Institut für Informatik — TU München
Lehrstuhl für Informatik mit Schwerpkt. Wiss. Rech.
Prof. Dr. H.-J. Bungartz, Dipl.-Inf. M. Moltenbrey,
Dipl.-Inf. D. Pflüger

WS 2008/09

PSE Verkehrssimulation
Aufgabenblatt 4: Kreuzungen

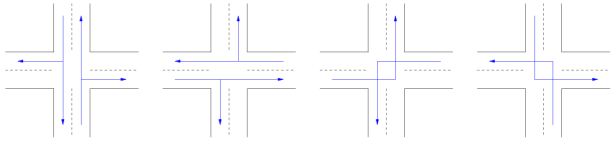
Ausgegeben 11.12.2008, Abgabe 07.01.2009 12:00 Uhr

Erweiterung der Simulationslogik

Bisher sind Sie in der Lage Verkehrssimulationen in vereinfachten Verkehrsnetzen durchzuführen. Momentan findet allerdings noch keine differenzierte Behandlung von Kreuzungen statt. Dies ist aber für eine „realistischere“ Simulation unerlässlich. Ziel dieses Aufgabenblattes ist es nun ein einfaches Verhalten an Kreuzungen zu entwerfen und zu implementieren.

Aufgabe 6 „Das 4-Phasen-Modell“

Für eine einfache Behandlung des Verhaltens an Kreuzungen wurde das so genannte *4-Phasen-Modell* eingeführt. Dieses beschreibt die Fahrtregeln an einer Kreuzung mit vier einmündenden Straßen. Dazu wird das Verhalten an Kreuzungen in vier Phasen unterteilt, die jeweils $p_1, \cdots, p_4$ Zeitschritte der Simulation umfassen.



Normalerweise gilt $p_1 = p_2 = p_3 = p_4$. Eine Variation ist allerdings möglich. Die vier Phasen sind in obigem Bild dargestellt. Die Bewegungen der einzelnen Phasen sind dabei so gewählt, dass es kaum zu Kollisionen kommen kann und falls doch werden diese vernachlässigt. Im Rahmen dieses PSE werden lediglich Kreuzungen mit höchstens vier Einmündungen betrachtet.

Setzen Sie dieses 4-Phasen-Modell in Ihrem Simulator um und überprüfen Sie das korrekte Verhalten mit geeigneten Testszenarien. Achten Sie dabei auch auf die Abbiegebeziehungen, die an den Kreuzungen gelten.

Aufgabe 7 „Verallgemeinerung des 4-Phasen-Modells“

Das Modell aus Aufgabe 5 ist lediglich auf Kreuzungen mit vier einmündenden Straßen ausgelegt. In unseren Verkehrsnetzen können allerdings auch Kreuzungen mit drei Straßen auftreten. Daher ist für diese Fälle eine gesonderte Behandlung notwendig. Entwerfen und implementieren Sie daher ein Modell analog zu dem aus Aufgabe 5, welches eine Behandlung solcher Kreuzungstypen erlaubt. Überprüfen Sie auch hier wieder die Korrektheit Ihrer Implementierung anhand geeigneter Testfälle.

Anhand dieser Aufgabe wollen wir exemplarisch das Vorgehen bei einer Modellvalidierung aufzeigen. Dabei geht es darum zu zeigen, wie realistisch und praxistauglich ein Modell für eine konkrete Problemstellung ist. Zuerst muß dieses entworfen werden und anschließend mit geeigneten Testfällen dessen Korrektheit untersucht werden. Die Ergebnisse der Untersuchung liefern Informationen inwieweit das ursprüngliche Modell ggf. verändert werden muß, um die gewünschte Funktionalität auzuweisen.

Dokumentieren Sie hierzu Ihr Modell, die Testfälle und deren Auswertung schriftlich. Welche Rückschlüsse lassen sich aus den Ergebnissen für Ihr Modell ziehen? Geben Sie die „Modelldokumentation“ bitte zusätzlich zu Ihrem Programmcode ab (gerne auch handschriftlich) oder präsentieren Sie diese bei der nächsten Abnahme.

Aufgabe 8 „Ampeln“

Als nächstes wollen wir die Behandlung von Ampeln in den Simulator integrieren. Dafür bedarf es einer weiteren, neuen Datenstruktur, die eben jene Ampeln beschreibt. Wir begnügen uns mit einem vereinfachten Ampel-Modell, in dem bspw. die Gelbphasen ignoriert werden. Die Ampeln sind über ihre Schaltfrequenz (rot ⇒ grün bzw. grün ⇒ rot) und ihre anfängliche Phase (*rot* oder *grün*) bestimmt. Die Ampeln müssen während der Simulation zum jeweiligen, richtigen Simulationszeitpunkt geschaltet werden. Das folgende Code-Fragment zeigt die XML-Beschreibung einer Ampel.

```
<TrafficLight>
   <nodeNr> </nodeNr>
   <fromNodeNr> </fromNodeNr>
   <initialPhase> </initialPhase>
   <cycle> </cycle>
   <initTime> </initTime>
</TrafficLight>
```

Hierbei gibt der Parameter `nodeNr` denjenigen Knoten (bzw. die Kreuzung) an, an der diese Ampel steht. Der folgende Parameter `fromNodeNr` gibt die Verbindungen vor, für die diese Ampel verantwortlich ist. Abb. 1 zeigt eine einfache Ampelsituation an einer Kreuzung mit vier einmündenden Straßen. Es handelt sich dabei um eine

vereinfachte Darstellung, ohne Berücksichtigung von mehrspurigem Verkehr. Abb. 2 zeigt einen Ausschnitt der XML-Beschreibung.

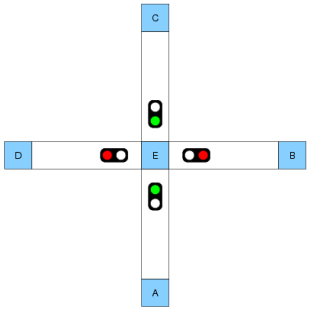


Abbildung 1: Einfache Ampelsteuerung bei einspurigem Verkehr

```
<TrafficLight>
   <nodeNr> E </nodeNr>
   <fromNodeNr> A </fromNodeNr>
   <initialPhase> green </initialPhase>
   <cycle> 5 </cycle>
   <initTime> 10 </initTime>
</TrafficLight>
...
<TrafficLight>
   <nodeNr> E </nodeNr>
   <fromNodeNr> B </fromNodeNr>
   <initialPhase> red </initialPhase>
   <cycle> 5 </cycle>
   <initTime> 10 </initTime>
</TrafficLight>
```

Abbildung 2: Auszug aus der XML-Beschreibung

Die beiden Parameter `initialPhase` und `cycle` dienen der eigentlichen Ampelsteuerung. Mit `initialPhase` wird die Startbeleuchtung der Ampel angegeben. `cycle` wiederum gibt die Anzahl an Simulationsschritten an bis die Ampel schaltet, d.h. ihr Signal ändert. Der letzte Parameter – `initTime` – gibt denjenigen Simulationszeitpunkt an, an dem die Ampel zum ersten Mal anfängt zu schalten und wird benötigt damit nicht alle Ampeln g;eichzeitig schalten.

Integrieren Sie das Ampelmodell in Ihren Simulator und testen Sie es anhand geeigneter Szenarien. Geben Sie eine (kurze) Szenarienbeschreibung und eine Auswertung der Ergebnisse bitte schriftlich ab.

Aufgabe 9 „Rechts-vor-Links an Kreuzungen“

Es existieren natürlich eine Vielzahl von Modellen für die Beschreibung von Kreuzungsverhalten. Eine gängige Verhaltensweise an Kreuzungen ist das „Rechts-vor-Links“. Dies zu modellieren gestaltet sich allerdings als etwas schwieriger. Eine korrekte Umsetzung ermöglicht allerdings die Erhöhung des Realitätsgrades einer Simulation.

Entwerfen und implementieren Sie ein solches Modell für Ihren Simulator. Sie können dabei von Kreuzungen mit maximal vier Straßen ausgehen.

Viel Spaß!