 *     Die Verdunstungszeit wird als die Zeit definiert, die es braucht, damit ein
 *     Duft der Intensität 1 auf 0.1 sinkt und damit verschwindet. Intensivere
 *     Düfte halten somit etwas, aber nur wenig relativ zu ihrem Intensitätsunterschied,
 *     länger an.
 */
public void decayDuft() {
	if (duft > 0)
		duft *= fVerdunstung;
```

```
        if (duft <= .1)
                duft = 0;
}
```

Die wohl größte Funktion des gesamten Programms stellt die Simulation für das Verhalten der Ameisen:

```
public void zug(){
        // Aktuelles Kartenfeld merken
        MapTile curTile = Model.getWorld()[pos.getX()][pos.getY()];

        // Laufrichtung notieren
        int dx = 0, dy = 0;

        // Nächstes Nest-Feld finden
        findNestNear();

        // Verhalten für Ameise ohne Futter
        if (!hatFutter){
                // Falls Ameise bereits auf Futterplatz, Futter aufheben und Zug beenden
                if (curTile.getFutter() > 0){
                        aufheben(curTile);
                        return;
                }

                // Wenn kein Futter gefunden wurde, umliegende Felder untersuchen
                MapTile high = null;

                // Überprüfen, ob umliegende Felder existieren. Felder, die näher ans Nest führen, werden ignoriert.
                MapTile tileLeft = null,
                                tileRight = null,
                                tileUp = null,
                                tileDown = null;
                // Nur links in Betracht ziehen, wenn das Nest auf gleicher x-Koordinate oder rechts davon liegt.
                if (pos.getX() > 0 && pos.getX() - nestNearX <= 0)
                        tileLeft = Model.getWorld()[pos.getX() - 1][pos.getY()];
                if (pos.getX() < 499 && pos.getX() - nestNearX >= 0)
                        tileRight = Model.getWorld()[pos.getX() + 1][pos.getY()];
                if (pos.getY() > 0 && pos.getY() - nestNearY <= 0)
                        tileUp = Model.getWorld()[pos.getX()][pos.getY() - 1];
                if (pos.getY() < 499 && pos.getY() - nestNearY >= 0)
                        tileDown = Model.getWorld()[pos.getX()][pos.getY() + 1];


                // Existierende umliegende Felder untersuchen
                        // Nur untersuchen, wenn Feld überhaupt existiert. Wenn noch kein anderes passendes
```

```
                // Feld gefunden wurde, nur prüfen, ob Duftpunkt vorhanden sind.
                // Für das erste Feld (tileLeft) kann die Überprüfung verkürzt werden, da noch kein
                // anderes passendes Feld gefunden worden sein kann.
        if (tileLeft != null && tileLeft.getDuft() > 0){
                high = tileLeft;
                dx = -1;
        }
        if (tileRight != null &&
                        ((high == null && tileRight.getDuft() > 0) ||
                        (high != null && tileRight.getDuft() > high.getDuft()))){
                high = tileRight;
                dx = +1;
        }
        if (tileUp != null &&
                        ((high == null && tileUp.getDuft() > 0) ||
                        (high != null && tileUp.getDuft() > high.getDuft()))){
                high = tileUp;
                dx = 0; dy = -1;
        }
        if (tileDown != null &&
                        ((high == null && tileDown.getDuft() > 0) ||
                        (high != null && tileDown.getDuft() > high.getDuft()))){
                high = tileDown;
                dx = 0; dy = +1;
        }

        // Falls keine Duftpunkte gefunden wurden, zufällig bewegen
        if (high == null){
                do {
                // x- oder y-Richtung
                if (rnd.nextBoolean())
                        // Links oder Rechts
                        if (rnd.nextBoolean())
                                dx = -1;
                        else
                                dx = +1;
                else
                        // Oben oder Unten
                        if (rnd.nextBoolean())
                                dy = -1;
                        else
                                dy = +1;
                }
                // Falls die Zufallsbewegung über die Weltgrenzen führen würde, Bewegung neu berechnen
                while (!(0 <= pos.getX() + dx && pos.getX() + dx <= 499 &&
```

```
                                0 <= pos.getY() + dy && pos.getY() + dy <= 499));
            }
        }


        // Verhalten für Ameise mit Futter
        else {
            // Falls die Ameise bereits auf dem Nest angekommen ist, Futter ablegen und Zug beenden
            if (curTile.isNest()){
                ablegen();
                return;
            }

            // Sonst Duftpunkt legen
            duftAbgeben(curTile);

            // Da Ameisen nicht diagonal laufen können, macht es keinen Unterschied, ob ein gerader Weg zum Nest gewählt
            // oder zuerst nur auf einer Dimension in Richtung Nest gelaufen wird.

            // Richtung (x oder y) mit größerer Differenz zuerst korrigieren
            if (Math.abs(pos.getX() - nestNearX) > Math.abs(pos.getY() - nestNearY))
                // Ein Feld in die entsprechende Richtung laufen
                dx = (int)Math.copySign(1, nestNearX - pos.getX());
            else
                dy = (int)Math.copySign(1, nestNearY - pos.getY());
        }

        // Zum gewählten Feld laufen
        pos.setX(pos.getX() + dx);
        pos.setY(pos.getY() + dy);
}
```

Beispiele und besondere Effekte bei bestimmten Werten

Einstellung	Effekt
Hohe Anzahl an Futterquellen	Es werden schneller Futterquellen gefunden. Duftwege zu einzelnen Futterquellen überschneiden sich teilweise, was dazu führt, dass einige Futterquellen nicht mehr verwendet werden, obwohl sie noch Futter tragen, weil die Ameisen stattdessen eine stärker duftende Abzweigung nehmen. Bei niedrigen Verdunstungszeit kann die Futterquelle komplett vergessen werden, bis sie neu entdeckt wird.
Niedrige Anzahl an Futterquellen	Es dauert sehr lange, bis Ameisen Futter finden.
Niedrige Verdunstungszeit und niedrige Ameisenzahl	Wenn der Weg von einer Futterquelle zum Nest so lang wird, dass die Ameisen, die diesen Weg betreten, nicht wieder zur Quelle zurückkehren können, bis der Duftweg verschwunden ist, wird die Futterquelle verloren und muss neu gefunden werden. Dies tritt ein, wenn die Wege sehr schnell verdunsten oder zu wenig Ameisen in der Nähe sind, um dem Weg zu folgen und weitere Duftstoffe abzugeben.
Niedrige Verdunstungszeit	Duftwege vom Nest zu bereits geleerten Futterquellen bleiben sehr lange erhalten und dienen als „Schnellstraßen“ in weiter entfernte Gebiete, da Ameisen diesem Weg folgen, bis sie zum Ende gelangt sind und von dort aus weitersuchen. Dies kann dazu führen, dass die Welt schneller komplett untersucht wird, da weniger Ameisen bereits abgesuchte Gebiete in Nestnähe durchlaufen (→ Abbildung 1).
Nestposition in der Mitte	Die Ameisen verteilen sich ungefähr gleichmäßig in alle Richtungen und die maximalen Entfernung zwischen Futterquellen und Nest ist minimal.
Nestposition in einer Ecke	Da sich die Ameisen mit höherer Wahrscheinlichkeit in den ersten Teilen der Simulation in der Start-Ecke aufhalten (→ Abbildung 3) und es möglich ist, dass Futterquellen am weit entfernten anderen Ende der Karte liegen, dauert es erheblich länger, bis die Ameisen alle Quellen gefunden haben. Aufgrund der großen Distanzen ist es wahrscheinlicher, dass die Duftwege verdampft sind, bis genug Ameisen diesen gesichert haben. Ist der Duftweg allerdings einmal stabil, befinden sich in der Nähe des Nestes mehr Ameisen als im Standard-Szenario, weshalb die Futterquelle trotzdem recht schnell abgebaut wird.
Sehr hohe Anzahl an Ameisen und Futterquellen	Bei extrem hohen Anzahlen an Ameisen und Futterquellen, beispielsweise 20.000 Ameisen und 200 Futterquellen entstehen sehr viele Duftwege nach außen und ein Großteil der Ameisenpopulation bewegt sich an die Kartenränder, wobei die Chance besteht, dass Futterquellen im Inneren der Karte übersehen werden oder große Gruppen an Ameisen von Duftwegen eingeschlossen werden

	und sich nur innerhalb dieser Grenzen aushalten, bis die Wege verdunstet sind (→ Abbildung 2).

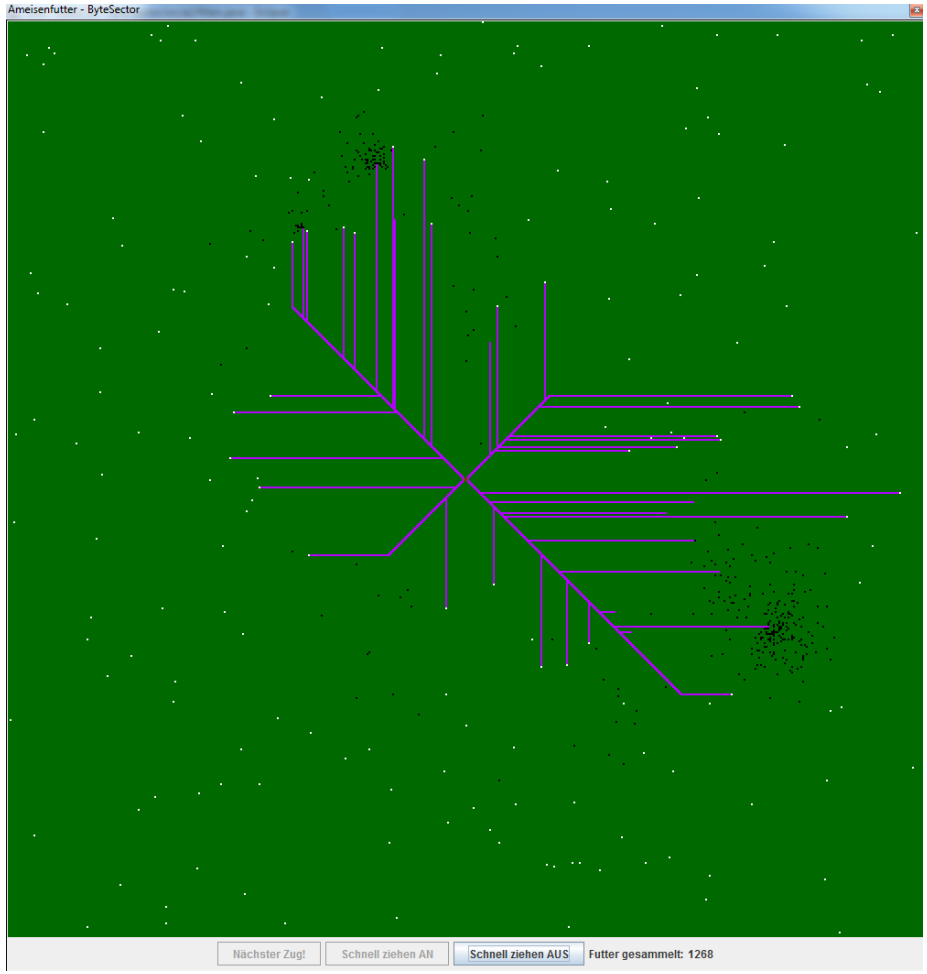

Abb. 1: „Schnellstraßen“

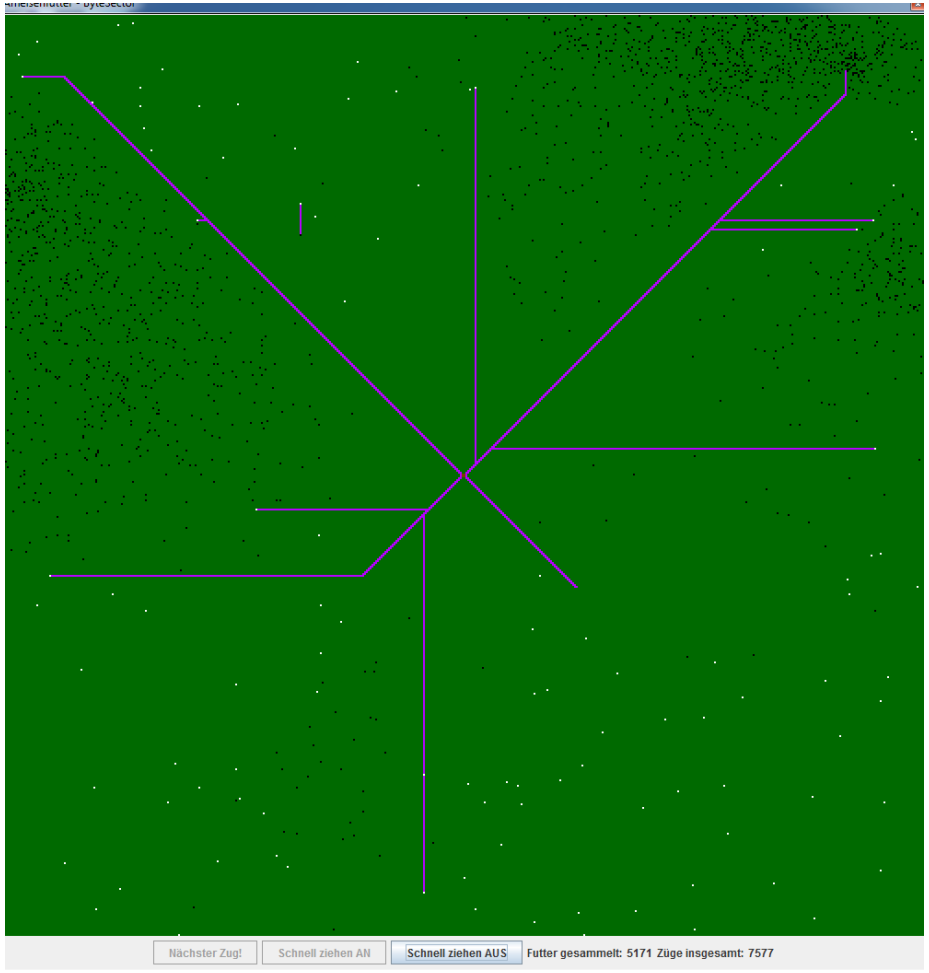

Abb. 2: Ameisen weit am Rand und zwischen Duftwegen eingeschlossen

Abb. 3: Mit dem Nest in der Mitte verteilen sich die Ameisen schneller auf die ganze Karte