Fachbereich 3 Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme
R. Kollmann, A. Lattner, S. Prelle,
M. Richters, H. Shi, A. Werner, M. Gogolla

9. April 2001

Softwarepraktikum 2001

Aufgabe „Scotland Yard“

“ ‘On the face of it the case is not a very complex one, though it certainly presents some novel and interesting features. A further knowledge of facts is necessary before I would venture to give a final and definite opinion.’ ”

1 Einführung

Wer kennt es nicht, das Spiel des Jahres 1983[1]. Im Verkehrsgewühl der pulsierenden Großstadt London treibt sich ein Agent, genannt Mr. X, herum. Die berühmten Detektive des Scotland Yard haben die Aufgabe, Mr. X zu finden. Keine leichte Aufgabe, da die Verkehrsverbindungen Londons sehr vielfältig und unübersichtlich geworden sind. Obwohl die Briten für ihre Traditionsliebe bekannt sind, sind sie doch neuen Dingen gegenüber aufgeschlossen, wenn sie damit die Hoffnung auf mehr Bequemlichkeit verbinden. So haben die Detektive beschlossen, ein Computerprogramm einzusetzen, das die Züge des unsichtbaren Mr. X vorausberechnet und ihnen so die Arbeit erleichtert. Die sportliche Fairness, ein weiteres Prinzip der Inselbewohner, gebietet allerdings, daß auch Mr. X einen Computer einsetzt, um sich seinen Häschern zu entziehen.

2 Aufgabenstellung

Im Laufe dieses Softwarepraktikums soll jede Gruppe ein Programm erarbeiten, das sinnvolle Spielzüge für Mr. X oder einen Detektiv berechnet. Mehrere Versionen dieses Programms – auch unterschiedlicher Gruppen –

[1]Wer Scotland Yard wirklich nicht kennt, sollte sich jemanden suchen, der das Spiel hat, und einmal ein paar Runden ausprobieren.

sollen dann auf verschiedenen Rechnern gemeinsam eine regelgerechte Partie Scotland Yard spielen können. Hierbei sollen sich die einzelnen Gruppen nach einem gemeinsamen Kommunikationsprotokoll richten.

Zur Verwaltung der Kommunikation der einzelnen Spieler soll der Scotland-Yard-Server eingesetzt werden. Eine detailliertere Erläuterung dieses Protokolls ist der Beschreibung des Servers zu entnehmen. Der Server soll mit allen dort angegebenen Funktionen realisiert werden.

Um die Programme universell einsetzen zu können, werden möglichst wenige der Spielregeln in dem Programmen selbst kodiert, sondern über von außen veränderbare Dateien festgelegt. So soll zum einen der Stadtplan variabel sein. Das Programm sollte auch in Paris, Tokio oder Amsterdam einsetzbar sein. Zum anderen sind die verschiedenen Verkehrsmittel nicht festgelegt. Warum sollte man in Paris nicht auch Metro fahren können oder in Amsterdam ein Stück zu Fuss gehen? Damit verbunden ist natürlich auch, daß verschiedene Fahrkarten mit verschiedenen Gültigkeitsbereichen zum Einsatz kommen. Selbstverständlich soll auch die Anzahl der Fahrkarten, die jedem Spieler zur Verfügung steht, von Spiel zu Spiel verändert werden können. Es gibt also verschiedene Dateien, die diese Parameter festlegen und daher dem Server vor Spielbeginn zur Verfügung stehen müssen.

2.1 Obligatorische Anforderungen

Die Spielumsetzung jeder Gruppe soll eine Menge von Mindestanforderungen erfüllen. Die Anforderungen sind im Einzelnen:

- **Client & Server.** Jede Gruppe muß einen Client und einen Server implementieren. Der Server dient zur allgemeinen Spielverwaltung. Hier können sich die Spieler anmelden und Züge durchführen. Der Client dient zur Interaktion eines Spielers mit dem Server. Der Client muß also Rückmeldung über den Spielstatus geben und mögliche Interaktionen mit dem Server darstellen und ausführen können.

- **Protokollunterstützung.** Die Kommunikation zwischen Clients und Servern erfolgt über das vorgegebene Protokoll. Um eine Kommunikation zwischen den Programmen unterschiedlicher Gruppen zu ermöglichen, muß das vorgegebene Protokoll unterstützt werden.

- **Parsing von Eingabedateien.** Das Programm muß die in dieser Aufgabenstellung vorgegebenen Eingabedateien parsen können und die Informationen in geeignete Datenstrukturen ablegen können.

- **Spielverwaltung.** Über die Spielverwaltung muß der regelgerechte Ablauf des Spiels garantiert werden. Informationen über die Position und die Reisemöglichkeiten der Agenten müssen verfügbar sein. Jeder Spielzug muß überprüft werden, so daß nur erlaubte Züge durchgeführt werden können.

- **ASCII-Interface.** Auf der Client-Seite muß mindestens ein ASCII-Interface geboten werden, um Spielzüge durchzuführen. Das User Interface sollte aber mehr leisten, als lediglich eine Telnet-Verbindung zu kapseln, d.h. es sollen also nicht die Spielzüge in Form des Protokolls eingegeben werden können, sondern ein Menü mit möglichen Interaktionen geboten werden.

- **GUI oder Spielintelligenz** Jede Gruppe muß entweder eine graphische Benutzungsoberfläche oder eine Spielintelligenz realisieren. Die beiden Punkte werden bei den optionalen Anforderungen beschrieben.

2.2 Optionale Anforderungen

Die optionalen Anforderungen sind:

- **GUI.** Für den Client soll eine graphische Oberfläche ein komfortableres Spielen ermöglichen. Die Durchführung von Spielzügen und anderen Interaktionen soll über geeignete GUI-Bausteine wie Menüs, Buttons, Auswahllisten, Comboboxen, etc. erleichtert werden.

- **Spielintelligenz.** Die Spielintelligenz soll für Mr.X oder einen Agenten die Menge der möglichen Züge analysieren und bewerten, so daß ein Spieler durch die Bewertung den besten (oder einen der besseren) Züge durchführen kann. Ergebnis der Bewertung ist also eine Liste von möglichen Spielzügen mit einem Wert, dessen Höhe eine Aussage über die Güte des Zuges macht.

- **Grafische Spielbrettdarstellung.** Eine Erweiterung für das GUI wäre, das Spielfeld durch eine Graphik oder durch ein eingescanntes Spielbrett darzustellen.

- **Computergegner.** Wenn sich eine Gruppe entschließt, eine Spielintelligenz zu programmieren, läge es nahe, diese auch für autonome Computerspieler zu verwenden. Bei einer Menge von bewerteten Spielzügen könnte der beste (oder einer der besten) vom Computerspieler automatisch durchgeführt werden.

- **Kommunikation unter Agenten.** Die Computerspieler könnten auch untereinander kommunizieren, in dem sie beispielsweise Vermutungen über den Aufenthalt des Mr.X, den eigenen Plan oder Befehle an andere Agenten abgeben.

3 Eingabedateien

> “ ‘Now, Mr. Holmes, you have got the facts.’ ‘They are singular, not to say grotesque,’ said Holmes. ”

3.1 Der Spielplan

Die Spieler bewegen sich in einem ungerichteten Graphen. Der Graph besteht aus Knoten und Verbindungen zwischen den Knoten, den sogenannten Kanten. Jeder Knoten hat eine eindeutige positive ganzzahlige Nummer, wobei wir der Nummerierung des Spielplans folgen. Die Gestalt eines Graphen wird durch eine Textdatei `<name>.map` festgelegt. Als Beispiel stellen wir die Datei `london.map` bereit. Jede Zeile in einer solchen Datei hat folgende Gestalt:

```
Knotennummer1 Knotennummer2 Verbindungsart
```

Die beiden Knotennummern geben an, welche Knoten miteinander verbunden sind. Die Verbindungsart ist der Name des Verkehrsmittels, welches die beiden Knoten verbindet. Als Beispiel hier ein Ausschnitt aus der `.map`-Datei:

```
126 127 taxi
127 133 bus
127 133 taxi
```

Man beachte, daß hier nicht festgelegt ist, welche Namen an der Stelle Verbindungsart überhaupt auftauchen können. Weiterhin ist zu beachten, daß “ungerichteter Graph” bedeutet, daß die Knoten stets in beiden Richtungen miteinander verbunden sind, auch wenn nur eine Verbindung in der Datei `.map`-Datei angegeben ist.

3.2 Fahrkarten & Mitspieler

> “ Holmes lit his cigar and leaned back in his chair. ‘Let us hear about it,’ said he. ”

Eine weitere Datei, die vom Server verwaltet wird, ist das sog. *Game Resource Model* mit der Endung `.grm`; in unserem Beispiel `london1.grm`. Diese Datei muß folgende Informationen enthalten:

- **Der zugehörige Graph**
  Ein Dateiname verweist auf den zugehörigen Stadtplan – also z.B. die Datei `london.map`.

- **Verbindungsarten und Tickets**
  Die Verbindungsarten und Tickets werden in Zeilen folgenden Formates bestimmt:

  ```
  Fahrkarte Verbindungsart ... Verbindungsart
  ```

  Am Anfang jeder Zeile steht also ein beliebiger Name einer Fahrkarte, gefolgt von einer Liste von Verbindungen, für die die so bezeichnete Fahrkarte gültig ist.

  Es kann also verschiedene Fahrkarten für eine Verbindung geben. Genauso können mit einer Karte verschiedene Verbindungen genutzt werden.

  Hier ein Beispiel:

  ```
  Taxi-Ticket taxi
  Bus-Ticket bus
  U-Bahn-Ticket underground
  Black-Ticket bus taxi underground schiff
  ```

- **Anzahl und Funktion der Spieler**
  Es muß geregelt werden, wie viele Spieler in welcher Funktion an einer Partie teilnehmen können. Diese Fragen werden in Kombination mit dem nächsten Punkt geklärt.

- **Startpositionen und Züge**
  Eine Liste von Zahlen bestimmt die Züge, in denen sich Mr. X zeigen muß. Da sich Mr. X immer auch im letzten Zug zeigen muß, bestimmt die letzte Zahl der Liste auch die Anzahl der Züge in der Runde. Darauf folgt für jeden Spieler je eine Zeile. Diese Zeilen haben folgende Bedeutung. Die erste Zahl ist die Nummer des Spielers. Die Nummer 0 ist immer für Mr. X reserviert. Auf die Spielernummer folgt das Startfeld des Spielers. Daran schließt sich eine Liste an, die anzeigt, wieviele Fahrkarten der betreffende Spieler am Anfang hat. Die Reihenfolge in der Liste entspricht dabei der Reihenfolge, in der die Fahrkarten eingeführt wurden. Es folgt ein Beispiel für einen solchen Abschnitt:

  ```
  3 8 13 18 24
  0 50 4 3 3 4
  ```

```
1 141 10 8 4 0
2 112 10 8 4 0
3 53 10 8 4 0
4 29 10 8 4 0
```

Wie wir sehen, startet Spieler 3 auf Feld 53 mit 10 Fahrkarten `Taxi-Ticket`, 8 Fahrkarten `Bus-Ticket` und 4 Fahrkarten `U-Bahn-Ticket`.

Kommentare werden in einer *Game Resource Model*-Datei durch Prozentzeichen eingeleitet und enden beim Newline. Sie werden überlesen und gelten als Whitespace. Eine `.grm`-Datei muß in fester Reihenfolge die Trennwörter `MAPNAME`, `TICKETS` und `GAME` enthalten. Eine Beispieldatei könnte das folgende Aussehen haben:

```
% Der benutzte Stadtplan
MAPNAME london.map

% Tickets und Verbindungsarten
TICKETS

Taxi-Ticket taxi
Bus-Ticket bus
U-Bahn-Ticket underground

% Dieses Ticket kann in diesem Beispiel
% fuer mehrere Verbindungen genutzt werden
Black-Ticket bus taxi underground schiff

GAME
% Hier stehen die Runden, in denen sich Mr. X
% zeigen muß.
% Die Anzahl der Runden eines Spiels wird durch
% die letzte Runde bestimmt, in der sich Mr. X zeigen
% muß. (Hier also 24)
3 8 13 18 24

% Mr. X startet auf Feld 50, hat 4 Taxi-, 3 Bus-, 3 U-Bahn
% und 4 Black-Tickets
0 50 4 3 3 4

% Detektiv 1 startet auf Feld 141, hat 10 Taxi-Tickets ...
1 141 10 8 4 0
```

```
2 112 10 8 4 0
3 53 10 8 4 0
4 29 10 8 4 0
```

4 Spielablauf

4.1 Spielbeginn

Kommen wir nun zum Spielverlauf. Jeder Spieler kann eine Partie mit `create` eröffnen, oder an einer bereits eröffneten Partie mit join teilnehmen, falls noch ein Platz frei ist. Das Programm mit der Nummer 0 spielt dabei die Rolle von Mr. X. Die Gesamtzahl der Mitspieler ist durch das verwendete *Game Resource Model* festgelegt. Gezogen wird in aufsteigender Reihenfolge (Detektiv 1, Detektiv 2, ...) wobei der Server den jeweiligen Mitspieler mit der Meldung

```
syserv: Party n: Your turn as role
```

darauf aufmerksam macht, daß er in Partie `n` in der Rolle `role` am Zug ist.

4.2 Ein Spielzug

Ist ein Spieler am Zug, so muß er berechnen, auf welches Feld er mit welchem Verkehrsmittel als nächstes zieht. Für die Berechnung ist es wichtig zu wissen, wo die Mitspieler stehen. Diese Informationen erhalten die Spieler vom Server durch eine Meldung der Form

```
syserv: Party n: Detective d moved to f using a t.
```

D.h. Detektiv d hat sich in Partie n mit einem t-Ticket nach Feld f bewegt. Also z.B.:

```
syserv: Party 3: Detective 2 moved to 42 using a Taxi-Ticket.
```

Der Server darf natürlich nur dann die Zielfelder von Mr. X angeben, wenn sich dieser zeigen muß.

Hat ein Spieler seinen Zug berechnet, so schickt er dem Server die Nachricht `move n t f`, d.h. ziehe in Partie `n` mit Ticket `t` auf Feld `f`. Illegale Züge müssen vom Server zurückgewiesen werden. Kann ein Spieler nicht ziehen, z.B. weil er keine Tickets mehr hat, so läßt er den Zielpunkt weg und gibt als Fahrkarte `nil` an.

- Achten Sie darauf, daß der Server trotzdem ein regelgerechtes Spielen erzwingt.
- Denken Sie darüber nach, welche Informationen Ihnen zur Berechnung Ihres nächsten Zuges dienlich sein können. Versuchen Sie, das Verhalten Ihres Programms von der Position der anderen Mitspieler abhängig zu machen. Überlegen Sie, welche Felder Ihre Mitspieler in den nächsten Zügen mit ihren Tickets erreichen können. Inwieweit kann das beobachtete Zugverhalten von Mr. X und den anderen Detektiven genutzt werden?

4.3 Spielende

Der Server kann das Spiel abbrechen, wenn ein wichtiger Grund vorliegt. Ein wichtiger Grund wäre gegeben, wenn z.B. ein Detektivprogramm auf ein Feld zieht, auf dem sich Mr. X befindet oder wenn sich alle Spieler vor Mr. X nicht mehr bewegen können, also `nil` als Ticket angegeben haben.

Um den Abbruch herbeizuführen, schickt der Server an alle Mitspieler die Meldung

```
syserv: Party n: Game over. r,
```

wobei r den Grund des Abbruchs angibt.

- Überlegen Sie sich, welche Gründe es gibt, eine Partie abzubrechen.

4.4 Spielregeln & Schummeln

> “ ‘Good God! This is awful! You don’t mean – you don’t mean that I am suspected?’ ”

Es gelten alle Spielregeln wie im Originalspiel (s. Spielanleitung) mit der einen Ausnahme, daß Mr. X keine Doppelzüge ausführen darf. Ihre Detektivprogramme sollten so ehrlich sein, von der Startposition des Mr. X keinen Gebrauch zu machen. Falls Sie nicht allen Mitspielern diese Ehrlichkeit zutrauen, können Sie sich als Zusatzaufgabe überlegen, wie diesem Missbrauch begegnet werden könnte. Ihr Server hat dafür Sorge zu tragen, daß nur regelgerechte Züge akzeptiert werden. Muß der Server ein Kommando eines Clients zurückweisen so schickt er dem Client die Meldung: `Error: r`, wobei r eine durch Newline terminierte Zeichenkette ist, die den Grund des Scheiterns angibt.

5 Allgemeine Hinweise

" 'You've seen the paper, Mr. Holmes?' he asked, holding one out to us. 'Yes, Baynes, I've seen it. Pray don't think it a liberty if I give you a word of friendly warning.' 'Of warning, Mr. Holmes?' 'I have looked into this case with some care, and I am not convinced that you are on the right lines. I don't want you to commit yourself too far, unless you are sure.' 'You're very kind Mr. Holmes.' 'I assure you I speak for your good.' "

- Entwerfen Sie geeignete Datenstrukturen für die benötigten komplexen Datentypen, wie z.B. Graphen, Pfade, Kartenvorrat. Definieren Sie dazu Funktionen, die einen einfachen Zugriff gestatten.
- Teilen Sie die Aufgabe sinnvoll in Pakete ein. Denken Sie an Pakete zur Kommunikation, zur Behandlung Ihrer Datenstrukturen und zur strategischen Berechnung. Achten Sie darauf, daß diese Pakete sowohl für die Detektive als auch für Mr. X brauchbar sind.
- Behalten Sie im Auge, daß Sie keine "Konventionen" in Ihre Strategiemodule einprogrammieren. So sollte z.B. nicht nach dem Wort `underground` gesucht werden, um einen gut erreichbaren Knoten zu finden (es könnte ja auch `metro` heissen). Die Erreichbarkeit von Knoten soll nur aus der Struktur des Graphen berechnet werden. Denken Sie darüber nach, für die Erreichbarkeit und andere strategische Überlegungen Metriken einzuführen, auf Grund derer Sie Ihre Entscheidungen treffen können.
- Schreiben Sie auch ein Programm, das es Ihnen erlaubt, als menschlicher Spieler an einer Runde teilzunehmen. Dieses Programm muß gegenüber den anderen Programmen das vereinbarte Verhalten aufweisen. Die Entscheidung, welches der nächste Zug ist, soll jedoch durch den Benutzer getroffen werden. Auch hierbei sollen bestehende Pakete benutzt werden.

" 'Now, Mr. Holmes, you are in full possession of the facts. What course do you recommend?' "