1. Programmieraufgabe

Programmierparadigmen

LVA-Nr. 194.023
2023/2024 W
TU Wien

Themen:

Aufbau der Zusammenarbeit in der Gruppe, Einrichten einer Arbeitsumgebung, Modularisierung, nominale Abstraktion

Ausgabe:

09. 10. 2023

Abgabe (Deadline):

16. 10. 2023, 14:00 Uhr

Abgabeverzeichnis:

`Aufgabe1-3`

Hochladen ins Git-Repository mittels `push`

auch eingebundene Pakete hochladen (außer Standard-Pakete)

Programmaufruf:

`java Test`

Grundlage:

Kapitel 1 des Skriptums

Raum und Felder

Kontext

Eine Ameise ist klein, oft lästig und dennoch faszinierend. Manchmal scheint sie orientierungslos herumzuirren. Ein andermal läuft sie sehr zielgerichtet entlang einer optimal angelegten Ameisenstraße und schleppt dabei nicht selten Beute mit einem Mehrfachen ihres eigenen Körpergewichts. Obwohl sie stets einem einfachen, fest verdrahteten Schema folgt, kann sie ganz unterschiedliche Verhaltensweisen zeigen und zusammen mit zahllosen anderen Ameisen einen komplex organisierten Staat bilden.

Ameisen scheiden kleine Mengen eines flüchtigen Duftstoffs aus. Hat eine Ameise Futter gefunden, trägt sie ein Stück davon zurück zum Bau und informiert so andere Ameisen über den Fund. Diese Ameisen folgen der auf dem Rückweg gelegten Duftspur (die jünger und besser wahrnehmbar ist als andere Spuren) und gelangen so zur Futterquelle. Wegen der Lückenhaftigkeit der Duftspur nehmen nicht alle Ameisen den gleichen Weg, sondern müssen immer wieder Umwege auf der Suche nach der Spur einlegen. Dabei hinterlassen die Ameisen viele Duftspuren zum gleichen Ziel. Wenn Sie Futter zum Bau zurückbringen und sich wieder zur Futterquelle begeben, folgen sie der am stärksten wahrgenommenen Duftspur. Das muss nicht der kürzeste Weg sein. Mit jedem Durchlaufen eines Wegs verstärkt sich die Duftspur. Duftspuren, die stärker vom optimalen Weg abweichen, sind bald kaum mehr wahrnehmbar, während sich Spuren in der Nähe des Optimums verstärken. Das liegt daran, dass längere Umwege länger dauern und sich während einer längeren Zeit mehr Duft verflüchtigt. Je öfter die Futterquelle besucht wird, desto stärker wird eine Duftspur, die nahe am optimalen Weg liegt. Das ist die Ameisenstraße. Wird die Ameisenstraße durch ein Hindernis blockiert, bildet sich bald ein gut optimierter Umgehungsweg, weswegen der von den Ameisen verfolgte Algorithmus recht stabil ist. Auch in der Informatik werden Algorithmen eingesetzt, die nach dem Vorbild der Ameisen gestaltet sind.

Welche Aufgabe zu lösen ist

Simulation Entwickeln Sie ein Java-Programm, das Bewegungen von Ameisen in einem zweidimensionalen, rechteckigen Raum simuliert, z. B. 100 Ameisen in einem Raum aufgeteilt auf etwa 50 000 Felder. Jedes Feld, dargestellt durch ein Pixel oder einige wenige Pixel, trägt eine Duftspur einer bestimmten Stärke (Stärke z. B. durch Farbintensität dargestellt), die mit jedem Simulationsschritt schwächer wird (durch Multiplikation mit einem Faktor kleiner eins, z. B. 0.98) und sich um einen konstanten Betrag erhöht, wenn sich eine Ameise darauf befindet. Aktuelle Positionen der Ameisen werden ebenso grafisch dargestellt, etwa durch je ein schwarzes Feld oder eine winzige Zeichnung einer Ameise. Zur Vereinfachung können sich auf jedem Feld beliebig viele Ameisen gleichzeitig aufhalten. Ränder des Raums sind so organisiert, dass eine Ameise, die den Raum auf einer Seite verlässt, auf der gegenüberliegenden Seite erscheint; d. h., der Raum bildet einen grenzenlosen Torus. Über den Raum

verteilt befinden sich einige (etwa 3 bis 10) Futterquellen, jeweils mit unendlichen Futtervorräten, irgendwo im Raum ist ein Ameisenbau, jeweils deutlich gekennzeichnet. Ameisen suchen Futter und transportieren es von den Quellen zum Bau.

Bewegung der Ameisen

Jede Ameise bewegt sich in jedem Simulationsschritt auf ein benachbartes, geradeaus bis seitlich liegendes Feld. Erlaubt sind also die Felder links, halblinks, vor, halbrechts oder rechts von der Ameise, jeweils von der letzten Bewegungsrichtung aus gesehen. Die tatsächliche Bewegungsrichtung wird zufällig aus den je fünf möglichen Richtungen gewählt, wobei die Wahrscheinlichkeiten der Richtungen von den Duftspuren und dem Zustand der Ameise abhängen. Davon ausgenommen sind Ameisen, die sich an einer Futterquelle befinden und davor kein Futter mit sich getragen haben (sie nehmen Futter auf) und Ameisen, die sich im Ameisenbau befinden (sie legen Futter ab, falls sie welches mitführen): Diese Ameisen beginnen zurückzulaufen, laufen also in die Richtung, aus der sie im letzten Schritt gekommen sind.

Eine Ameise ist stets in einem der folgenden Zustände:

Zustände der Ameisen

Erkundung: Die Ameise trägt kein Futter mit sich und sucht nach einer noch unentdeckten Futterquelle. Um neue Gegenden zu erforschen, bevorzugt sie Richtungen mit schwacher oder keiner Duftspur. Wenn sie ein Nachbarfeld mit einer sehr starken Duftspur findet oder auf eine Ameise trifft, die Futter mit sich trägt, wechselt sie in den Zustand „Futtersuche“. Wenn sie auf eine Futterquelle stößt, nimmt sie Futter auf und wechselt in den Zustand „Futterbringung“.

Futtersuche: Die Ameise trägt kein Futter mit sich, nimmt aber an, dass Duftspuren sie zu einer Futterquelle führen. Bevorzugt werden Richtungen mit möglichst starker Duftspur. Wenn die Ameise auf eine Futterquelle stößt, nimmt sie Futter auf und wechselt in den Zustand „Futterbringung“. Wenn sie über eine bestimmte Anzahl von Schritten auf ihren Nachbarfeldern jeweils nur schwache Duftspuren findet, wechselt sie in den Zustand „Erkundung“.

Futterbringung: Die Ameise trägt Futter mit sich. Bevorzugt werden Richtungen mit möglichst starker Duftspur. Wenn die Ameise auf den Bau stößt, legt sie das Futter ab und wechselt in den Zustand „Futtersuche“.

Zu Programmbeginn sind die Ameisen im oder in unmittelbarer Nähe zum Bau platziert und befinden sich im Zustand „Erkundung“.

Der zweidimensionale Raum soll als Bildschirmfenster dargestellt werden, z. B. durch das AWT/Swing-Framework. Sie können aber auch das aus EP1 und EP2 bekannte `CodeDraw` verwenden. Fensterinhalte sollen regelmäßig, aber nicht notwendigerweise nach jedem Simulationsschritt aktualisiert werden, damit sich Bewegungen beobachten lassen.

passende Parameterwerte durch Ausprobieren finden

Details bestimmen, wie sich die Ameisen bewegen. Möglicherweise kommt es auf die Größe des Raums, die Anzahl der Ameisen, die Stärke der gesetzten Duftmarken, die Geschwindigkeit von deren Abnahme, die Anzahl und Entfernung der Futterquellen und so weiter an. Um dem Rechnung zu tragen, soll Ihr Programm hintereinander mindestens drei Simulationen mit unterschiedlichen Details ausführen, sodass die Abläufe verschieden voneinander wirken.

drei Simulationsläufe

Testen Testen Sie Ihr Programm sorgfältig. Sorgen Sie dafür, dass nach dem Programmstart keine Benutzerinteraktion (das heißt, keine Eingabe über die Tastatur oder Maus) nötig ist, sondern einfach nur ein Fenster erscheint, in dem die Simulationen betrachtet werden können. Es ist aber erlaubt, dass auf der Konsole im Hintergrund Informationen einschließlich Debug-Output ausgegeben werden und das Fenster am Ende mit einem Mausklick geschlossen werden muss. Das Programm soll (nach neuerlicher Übersetzung aller vorhandenen `.java`-Dateien) mittels `java Test` vom Verzeichnis `Aufgabe1-3` aus aufrufbar sein.

Programmname „`Test`“ aus gutem Grund gewählt

im richtigen Verzeichnis ausführbar?

Kommentare Betrachten Sie jede Definition und Deklaration einer Methode als nominale Abstraktion. Beschreiben Sie diese Abstraktionen durch kurze, aber informative Kommentare. Versehen Sie auch jede Klasse und jedes Interface mit entsprechenden Kommentaren. Eine Java-Klasse kann Elemente unterschiedlicher Modularisierungseinheiten enthalten (Modul, Klasse, Objekt). Beschreiben Sie in den Kommentaren deutlich, welche Elemente zu welchen Modularisierungseinheiten gehören und welche Abstraktionen hinter den Modularisierungseinheiten stecken.

Abstraktionen und Modularisierungseinheiten kurz beschreiben (siehe Skriptum)

Neben Programmtext soll die Datei `Test.java` als Kommentar eine kurze, aber verständliche Beschreibung der Aufteilung der Arbeiten auf die einzelnen Gruppenmitglieder enthalten – wer hat was gemacht. Beschreibungen wie die folgende reichen nicht: „Alle haben mitgearbeitet.“

Arbeitsaufteilung kurz, verständlich beschreiben

Wie die Aufgabe zu lösen ist

einfach halten

Der Programmtext Ihrer Lösung soll möglichst einfach sein und keine unnötige Funktionalität haben. Er soll wiederverwendbar sein, da die nächste Aufgabe auf Teilen davon aufbaut. Vermeiden Sie jedoch Vorgriffe, das heißt, schreiben Sie keine Programmteile aufgrund der Vermutung, dass diese Teile in der nächsten Aufgabe verlangt sein könnten.

experimentieren

Zufallszahlen spielen eine große Rolle, vor allem für die Wahl der Bewegungsrichtung. Allerdings darf die Auswahl der Richtung nicht rein zufällig sein, sondern muss sich nach dem Zustand und den Duftspuren richten. Es kommt auf ein ausgewogenes Verhältnis zwischen Zufall und den algorithmischen Vorgaben an. Um gute Einstellungen zu finden, wird es notwendig sein, mit verschiedenen Werten und Vorgehensweisen zu experimentieren. Bei Simulationen geht es meist, nicht nur in dieser Aufgabe darum, passende Parametrisierungen zu finden, sodass Simulationsergebnisse das möglichst gut widergeben, was wir in der Natur vorfinden. Im Idealfall werden Details des Programms (allgemein als Parameter bezeichnet, nicht nur Methodenparameter) so gewählt, dass tatsächlich bald Ameisenstraßen entstehen, die nahe am Optimum liegen.

alle arbeiten mit

Diese Aufgabe hilft Ihren Tutor_innen, Ihre Kenntnisse sowie die Zusammenarbeit in der Gruppe einzuschätzen. Bitte sorgen Sie in Ihrem eigenen Interesse dafür, dass jedes Gruppenmitglied etwa in gleichem Maße mitarbeitet. Sonst könnten Sie bei einer Fehleinschätzung wertvolle Zeit verlieren. Scheuen Sie sich bitte nicht, Ihre Tutorin oder Ihren Tutor um Hilfe zu bitten, falls Sie bei der Lösung der Aufgabe Probleme haben oder keine brauchbare Zusammenarbeit in der Gruppe zustandekommt.

Warum die Aufgabe diese Form hat

Umfang und Schwierigkeitsgrad der Aufgabe wurden so gewählt, dass die eigentliche Programmierung bei guter Organisation und entsprechendem Vorwissen nicht zu viel Zeit in Anspruch nimmt, aber eine Einarbeitung in neue Themen nötig ist und Diskussionsbedarf entsteht. Nutzen Sie die Gelegenheit, um die Aufgabenverteilung und interne Abläufe innerhalb der Gruppe zu organisieren. Details der Aufgabe bleiben bewusst offen:

- Sie sollen in der Gruppe diskutieren, wie Sie die Aufgabe verstehen und welche Lösungswege geeignet erscheinen.
- Sie sollen sich daran gewöhnen, dass Aufgaben nicht vollständig spezifiziert sind, aber trotzdem Vorgaben eingehalten werden müssen.
- Sie sollen sich eine eigene brauchbare Faktorisierung überlegen und sich dabei vorerst einmal nur von Abstraktionen leiten lassen.
- Sie sollen auch die Verantwortung für die Korrektheit Ihrer Lösung (so wie Sie sie selbst verstehen) übernehmen, indem Sie entsprechende Tests durchführen.

Allgemeine Informationen zur Übung

Folgende Informationen betreffen diese und auch alle weiteren Aufgaben.

Was bei der Lösung der Aufgabe zu beachten ist

Schwerpunkte beachten

Unter der Überschrift „Wie die Aufgabe zu lösen ist“ finden Sie Hinweise darauf, wie Sie die Lösung der Aufgabe vereinfachen können und welche Fallen Sie umgehen sollen, erfahren aber auch, welche Aspekte bei der Beurteilung als wichtig betrachtet werden. In der ersten Aufgabe kommt es beispielsweise auf Datenabstraktion und die Einfachheit der Lösung an. Das heißt, in späteren Aufgaben können Ihnen bei solchen Hinweisen auch für unnötig komplizierte oder umfangreiche Lösungen Punkte abgezogen werden, weil Sie sich nicht an die Vorgaben gehalten haben. Unterschiedliche Aufgaben haben unterschiedliche Schwerpunkte. Die nächste Aufgabe wird nicht nach dem gleichen Schema beurteilt wie die vorige. Richten Sie sich daher nach der jeweiligen Aufgabenstellung.

strukturiert vorgehen

Ein häufiger Fehler besteht darin, eine Aufgabe nach Gefühl zu lösen ohne zu verstehen, worauf es ankommt. Meist bezieht sich die Aufgabe auf ein Thema, das zuvor theoretisch behandelt wurde. Versuchen Sie, eine Beziehung zwischen der Aufgabenstellung und davor behandeltem Stoff herzustellen. Achten Sie darauf, Fachbegriffe nicht nur umgangssprachlich zu interpretieren, sondern so wie im Skriptum beschrieben. Die ersten Aufgaben sind vermutlich auch ohne Skriptum lösbar, spätere aber kaum. Als Hilfestellung sind in jeder Aufgabenstellung Teile des Skriptums genannt, in denen die relevantesten Themen behandelt werden, bei komplizierten Themen oft nur wenige Seiten.

Versuchen Sie nicht, Teile der Aufgabenstellung durch Tricks oder Spitzfindigkeiten zu umgehen. Beispielsweise gibt es immer wieder Lösungsversuche, in denen die Test-Klasse nur den String „Test erfolgreich“

ausgibt, statt tatsächlich sinnvolle Tests durchzuführen. Solche Versuche werden durch händische Beurteilungen mit hoher Wahrscheinlichkeit erkannt. Spätere Aufgaben enthalten oft Schwierigkeiten, die mit Allgemeinwissen alleine oder über aufgabenbezogene Web-Recherchen kaum zu lösen sind. Gerade in solchen Fällen ist davon abzuraten, die Schwierigkeiten durch Tricks zu umgehen. Hinweise zur richtigen Lösung lassen sich im Skriptum und auf den Vorlesungsfolien finden.

keine Spitzfindigkeiten

Ihre Lösung bestehend aus `.java`-Dateien muss am Tag der Abgabe um 14:00 Uhr pünktlich im Git-Repository Ihrer Gruppe stehen. Übersetzte `.class`-Dateien sollen nicht ins Repository gestellt werden, da sie die Zusammenarbeit in der Gruppe erschweren und vor der Beurteilung ohnehin neu generiert werden. Zugangsinformationen zum Repository erhalten Sie in kürze. Informationen zum Umgang mit dem Repository finden Sie in TUWEL. Es wird empfohlen, rechtzeitig vor der Deadline die Lösung auf dem Übungsrechner auszuprobieren und die Daten dabei durch `pull` aus dem Repository zu laden. So können Sie typische Fehler bei der Abgabe (z.B. auf `push` vergessen, Lösung im falschen Verzeichnis, falsche `package`- und `include`-Anweisungen, Klassen aus Nicht-Standard-Paketen verwendet und nicht mit abgegeben) sowie Inkompatibilitäten aufgrund unterschiedlicher Versionen und Betriebssystemeinstellungen (z. B. Dateinamen mit Umlauten sowie neueste Sprach-Features nicht unterstützt) erkennen und beseitigen.

ausprobieren

Was Ihr_e Tutor_in von Ihnen wissen möchte

Ihre Tutorin oder Ihr Tutor wird Ihnen in kürze eine Mail schreiben, um sich vorzustellen und um Informationen über Sie zu bitten. Geben Sie diese Informationen möglichst bald, damit die für Sie am besten geeignete Form der Betreuung gewählt werden kann. Unabhängig von der Form der Betreuung kann natürlich jedes Gruppenmitglied jederzeit konkrete Fragen an Tutor_innen richten. Scheuen Sie sich bitte nicht, sich auch mit organisatorischen oder gruppeninternen Problemen, die Sie möglicherweise nicht selbst lösen können, an Tutor_innen zu wenden. Früh im Semester sind Probleme meist einfacher zu lösen als im bereits weit fortgeschrittenen Semester.