

Objektorientierte mikroskopische Verkehrsflusssimulation

Von der Fakultät für Bauingenieurwesen
der Ruhr-Universität Bochum genehmigte

Dissertation

zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)

von
Kai Erlemann

Bochum, 2007

Vorwort

Die vorliegende Arbeit entstand in den Jahren 2002 bis 2007 während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Ingenieurinformatik im Bauwesen. Ich möchte mich herzlich bedanken bei allen, die zum Gelingen dieser Arbeit beigetragen haben. Allen voran gebührt mein Dank natürlich Professor Hartmann, der mich von Anfang an in meiner Arbeit stets unterstützt und gefördert hat, und der mir den Weg in die wunderbare Welt der Objektorientierung gewiesen hat. Auch Professor Brilon danke ich für Übernahme des Koreferats, und dafür, dass er die sehr erfolgreiche Kooperation auf dem Gebiet der Verkehrssimulation überhaupt erst ermöglicht hat.

Des Weiteren sollen natürlich auch die zahlreichen Kollegen nicht unerwähnt bleiben, die mir während der Entwicklung des Programmsystems und dem Verfassen dieser Arbeit geduldig zur Seite gestanden haben. Vielen Dank an mein verkehrstechnisches Gewissen, den unermüdlichen Jochen Harding, für die vielen Stunden, die er mit mir gemeinsam an den Fahrverhalten getüftelt hat. Stefan Seifarth danke ich für seine Diagrammkomponente und die Optimierungsroutinen, Monika Rotthaus für die erste Dokumentation des Programms. Rüdiger Schütz danke ich dafür, dass er mit seiner Zuflussregelung die Erweiterbarkeit des Programms bewiesen hat. Außerdem danke ich allen anderen Mitarbeitern des Lehrstuhls, besonders Kay Smarsly, Ingo Mittrup, Karlheinz Lehner und Matthias Baitsch für die zahllosen produktiven Diskussionen.

Zu guter Letzt möchte ich meiner Freundin Sulamith, meiner Familie und meinen Freunden danken, die stets hinter mir standen und mich auf meiner langen Reise durch die Untiefen der Verkehrssimulation begleitet haben.

Bochum, im September 2007

Kai Erlemann

Tag der Einreichung: 28.03.2007

Tag der mündlichen Prüfung: 19.07.2007

1. Gutachter: Prof. Dr.-Ing. D. Hartmann
Lehrstuhl für Ingenieurinformatik im Bauwesen
Ruhr-Universität Bochum

2. Gutachter: Prof. Dr.-Ing. W. Brilon
Lehrstuhl für Verkehrswesen
Ruhr-Universität Bochum

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.3 Aufbau der Arbeit	3
2 Verkehrstechnische Grundlagen	5
2.1 Einleitung	5
2.2 Simulation	6
2.2.1 Verkehrssimulation	6
2.3 Definitionen	7
2.3.1 Verkehr	7
2.3.2 Verkehrsinfrastruktur	8
2.3.3 Fahrzeuge	8
2.3.4 Dynamische Grundgrößen eines Fahrzeugs	8
2.3.5 Verkehrsstärke	9
2.3.6 Verkehrsdichte	9
2.3.7 Mittlere Geschwindigkeit	9
2.3.8 Kapazität und Auslastung	10
2.3.9 Mittlere Reisezeit	10
2.3.10 Fahrzeugabstand	11
2.4 Grundlagen der Verkehrssimulation	11
2.4.1 Detaillierungsgrad	11
2.4.2 Zeitverhalten der Simulation	13
2.5 Existierende Verkehrssoftware	14
2.5.1 Überblick	14

2.5.2	VISSIM	14
2.5.3	Paramics	15
2.5.4	AIMSUN	15
2.5.5	CORSIM	16
2.5.6	HUTSIM	17
2.5.7	PELOPS	17
2.5.8	OLSIM	18
2.6	Einschränkungen existierender Lösungen	19
2.7	Verbesserter Lösungsansatz	20
3	Modellierung	21
3.1	Einleitung	21
3.2	Anforderungen	21
3.3	Simulationsablauf	22
3.4	Fahrer-Fahrzeug-Elemente	22
3.4.1	Fahrer	24
3.4.2	Fahrverhalten	25
3.4.3	Fahrzeuge	26
3.4.4	Fahrzeugtypen	27
3.4.5	Pkw	29
3.4.6	Lkw	30
3.5	Streckennetz	31
3.5.1	Fahrspuren	33
3.5.2	Quellen	34
3.5.3	Vorlauf	37
3.5.4	Senken	38
3.5.5	Mehrspurbereiche	39
3.6	Strecken	40
3.7	Hindernisse	41
3.8	Simulationsauswertungen	44
3.8.1	Lokale Messungen	44
3.8.2	Momentane Messungen	46
3.8.3	Reisezeiten	46

4 Verhaltensmodelle	49
4.1 Einleitung	49
4.2 Abstandsmodelle	50
4.2.1 Fahrzeugfolge-Theorie	50
4.2.2 Optimal-Velocity-Model	51
4.2.3 Psycho-physisches Modell	52
4.3 Spurwechselmodelle	55
4.3.1 Sparmann	56
4.3.2 Theis	57
4.4 Beschränkungen existierender Modelle	60
5 Absichtsbasiertes Fahrverhalten	63
5.1 Grundidee	63
5.2 Lösungsansatz	65
5.3 Mathematische Betrachtung	66
5.4 Implementierung	68
5.5 Diskussion	72
6 Implementierung	75
6.1 Einleitung	75
6.2 Programmstruktur	75
6.2.1 Programmbedienung	77
6.2.2 Zweidimensionale Darstellung des Streckennetzes	79
6.2.3 Dreidimensionale Darstellung des Streckennetzes	80
6.2.4 Animationen	83
6.3 Datenaustausch	85
6.3.1 Geographical Data Format (GDF)	85
6.3.2 Web Map Service (WMS)	88
6.3.3 Keyhole Markup Language (KML)	90
6.3.4 Scalable Vector Graphics (SVG)	92
6.3.5 Sonstige Formate	93
6.4 Parallelisierung der Verkehrssimulation	95
6.4.1 Parallelisierungsarten	95

6.4.2	Asynchrone Parallelisierung	96
6.4.3	Synchrone Parallelisierung	97
6.4.4	Message Passing Interface	99
6.4.5	Implementierung der verteilten Simulation	99
7	Erweiterungen des Objektmodells	105
7.1	Kreisverkehr	105
7.1.1	Verhalten am Kreisverkehr	105
7.1.2	Approximation von Kreisbögen	106
7.1.3	Modellierung eines Kreisverkehrs	107
7.2	Parkplätze	109
7.2.1	Modellierung von Stellplätzen	109
7.2.2	Auswahl eines Stellplatzes	109
7.2.3	Nachbildung des Einparkvorgangs	111
7.3	Lichtsignalanlagen	113
7.4	Fußgänger	115
7.4.1	Fußgängermodell	116
7.4.2	Kopplung mit dem Straßenverkehr	119
7.5	ÖPNV	121
7.5.1	Linienbusse	121
7.5.2	Straßenbahnen	123
8	Simulation großflächiger Straßennetze	125
8.1	Einordnung des Anwendungsspektrums	125
8.2	Netzerstellung	126
8.2.1	Hauptfahrspuren	128
8.2.2	Zu- und Abfahrten	129
8.2.3	Verbinden der Teilnetze	130
8.3	Festlegung der Verkehrsstärken	130
8.3.1	Streckenerstellung	131
8.3.2	Streckenelimination	132
8.3.3	Setzen der Verkehrsstärken	132
8.4	Einzelplatz-Simulation	134

8.4.1	Hardwareanforderungen	134
8.4.2	Simulationablauf	134
8.4.3	Zeitverhalten	135
8.5	Parallele Simulation	136
9	Zusammenfassung und Ausblick	139
9.1	Zusammenfassung	139
9.2	Ausblick	141
A	Programmgesteuerte Erzeugung eines Streckennetzes	143
A.1	Erstellung einer Einfahrt	143
A.2	Erzeugung der Fahrspuren	144
A.3	Quellen und Senken	146
A.4	Verkehrsstärken	147
	Literaturverzeichnis	149

Kapitel 1

Einleitung

1.1 Motivation

Mobilität ist ein wichtiger wirtschaftlicher Kernfaktor der Infrastruktur einer Industrienation, und wird noch immer zum größten Teil durch den Verkehrsträger Straße bereitgestellt. Die ständig steigenden Verkehrsichten auf den Autobahnen führen dazu, dass selbst geringfügige lokale Störungen starke Kapazitätseinbrüche des Gesamtstraßennetzes hervorrufen können. Die wirtschaftlichen Folgen durch Staus, Lärmentwicklung und Schadstoffemission sind immens.

Seit Beginn der siebziger Jahre ist der Gesamt-Kraftfahrzeugbestand in der Bundesrepublik um mehr als das Doppelte auf heute 54,1 Millionen Fahrzeuge angewachsen [21]. Das durchschnittliche tägliche Verkehrsaufkommen hat sich in dieser Zeit auf 51.600 Fahrzeuge pro Tag erhöht, und damit mehr als verdoppelt. Ein Ende dieses Trends ist trotz Einführung von Autobahn-Maut-Gebühren und steigenden Energiekosten nicht zu erwarten.

Laut einer Projektstudie der Technikakademie acatech ([2]) wird das Gesamtverkehrsaufkommen bis zum Jahr 2020 um ca. 21 Prozent zunehmen, die Fahrleistung im Straßen-Güterverkehr wird gegenüber dem Jahr 2002 sogar um etwa 34 Prozent steigen. Für Bundesautobahnen kann von einer Verkehrszunahme von 30 Prozent für Pkw bzw. 45 Prozent für Lkw ausgegangen werden. Auf einzelnen Autobahnen wird, in Folge des innereuropäischen Strukturwandels, sogar ein Anwachsen des Lkw-Transitverkehrs um bis 180 Prozent erwartet. Der Bundesverkehrswegeplan 2003 sieht vor, das Bundesautobahnnetz um insgesamt 1900 km zu erweitern, und bestehende 2200 km des Netzes weiter auszubauen.

Zur kosteneffizienten Planung von Neu- und Ausbaustrecken ist es deshalb unerlässlich, die signifikanten Faktoren, die für einen reibungslosen Verkehrsablauf, aber auch für einen Zusammenbruch des Verkehrsstroms verantwortlich sind, genau zu kennen. Insbesondere bei der Bemessung von Verkehrs-Knotenpunkten mit geringen räumlichen Abständen kommt es zu komplexem Systemverhalten, das mit herkömmlichen analytischen Bemessungsverfahren nicht abgebildet werden kann. In solchen Fällen kann der Einsatz einer computergestützten stochastischen Simulation des Verkehrsflusses ein wertvolles Werkzeug sein.

Derartige Computersimulationen versetzen den Verkehrsingenieur in die Lage, verschiedene Planungsentwürfe einander gegenüberzustellen und die Auswirkungen von Planungsänderungen direkt untersuchen zu können. Die Leistungsfähigkeit von bestehenden Straßennetzen kann für zukünftig erwartete Verkehrsstärken abgeschätzt und beurteilt werden. Auch der direkte

Eingriff in den Verkehrsfluss, z. B. durch Geschwindigkeitsbegrenzungen, Überholverbote oder Verkehrsregelungsanlagen, kann durch die mikroskopische Verkehrssimulation nachgebildet werden.

Im Verkehrswesen werden seit vielen Jahrzehnten computergestützte Simulationen eingesetzt, bereits Ende der 1950er Jahre wurden erste Modelle entwickelt. Mit der steigenden Leistungsfähigkeit der Computer stieg auch der Anteil der mikroskopischen Simulationsmodelle, die eine detaillierte Abbildung einzelner Fahrzeuge und ihrer Verhaltensmuster ermöglichen. Die realitätsgetreue Umsetzung des Fahrer-Verhaltens in eine Simulation erfordert dabei leistungsfähige Modelle. So müssen sowohl mikroskopische Aspekte einzelner Fahrzeuge (zum Beispiel korrekte Sicherheitsabstände, Beschleunigungen, Reaktionszeiten einzelner Fahrzeuge) als auch makroskopische Gesichtspunkte (Durchschnittsgeschwindigkeiten, Fahrzeugdichten, Spurverteilungen von Fahrzeugströmen) detailliert abgebildet werden. Zahlreiche mikroskopische Verhaltensmodelle sind in den letzten Jahrzehnten entwickelt worden, manche davon eher minimalistisch, andere hochkomplex und rechenintensiv.

Die Entwicklung oder Erweiterung eines mikroskopischen Verhaltensmodells ist alles andere als trivial. Zahlreiche situationsabhängige Sonderfälle müssen erfasst werden, um in der Simulation auch auf außergewöhnliche Verkehrssituationen angemessen zu reagieren. Andererseits erhöhen zusätzliche Fallunterscheidungen die Komplexität der Verhaltensmodelle, vermindern die Lesbarkeit des Quelltextes und erschweren die Kalibrierung und Fehlersuche. Daher ist ein Ansatz erforderlich, der die Entwicklung komplexer, erweiterbarer Verhaltenssysteme erlaubt, gleichzeitig aber ein Mindestmaß an Übersichtlichkeit und Modularität gewährleistet.

1.2 Zielsetzung

Diese Arbeit beschreibt die Entwicklung eines neuen eigenständigen Programm Pakets, das die mikroskopische Simulation des Verkehrs auf Bundesautobahn-Streckenabschnitten erlaubt.

Grundgerüst der entwickelten Verkehrssimulation ist ein streng objektorientiertes Klassenmodell, das die realitätsnahe Nachbildung der Verkehrsgegebenheiten durch Objektentsprechungen erlaubt. Durch konsequente Anwendung von Vererbungsmechanismen und Assoziationsgeflechten orientiert sich das Objektmodell stark an der realen Welt und erleichtert so die Einarbeitung in die Nutzung und die Erweiterung des Simulationspaketes. Die generelle Herangehensweise bei der Entwicklung einer Verkehrssimulations-Software soll ebenso dargestellt werden wie die detaillierte Modellierung einzelner spezieller Anwendungsfälle.

Im Rahmen dieser Arbeit werden bestehende Verhaltensmodelle verschiedener Komplexität untersucht, ihre gemeinsamen Schnittstellen identifiziert und an das gemeinsame Objektmodell angepasst. Aufbauend auf einer Analyse der Vor- und Nachteile bestehender Modelle wird ein neues Fahrerhalten entwickelt, das die menschlichen Entscheidungsfindung in Form von modularen Absichten nachbildet. Das absichtsbasierte Verhaltensmodell zeichnet sich durch eine leichte Erweiterbarkeit und erhöhte Robustheit der Simulation aus; die Modellierung und Implementierung dieses Verhaltens wird dargestellt.

Neben dem Verkehr auf Bundesautobahnen kann das entwickelte Klassenmodell auch für die Abbildung innerstädtischen Verkehrs genutzt werden. Das objektorientierte Modell kann beliebig erweitert werden, um z.B. weitere Verkehrsteilnehmer wie Fußgänger oder ÖPNV-Fahrzeuge in die Simulationswelt zu integrieren. Die konkrete Vorgehensweise zur einfachen Integration

neuer Simulationselemente wird anhand von Beispielen erläutert.

Die großen Datenmengen, die bei der mikroskopischen Simulation verarbeitet werden, übersteigen oft das Rechenvermögen einzelner Prozessoren. Es muss daher untersucht werden, wie die Rechenlast bei Bedarf aufgeteilt und in parallelen Berechnungsverfahren auf verteilten Rechnersystemen abgearbeitet werden kann. Die Probleme und Möglichkeiten der Simulations-Parallelisierung werden diskutiert.

1.3 Aufbau der Arbeit

Der Aufbau der Arbeit orientiert sich an der Vorgehensweise bei der Erstellung und der Nutzung der Simulationsumgebung. Die Arbeit ist unterteilt in neun Kapitel, in denen der gesamte Entstehungsprozess einer mikroskopischen Verkehrssimulation geschlossen aufgezeigt wird.

Im zweiten Kapitel werden zunächst die verkehrstechnischen Grundlagen behandelt, die für die Simulation relevant sind. Der Begriff der Verkehrssimulation wird definiert, verschiedene existierende Simulationspakete sowie deren Stärken und Einschränkungen untersucht.

Kapitel drei beinhaltet eine Beschreibung des verwendeten Klassenmodells, mit dem die für die Simulation relevanten Objekte abgebildet werden. Der Aufbau des Modells wird anhand von UML-Diagrammen festgelegt, die auch die Interaktionen der beteiligten Objekte beschreiben.

Die Beschreibung der implementierten Fahrverhalten und Spurwechselverhalten sowie ihrer gemeinsamen generischen Schnittstelle zur Einbindung in die Simulation erfolgt im vierten Kapitel. Hier werden auch die Besonderheiten bei der Umsetzung der Modelle identifiziert.

Im fünften Kapitel wird – aufbauend auf der Analyse der existierenden Verhaltensmodelle – ein neuer Ansatz vorgestellt, der die Fahrzeugsteuerung einem kooperativen Abstimmungsprozess verschiedener gleichberechtigter Absichtsmodule unterwirft. Das mathematische Verfahren und die Vorteile bei der Kalibrierung und Wartung des Fahrverhaltens werden beschrieben.

Das sechste Kapitel beschäftigt sich mit der Umsetzung der zuvor entwickelten Modelle in eine konkrete Java-Applikation. Dabei werden die verwendeten Programmiermethoden, insbesondere die Java-spezifischen Ansätze, dargelegt. Auf spezielle Implementierungsbereiche, wie die grafische Darstellung, die Benutzerführung und den Datenaustausch mit anderen Programmen, wird detailliert eingegangen. Auch die verteilte Berechnung der Simulation und die Implementierung einer Parallelisierungsstrategie wird beschrieben.

Kapitel sieben untersucht die Erweiterbarkeit des für Autobahnen konzipierten Klassenmodells anhand einiger konkreter Beispiele aus dem innerstädtischen Straßenverkehr, behandelt werden z.B. Kreisverkehre, Lichtsignalanlagen, Fußgänger- oder ÖPNV-Verkehr.

Die Leistungsfähigkeit des erstellten Programmsystems wird in Kapitel acht an einem konkreten Praxisbeispiel eines räumlich Autobahnnetzes überprüft. Dabei werden sowohl die Modellierung, die Parametereinrichtung, als auch die sequentielle und parallele Simulation beschrieben und auf ihre Praxistauglichkeit untersucht.

Im abschließenden neunten Kapitel wird eine kurze Zusammenfassung der Ergebnisse sowie ein Ausblick auf mögliche zukünftige Einsatzgebiete und Erweiterungen des Simulationspaketes gegeben.

Kapitel 2

Verkehrstechnische Grundlagen

2.1 Einleitung

Um eine Nachbildung des realen Verkehrsgeschehens im Computer zu erreichen, ist es zunächst erforderlich, die grundlegenden Vorgänge im Straßenverkehr zu betrachten. Das Lenken eines Fahrzeuges ist ein komplexer Vorgang, der von zahlreichen äußeren Einflüssen, Erfahrungswerten des jeweiligen Fahrers und vorgegebenen Gesetzmäßigkeiten beeinflusst wird. Während sich die meisten Autofahrern, besonders wenn sie routinemäßig weite Strecken mit dem Fahrzeug bewältigen, dieser Komplexität nicht bewusst sind, kann sie am Beispiel der Lernphase eines Führerscheinanwärters sehr einfach verdeutlicht werden: Jeder Fahrzeugführer muss sich vor dem Erhalt der Fahrerlaubnis zunächst einer mehr oder weniger langen theoretischen und praktischen Ausbildungsphase unterwerfen. Das Abschätzen von Geschwindigkeiten, Brems- und Beschleunigungswegen, geometrischen Abmessungen des Fahrzeugs, Interaktionen mit anderen Fahrzeugen, Beachtung von Verkehrszeichen, Fahrbahnmarkierungen, Lichtsignalen und Vorfahrtsregeln – all diese Informationen müssen erlernt und stetig trainiert werden. Auch nach Erwerb der Fahrerlaubnis muss noch ein gewisse Fahrerfahrung gesammelt werden, bevor eine zufriedenstellende Sicherheit im Steuern eines Fahrzeugs erreicht wird.

Zusätzlich zu diesem aufwändigen Lernprozess sollte nicht außer Acht gelassen werden, dass ein volljähriger Fahranfänger bereits zahlreiche physiologische Grundvoraussetzungen mitbringt, die er sich im Laufe seines Heranwachsens unbewusst angeeignet hat. Dazu zählen unter anderem ein räumliches Vorstellungsvermögen, die Schätzung von Distanzen und Relativgeschwindigkeiten, taktile und sensorische Fähigkeiten zur Beurteilung von Umgebungseinflüssen, eine Wissensbasis über das zu erwartende Verhalten anderer Fahrzeuge, eine Filterung und Auswertung der wahrgenommenen Informationen sowie nicht zuletzt die kreative Fähigkeit zur spontanen Reaktion auf unvorhergesehene Situationen. Das Zusammenspiel aus all diesen Fähigkeiten und Erfahrungswerten stellt sicher, dass ein menschlicher Fahrer mit relativ hoher Zuverlässigkeit das Ziel seiner Fahrt planmäßig und ohne Unfälle zu verursachen erreichen kann.

Soll das menschliche Verhalten des Fahrers im Computer nachgebildet werden, müssen die Gegebenheiten der realen Welt identifiziert und in einer vom Rechner verständlichen Form strukturiert beschrieben werden. Es ist offenkundig, dass die vollständige Nachbildung der oben skizzierten Einflussfaktoren die Fähigkeiten der modernen Rechnergeneration (und der Programmierparadigmen) bei Weitem übersteigen dürfte. Jede Simulation eines realen Phänomens

erfordert einen zusätzlichen Aufwand bei der Programmierung und insbesondere Kalibrierung des Systems. Um einem Computer die Fähigkeit zum Lenken eines (wenn auch virtuellen) Fahrzeuges zu vermitteln, muss daher das reale Verkehrsgeschehen adäquat abstrahiert und auf die verkehrstechnisch relevanten Aspekte kondensiert werden. Dies ist die Aufgabe der Verkehrssimulation.

2.2 Simulation

Eine Simulation ist grundsätzlich die Nachbildung eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell [116]. Die Hoffnung dabei ist, die Erkenntnisse, die bei der Untersuchung dieses Modells gefunden werden, möglichst genau auf das reale System übertragen zu können. Im Allgemeinen wird durch Simulation das Verhalten von Systemen untersucht, die sich z. B. aufgrund ihrer Größe oder Komplexität, ihres nicht-deterministischen Verhaltens, der schwierigen Beobachtung oder des hohen damit verbundenen Aufwandes nicht oder nur mühsam in der Realität beobachten lassen. Unerlässlich werden Simulationen spätestens dann, wenn das zu untersuchende Ereignis in der Realität mit großen Gefahren für Menschen oder Umwelt verbunden ist, oder aber sehr selten eintretende Situationen betrachtet werden sollen. Auch für die Untersuchung von noch nicht existenten Systemen, die sich beispielsweise noch in der Entwicklung befinden (Prototyping), bietet sich eine vorherige Simulation an, um unangenehme Überraschungen nach der Fertigstellung eines Projektes zu vermeiden.

Häufige Einsatzgebiete der Simulation aus dem Bereich des Ingenieurwesens sind zum Beispiel das dynamische Verhalten von Bauwerken, aerodynamische Strömungsuntersuchungen an Bauwerksmodellen, das Verhalten großer Menschenmassen in Paniksituations, oder das Testen neuer Fahrzeugkarosserien in der Automobilindustrie. Während ein Teil dieser Untersuchungen auch allein durch Experimente an realen Modellen durchgeführt werden könnte, haben sich in den letzten Jahrzehnten die computergestützten Simulationsverfahren mehr und mehr durchgesetzt, und in vielen Bereichen die realen Modelle weitgehend verdrängt. Einer der Hauptgründe hierfür ist sicherlich der finanzielle Aspekt: die Durchführung einer numerischen Simulation am Rechner erfordert, im Gegensatz zu einem realen Experiment, neben Personalkosten nur relativ geringe finanzielle Aufwendungen für Hard- und Softwareausstattung und Betrieb der eingesetzten Rechnersysteme. Die stetige Weiterentwicklung moderner Rechnergenerationen ermöglicht es, heute Simulationen auf Standard-PCs durchzuführen, für die noch vor wenigen Jahren Großrechner erforderlich gewesen wären. Hinzu kommt, dass eine Computersimulation, einmal eingerichtet, praktisch beliebig oft in verschiedenen Varianten durchgeführt werden kann, während reale Experimente meist für jede Durchführung eine vergleichsweise lange Vor- und Nachbereitungszeit benötigen und folglich sehr viel kostenintensiver sind.

2.2.1 Verkehrssimulation

Unter einer Verkehrssimulation wird im Folgenden ein computergestütztes Softwaresystem zur Nachbildung des Verhaltens von Verkehrsteilnehmern auf einem Streckennetz verstanden. Die Art der abgebildeten Verkehrsteilnehmer ist dabei von der betrachteten Problemdomäne abhängig; prinzipiell sind Simulationen beliebiger Verkehrsträger möglich. So existieren für die wichtigsten Verkehrsträger Luftverkehr, Schifffahrt, Eisenbahn, Straßenverkehr oder Fußgänger eigene Simulationssysteme, die jeweils eigenen Randbedingungen und Verhaltensmustern un-

terliegen. Es leuchtet z. B. ein, dass ein Programm zur Simulation von Massenevakuierung aus Stadien anderen Anforderungen an das Softwaremodell unterliegt als ein Werkzeug zur Simulation von Flugzeugbewegungen. Jedes Verkehrssystem erfordert folglich ein eigenes Regelsystem zur Steuerung der einzelnen Verkehrsteilnehmer, dem beim Software-Entwurf Rechnung getragen werden muss.

Neben der Simulation des Fahrzeugverkehrs kann es sinnvoll sein, auch Fußgänger als Teile der Simulationswelt zu integrieren, da zwischen beiden Verkehrsträgern Wechselwirkungen bestehen (z. B. in Form von Fußgängerampeln oder Fußgängerüberwegen). Üblich sind auch Systeme, mit denen der Öffentliche Personennahverkehr (ÖPNV) einer Region mit seinen unterschiedlichen Fahrzeugtypen (Busse, U-Bahnen, S-Bahnen, Fernzüge) abgebildet werden kann. Die verschiedenen Verhaltensformen dieser Fahrzeugtypen müssen bereits bei der Konzeption eines multimodalen Simulationssystems beachtet werden.

Endziel einer Verkehrssimulation muss es sein, den Verkehrsablauf in seiner Gesamtheit möglichst realitätsnah abzubilden. Dazu muss sichergestellt sein, dass die dem Verkehr zu Grunde liegenden Gesetzmäßigkeiten so genau erfasst werden, dass die von der Simulation erzielten Ergebnisse sowohl für das Verhalten der einzelnen Fahrzeuge, als auch auf der Ebene des Gesamtverkehrs, in möglichst gut mit der Realität übereinstimmen.

Das Gesamtsystem einer Verkehrssimulation kann grob in zwei große Bereiche unterteilt werden, die beide mit ausreichender Genauigkeit im Computer nachgebildet werden müssen: die Verkehrsinfrastruktur und der eigentliche Verkehr; beide Begriffe werden im Folgenden definiert.

2.3 Definitionen

Um ein einheitliches Verständnis sicherzustellen, werden nachfolgend alle Begrifflichkeiten, die für eine Nachbildung des Straßenverkehrs im Computer erforderlich sind, definiert.

2.3.1 Verkehr

Der Begriff „Verkehr“ beschreibt die Gesamtheit der Bewegungen von Personen, Waren oder Informationen in einem zeitlich und räumlich definierten Verkehrsgebiet. Verkehrsbewegungen können dabei unter Verwendung von Verkehrsmitteln erfolgen, und unterliegen zumeist klar definierten Bewegungs- und Verhaltensvorschriften. Verkehr ist in der Regel zweckgebunden und verfolgt die zielgerichtete Bewegung zu einem bestimmten räumlichen Gebiet.

Voraussetzung für den Verkehr ist eine entsprechende Verkehrsinfrastruktur, die die Bewegung der Verkehrsteilnehmer unterstützt und ermöglicht. Die Art und Ausprägung der Verkehrsinfrastruktur ist dabei in starkem Maße abhängig von der Fortbewegungsmethode und den eingesetzten Verkehrsmitteln. So werden für Straßen-, Schienen- und Schifffahrtsverkehr grundlegend verschiedene Infrastrukturen und Verkehrsmittel benötigt. Gruppen von einheitlichen Verkehrsmitteln und -infrastrukturen werden unter dem Begriff Verkehrsträger zusammengefasst. Nehmen verschiedene Verkehrsträger an der Ortsveränderung von Gütern oder Personen teil, spricht man von multimodalem Verkehr. Die Aufteilung des Transportaufkommens auf die verschiedenen Verkehrsträger wird als „Modal Split“ bezeichnet.

2.3.2 Verkehrsinfrastruktur

Die Verkehrsinfrastruktur ermöglicht oder vereinfacht den am Verkehr teilnehmenden Fahrzeugen oder Verkehrsteilnehmern die Bewegung, also das Herbeiführen der gewünschten Ortsveränderung. Wichtigster Teil der Infrastruktur ist dabei zumeist ein Netz physischer Fahrwege, auf denen sich die Fahrzeuge möglichst sicher und reibungslos bewegen können, also die eigentliche Fahrbahn bzw. das Transportmedium. Zu den Verkehrsnetzen zählen sowohl Straßen und Gehwege, das Schienennetz, die Fluss- und Kanalnetze der Binnenschiffahrt sowie die internationales Seeschiffahrtsrouten und Luftfahrtkorridore. Ein gut ausgebautes und in gutem Zustand erhaltenes Streckennetz ist eine Grundvoraussetzung um eine hohe Mobilität zu gewährleisten und den Grunddurchsatz des Netzes zu maximieren.

Zur Infrastruktur zählen neben dem reinen Streckennetz auch weitere Elemente, die die Qualität des Verkehrsablaufes gewährleisten und den Verkehrsteilnehmer beim Erreichen ihres Ziels unterstützen. Insbesondere die Straßenausstattung (Beschilderungen, Markierungen, Lichtsignalanlagen, Verkehrsleitsysteme) ist als Teil der Verkehrsinfrastruktur zu betrachten.

2.3.3 Fahrzeuge

Fahrzeuge ermöglichen und vereinfachen den für den Transport von Gütern und Personen angestrebten Ortswechsel. Grundsätzlich erfolgt die Steuerung eines Fahrzeugs stets durch einen menschlichen Fahrer; lediglich einige wenige Ausnahmen existieren in Form von computergesteuerten Systemen, wie sie z. B. zur Passagierbeförderung bei Flughäfen eingesetzt werden. Das Fahrzeug bewegt sich entlang des ihm zugedachten Fahrweges, der durch die Verkehrsinfrastruktur bereit gestellt wird. Analog zur Klassifizierung der Infrastruktur lassen sich Fahrzeuge in die Gruppe Straßenfahrzeuge, Schienenfahrzeuge, Wasserfahrzeuge und Flugzeuge unterteilen.

Die Bewegung der Fahrzeuge wird durch die Fahreigenschaften des Fahrzeugs, durch die Steuerung des Fahrers, durch externe Beeinflussungsfaktoren und durch die Gesetze der Newtonschen Physik beeinflusst. Je nach Detailgrad der Simulation können diese Einflussgrößen unterschiedlich genau nachgebildet werden. Nachfolgend werden die für die Simulation relevanten Kenngrößen beschrieben.

2.3.4 Dynamische Grundgrößen eines Fahrzeugs

Die Bewegung eines Fahrzeugs wird durch die dynamischen Grundgleichungen beschrieben, wobei das Fahrzeug idealisiert als Punktmasse betrachtet wird. Für ein Fahrzeug, das sich vom Zeitpunkt t_i bis zum Zeitpunkt t_{i+1} mit einer Beschleunigung $a(t)$ bewegt, gelten für die Position x und die Geschwindigkeit v die folgenden Gesetzmäßigkeiten:

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \int_{t_i}^{t_{i+1}} \mathbf{v}(t) dt \quad (2.1)$$

$$\mathbf{v}(t_{i+1}) = \mathbf{v}(t_i) + \int_{t_i}^{t_{i+1}} \mathbf{a}(t) dt \quad (2.2)$$

Erfolgt die Bewegung des Fahrzeugs in diskreten Zeitschritten (vgl. Abschnitt 2.4.2), werden Beschleunigung und Geschwindigkeit näherungsweise für die Dauer eines Zeitschritts Δt als konstant behandelt. Daraus ergeben sich die beiden für die zeitschritt-basierte Simulation wesentlichen Bewegungsgleichungen:

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_i) + \Delta t \cdot \mathbf{v}(t_{i+1}) \quad (2.3)$$

$$\mathbf{v}(t_{i+1}) = \mathbf{v}(t_i) + \Delta t \cdot \mathbf{a}(t_{i+1}) \quad (2.4)$$

2.3.5 Verkehrsstärke

Die Verkehrsstärke entspricht der Anzahl derjenigen Fahrzeuge eines Verkehrsstroms, die pro Zeiteinheit einen lokal feststehenden Meßquerschnitt in eine Richtung passieren. Die Ermittlung der Verkehrsstärke erfolgt durch eine lokale Messung über eine Zeitdauer Δt , während der alle Fahrzeuge am Messpunkt x_0 registriert werden. Die Verkehrsstärke q ergibt sich aus der lokal registrierten Anzahl der Fahrzeuge n_l und der Dauer Δt :

$$q = \frac{n_l}{\Delta t} \quad [\text{Fz/h}] \quad (2.5)$$

Bei der Ermittlung der Verkehrsstärke ist die Dauer des Meßintervalls entscheidend. Wird die Dauer einer Messung zu klein gewählt (z. B. 10 Sekunden), treten sehr große Schwankungen zwischen einzelnen Messungen auf, und der Einfluß eines einzelnen Fahrzeugs auf das Meßergebnis ist sehr groß. Wird hingegen ein sehr großes Zeitintervall gewählt (z. B. 10 Stunden), können Spitzenwerte durch die zeitliche Mittelung verloren gehen. Üblich Werte für Meßintervalllängen in der Verkehrssimulation liegen zwischen fünf Minuten und einer Stunde.

2.3.6 Verkehrsdichte

Die Verkehrsdichte gibt an, wie viele Fahrzeuge eines Verkehrsstroms sich pro Wegeinheit zu einem Beobachtungszeitpunkt in einem bestimmten Beobachtungsgebiet befinden. Im Gegensatz zur Verkehrsstärke, bei der eine lokale Messung über eine Zeitintervall Δt erfolgt, wird die Bestimmung der Dichte zu einem vorgegebenen Zeitpunkt t_0 durchgeführt. Die Verkehrsdichte k ist das Verhältnis der momentanen Anzahl der Fahrzeuge im Meßbereich n_m zur Länge des Messbereiches Δx :

$$k = \frac{n_m}{\Delta x} \quad [\text{Fz/km}] \quad (2.6)$$

2.3.7 Mittlere Geschwindigkeit

Werden zusätzlich zur Anzahl der Fahrzeuge auch die Geschwindigkeiten der Fahrzeuge v_i bei jeder Fahrzeugerfassung i registriert, lassen sich daraus die Durchschnittsgeschwindigkeiten bestimmen. Je nach Art der Messung (lokale oder momentane Beobachtung, Abbildung 2.1) wird zwischen mittlerer lokaler Geschwindigkeit \bar{v}_l und mittlerer momentaner Geschwindigkeit

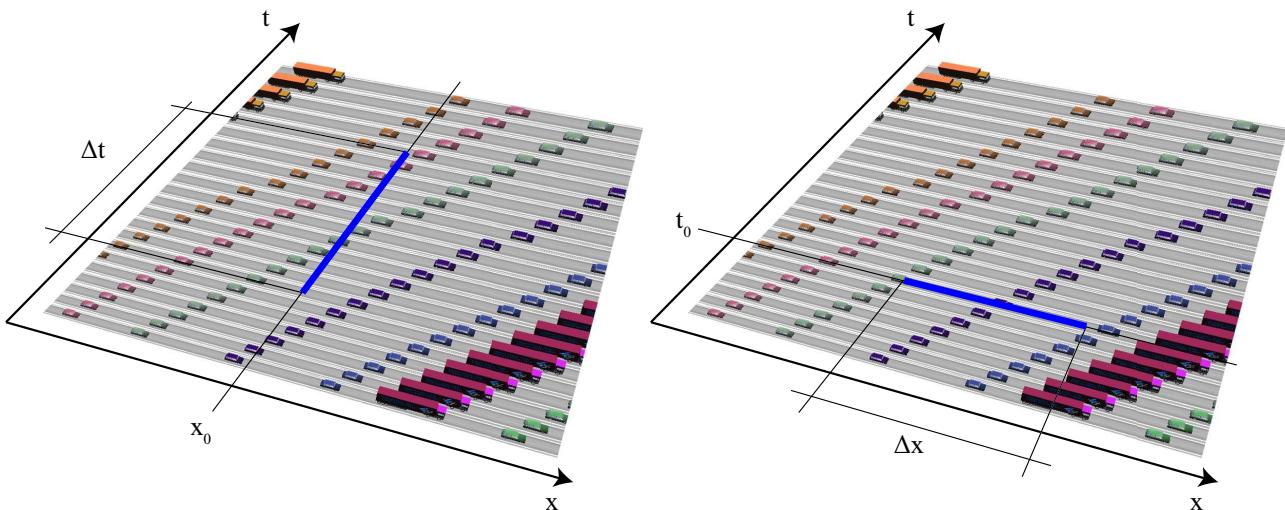


Bild 2.1: Vergleich von lokaler (links) und momentaner Messung (rechts)

\bar{v}_m unterschieden.

$$\bar{v}_l = \frac{1}{n} \sum_{i=1}^n v_{li} = \frac{1}{n} \sum_{i=1}^n \frac{\Delta x}{\Delta t_i} \quad (2.7)$$

$$\bar{v}_m = \frac{1}{n} \sum_{i=1}^n v_{mi} = \frac{1}{n} \sum_{i=1}^n \frac{\Delta x_i}{\Delta t} \quad (2.8)$$

2.3.8 Kapazität und Auslastung

Die Kapazität C bezeichnet die größtmögliche Verkehrsstärke, die ein Verkehrsstrom an einem bestimmten Querschnitt erreichen kann.

$$C = q_{max} \quad [\text{Fz/h}] \quad (2.9)$$

Da der Verkehr nur in den Spitzentunden die maximale Auslastung einer Strecke erreicht, ist es sinnvoll, einen Auslastungsgrad a zu definieren, der den momentanen Anteil der Verkehrsstärke an der erreichbaren Kapazität beschreibt.

$$a = \frac{q}{C} \quad [-] \quad (2.10)$$

2.3.9 Mittlere Reisezeit

Die Reisezeit ist die Gesamtzeit T_i , die ein Fahrzeug für das Zurücklegen einer Strecke L_i benötigt. Um die Reisezeiten verschiedener Fahrzeuge vergleichen zu können, werden sie auf eine durchschnittliche Reisezeit pro Streckeneinheit normiert. Die mittlere Reisezeit pro Streckenabschnitt T_R ergibt sich aus der Mittelung aller normierten Einzelzeiten.

$$T_R = \frac{1}{n} \sum_{i=0}^n \frac{T_i}{L_i} \quad [\text{min}/100 \text{ km}] \quad (2.11)$$

2.3.10 Fahrzeugabstand

Bei der Angabe eines Abstands zwischen zwei Fahrzeugen werden grundsätzlich, je nach Anwendungsfall, zwei Art von Abstand unterschieden: Ist die kürzeste räumliche Distanz zwischen der Geometrie eines Fahrzeugs und dem betrachteten Hindernis gemeint, spricht man vom Nettoabstand dx_{netto} . Der Bruttoabstand dx_{brutto} hingegen kennzeichnet die Distanz zwischen zwei Vergleichspunkten beider Fahrzeuge, im Allgemeinen der vorderen Kante (vgl. Abbildung 2.2).

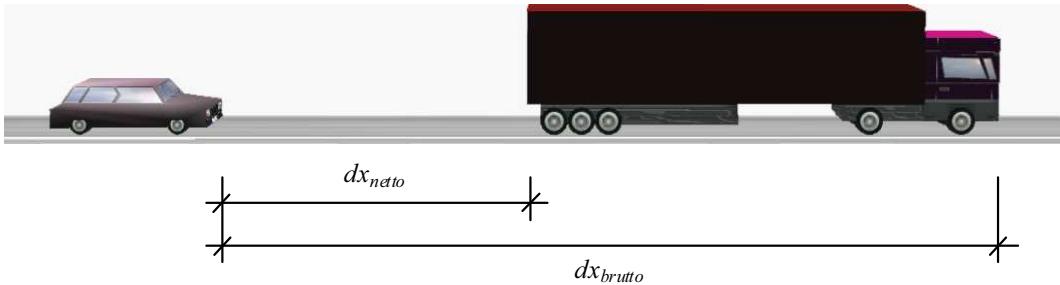


Bild 2.2: Brutto- und Nettoabstand zum Vorderfahrzeug aus Sicht eines PKW

Der Bruttoabstand wird bevorzugt für makroskopische Betrachtungen des Verkehrsstroms herangezogen, da der mittlere Bruttoabstand dem Kehrwert der Verkehrsdichte entspricht. In den Fahrzeugfolgemodellen (vgl. Kapitel 4) wird, soweit nicht anders angegeben, bevorzugt der Nettoabstand verwendet, da dieser die tatsächliche freie Weglänge zwischen einem Fahrzeug und einem Hindernis darstellt und somit auch für andere, nicht fahrzeugbezogene Hindernisse verwendet werden kann (vgl. Abschnitt 3.7).

2.4 Grundlagen der Verkehrssimulation

2.4.1 Detaillierungsgrad

Bei der Konzeption eines Simulationsmodells spielt allgemein der angestrebte Realitätsgrad eine große Rolle. Je genauer und detaillierter das Verhalten der Fahrzeuge nachgebildet werden soll, desto mehr Aufwand ist für die Kalibrierung und die Durchführung der Simulation aufzubringen. Selbst moderne Hochleistungsrechner stoßen bei komplexen Simulationsmodellen schnell an ihre Rechenkapazitäten. Daher ist es angebracht, den Detailierungsgrad einer Simulation frühzeitig an die Erfordernisse des späteren Einsatzgebiets anzupassen.

In der Verkehrssimulation werden üblicherweise vier Grade der Realitätstreue unterschieden: makroskopische, mesoskopische, mikroskopische und submikroskopische Modelle. Der Detailierungsgrad, und damit auch die benötigte Rechenleistung, steigt bei dieser Unterteilung von den stark abstrahierenden makroskopischen Modellen zu den hochkomplexen submikroskopischen Modellen drastisch an. Gleichzeitig wird die Größe der mit den jeweiligen Modellen simulierbaren Streckennetze kleiner. Jedes der im Folgenden beschriebenen Modelle besitzt daher spezifische Einsatzgebiete. Auch wenn in dieser Arbeit ausschließlich die mikroskopische Simulation im Vordergrund steht, ist es hilfreich, zunächst die mikroskopische Betrachtungsweise gegenüber den anderen abzugrenzen.

Makroskopische Modelle

Bei diesem Ansatz wird nicht das einzelne Fahrzeug mit seinen Fahreigenschaften betrachtet, sondern vielmehr die Gesamtheit aller Fahrzeuge eines Streckenabschnitts als kontinuierlicher Volumenstrom. Dieser Volumenstrom fließt durch das Streckennetz und kann damit als den bekannten Gesetzen der Hydrodynamik unterworfen angesehen werden. Die einzelnen Fahrzeuge werden nicht mehr „individuell“ nachgebildet, sondern repräsentieren Flüssigkeits- oder Gas-Atome in einem Kontinuum, das an den Knotenpunkten des Netzes zusammenfließt oder sich entsprechend den Druckverhältnissen auf verschiedene Straßen aufteilt. Die verkehrstechnischen Kenngrößen Verkehrsstärke, Verkehrsichte und mittlere Geschwindigkeit lassen sich in diesem Modell äquivalent zu den Strömungsgrößen Volumenstrom, Dichte und Strömungsgeschwindigkeit in der Strömungsmechanik behandeln.

Die grundlegenden Arbeiten zu den makroskopischen Modellen stammen von Lighthill und Whitham ([74][75]) und wurden seither beständig erweitert ([86][67][98]). Haupteinsatzgebiet für den makroskopischen Ansatz ist die Simulation großflächiger Straßennetze, die durch den vergleichsweise geringen Rechenaufwand auch ganze Städte, Regionen oder gar Länder umfassen können. Für die überregionale Stauprognose sind makroskopische Modelle gut geeignet; eine detaillierte Simulation zur realitätsnahen Straßenplanung ist mit ihnen aber nicht möglich.

Mesoskopische Modelle

Mesoskopische Modelle stellen eine Verbindungsebene zwischen der makroskopischen und der mikroskopischen Betrachtungsweise dar. Hierbei können zwei unterschiedliche Ansätze verfolgt werden: Zum einen die datenorientierte Kopplung von zwei parallel agierenden makroskopischen bzw. mikroskopischen Simulationsebenen (z. B. die mikroskopische Simulation von Knotenpunkten oder die makroskopische Simulation der freien Strecke), zum anderen die direkte Überführung einer mikroskopischen Theorie in ein makroskopisches Modell ([89][98]). Einige der existierenden Simulationsprogramme erlauben dabei die direkte Kopplung verschiedener Simulationsebenen zu einem mesoskopischen Ansatz (vgl. auch Abschnitte 2.5.2 und 2.5.4).

Mikroskopische Modelle

Ein mikroskopisches Modell liegt dann vor, wenn jedes einzelne Fahrzeug als singuläre Einheit innerhalb der Simulationswelt repräsentiert wird. Durch ein Verhaltensmodell gesteuert, interagiert das Fahrzeug mit seinen Nachbarfahrzeugen und bildet so auf „atomarer“ Ebene das Strömungsverhalten des Verkehrsstroms nach. Makroskopische Kenngrößen ergeben sich durch Beobachtung der Fahrzeuge und Mittelung (Homogenisierung) der so gewonnenen Bewegungsdaten. Vorteil der mikroskopischen Simulation ist der hohe erreichbare Realitätsgrad, der die direkte Untersuchung des Verkehrsverhaltens für einzelne Streckenabschnitte erlaubt. Auch der Einfluss realer Verkehrssteuerungsmaßnahmen, z. B. durch Lichtsignalanlagen, Zuflussregelungen oder Vorfahrtregelungen, kann auf der mikroskopischen Ebene weitaus präziser abgebildet werden als mit makroskopischen Modellen. Dabei steigt jedoch durch die zahlreichen Objekte und deren Interaktionen auch die Zahl der erforderlichen Rechenschritte pro Zeitintervall sowie der Speicherverbrauch. Dennoch sind mikroskopische Verkehrssimulationen vielseitig einsetzbar und werden deshalb auch zunehmend verwendet. Einen Überblick über mikroskopische Modelle liefert Kapitel 4, eine objektorientierte Referenzimplementierung wird in Kapitel 3 dargestellt.

Submikroskopische Modelle

Eine weitergehende Verfeinerung der mikroskopischen Simulationsebene ist das submikroskopische Modell, das nicht nur einzelne Fahrzeuge auf einem relativ abstrakten Niveau beschreibt, sondern darüber hinaus auch akribisch einzelne Funktionsgruppen der Fahrzeugmechanik und -steuerung nachbildet. Jedes einzelne fahrrelevante Bauteil und sein Zusammenspiel mit den übrigen Komponenten wird dabei so detailliert und physikalisch korrekt wie möglich simuliert. Eine der wenigen submikroskopischen Simulationsprogramme aus dem deutschen Raum – das Programmpeaket PELOPS ([76]) – wird in Abschnitt 2.5.7 beschrieben. Durch den sehr großen Rechen- und Kalibrierungsaufwand wird die submikroskopische Simulation hauptsächlich für den Automobilbau eingesetzt, und ist derzeit verkehrsplanerisch von geringer Relevanz.

2.4.2 Zeitverhalten der Simulation

Soll das reale Verkehrsgeschehen in einem Computerprogramm abgebildet werden, muss dabei die unterschiedliche Interpretation des Zeitbegriffs beachtet werden. Während für den menschlichen Beobachter in der realen Welt die Zeit einen kontinuierlichen Verlauf annimmt, kann ein Computer konstruktionsbedingt nur diskrete Zeitschritte verarbeiten. Um also den realen Verkehrsablauf in die Simulationswelt zu transferieren, muss die Zeit in sinnvoller Weise in Form von Zeitschritten diskretisiert werden, und läuft damit in Form von Zeitschritten ab. Um eine einheitliche Terminologie zu gewährleisten, wird der Begriff *Simulationszeit* im Folgenden für den Zeitablauf aus Sicht der Fahrzeuge verwendet (*simulierte Zeit*), während der Begriff *Realzeit* die Zeitempfindung des menschlichen Anwenders der Simulationssoftware definiert (*benötigte Zeit für die Simulationsberechnung*). Beide Zeitskalen laufen nicht zwangsläufig synchron zueinander ab; vielmehr wird angestrebt, das Verhältnis von Simulationszeit zu Realzeit möglichst groß zu halten, um den Zeitaufwand für die Simulation klein zu halten. Entspricht ein Zeitschritt in der Simulationszeit genau einem gleichgroßen Zeitintervall in der Realzeit, spricht man von einer Echtzeitsimulation.

Für die Festlegung der Dauer eines Zeitschritts in der Simulationszeit existieren im Wesentlichen zwei verschiedene Ansätze: Systeme mit konstanter Zeitschrittweite (zeitschrittorientiert) und solche mit variablen Simulationszeit-Dauern zwischen den einzelnen Aktualisierungsschritten (ereignisorientiert).

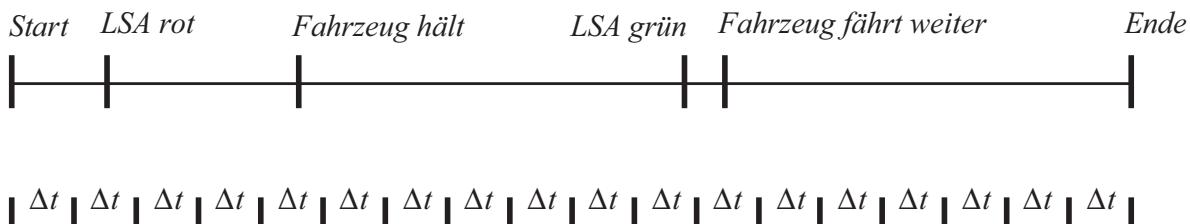


Bild 2.3: Ereignisbasierte (oben) und zeitschrittorientierte Simulation (unten)

Beim ereignisorientierten Ansatz dienen bestimmte Ereignisse (z. B. das Erreichen einer Ausfahrt, die Durchführung eines Spurwechselmanövers, der Beginn der Grünphase einer Lichtsignalanlage, etc.) als Signalgeber. Zwischen diesen Ereignissen behalten die Fahrzeuge ihren Bewegungszustand bei; ihre neue Position wird dann über Bewegungsgleichungen extrapoliert. Alle

Änderungen des Bewegungszustands werden hierzu als Ereignis interpretiert. Die Berechnung von Zwischenstufen kann bei diesem Modell entfallen, was den Rechenaufwand verringert. Steigt mit der Komplexität des simulierten Systems auch die Zahl der auftretenden Ereignisse, verschlechtert sich allerdings das Laufzeitverhalten zu Ungunsten der ereignisorientierten Berechnung und kann sogar den Rechenaufwand zeitschrittorientierter Verfahren übersteigen. Insgesamt gibt es nur wenige ereignisbasierten Simulationsprogramme ([23][29]).

Sehr viel häufiger findet die zeitschrittisierte Simulation Verwendung, da sie einfacher in der Programmierung ist und vielseitiger eingesetzt werden kann. Es wird während der Simulation eine feste Zeitschrittweite vorgegeben, mit der die Simulationszeit fortschreitet. Alle Fahrzeuge werden zu jedem Zeitschritt mit ihrer aktuellen Geschwindigkeit in diskreten Schritten durch das Netz bewegt. Die Größe des Zeitschrittintervals Δt hängt dabei vom verwendeten Simulationsmodell oder von eventuell möglichen Benutzereinstellungen ab; sie liegt meist im Bereich zwischen 0,1 und 1,0 Sekunden. Je geringer die Zeitschrittweite, desto „flüssiger“ kann die Simulation erfolgen, aber desto größer wird auch der Rechenaufwand. Außerdem kann durch eine übermäßige Verringerung der Zeitschrittweite (<0,1 Sekunden) keine künstliche Erhöhung der Simulationsgenauigkeit erreicht werden, da die meisten Verhaltensmodelle für diese Zeitschritte nicht kalibriert wurden und die Reaktionszeiten menschlicher Fahrer deutlich über 0,1 Sekunde liegen.

2.5 Existierende Verkehrssoftware

2.5.1 Überblick

Im Laufe der letzten Jahrzehnte wurden zahlreiche mikroskopische Simulationsprogramme entwickelt, die heute noch im Einsatz sind und kommerziell vertrieben werden. Da der Markt für Verkehrs-Simulationssoftware sehr überschaubar ist, gibt es nur einige wenige kommerzielle Anbieter, die eine signifikante Nutzerbasis vorweisen können: VISSIM, PARAMICS und AIM-SUN, um die drei in Europa bekanntesten Programme zu nennen, haben sich im Laufe der Zeit von universitären Projekten zu kommerziell erfolgreichen, inzwischen auch weltweit vertriebenen Allzweck-Lösungen entwickelt. In Funktionsumfang, Bedienkomfort und Leistungsfähigkeit sind die Unterschiede zwischen den Programmen gering.

2.5.2 VISSIM

Das Simulationspaket VISSIM (*Verkehr in Städten-Simulation*) basierte auf den Arbeiten von Wiedemann et. al. ([124]) an der Universität Karlsruhe und ist die Weiterentwicklung des MISSION-Systems, das im Rahmen des damaligen EG-Projektes ICARUS entstanden ist [125]; vertrieben wird VISSIM von der ptv AG in Karlsruhe. Aufgrund seiner langen Entstehungsgeschichte ist VISSIM ein ausgereiftes und leistungsstarkes Paket, und nimmt heute die Rolle des Marktführers auf dem Bereich der Verkehrssimulation in Deutschland ein ([40][39][38]).

VISSIM basiert auf dem psycho-physischen Fahrverhalten nach Wiedemann [124]; Fahrstreifenwechsel erfolgen nach der Methodik von Sparmann [105]. Wie der Name andeutet, ist VISSIM auf den innerstädtischen Verkehr spezialisiert, kann aber auch Autobahnverkehr nachbilden. Weiterhin werden die Einbindung von ÖPNV-Fahrzeugen, Fahrrädern und Fußgängern unter-

stützt. Erzeugte Simulationsdaten können direkt mit der ebenfalls von ptv vertriebenen makroskopischen Planungssoftware VISUM ausgetauscht werden. Externe Anwendungen können auf VISSIM über eine COM-Schnittstelle zugreifen, eine quelloffene Version von VISSIM ist jedoch nicht erhältlich.



Bild 2.4: Bedienoberfläche und 3D-Ansicht von VISSIM (ptv AG, Karlsruhe)

2.5.3 Paramics

Ein weiterer Vertreter der kommerziell erfolgreichen Komplettsysteme ist das von der Firma Quadstone im schottischen Edinburgh vertriebene Paramics, das ursprünglich auf ein Parallel Computing-Projekt am Edinburgh Parallel Computing Centre (EPCC) und der Universität von Edinburgh zurückgeht ([90]). Kern der Entwicklung war die verteilte Berechnung von großen Straßennetzen auf Parallelrechnern ([26]). Nach und nach kamen Module hinzu, die auch eine komfortable Eingabe, Berechnung und Auswertung auf Einzelplatz-Rechnern ermöglichen. Dies hat dazu geführt, dass Paramics heute eine der meistverbreiteten Simulationsumgebungen ist, und – nach Angaben des Herstellers – von ca. 500 Kunden in 40 Ländern eingesetzt wird.

Das Fahrverhalten von Paramics beruht dabei auf einem von Fritzsche entwickelten Modell ([45]), das beständig weiterentwickelt wird ([90]).

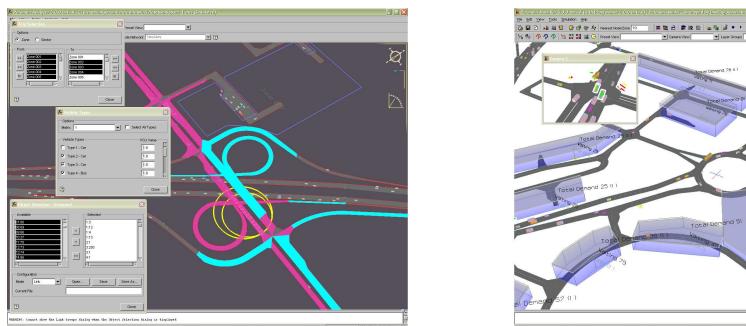


Bild 2.5: Analyser und Modelleur von Paramics (Quadstone, Edinburgh)

2.5.4 AIMSUN

Ebenfalls zu den Marktführern zählt das System AIMSUN, das in Kooperation zwischen der Polytechnischen Universität von Katalonien mit der Firma Transport Simulation Systems (TSS,

Barcelona) entwickelt wurde ([8]). AIMSUN ist Teil der GETRAM-Simulationsumgebung, die neben der mikroskopischen Simulation mit AIMSUN auch die Einbindung makroskopischer Modelle (z. B. EMME/2) erlaubt.

Die Erstellung von Straßennetzen in AIMSUN erfolgt mit einem vergleichsweise komfortablen grafischen Editor (AIMSUN Modeller), der neben der eigentlichen Netzerstellung auch die Bearbeitung GIS-ähnlicher Informationen erlaubt. Als Erweiterungsmöglichkeit steht eine C++- und eine Python-Schnittstelle zur Verfügung.

Das von AIMSUN verwendete Fahrzeugfolgemodell basiert auf den Arbeiten von Gipps ([49]), ebenso wie das Spurwechselmodell ([50]). Eine Zusammenfassung der Verhaltensimplementierung in AIMSUN liefert Barceló ([6]).

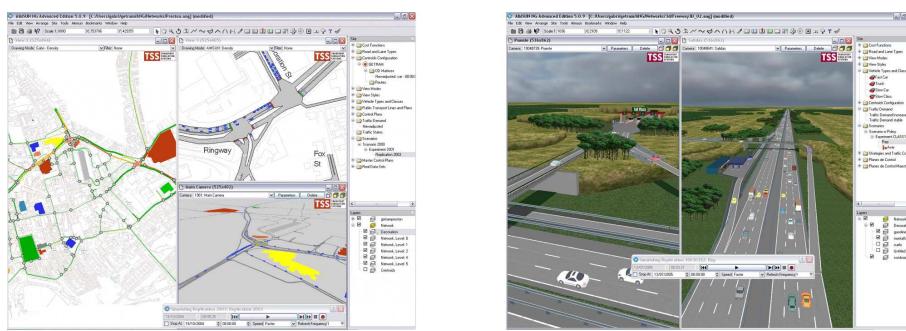


Bild 2.6: Multi-View Oberfläche von AIMSUN NG (TSS, Barcelona)

2.5.5 CORSIM

In den USA ist das Programm CORSIM (*Corridor Traffic Simulation Model*) relativ stark verbreitet, was insbesondere darauf zurückzuführen ist, dass es von der Federal Highway Association (FHWA) in Auftrag gegeben wurde und relativ kostengünstig vertrieben wird. CORSIM basiert auf zwei separat entwickelten Paketen, eines für den innerstädtischen Verkehr (NETSIM) und eines für den Verkehr auf Autobahnen (FRESIM). Insgesamt werden die von der FHWA herausgegebenen Simulationsprogramme bereits seit den 1970er Jahren eingesetzt, und besitzen dementsprechend eine vergleichsweise hohe Benutzerzahl und robuste Simulationsstrategien ([85]).

Das lange Tradition von CORSIM ist aber andererseits ein Nachteil, da viele der verkehrstechnischen und informatischen Grundlagen mittlerweile überholt sind. So leidet CORSIM unter einer veralteten und schwer wartbaren Code-Basis in Fortran, einer nicht mehr konkurrenzfähigen Bedienoberfläche und teilweise nicht mehr zeitgemäßen Modellierungsansätzen. Obwohl noch immer neue Versionen von CORSIM erscheinen (aktuell ist Version 6.0), und auch neue Funktionen eingebaut werden (ÖPNV, Zuflusskontrollen, größere Straßennetze), ist der Einsatz von CORSIM mittelfristig gefährdet. Als Nachfolger wurde von der FHWA inzwischen das Projekt NGSIM (*Next Generation Simulation*) initiiert, das zum Ziel hat, ein einheitliches und offenes System von Algorithmen und Datenstrukturen für künftige Simulationssoftware zu definieren.



Bild 2.7: CORSIM mit dem Grafikmodul TRAFVU (McTrans, Florida)

2.5.6 HUTSIM

An der Technischen Universität Helsinki (HUT) wird seit 1989 das eigene Simulationspaket HUTSIM entwickelt und eingesetzt ([69]). HUTSIM verwendet intern *Pipes*, durch die Fahrzeuge, Straßenbahnen und Fußgänger mit einer Zeitschrittauflösung von bis zu 20 Schritten pro Sekunde bewegt werden. Ursprünglich nur für den innerstädtischen Bereich entwickelt, wurde das Modell mittlerweile auch auf Autobahn- und Landstraßenverkehr erweitert (u. a. in Kooperation mit dem Lehrstuhl für Verkehrswesen der Ruhr-Universität Bochum). HUTSIM wurde in der Programmiersprache Pascal unter MS-DOS entwickelt, und läuft in der aktuellen Version sowohl unter Linux als auch in einer Windows-Umgebung. Im Vergleich zu anderen kommerziellen Paketen, verfügt HUTSIM allerdings über eine relativ schlichten Benutzeroberfläche. Die Möglichkeit einer realitätsnahen grafischen Darstellung ist nur mit externen Visualisierungspaketen von Drittanbietern möglich ([36]).

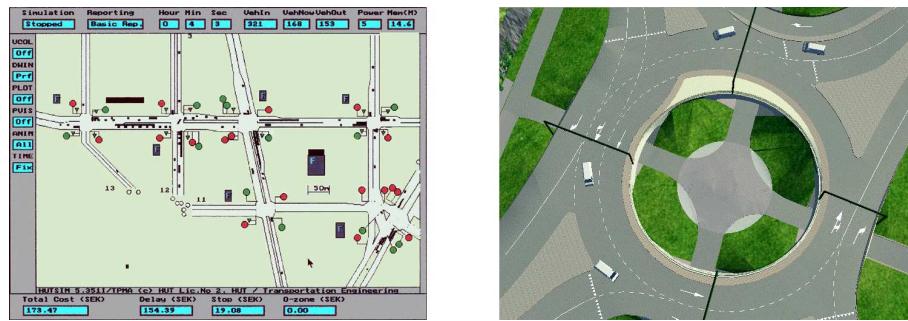


Bild 2.8: HUTSIM (HUT) und 3D-Visualisierung mit Virtual Map (Novapoint)

2.5.7 PELOPS

Anders als die meisten heute existierenden Verkehrssimulationen verfolgt PELOPS (**P**rogrammsystem zur **E**ntwicklung **l**ängsdynamischer **m**ikroskopischer **V**erkehrsprozesse in **S**ystemrelevanter **U**mgebung) einen submikroskopischen Ansatz, d. h. Fahrzeuge werden intern durch zusätzliche Steuerungskomponenten modelliert, die das komplette Fahrzeug vom Gaspedal über Motorblock, Kupplung, Getriebe und Differential bis auf die Kraftübertragung der Reifen nachbilden [76]. Entwickelt wurde PELOPS ab 1989 in Kooperation zwischen der RWTH Aachen und der BMW AG; es wurde u.a. im Rahmen des PROMETHEUS-Projektes eingesetzt. Vertrieben wird es heute von der fka - Forschungsgesellschaft Kraftfahrwesen mbH in Aachen.

Durch den submikroskopischen Ansatz ist PELOPS zwar in der Lage, ein sehr realitätsgetreues Verhalten von Fahrzeugen zu erzielen; beispielsweise können Treibstoffverbrauch und Schadstoffemissionen gut nachgebildet werden. Andererseits begrenzt der hohe Detaillierungsgrad den Einsatz von PELOPS bei der Simulation größerer Netze. Das Haupteinsatzgebiet des Systems sind deshalb, auch aufgrund der hohen Lizenzgebühren, die Forschungs- und Entwicklungsabteilungen der Automobilindustrie und weniger die Verkehrsplanung.

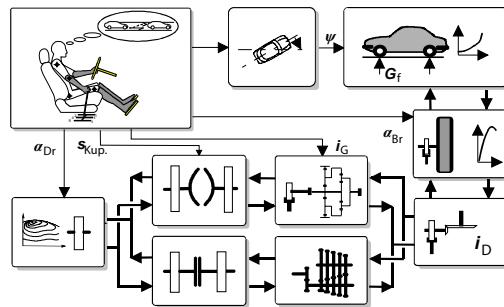


Bild 2.9: Submikroskopische Fahrzeugmodellierung in PELOPS (fka, Aachen)

2.5.8 OLSIM

Einen Sonderfall nimmt das OLSIM-System (*Online-Simulation*) ein, das an der Universität Duisburg-Essen entwickelt wurde. OLSIM bildet die Basis für das seit März 2003 erfolgreich betriebene, internetgestützte Stauprognoze-System des Landes NRW. OLSIM verwendet Zellulärautomaten nach Nagel-Schreckenberg ([77]) und kann ein Streckennetz von insgesamt 2250 Autobahn-Kilometern im Rechner abbilden. Dabei werden in Echtzeit Messdaten von 2500 Messstellen in die Simulation eingebunden, die halbstündliche bzw. stündliche Verkehrsprognosen erlauben. Die Akzeptanz dieses WWW-gestützten Informationsdienstes ist überdurchschnittlich hoch, täglich registriert die Internetseite bis zu 220.000 Zugriffe ([88]).

Möglich wird eine derart massive Echtzeit-Simulation durch das stark vereinfachende Prinzip des Zellulärautomaten. Hiermit können zwar enorm viele Fahrzeuge in sehr kurzer Zeit simuliert werden, und dabei auch makroskopisch sehr gute Ergebnisse erzielt werden; leider geschieht dies jedoch auf Kosten der mikroskopischen Genauigkeit. Für die Bemessung und Bewertung von strassenplanerischen Projekten ist dieses Projekt daher nur bedingt geeignet.



Bild 2.10: Prognose des Verkehrs im Ruhrgebiet mit OLSIM (Universität Duisburg-Essen)

2.6 Einschränkungen existierender Lösungen

Universitäten, Forschungsinstitute, Verkehrsbehörden und privatwirtschaftliche Planungsbüros bilden zusammen die Hauptabnehmer für verkehrstechnische Simulationssoftware. Da dieser Markt überschaubar ist, und selbst die kommerziell erfolgreichsten Softwaresysteme in diesem Bereich nur einen kleinen Kundenstamm vorweisen können, sind die Kosten für die Softwarelizenzen dementsprechend hoch. Für den mittelständischen Verkehrsingenieur wirkt sich dieser Kostenfaktor hinderlich aus und unterbindet die eigentlich wünschenswerte Verwendung der Verkehrssimulation als Planungswerkzeug.

Ein weiterer Effekt des kleinen Marktes ist der intensive Wettbewerb um die wenigen potentiellen Kunden, der zwischen den kommerziellen Anbietern herrscht. Neben Bedienbarkeit, Ausführungsgeschwindigkeit und Funktionsumfang stellt dabei die Güte der mit den Simulationsmodellen erzielbaren Ergebnisse das wichtigste Alleinstellungskriterium dar. Alle Anbieter sind dementsprechend bemüht, die Leistungsfähigkeit ihrer Modelle und Algorithmen hervorzuheben, ohne dabei zu viele Implementierungsdetails für die Konkurrenz offenzulegen. Ein programmatischer Zugriff auf die Softwarepakete durch externe Anwendungen kann oft nur über externe Schnittstellen erfolgen. Direktes Einsehen, Verifizieren und Modifizieren des Quelltextes wird durch eine strikte „Closed-Source“-Politik kategorisch unterbunden. Die Simulationssoftware stellt somit in der Regel eine „Black-Box“ dar, deren innere Struktur vom Anwender nicht eingesehen werden kann. Ein vollkommen quelloffenes Simulationspaket mit ausreichendem Funktionsumfang existiert bisher noch nicht.

Aufgrund der heterogenen Struktur der verschiedenen Programmsysteme ist es darüber hinaus in den seltensten Fällen mögliche, bestehende Streckennetze von einem System auf ein anderes zu übertragen - zumindest nicht, ohne dabei massive manuelle Änderungen vornehmen zu müssen. Ein gemeinsamer Standard für den Datenaustausch in der mikroskopischen Verkehrssimulation existiert nicht und wird aufgrund der gegenläufigen Geschäftsinteressen der Hersteller wohl auch in absehbarer Zeit nicht vorliegen. Daher begibt sich der Nutzer eines kommerziellen Programms in eine starke Abhängigkeit vom jeweiligen Anbieter; insbesondere muss er sich auf beständigen Support und Weiterentwicklung des Programms verlassen können. Es muss sichergestellt sein, dass einmal getätigte Investitionen in die ausgewählte Software zukunftssicher bleiben; eine eventuelle Umstellung des Betriebssystems darf nicht automatisch auch eine teure Umstellung der Simulationssoftware erfordern.

Die lange Entwicklungsgeschichte einiger Programmprodukte, deren erste Versionen oftmals viele Jahre zurückliegen, bedingt auch das gehäufte Vorhandensein von sogenanntem „legacy code“, also veralteten Quelltext-Abschnitten und -Strukturen, die von Version zu Version weitergereicht werden. Dieser Umstand erschwert einerseits stark die Weiterentwicklung, da der Quelltext im Laufe der Jahre unübersichtlich zu werden pflegt; andererseits wird hierdurch der Einsatz modernerer Programmier-Paradigmen unterbunden. Programmiersprachen wie Pascal und Fortran haben durchaus immer noch ihre Berechtigung bei kleineren Software-Projekten, stellen aber für große, sich ständig weiterentwickelnde Projekte oftmals eine gravierende Behinderung dar. Auch die oftmals „nostalgisch“ anmutenden Benutzungsoberflächen älterer Programme sind für den heutigen Anwender meist unzumutbar. Der einzige Weg, veraltete Programmstrukturen auf einen aktuellen Stand zu bringen, ist meist die komplette Neuentwicklung des Systems von Grund auf, was allerdings sehr kostenintensiv und daher nur in den seltensten Fällen durchführbar ist.

2.7 Verbesserter Lösungsansatz

Unter Berücksichtigung der oben aufgeführten Nachteile wurde für die zu entwickelnde Simulationsumgebung das folgende Anforderungsprofil aufgestellt:

- **Objektorientiertes Modell:** Das komplette System soll aus miteinander interagierenden Klassen bestehen, Vererbungshierarchien nutzen und sich dabei so weit wie möglich an den realen Gegebenheiten orientieren.
- **Erweiterbarkeit:** Es müssen geeignete Schnittstellen und Entwurfsmuster verwendet werden, um auch eine spätere Erweiterbarkeit des Klassenmodells zu gewährleisten.
- **Plattformunabhängig:** Der Einsatz der Software soll nicht auf bestimmte Plattformen festgelegt sein, sondern möglichst flexibel auch auf anderen Rechnerarchitekturen und Betriebssystemen verwendet werden können. Der Einsatz plattformspezifischer Bibliotheken ist zu vermeiden.
- **Zugänglichkeit:** Um den späteren Einsatz für weitergehende Projekte zu gewährleisten und einen transparenten Einblick in die verwendeten Algorithmen zu ermöglichen, muss der gesamte Quelltext bei Bedarf jederzeit nachvollziehbar verfügbar gemacht werden. Ein geschlossener Softwareentwurf, der nur durch Schnittstellen einen Zugriff durch externe Programme erlaubt, ist nicht zielführend.
- **Benutzerführung:** Das Programm sollte über eine einfach zu bedienende grafische Benutzeroberfläche verfügen und die durch das Objektmodell bereitgestellte Semantik dazu nutzen, die intuitive Bedienbarkeit der Anwendung zu verbessern.
- **Parametrisierbarkeit:** Da das Programm unter Umständen auch von Personen mit geringer Computererfahrung verwendet werden wird, muss eine Anpassung der Fahrmodelle auch ohne Eingriff in den Quelltext durchführbar sein. Geeignete Parameter und Einstellungsmenüs sind vorzusehen.
- **Datenaustausch:** Es sollten Schnittstellen vorgesehen werden, um einen Datenaustausch mit externen Datenquellen und Anwendungsprogrammen zu ermöglichen.
- **Skalierbarkeit:** Neben dem Betrieb auf einem einzelnen Rechner soll das System die gleichzeitige parallele Simulation auf mehreren Rechnern unterstützen. Auf diese Weise können auch große, räumlich umfassende Streckennetze erfasst werden. Die theoretisch maximal simulierbare Netzgröße soll dabei in erster Linie von der Zahl der an der Simulation beteiligten Rechner abhängen.
- **Neue Verhaltensmodelle:** Gegenüber existierenden Lösungen soll die Modifikation bestehender Verhaltensmodelle sowie die Einbindung neuer Modelle vereinfacht werden. Ein neues, modular erweiterbares, auf Selbstorganisationsprinzipien basierendes Verhaltensmodell für verschiedene Verkehrssituation soll implementiert werden.

Die Umsetzung einer solchen generisch verwendbaren Simulations-Software wird in den folgenden Kapiteln dargestellt.

Kapitel 3

Modellierung

3.1 Einleitung

Auf den folgenden Seiten soll die Entwicklung eines durchgängig objektorientierten Computermodells für die Verkehrssimulation beschrieben werden. Grundlage der Modellierung bildet dabei die Unified Modeling Language (UML) als Beschreibungskalkül, die eine weitgehend Programmiersprachen-unabhängige, jederzeit nachvollziehbare . Alle im Folgenden entwickelten Diagramme folgen der UML 2.0 Spezifikation vom Oktober 2004 ([55]). Um die Übersichtlichkeit zu wahren, werden nur die für das Verständnis des Modells wesentlichen Elemente in den UML-Diagrammen dargestellt. Die Details der Umsetzung des Klassenmodells in die Programmiersprache Java wird im Kapitel 6 beschrieben.

Das hier beschriebene Modell bezieht sich allein auf den verkehrstechnischen Teil der Simulation. Alle Klassen, die sich mit informationstechnischen Aspekten, wie der Benutzerführung, der grafischen Darstellung, der Datenauswertung, dem Datenaustausch etc. beschäftigen, werden ebenfalls im Implementierungs-Kapitel erläutert.

3.2 Anforderungen

Ziel des hier beschriebenen Modells ist die Nachbildung aller wesentlichen realen Elemente des Verkehrsablaufs. Obwohl die Software primär für die Simulation des Verkehrs auf Bundesautobahnen ausgelegt ist, wird das Modell so konzipiert, dass auch der innerstädtische Verkehr realitätsnah nachgeahmt wird. Hauptaugenmerk der Modellierung liegt daher auf der Erweiterbarkeit des Systems, die auch während der Implementierung und Wartung des Programms gewährleistet bleiben muss.

Da die Verwendung eines objektorientierten Modells bei falscher Verwendung zu einem größeren Speicher- und Rechenzeitbedarf (z.B. für Objektinstanziierung oder Speicherbereinigung) im Vergleich zu traditionellen prozeduralen Ansätzen führen kann , ist schon in der Projektentwurfsphase auf die Performanz zu achten. Potentielle „Flaschenhälse“ (d. h. ressourcenintensive Programmteile mit Verzögerungseffekten) sind daher so früh wie möglich zu identifizieren und zu eliminieren. Modellierungsentscheidungen, die aufgrund von Performanz-Überlegungen getroffen wurden, werden daher im weiteren Text entsprechende herausgestellt.

3.3 Simulationsablauf

Bevor die einzelnen Elemente der Simulationswelt beschrieben werden, wird zunächst der grund-sätzliche organisatorische Ablauf der Simulation erörtert. Vor Durchführung des eigentlichen Simulationslaufs muss sichergestellt sein, dass alle der Simulation zu Grunde liegenden Daten (insbesondere das Straßennetz mit Streckeninformationen, Verkehrsstärken etc.) bereits vorliegen. Dabei ist es unerheblich, ob diese Daten durch ein Programm erzeugt, von einem Benutzer eingegeben oder aus externen Dateiformaten importiert werden – wichtig ist lediglich, dass die vollständige Beschreibung des zu simulierenden Netzes durch Instanzierung des hier beschriebenen Objektmodells erfolgt.

In der zeitschrittbasierten Simulation setzt sich ein Simulationslauf aus einer Folge von diskreten Simulationsschritten konstanter Dauer zusammen. Für das vorliegende Modell werden dabei Zeitschrittweiten von 0,1 bis 1,0 Sekunden verwendet. Die Zeit, die für die Berechnung eines Zeitschritts durch den Computer benötigt wird, kann dabei stark von der Zeitschrittdauer in der Simulation abweichen. Im Idealfall sollte die Berechnungszeit deutlich unter der simulierten Zeit liegen, um die schnelle Prognose zukünftiger Verkehrsstärken zu ermöglichen. Bei komplexen Netzen mit hohen Verkehrsstärken kann der Berechnungsaufwand – und damit die Rechenzeit – allerdings deutlich ansteigen, wodurch der Verkehr in der Simulation im ungünstigsten Fall langsamer abläuft als in der Realität.

Jeder einzelner Simulationsschritt setzt sich aus verschiedenen Phasen zusammen, die nacheinander in einer festen Reihenfolge durchlaufen werden müssen (vgl. Abbildung 3.1):

- **Hindernisse erkennen:** Jedes Fahrzeug muss sich zunächst Informationen über alle Hindernisse in seiner Umgebung verschaffen, die sein Fahrverhalten potentiell beeinflussen. Dies können z.B. andere Fahrzeuge, Lichtsignale, oder Haltelinien sein.
- **Beschleunigung wählen:** Die zur Verfügung stehenden Umgebungsinformationen werden durch das Fahrverhalten ausgewertet und eine neue Beschleunigung bestimmt, die jedoch erst in der Bewegungsphase angewendet wird.
- **Spurwechsel durchführen:** Auf mehrspurigen Straßen kommen Spurwechsel in Betracht; das Spurwechselverhalten entscheidet, ob ein Spurwechsel eingeleitet werden soll.
- **Fahrzeug weiterbewegen:** Die Geschwindigkeit des Fahrzeugs wird um die gewählte Beschleunigung geändert, und das Fahrzeug um die Strecke verschoben, die dem Produkt aus der Zeitschrittdauer und der aktuellen Geschwindigkeit entspricht.

Die Einteilung in die beschriebenen vier Phasen ist erforderlich, um sicherzustellen, dass sich alle Fahrzeuge erst dann bewegen, wenn alle ihre neue Wunschbeschleunigung gewählt haben. Andernfalls sind Inkonsistenzen bei der Entfernungsbestimmung die Folge (frühere Fahrzeuge wären den später aufgesetzten Fahrzeugen immer um einen Zeitschritt voraus). Die konkrete Umsetzung einer Zeitschritt-Berechnung wird im folgenden Sequenzdiagramm dargestellt:

3.4 Fahrer-Fahrzeug-Elemente

Die kleinste und wichtigste Grundeinheit der mikroskopischen Verkehrssimulation ist das Fahrer-Fahrzeug-Element (FFE), das einen einzelnen motorisierten Verkehrsteilnehmer umfasst. Da-

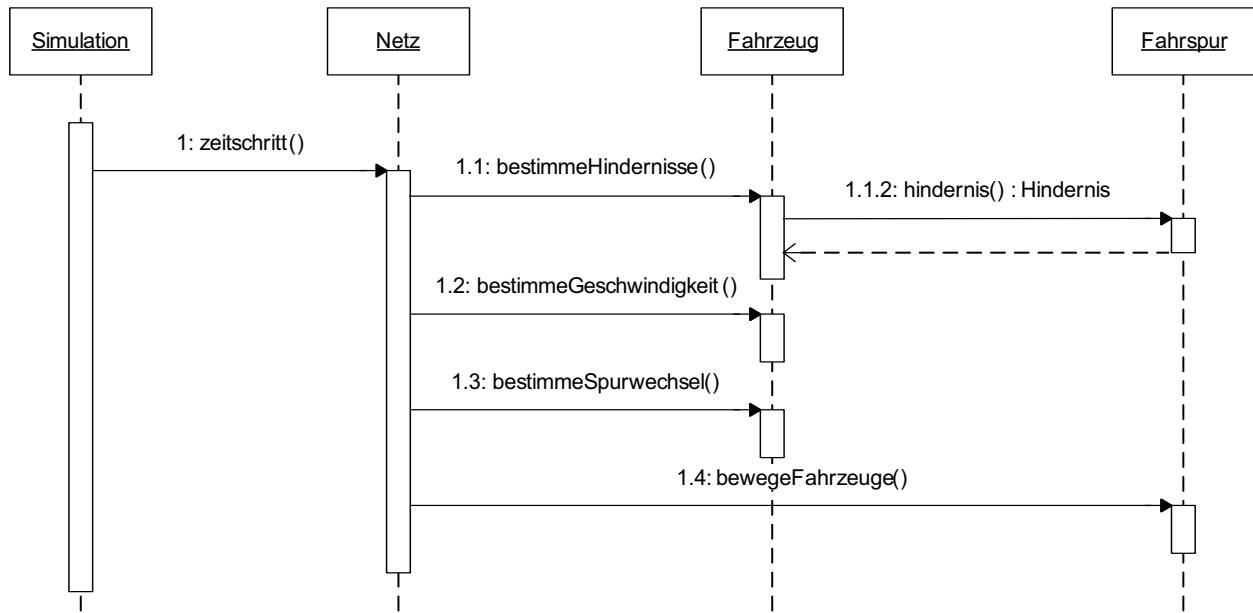


Bild 3.1: Sequenzdiagramm der Haupt-Simulationsschleife

bei beschreibt das Fahrzeug alle Eigenschaften des Fortbewegungsmittels (z.B. geometrische Abmessungen, Beschleunigungsvermögen, Position und Fahrzustand), während der Fahrer Informationen für das Fahrverhalten bereitstellt (z.B. Sicherheitsbedürfnis oder Beschleunigungswille). Da sowohl Fahrzeug als auch Fahrer für die Berechnung eines Simulationsschritts erforderlich sind, treten sie in der Simulation nie isoliert, sondern ausschließlich in Form eines FFE auf.



Bild 3.2: Klassendiagramm eines Fahrer-Fahrzeug-Elements

Prinzipiell ist ein Austauschen des Fahrers und/oder des Fahrzeugs möglich; z. B. könnte man ein und denselben Fahrer nacheinander mit verschiedenen Fahrzeugen fahren lassen, um den Einfluss des Fahrzeugtyps auf die Simulation zu überprüfen. Dieser Fall wird hier jedoch nicht weiter verfolgt, so dass Fahrer und Fahrzeug im Regelfall während ihrer ganzen Existenz fest verbunden bleiben. Beide besitzen eine „gemeinsame Lebenslinie“, d.h. sie werden zur selben Zeit erzeugt und auch gleichzeitig aus der Simulation entfernt.

Innerhalb der FFE übernehmen sowohl Fahrer als auch Fahrzeug klar definierte Funktionen, die den realen Gegebenheiten entlehnt sind. So ist die Klasse `Fahrer` hauptsächlich für die Steuerung eines Fahrzeugs zuständig, während die Klasse `Fahrzeug` die physikalische Bewegung entlang der jeweiligen Fahrspur realisiert. Die realitätsnahe Nachbildung des tatsächlichen Verkehrsgeschehens ergibt sich aus der Kommunikation zwischen Fahrer und Fahrzeug und der Interaktion zwischen den verschiedenen Fahrer-Fahrzeug-Elementen.

3.4.1 Fahrer

Die Klasse Fahrer ist, analog zu ihrem menschlichen Pendant, für die Steuerung des Fahrzeugs zuständig. Dazu werden für jeden Fahrer individuelle Charaktereigenschaften definiert, die seine Fahrweise bestimmen. Zusätzlich besitzt jeder Fahrer eine ihm eigene Instanz eines Fahrverhaltens, die Steuerungsinformationen aus den über das Fahrzeug aufgenommenen Umgebungsinformationen und den Eigenschaften des Fahrers ableitet.

Insgesamt werden jedem Fahrer fünf verschiedene Eigenschaften zugeordnet, die sich an dem psycho-physischen Modell von Wiedemann anlehnen ([124]). Jede dieser Eigenschaften wird durch eine normalverteilte Fließkommazahl der folgenden Form definiert:

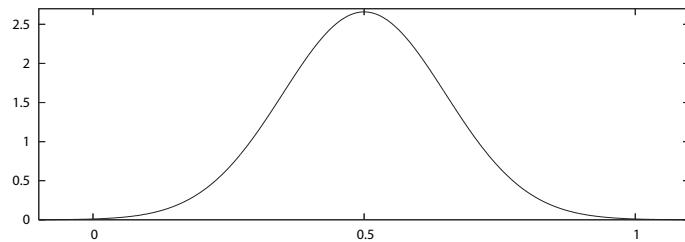


Bild 3.3: Diagramm einer $(0.5,0.15)$ -Normalverteilung

Die folgenden Eigenschaften wurden von Wiedemann übernommen und entsprechen den dort verwendeten Zufallsvariablen ZF0 bis ZF4:

- **Wunschgeschwindigkeit:** Diese gibt qualitativ an, welche Wunschgeschwindigkeit der Fahrer anstrebt. Je größer dieser Wert ist, desto höher ist die angestrebte Geschwindigkeit. Der absolute Wert der Wunschgeschwindigkeit hängt vom Fahrzeugtyp und der erlaubten Höchstgeschwindigkeit ab (vgl. Abschnitte 3.4.5 und 3.4.6).
- **Sicherheitsbedürfnis:** Hiermit wird das individuelle Sicherheitsbedürfnis des Fahrers beschrieben. Ein großes Sicherheitsbedürfnis impliziert einen großen angestrebten Sicherheitsabstand zu anderen Fahrzeugen oder Hindernissen.
- **Schätzvermögen:** Hierdurch lässt sich erfassen, wie gut Fahrer Entfernungen oder Geschwindigkeiten eines Hindernisses einschätzen können. Je höher der Wert, desto genauer schätzt der Fahrer Distanzen ein. Das Schätzvermögen wird verwendet, um Wahrnehmungsschwellen für das Wiedemannsche Abstandsmodell zu berechnen (vgl. 4.2.3).
- **Beschleunigungswille:** Hierüber wird das relative Beschleunigungsvermögen des Fahrzeugs repräsentiert und festgelegt, wie sehr der Fahrer gewillt ist, dieses auch auszunutzen. Es wird unterstellt, dass „sportliches“ Fahren meist auch mit einer erhöhten maximalen Bremsbeschleunigung einher geht, wodurch dieser Parameter sowohl für Beschleunigungs- als auch Bremsmanöver verwendet werden kann. Große Werte bedeuten dabei hohe maximale Absolut-Beschleunigungen.
- **Gaspedal-Kontrolle:** Diese ist ein Maß dafür, wie präzise das Gaspedal beim Beschleunigen reagiert. Liegt dieser Wert hoch, beschleunigt das Fahrzeug stärker als vom Fahrer eigentlich beabsichtigt war.

Obwohl die Fahrer-Eigenschaften in erster Linie zur Umsetzung des Wiedemannschen Fahrverhaltens implementiert wurden (siehe Kapitel 4.2.3), können sie auch allgemein für andere Fahrverhalten genutzt werden. Auch die Einbindung weiterer Attribute zur detaillierteren Beschreibung des Fahrers ist problemlos möglich. So könnten beispielsweise auch zeitlich veränderliche Eigenschaften eines Fahrers, wie Ungeduld oder Übermüdung, implementiert werden. Da das Hinzufügen weiterer Parameter, insbesondere wenn sie sich zeitlich ändern, aber immer mit zusätzlichem Kalibrierungsaufwand verbunden ist und sich die vorhandenen Parameter als hinreichend erwiesen haben, wurde hier auf die Einbindung weiterer Fahrereigenschaften verzichtet.

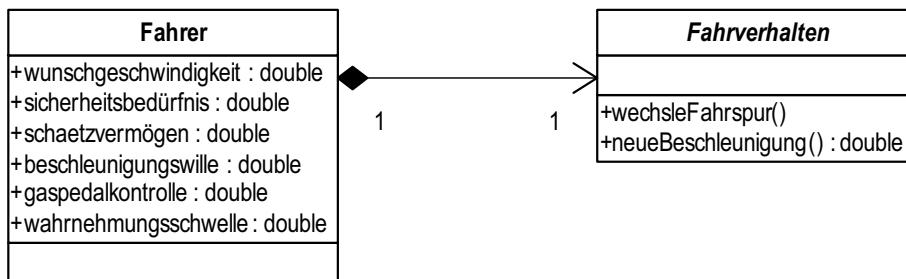


Bild 3.4: Klassendiagramm des Fahrers und seines Verhaltens

3.4.2 Fahrverhalten

Über das Fahrverhalten erfolgt die eigentliche Steuerung eines Fahrzeugs. Nach Auswertung der verfügbaren Informationen über den Fahrzeugzustand, etwaige Hindernisse, Fahrereigenschaften, etc., wird entschieden, wie sich das Fahrzeug in einem betrachteten Zeitschritt verhält. Da das Simulationsmodell bewusst erweiterbar ausgelegt wurde, lassen sich beliebig viele Arten von Fahrverhalten definieren. Für jedes neue Fahrverhalten wird dazu eine neue Unterklasse der abstrakten Klasse *Fahrverhalten* definiert.

Primäre Aufgabe des Fahrverhaltens ist die Bewegung des Fahrzeuges entlang seiner Fahrspur. Hierfür wird durch die Simulation zu jedem Zeitschritt vom Fahrverhalten eine neue Längsbeschleunigung des Fahrzeugs angefordert. Durch Euler-Integration führt diese anschließend zu einer Geschwindigkeitsänderung des Fahrzeugs. Durch eine zweite Integration wird die neue Geschwindigkeit dann in eine Positionsänderung des Fahrzeuges umgerechnet. Wie das Fahrverhalten den neuen Wert der Längsbeschleunigung bestimmt, ist von der jeweiligen Implementierung der Fahrverhaltensklasse abhängig. Eine detaillierte Übersicht über verschiedene Fahrverhalten wird in Kapitel 4.2 gegeben.

Befindet sich das Fahrzeug auf einem Streckenabschnitt mit mehreren Richtungsfahrbahnen, ist ein Spurwechsel möglich. Auch die Entscheidung über einen Spurwechsel ist Aufgabe der Fahrverhaltensklasse. Im Gegensatz zur Längsbeschleunigung, die zu jedem Zeitpunkt einen neuen Wert annehmen kann, ist ein Spurwechsel ein Prozess, der, einmal eingeleitet, über mehrere Zeitschritte hinweg aktiv bleibt. Löst ein Fahrverhalten einen Spurwechsel aus, muss dieser erst abgeschlossen werden, bevor ein neuer Spurwechselvorgang eingeleitet werden kann. Abbildung 3.5 verdeutlicht den Ablauf eines Spurwechselvorgangs.

Jedem Fahrer wird bereits bei der Erzeugung ein Fahrverhalten zugewiesen, das er während der gesamten Simulationszeit beibehält. Von welcher Art dieses Fahrverhalten ist, entscheidet die

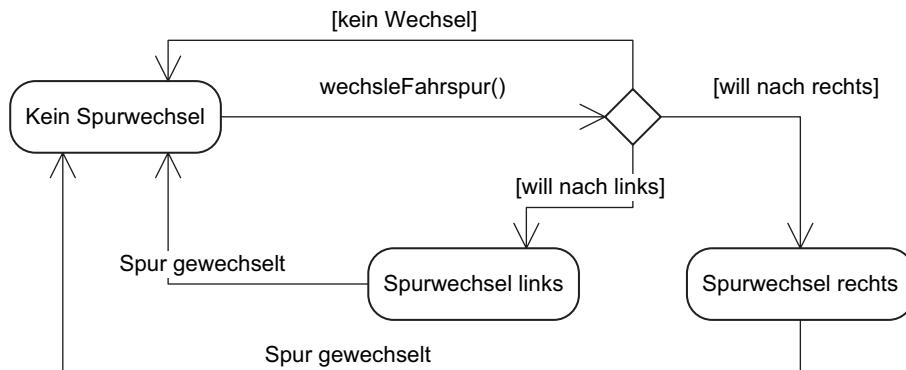


Bild 3.5: Zustanddiagramm des Spurwechselverhaltens

Klasse *VerhaltensManager*, die für jedes Fahrzeug eine neue Instanz einer Fahrverhaltens-Klasse erzeugt. Wird während der Simulation ein anderes Verhaltensmodell gewählt, betrifft diese Änderung nur neu erstellte Fahrer-Fahrzeug-Einheiten – bereits in der Simulation befindliche Fahrzeuge verwenden weiterhin das ursprüngliche Fahrverhalten.

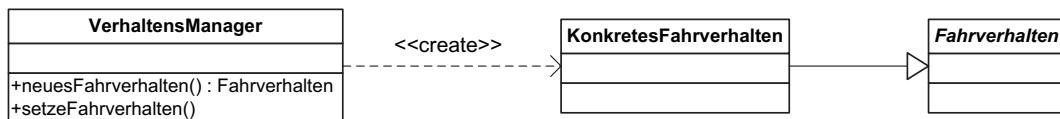


Bild 3.6: Erzeugen von Fahrverhalten durch den VerhaltensManager

3.4.3 Fahrzeuge

Naturgemäß bilden Fahrzeuge den wichtigsten Teil des Simulationsmodells: Das gesamte Verkehrsgeschehen wird durch die Bewegung der Fahrzeuge nachgebildet, und die Analyse dieser Bewegung liefert die für den Verkehringenieur interessanten Daten. Daher muss ein Fahrzeug ausreichende Funktionen zur Verfügung stellen, um seinen aktuellen Zustand zu beschreiben und während der Fahrt Informationen sammeln zu können. Des Weiteren muss über das Fahrzeug dem Fahrer bzw. dem Fahrverhalten die Möglichkeit geben werden, den Bewegungszustand des Fahrzeugs zu beeinflussen.

Der dynamische Zustand eines Fahrzeuges wird dabei über seine Position, seine Geschwindigkeit und die aktuelle Beschleunigung beschrieben. Letztere wird vom Fahrverhalten für jeden Zeitschritt neu festgelegt, Geschwindigkeit und Position werden anschließend aktualisiert. Eine Übersicht über den Ablauf eines Simulationsschrittes liefert Abbildung 3.1.

Die Lokalisierung eines Fahrzeugs in einem Weltkoordinatensystem erfolgt über die Fahrspur, auf der sich das Fahrzeug befindet. Jede Fahrspur besitzt ein lokales linienförmiges Koordinatensystem, das entlang der Fahrbahnmittellinie verläuft und seinen Ursprung im Anfangspunkt der Fahrspur besitzt. Entlang dieses Koordinatensystems wird die Position des Fahrzeugmittelpunktes als Distanz angegeben (vgl. Bild 3.7). Zur Umrechnung dieser lokalen Koordinaten in das globale Koordinatensystem der Simulationswelt stehen in der Klasse *Fahrspur* entsprechende Methoden zur Verfügung (siehe Abschnitt 3.5.1).

Damit ein Fahrzeug stets über alle anstehenden notwendigen Spurwechsel informiert ist, wird ihm direkt zur Erzeugungszeit eine Strecke mitgegeben, entlang der das Fahrzeug fahren möchte.

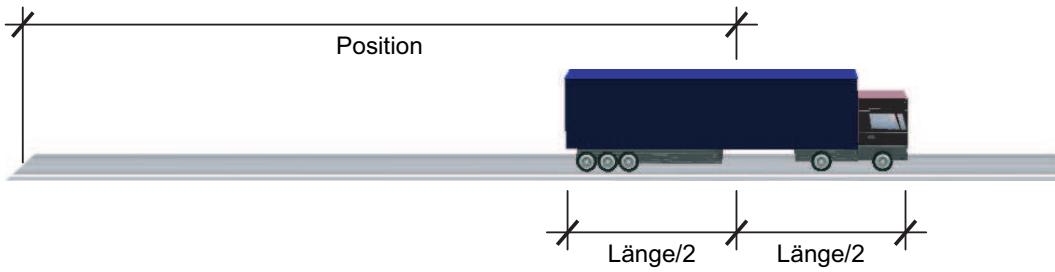


Bild 3.7: Positionierung eines Fahrzeugs relativ zum Beginn einer Fahrspur

Anhand dieser Streckeninformationen kann das Fahrzeug jederzeit die Entfernung zur nächsten Ausfahrt bestimmen (siehe Abschnitt 3.6). Für Auswertungszwecke wird dem Fahrzeug außerdem eine Instanz der Klasse *Reisezeit* zugewiesen, die Informationen über Reisezeiten, Geschwindigkeiten und zurückgelegte Distanzen kapselt. Wird ein Fahrzeug aus der Simulation entfernt, können die während der Fahrt aufgenommenen Werte über das Reisezeit-Objekt eingesehen und ausgewertet werden.

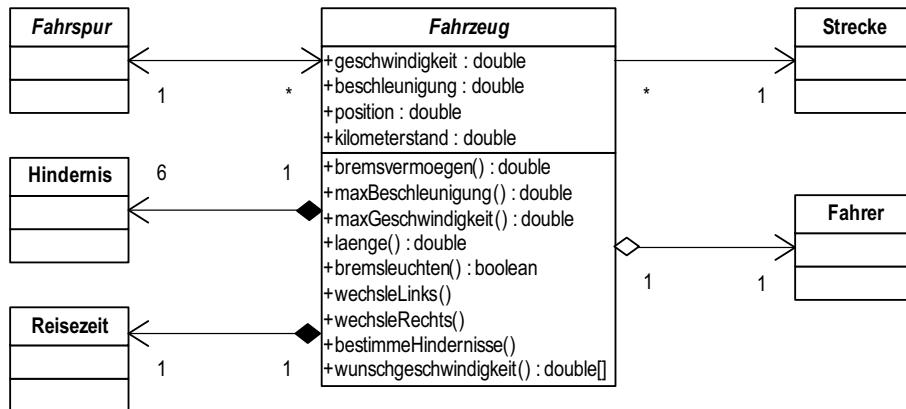


Bild 3.8: Klassendiagramm eines Fahrzeugs

3.4.4 Fahrzeugtypen

Fahrzeuge lassen sich je nach Bauart, Verwendungszweck und Fahreigenschaften in verschiedene Fahrzeugtypen einteilen. Jeder Fahrzeugtyp weist eine ihm eigene, charakterische Fahrweise auf, die auch auf der makroskopischen Ebene messbare Auswirkungen zeigt. So hat beispielsweise ein hoher Schwerlastanteil einen starken Einfluss auf die Leistungsfähigkeit einer Straße. Diese vom Fahrzeugtyp abhängigen Effekte sollten bestmöglich in der Verkehrssimulation nachgebildet werden, um die Realität gut widerzuspiegeln.

Der Begriff Fahrzeugtyp ist in diesem Zusammenhang nicht mit einem einzelnen Fabrikat oder Modell eines Automobilherstellers zu verwechseln. Vielmehr werden unter dem Oberbegriff Fahrzeugtyp verschiedene Fahrzeuge mit vergleichbaren Eigenschaften zusammengefasst. Mögliche Fahrzeugtypen sind z. B. Pkw, Lkw, Bus oder Krad. Jeder dieser Fahrzeugtypen weist spezielle Eigenarten auf, die in einer eigenen Unterklasse innerhalb des Computermodells repräsentiert werden.

Jeder Fahrzeugtyp erfordert einen eigenen Satz an Parametern und Regeln. So müssen für

jede neue Fahrzeug-Klasse Informationen über Wunschgeschwindigkeitsverteilungen, typische Beschleunigungs- und Bremsvermögen und die Größenverteilungen der Fahrzeuggeometrien bekannt sein. Implementiert wurden die beiden Fahrzeug-Klassen Pkw und Lkw.

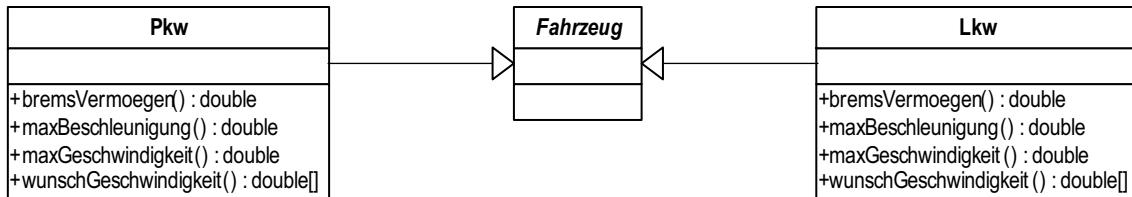


Bild 3.9: Realisierung der verschiedenen Fahrzeug-Unterklassen

Auf die Einbindung von Krafträder wurde bewusst verzichtet, da sie einerseits im Werktagsverkehr durch vergleichsweise geringe Verbreitung vernachlässigbaren Einfluss auf das Verkehrsgeschehen haben und andererseits nur wenige wissenschaftliche Untersuchungen zum Thema „Motorräder in der Verkehrssimulation“ existieren (z. B. [83] und [30]). Es wird deshalb hier davon ausgegangen, dass der makroskopische Einfluss von Krafträder sich nur geringfügig von dem eines Pkw unterscheidet, und daher alle Krafträder annähernd durch Pkw ersetzt werden können. Nicht motorisierte Fahrzeuge, wie z. B. Fahrräder, wurden ebenfalls nicht in die Simulationsumgebung integriert, da sie für das Haupteinsatzgebiet – die Simulation des Autobahnverkehrs – nicht relevant sind. Eine spätere Implementierung ist aber ohne Schwierigkeiten realisierbar.

Fahrzeuge des öffentlichen Personennahverkehrs (ÖPNV) erfordern zusätzlich zur normalen Fahrzeugsteuerungslogik die Implementierung eines Fahrplanmodells in Form von Fahrtrouten und Haltestellen. Im Kapitel 7.5 wird ein ÖPNV-Modell beschrieben, das die beiden zusätzlichen Fahrzeugtypen *Bus* und *Straßenbahn* beschreibt. Beide Klassen basieren dabei auf dem im folgenden beschriebenen Modell für Personen- und Lastkraftwagen. Abbildung 3.10 zeigt alle vier implementierten Fahrzeugtypen in der dreidimensionalen Ansicht der Simulationsumgebung.



Bild 3.10: Die verschiedenen Fahrzeugtypen des Simulationsmodells

3.4.5 Pkw

Unter dem Begriff Pkw (Personenkraftwagen) werden im Folgenden alle Fahrzeuge zusammengefasst, deren Gesamtgewicht 3,5 Tonnen nicht überschreitet. Auch Kleintransporter und Krafträder werden bei der Simulation vereinfachend als Pkw behandelt, um die Anzahl der zu implementierenden und kalibrierenden Sonderklassen zu beschränken.

Bei der Erzeugung eines Pkw werden die Breite und Länge jeweils über eine eigene Normalverteilung zufällig bestimmt. Die Breite eines Fahrzeugs hat direkten Einfluss auf das Wahrnehmungsverhalten nach dem Modells von Wiedemann; die Länge hingegen ist für alle Abstandsmodelle relevant, da diese direkt den Nettoabstand beeinflusst. Basierend auf einer Arbeit von Weiser ([122]), der eine durchschnittliche Pkw-Breite von 1,74 m vorschlägt, und der Annahme, dass die Durchschnittsabmessungen von Pkws seitdem gestiegen sind, werden für die Simulation Verteilungsfunktionen gemäß Tabelle 3.1 angesetzt.

	Breite	Länge
Verteilung	normalverteilt	gleichverteilt
Bereich	$1,80 \pm 0,1m$	$4,0 - 5,0m$

Tabelle 3.1: Abmessungen eines Pkw

Die im vorliegenden Modell verwendete Wunschgeschwindigkeitsverteilung für Pkw basiert auf einer Untersuchung von Verkehrsbeeinflussungsanlagen auf der A44 aus dem Jahre 2000 ([20]):

Tempolimit	kein	120	100	80	60	≤ 50
Mittelwert μ	142	120	110	100	80	50
Standardabw. σ	20	20	18	15	15	10

Tabelle 3.2: Wunschgeschwindigkeitsverteilung für Pkw

Als endgültige Wunschgeschwindigkeit v_{Wunsch} eines Fahrzeugs ergibt sich dann aus der normalverteilten Wunschgeschwindigkeitsparameter w des Fahrers (vgl. Abschnitt 3.4.1) und den Werten aus Tabelle 3.2:

$$v_{Wunsch} = \mu + w * \sigma$$

Der Wunschgeschwindigkeitsparameter w des Fahrers wird während der gesamten Simulation beibehalten; eine dynamische Anpassung – beispielsweise infolge Ungeduld – ist nicht vorgesehen, da eine Kalibrierung solcher zeitlich veränderlicher Fahrerparameter sehr schwierig zu realisieren ist. Bei Änderung der erlaubten Höchstgeschwindigkeit wird die Wunschgeschwindigkeit des Fahrers neu bestimmt.

Das effektive Beschleunigungsvermögen eines Pkw ist stark vom Beschleunigungswillen des Fahrers abhängig. Im Gegensatz zu Lkw-Fahrern nutzen Fahrer von Personenkraftwagen die Leistungsfähigkeit ihres Fahrzeugs weniger stark aus [13]. Der Wert des Beschleunigungsvermögens wird in Anlehnung an [13] aus der Geschwindigkeit v (in m/s) wie folgt berechnet:

$$\text{beschleunigung}_{max,Pkw} = (0,2 + 0,8 \cdot \text{beschleunigungswille}) \cdot (8 - \sqrt{v})$$

Für das Abbremsen eines Pkws wird ein linearer Zusammenhang zwischen der maximalen Bremsverzögerung und der Geschwindigkeit angenommen; detaillierte Informationen zur Wahl

der zugehörigen Parameter finden sich unter [18].

$$bremsen_{max,Pkw} = -8 - 2 \cdot beschleunigungswille + 0,036 * v$$

Die oben angegebenen positiven und negativen Beschleunigungen gelten nur für die ebene Strecke. Bei Steigungs- oder Gefällestrecken muss der Einfluss der Erdbeschleunigung auf das Fahrzeug und sein Beschleunigungsverhalten beachtet werden. Aus der momentanen Steigung m , die dem Fahrzeug für die jeweilige Fahrspur bekannt ist, wird der tangentiale Anteil der Erdbeschleunigung, in Abbildung 3.11 mit g_t bezeichnet, wie folgt berechnet:

$$q_t = g \cdot \sin(\arctan y/x) = g \cdot \sin(\arctan m) \approx g \cdot m$$

Für Steigungen unter 10 Prozent liefert die obige Approximation hinreichend genaue Werte, der relative Fehler zwischen exakter und approximierter Lösung liegt unter einem halben Prozentpunkt. Für Steigungen bis 20 Prozent liegt der Fehler bei ca. 2 Prozent, und liefert somit für die Simulation eine noch ausreichende Genauigkeit.

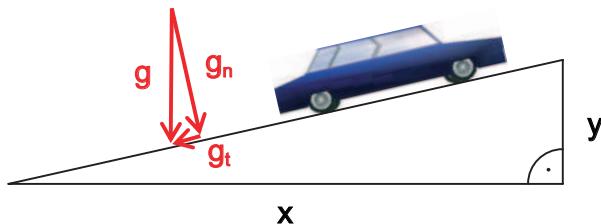


Bild 3.11: Einfluss der Hangabtriebsbeschleunigung

3.4.6 Lkw

Durch die Klasse *Lkw* werden im Klassenmodell alle Arten von Lastkraftwagen, Lastzügen, Reisebussen und vergleichbaren Großfahrzeugen mit einem Gewicht von mehr als 3,5 Tonnen zusammengefasst. Lkw unterscheiden sich aufgrund ihrer größeren Abmessungen und geringeren Maximalgeschwindigkeiten von Personenkraftwagen. Die Unterscheidung zwischen Pkw und Lkw ist für die Simulation deshalb wichtig, weil für Lkw besondere Regelungen der Straßenverkehrsordnung gelten (z. B. Geschwindigkeitsbegrenzungen, Überholverbote, Streckensperrungen); diese müssen durch die Klassenstruktur und die Verhaltensmodelle nachgebildet werden müssen.

Für die Abmessungen eines durchschnittlichen Lkw werden, entsprechend dem Vorgehen bei der Erzeugung der Pkw, zufallsverteilte Werte nach Tabelle 3.3 gewählt.

	Breite	Länge
Verteilung	gleichverteilt	gleichverteilt
Bereich	2,0 – 2,55m	13,0 – 15,0m

Tabelle 3.3: Abmessungen eines Lkw

Für die Wunschgeschwindigkeitsverteilung der Lkw werden ebenfalls die in [20] angegebenen Werte auf der A44 übernommen (Tabelle 3.4).

Tempolimit	kein	120	100	80	60	≤ 50
Mittelwert μ	92	91	90	85	75	50
Standardabw. σ	5	5	5	4	3	6

Tabelle 3.4: Wunschgeschwindigkeitsverteilung für Lkw

Bei Lastkraftwagen kann generell beobachtet werden, dass viele Fahrer das maximale Beschleunigungsvermögen ihrer Fahrzeuge stärker ausnutzen als Pkw-Fahrer ([10]). Zwischen Maximalbeschleunigung und Geschwindigkeit wird ein linearer Ansatz nach [14] verwendet, der den vom jeweiligen Fahrer abhängigen Beschleunigungswillen berücksichtigt:

$$\text{beschleunigung}_{max,Lkw} = 1.581 - 0.057 \cdot v + (0.6 - 0.01 \cdot v) \cdot \text{beschleunigungswille}$$

Das Bremsvermögen eines Lkw liegt bei modernen Bremsanlagen zwischen 5 und 7 m/s^2 ([18]); nach [14] wird eine lineare Funktion angesetzt:

$$\text{bremsen}_{max,Lkw} = -6 - 2 \cdot \text{beschleunigungswille} + 0,09 \cdot v$$

Insbesondere bei nicht ebenen Fahrspuren muss für Lkw der Einfluss der Erdbeschleunigung auf das Beschleunigungs- bzw. Bremsvermögen angerechnet werden (siehe Abschnitt 3.4.5). Die Möglichkeit einer Überhitzung der Bremsanlagen, wie sie bei längeren Gefälestrecken auftreten kann, wird hingegen im vorliegenden Modell nicht abgebildet.

3.5 Streckennetz

Das Straßen- oder Streckennetz liefert die Infrastruktur für die Fortbewegung der Fahrzeuge. Die Klasse *Netz* stellt einen Datencontainer dar, der alle für die Simulation relevanten topografischen und topologischen Informationen umfasst. Jeder Simulation kann zu einem gegebenen Zeitpunkt nur ein Netz zugeordnet sein; eine gleichzeitige Simulation mehrerer Netze in einer einzigen Simulationsinstanz ist nicht vorgesehen.

Zur Nachbildung von realen Stadtstraßen-, Bundesstraßen- und Autobahnnetzen in der Simulationswelt muß zunächst eine systematische Zerlegung dieser Netze in atomare Basiseinheiten erfolgen. Hierzu wird eine Grundmenge an physischen und logischen Fahrbahnelementen definiert, aus denen jedes beliebige Streckennetz nach dem Baukastenprinzip zusammengesetzt werden kann. Durch assoziative Verknüpfung dieser Bausteine – im Folgenden als Fahrspuren bezeichnet – entsteht ein gerichteter Graph, der die topologische Entsprechung des realen Streckennetzes bildet.

In diesem Graphen werden die Fahrspuren als Knoten, die Verbindungen zwischen den Fahrspuren als Kanten dargestellt. Alle Fahrspuren werden dabei unidirektional ausgelegt, entsprechen also den Einrichtungsfahrbahnen der realen Welt. Eine direkte Umsetzung einer in beide Richtungen befahrbaren Fahrspur ist im Modell nicht vorgesehen, kann aber dennoch durch zwei entgegengesetzt verlaufende und miteinander verknüpfte Fahrspuren erfolgen, was allerdings zusätzlich die Umsetzung einer Programmlogik zur Vermeidung frontaler Zusammenstöße erfordert.

Jedes Netz enthält Knoten, deren Kanten ausschließlich vom Knoten ausgehen (Knoten 1 und 2 in Abbildung 3.12). Diese Knoten werden als Quellen bezeichnet und dienen als Einspeisungspunkte für Fahrzeuge in das Netz. Analog dazu werden Knoten mit ausschließlich auf sie zulaufenden Kanten als Senken bezeichnet; Senken dienen dazu, die Fahrzeuge wieder aus dem Netz zu entfernen (Knoten 3 in Abbildung 3.12). Grundsätzlich muss jedes Fahrzeug immer an einer Quelle erzeugt und an einer Senke wieder entfernt werden.

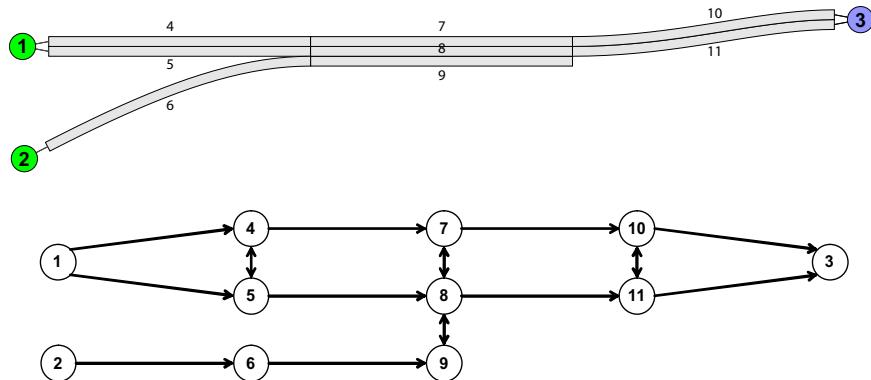


Bild 3.12: Eine Auffahrt in der Aufsicht (oben) und als Graph (unten)

Den Weg, den ein Fahrzeug zwischen einer Quelle und einer Senke zurücklegt, bezeichnet man als Strecke. Welcher Strecke ein Fahrzeug folgt, wird durch die Quelle zur Erzeugungszeit festgelegt; diese Streckenzuordnung bleibt bis zum Entfernen des Fahrzeuges erhalten. Die Strecke liefert allen Fahrzeugen auf Anfrage Informationen über zukünftige Abbiegeentscheidungen, Ausfahrttypen, Entfernungen bis zur nächsten Ausfahrt usw. Dadurch ist gewährleistet, dass sich jedes Fahrzeug frühzeitig auf bevorstehende Fahrtrichtungsentscheidungen einstellen kann.

Vor dem Start der Simulation muss eine Liste aller möglichen Strecken erzeugt und dem Netz zugeordnet werden. Auch wenn die Erzeugung von Strecken prinzipiell durch den Benutzer erfolgen könnte (sinnvoll zum Beispiel bei der Erstellung von ÖPNV-Routen), wird die Streckenerstellung im Allgemeinen automatisiert durch eine eigene Klasse *Router* durchgeführt. Für jedes Quelle-Senke-Paar ermittelt der Router die kürzestmögliche Strecke und fügt diese dem Netz hinzu.

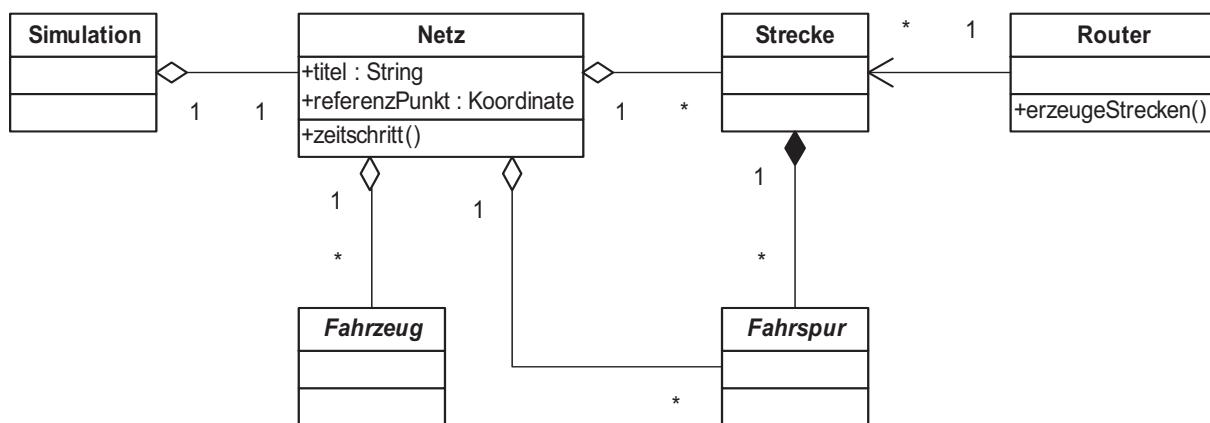


Bild 3.13: Klassendiagramm eines Streckennetzes

3.5.1 Fahrspuren

Fahrspuren repräsentieren Abschnitte einzelner Richtungsfahrbahnen. Je nach Art der Fahrspur kann diese dabei verschiedene geometrische Formen annehmen und/oder spezifische Aufgaben erfüllen. Wird ein ausreichend großer Satz an verschiedenen Fahrspurtypen vorgehalten, lassen sich beliebige Abschnitte einer gegebenen realen Straße in ein Modell aus einer oder mehreren Fahrspuren gleicher Semantik überführen.

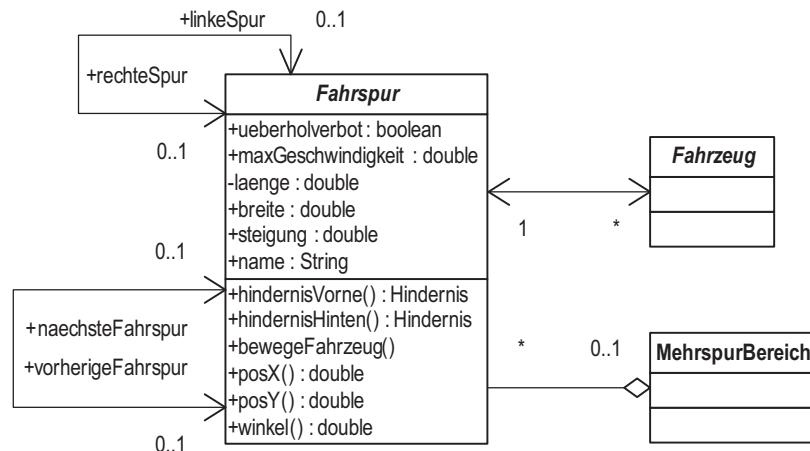


Bild 3.14: Klassendiagramm einer Fahrspur

Prinzipiell wird im vorliegenden Modell zwischen zwei verschiedene Arten von Fahrspuren unterschieden: Zum einen befahrbare Fahrspuren, die eine geometrische Ausdehnung besitzen – und damit am ehesten dem intuitiven Verständnis des Begriffs „Fahrspur“ entsprechen –, zum anderen infinitesimal kleine (d.h. punktförmige) Fahrspuren, die lediglich zur Umsetzung bestimmter semantischer Aspekte dienen. Zur letzteren Gruppe zählen beispielsweise Quellen und Senken, aber auch Zähl- und Haltepunkte sowie Abzweige zwischen mehreren Fahrspuren. Eine Übersicht über die implementierten Fahrspuren bietet Tabelle 3.5.

Fahrspurtyp	Geometrie	Beschreibung
Gerade	befahrbar	gerade Fahrspur
Kurve	befahrbar	gebogene Fahrspur (Bézier-Spline)
Vorlauf	befahrbar	virtuelle Fahrspur zur Harmonisierung neuer Fahrzeuge
Quelle	punktförmig	erzeugt neue Fahrzeuge
Senke	punktförmig	entfernt Fahrzeuge
Zählpunkt	punktförmig	registriert vorbeifahrende Fahrzeuge
Haltepunkt	punktförmig	Haltelinie an einer bevorrechtigten Straße
Stopschild	punktförmig	Haltelinie an einem Stoppschild
Signalgeber	punktförmig	Haltelinie für Lichtsignalanlagen
Abzweig	punktförmig	verbindet mehr als zwei Fahrspuren
Fußgängerüberweg	punktförmig	Haltelinie an Fußgängerüberwegen

Tabelle 3.5: Übersicht über alle implementierten Fahrspurtypen

Jede Fahrspur bietet Raum für eine beliebige, zur Simulationslaufzeit veränderliche Anzahl an Fahrzeugen. Über die Fahrspur können die Fahrzeuge Informationen über ihre Umgebung, also über benachbarte Fahrzeuge und sonstige Hindernisse, erfragen. Damit das Datenmodell zu jedem Zeitpunkt der Simulation konsistent ist und ein störungsfreier Verkehrsfluss gewährleistet

ist, muss durch die Klassen *Fahrspur* und *Fahrzeug* sichergestellt werden, dass die Assoziationen zwischen beiden Klassen immer gleichzeitig aktualisiert werden. Nur dann kann sichergestellt werden, dass dem Verhaltensmodell von der Fahrspur korrekte Hindernisinformationen gemeldet werden. Weitere Informationen zur Hindernisfindung werden in Kapitel 3.7 gegeben.

Um aus einzelnen Fahrspuren ein komplexes Netz aufbauen zu können, müssen die Fahrspuren untereinander verbunden werden können: In axialer Richtung kann an jedes Ende einer Fahrspur jeweils eine andere Fahrspur angeschlossen werden. Die in Fahrtrichtung vor der Fahrspur liegende Spur wird als „Vorgänger“, die in Fahrtrichtung folgende Fahrspur als „Nachfolger“ bezeichnet. Generell haben alle Fahrspuren sowohl mindestens einen Vorgänger als auch mindestens einen Nachfolger – Ausnahmen dieser Regel sind Quellen, Senken, Beschleunigungs- und Verzögerungsstreifen sowie Fahrstreifenreduktionen. Abzweige, Quellen und Senken sind die einzigen Fahrspur-Klassen, die mehr als einen Vorgänger bzw. Nachfolger besitzen dürfen.

Die ausschließliche Implementierung von Verbindungen in Fahrspurlängsrichtung ist nur so lange ausreichend, wie Spurwechsel auf seitlich benachbarte Fahrspuren ausgeschlossen werden können. Für den Verkehr auf Bundesautobahnen jedoch – die als mindestens zweistufige Fahrspuren ausgelegt sind – sind Spurwechsel zwingend erforderlich. Zu diesem Zweck können alle Fahrspuren beiderseits der Fahrbahnmitte mit Nachbarfahrspuren verbunden werden. Sobald eine Fahrspur mindestens eine Nachbarfahrspur besitzt, ist über die Spurwechselverhalten eine Spurwechselvorgang auf die benachbarten Spuren möglich.

Sobald mindestens zwei Fahrspuren seitlich miteinander verbunden sind, werden sie zu einem sogenannten Mehrspurbereich zusammengefasst. Ein Mehrspurbereich stellt informatisch gesehen eine Kompositionsklasse dar, die einen vereinfachten Zugriff auf benachbarte Fahrspuren erlaubt. Relevant wird diese Vereinfachung insbesondere in Hinblick auf das Laufzeitverhalten der Spurwechselverhalten, da diese oft und schnell auf die möglichen Nachbarfahrspuren eines Fahrzeugs zugreifen müssen. Die korrekte Zuordnung der Fahrspuren zu einem Mehrspurbereich muss bereits zur Erstellungszeit des Netzes sichergestellt sein. Dies ist daher Aufgabe der netzerstellenden Klasse – im Normalfall der grafischen Benutzungsumgebung und ihrer Werkzeuge.



Bild 3.15: Fahrbahnaufteilung als Abzweig (links) und als Mehrspurbereich (rechts)

3.5.2 Quellen

Quellen stellen Einspeisungspunkte für neue Fahrzeuge dar. In Abhängigkeit von der aktuellen Verkehrsstärke und der Aufsetzzeit des letzten Fahrzeugs entscheidet die Quelle, ob ein neues Fahrzeug erzeugt und aufgesetzt werden soll. Da der zeitliche und räumliche Abstand der Fahrzeuge, ihre Geschwindigkeit und die Fahrstreifenverteilung der Fahrzeuge einen großen Einfluss auf das Fahrverhalten und die maximal erreichbare Verkehrsstärke haben, muss bei der Implementierung der Quelle mit besonderer Sorgfalt vorgegangen werden. Bei der Entwicklung des Simulationsmodells hat sich gezeigt, wie wichtig und kritisch leistungsfähige Quellen für den gesamten Simulationsablauf sind. Daher wird an dieser Stelle ausführlich auf den Vorgang der Fahrzeugaufsetzung eingegangen.

Einer Quelle kann entweder eine einzelne Fahrspur oder ein aus mehreren Fahrspuren bestehender Mehrspurbereich zugeordnet sein. Bei jedem Simulationsschritt entscheidet die Quelle für jede angeschlossene Fahrspur, ob ein neues Fahrzeug aufgesetzt werden soll. Maßgebend für diese Entscheidung ist die Zeit, die seit dem Aufsetzen des letzten Fahrzeugs vergangen ist. Ein Fahrzeug kann erst dann aufgesetzt werden, wenn eine ausreichend große Zeitlücke zum Vordermann besteht. Die Größe der Zeitlücke ist dabei abhängig vom Sicherheitsbedürfnis des Fahrers und von der Verkehrsstärke, die der Quelle zugeordnet ist.

Verschiedene Zeitlückenverteilungs-Funktionen kommen für den Aufsetzmechanismus in Frage; für eine erste Untersuchung wurde der Ansatz von Dawson ([32]) implementiert, der, aufbauend auf den Arbeiten von Schuhl ([103]) und Leutzbach ([71]), eine sogenannte Hyper-Erlang-Verteilung vorschlägt. Diese besteht aus zwei superponierten Einzelverteilungen der Zeitlücken: einer verschobenen negativen Exponentialfunktion für den freien Verkehr, und einer verschobenen Erlangverteilung für Fahrzeuge im gebundenen Verkehr. Unter Verwendung des Hyper-Erlang-Ansatzes wurde von Großmann ([54]) eine allgemeine Verteilungsfunktion aufgestellt, die eine gute Näherung an real gemessene Zeitlückenverteilungen aufweist.

Die mit dieser Verteilungsfunktion erreichten Fahrzeugfolgeabstände entsprechen definitionsgemäß den statistisch zu erwartenden Abständen; allerdings stimmen sich nicht mit den vom Fahrzeugfolgemodell angestrebten Abständen überein. Wird der vom Fahrzeug gewünschte Folgeabstand durch eine zu klein dimensionierte Zeitlücke stark unterschritten, kommt es zu Bremsmanövern infolge zu kleiner Sicherheitsabstände. Durch Aufschauklungseffekte entstehen bei hoher Verkehrsstärke bereits nach kurzer Simulationszeit starke Rückstauungen im Bereich der Quelle. Die vorgegebene Verkehrsstärke wird dabei stets deutlich unterschritten. Auch durch eine gleiche Aufsetzgeschwindigkeit aller Fahrzeuge lässt sich dieser Effekt nicht verhindern. Die höchste stabil zu haltende Verkehrsstärke liegt bei etwa 1500 Fz/h, was für praxisrelevante Untersuchungen deutlich zu niedrig ist. Die Zeitlücken-Verteilungsfunktion muss deshalb so angepasst werden, dass die vom Abstandsmodell angestrebten Sicherheitsabstände bereits beim Aufsetzen der Fahrzeuge eingehalten werden.

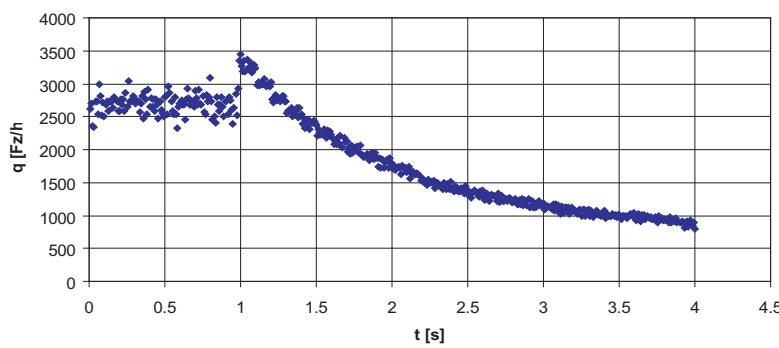


Bild 3.16: Erreichbare Verkehrsstärke in Abhängigkeit der Aufsetz-Zeitlücke

Der Einfluss der Zeitlücke auf die erreichbare Kapazität wurde in einem Simulationslauf untersucht. Hierfür wurde an einer einspurigen Quelle die Folgezeitlücke t langsam abgesenkt. Bis zu einer Folgezeitlücke von ca. 1,0-1,2 Sekunden bleibt der Verkehrsstrom stabil (Abbildung 3.16); wird diese Zeitlücke unterschritten, kann der beschriebene Selbstverstärkungseffekt beobachtet werden – der Verkehr bricht zusammen. Um dieses Problem klein zu halten, sind zu kleine Fahrzeugfolgeabstände zu vermeiden. Als pragmatischer Ansatz wurde deshalb für jeden Fahrstreifen eine Gleichverteilungsfunktion gewählt, die jedem Fahrstreifen i bei einer gegebenen

Verkehrsstärke q_i des Fahrstreifens eine feste Zeitlücke t_i zuordnet:

$$t_i = \frac{3600}{q_i} \quad [s] \quad (3.1)$$

Aus Abbildung 3.16 lässt sich außerdem ablesen, dass durch diesen Ansatz einer Gleichverteilung maximal eine Verkehrsstärke von ca. 3000 Fz/h erreicht werden kann, was einer Zeitlücke von 1,2 Sekunden entspricht. Dieser Wert entspricht in etwa den kleinsten zu erwartenden Zeitlücken in der Realität.

Sind einer Quelle zwei oder mehr Fahrspuren zugeordnet, wird die Gesamtverkehrsstärke q_{ges} auf alle Fahrspuren verteilt. Auch hier hat sich in den Versuchsläufen eine gleichmäßige Verteilung der Fahrzeuge bewährt. Für n Fahrstreifen ergibt sich für jede Fahrspur eine Verkehrsstärke q_i :

$$q_i = \frac{q_{ges}}{n} \quad [Fz/h] \quad (3.2)$$

In der Standardimplementierung werden von der Quelle nur die Fahrzeugtypen Pkw und Lkw generiert. Alle Lkw werden generell auf die rechte Fahrspur gesetzt, da ansonsten durch das Rechtsfahrgebot für langsamere Fahrzeuge unnötige Spurwechsel initiiert werden. Für alle übrigen Fahrspuren eines Mehrspurbereiches werden ausschließlich Pkw erzeugt.

Die Wahrscheinlichkeit $p_{i,Lkw}$, dass es sich bei einem neu aufgesetzten Fahrzeug auf der i -ten Fahrspur ($i = 0$ entspricht der rechten Fahrspur) um einen Lkw handelt, ergibt sich aus dem prozentualen Lkw-Anteil k_{Lkw} und der Gesamtzahl n der Fahrspuren zu:

$$p_{i,Lkw} = \begin{cases} n \cdot k_{Lkw} & \text{wenn } i = 0 \\ 0 & \text{wenn } i > 0 \end{cases} \quad (3.3)$$

Analog dazu ergibt sich die Wahrscheinlichkeit, dass ein neuer Pkw erzeugt wird, zu:

$$p_{i,Pkw} = \begin{cases} 1 - n \cdot k_{Lkw} & \text{wenn } i = 0 \\ 1 & \text{wenn } i > 0 \end{cases} \quad (3.4)$$

Beim Aufsetzen erhalten alle Fahrzeuge eine Startgeschwindigkeit, die – soweit möglich – der Maximalgeschwindigkeit des Fahrzeugs entspricht. Befindet sich vor dem aufgesetzten Fahrzeug bereits ein langsameres Fahrzeug auf der selben Fahrspur, wird dessen aktuelle Geschwindigkeit verwendet. Hierdurch wird verhindert, dass Fahrzeuge direkt nach dem Aufsetzen Bremsmanöver einleiten müssen, um einer Kollision mit dem Vordermann zu entgehen.

Ein plötzliches Abbremsen des neu erzeugten Fahrzeugs kann allerdings auch dann notwendig werden, wenn die Netto-Weglücke zwischen dem Fahrzeug und seinem Vordermann zu klein ausfällt. Daher wurde zusätzlich zur Mindestzeitlücke eine Sicherheitsabfrage implementiert, die ein Fahrzeug erst dann aufsetzt, wenn eine gewisse Mindestdistanz zwischen beiden Fahrzeugen vorhanden ist. Aus pragmatischen Gründen wurde hierfür die Entfernung bx des Wiedemannschen Modells gewählt (siehe Kapitel 4.2.3).

Jedes Fahrzeug, das an einer Quelle erzeugt wird, bekommt zufällig eine der Strecken zugeordnet, die an der Quelle beginnen. Die Wahrscheinlichkeit, dass ein Fahrzeug einer bestimmten Strecke zugeordnet wird, entspricht dem Anteil der Verkehrsstärke der Strecke an der Gesamtverkehrsstärke. Während der Simulation können die Strecken bzw. deren Verkehrsstärken geändert werden; deratige Änderungen werden von der Quelle automatisch berücksichtigt.

Wie beschrieben lag das Hauptaugenmerk bei der Entwicklung der Quelle auf einer Maximierung der aufbringbaren Verkehrsstärke: die Fahrzeuge werden hierzu mit weitgehend homogenen Geschwindigkeiten und Folgeabständen aufgesetzt. Damit sich dennoch eine realitätsnahe Verteilung der Geschwindigkeiten und Fahrstreifen einstellen kann, wird zwischen die Quelle und ihre Anschlussfahrspuren eine weitere Fahrspur, der sogenannte Vorlauf, eingefügt, der für eine Harmonisierung des Fahrzeugstroms sorgt und nachfolgend erläutert wird.

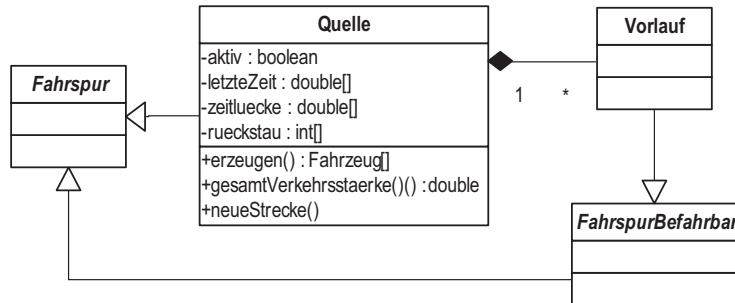


Bild 3.17: Klassendiagramm von einer Quelle und ihrem Vorlauf

3.5.3 Vorlauf

Ein Vorlauf ist eine gedachte gerade Fahrspur mit einer definierbaren Länge, auf der die Fahrzeuge direkt nach ihrer Erzeugung fahren. Erst wenn sie das Ende der Vorlaufstrecke erreicht haben, werden sie auf das eigentliche (sichtbare) Straßennetz aufgesetzt. Ziel dieser zusätzlichen Fahrspur ist die Harmonisierung der Fahrzeuggeschwindigkeiten und der Fahrspurverteilungen, bis sich ein ausreichend stationärer Fahrzustand eingestellt hat.

Vorläufe sind hypothetische Fahrspuren, d.h. sie besitzen keine Entsprechungen in der Realität und können nicht vom Benutzer angezeigt oder editiert werden. Sie dienen einzig und allein zur Sicherstellung eines gleichmäßigen Verkehrsstroms. Jeder Quelle ist für jede angeschlossene Fahrspur je ein Vorlauf zugeordnet; besitzt eine Quelle zwei oder mehr Vorläufe, werden diese zu einem Mehrspurbereich zusammengefasst. Alle Spurwechsel, die ein Fahrzeug direkt nach dem Aufsetzen durchführen muss, um seine gewünschte Fahrspur zu erreichen, sollten dabei noch auf dem Vorlauf erfolgen, um so nicht den Verkehr auf der eigentlichen Messstrecke zu beeinflussen.

Die gewünschte Harmonisierung erfolgt in zwei Stufen: Auf den ersten 200 Metern der Vorlaufstrecke werden zunächst nur die Geschwindigkeiten der Fahrzeuge aneinander angeglichen; Spurwechsel in diesem Bereich werden unterbunden. In Versuchen hat sich gezeigt, dass diese Distanz im Normalfall ausreicht, um eine ausreichende Angleichung der Fahrzeuge an ihre bevorzugte Geschwindigkeit zu erreichen. Erst nach 200 Metern werden in einer zweiten Stufe die Spurwechsel gestattet, so dass sich die Fahrzeuge auf ihre bevorzugten Fahrspuren verteilen können. Eine Anpassung der Geschwindigkeiten kann in diesen Bereichen natürlich auch weiterhin erfolgen, die Fahrzeuge bewegen sich unter Nutzung der normalen Verhaltensmodelle. Die Gesamtlänge des Vorlaufs sollte so dimensioniert sein, dass sich beim Verlassen des Vorlaufs alle Fahrzeuge in einem realitätsnahen Fahrzustand befinden.

Für die Vorlaufstrecke ist bei Quellen mit maximal zwei Fahrstreifen eine Mindestlänge von 1000 Metern anzusetzen. Sind mehr Fahrstreifen vorhanden, sollte die Vorlauflänge pro Fahrspur um 500 Meter verlängert werden, um Raum für die eventuell erforderlich werdenden zusätzlichen

Spurwechsel zu bieten. Eine darüber hinaus gehende Verlängerung der Vorlaufstrecken bringt keine spürbare Verbesserung mit sich.

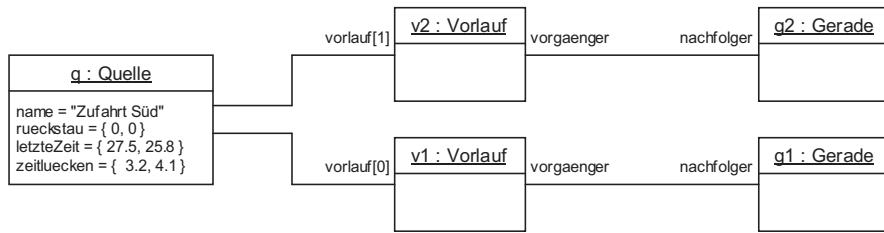


Bild 3.18: Objektdiagramm einer Quelle mit zweispurigem Vorlauf

Die Funktion von Nachlauffahrspuren, die – entsprechend dem Vorlauf am Fahrbahnbeginn – für realistische Randbedingungen am Fahrbahnende sorgen, wird von den Senken übernommen.

3.5.4 Senken

Jeder Autofahrer verfolgt stets eine feste Absicht: zu gegebener Zeit sein Fahrziel zu erreichen. Dieses Fahrtziel ist in der Simulation die Senke. Sie stellt den Punkt dar, an dem die Fahrzeuge das Simulationsnetz verlassen; außerdem bilden Senken die Endpunkte aller Strecken. Jede Fahrspur, die weder eine Senke noch eine andere Fahrspur als Nachfolger hat, stellt eine Sackgasse dar und wird von keinem Fahrzeug befahren, da der Routing-Algorithmus diese Fahrspuren nicht in Strecken einbindet.

Hat ein Fahrzeug eine Senke erreicht, ist seine Fahrt beendet. Das Fahrzeug wird als gelöscht gekennzeichnet und aus der Simulation entfernt. Alle Ressourcen, die das Fahrzeug zur Laufzeit belegt hat, müssen nun entfernt werden, um ein Überlaufen des Arbeitsspeichers zu verhindern. Aufgezeichnet werden lediglich seine Reisezeit, die in einer zentralen Tabelle abgelegt wird, sowie eventuell noch bestehende Messwerte an Zählpunkten oder aus Streckenmessungen. Alle anderen Referenzen des Fahrzeugs werden gelöscht und können auch nicht wiederhergestellt werden.

Sobald ein Fahrzeug gelöscht wurde, stellt es kein Hindernis für den nachfolgenden Verkehr mehr dar. Das letzte auf der Strecke befindliche Fahrzeug wird daraufhin beschleunigen, um seine Wunschgeschwindigkeit zu erreichen. Durch diesen Effekt würde die Durchschnittsgeschwindigkeit verfälscht, und auf allen Fahrspuren im Bereich vor den Senken zu hohe durchschnittliche Geschwindigkeiten gemessen. Um dies zu unterbinden, täuscht die Senke dem nachfolgenden Verkehr vor, dass der Verkehrsstrom wie zuvor weiterfliesst und das soeben entfernte Fahrzeug noch immer vorhanden ist. Dazu wird an jeder Senke die Geschwindigkeit eines gelöschten Fahrzeugs und der Zeitpunkt der Entfernung gespeichert. Aus diesen beiden Informationen kann dann die Position extrapoliert werden, die ein Fahrzeug einnehmen würde, wenn es seinen Fahrzustand beibehalten hätte und nicht gelöscht worden wäre.

Der Nettoabstand $dx_{virtuell}$ zum virtuellen Fahrzeug ergibt sich aus dem Abstand zur Quelle dx_{real} , der Geschwindigkeit v des entfernten Fahrzeugs zum Zeitpunkt des Löschens und der seither vergangenen Zeit Δt :

$$dx_{virtuell} = dx_{real} + v \cdot \Delta t - \text{länge}/2 \quad (3.5)$$

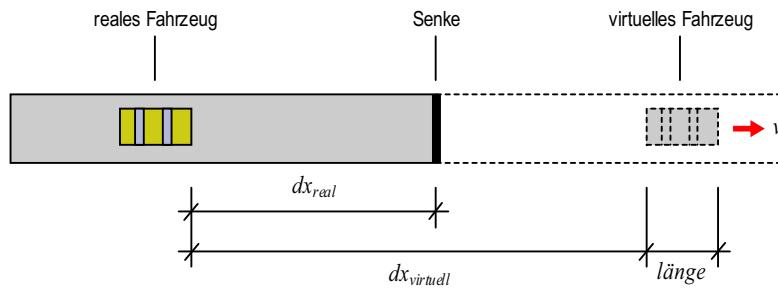


Bild 3.19: Ein virtuelles Fahrzeug an einer Senke

Die Senke meldet dem letzten Fahrzeug auf Anfrage ein Hindernis zurück, das sich in einer Entfernung $dx_{virtuell}$ mit einer Geschwindigkeit v bewegt. Durch dieses virtuelle Hindernis wird das normale Fahrzeugfolgemodell aktiviert und die Geschwindigkeit des letzten Fahrzeuges auf realistische Werte gedrosselt.

3.5.5 Mehrspurbereiche

Mehrspurbereiche sind Gruppen von lateral benachbarten Fahrspuren, zwischen denen Fahrzeuge Spurwechsel durchführen können. Jede Fahrspur, die mit einer rechts oder links angrenzenden Fahrspur verbunden ist, ist definitionsgemäß Teil eines Mehrspurbereiches. Diese Zuordnung ermöglicht es, auf relativ einfacher Weise Informationen über benachbarte Fahrspuren zu erhalten, was insbesondere für die Spurwechselverhalten von Interesse ist. Auch bei der Netzerstellung, der Grafikdarstellung und der Streckenfindung erleichtert die Klasse *Mehrspurbereich* den Zugriff auf benachbarte Fahrspuren.

Damit ein Spurwechsel zwischen zwei Fahrspuren durchgeführt werden kann, muss zwischen ihnen eine räumliche Verbindung bestehen. Nur so lange sich zwei Fahrspuren direkt berühren und sie in Form und Länge miteinander vergleichbar sind, ist ein Spurwechsel sinnvoll. Daraus wird festgelegt, dass innerhalb eines Mehrspurbereiches ausschließlich Fahrspuren gleichen Typs zusammengefasst werden können (vgl. Abbildung 3.20 links); eine Mischung von verschiedenen Fahrspurtypen (z. B. Geraden und Kurven, mittlere Abbildung) oder stark in der Länge differierenden Fahrspuren (rechte Abbildung) ist nicht gestattet.

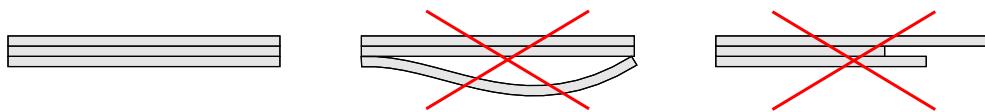


Bild 3.20: Mehrspurbereiche müssen in Typ und Länge übereinstimmen

Innerhalb eines Mehrspurbereichs werden die Fahrspuren ihrer Lage nach von rechts nach links geordnet und durchnummieriert. Die rechte Fahrspur eines Mehrspurbereichs erhält stets die Nummer 0, die am weitesten links gelegene Spur die Nummer $n - 1$, wobei n die Anzahl der Fahrstreifen darstellt. Die Assoziationen zwischen Fahrspuren und Mehrspurbereichen werden in Abbildung 3.21 dargestellt.

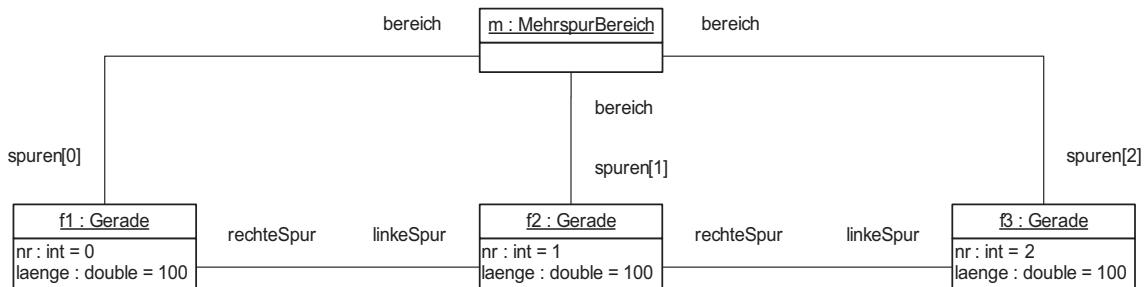


Bild 3.21: Objektdiagramm eines dreispurigen Mehrspurbereichs

3.6 Strecken

Damit sich ein Fahrzeug zielgerichtet durch ein Straßennetz bewegen kann, werden an allen Abzweigmöglichkeiten Fahrtrichtungsentscheidungen erforderlich. Beispielweise muss entschieden werden, ob das Fahrzeug an der nächsten Ausfahrt die Autobahn verlassen wird, oder noch länger auf der aktuellen Spur bleiben kann. Diese und ähnliche Entscheidungen können von der FFE entweder zufällig über eine Abbiegewahrscheinlichkeit getroffen werden (dieses Verfahren wird u.a. von VISSIM verwendet [39]), oder aber vorab festgelegt werden. Dem Fahrzeug wird hierfür eine feste Strecke zugewiesen, aus der die Abbiegevorgänge abgeleitet werden können. Für das hier vorgestellte Modell wird das letztere Verfahren verwendet.

Eine Strecke wird im Wesentlichen als Liste von Fahrspuren realisiert, auf denen sich FFE bewegen. Liegt eine Fahrspur nicht auf der dem Fahrzeug zugewiesenen Strecke, ist es dem Fahrzeug nicht erlaubt diese zu befahren. Fahrzeuge, die sich nicht mehr auf ihrer Strecke befinden, sind offensichtlich vom geplanten Weg abgekommen (z. B. weil sie eine Ausfahrt verpasst haben), und werden entweder aus der Simulation entfernt oder auf eine neue, alternative Strecke gesetzt, die der aktuellen Fahrspur entspricht. Nur wenn sich mindestens eine der Fahrspuren in der direkten Nachbarschaft des Fahrzeugs auf der Strecke dieses Fahrzeugs befindet, kann das Spurwechselverhalten eine Entscheidung über die gewünschte Fahrtrichtung treffen.

Um Spurwechselverhalten auf Mehrspurbereichen zu jedem Zeitpunkt mit Informationen über den weiteren Streckenverlauf zu versorgen, die für taktische Entscheidungen notwendig sind, können über die Strecke die folgenden Daten abgefragt werden:

- **Entfernung bis Spurende:** Gibt an, wie viele Meter noch zwischen dem Fahrzeug und dem Ende des aktuellen Fahrstreifens liegen.
- **Nächste Ausfahrt:** Gibt an, welche Ausfahrt für dieses Fahrzeug die nächste Ausfahrt ist, an der es ausfahren muss.
- **Entfernung nächste Ausfahrt:** Gibt an, wie viele Meter das Fahrzeug noch auf der aktuellen Fahrspur bleiben kann, bis die nächste Ausfahrt beginnt, an der es die Fahrspur verlassen muss.
- **Verhalten an nächster Verzweigung:** Gibt an, wie sich das Fahrzeug an der nächsten Einfahrt, Ausfahrt oder Verflechtungsstrecke verhalten wird, ob es einfahren, ausfahren oder auf seiner jetzigen Fahrspur bleiben wird, und wie viele Fahrspuren das Fahrzeug wechseln muss, um auf seiner Strecke zu bleiben.

Diese für das Spurwechselverhalten essentiellen Daten müssen während der Simulation ständig abgefragt werden, und sind ein wesentlicher Flaschenhals für die Performanz der Simulation. Da das Routing der Strecken jedoch weitgehend statisch umgesetzt wurde, d.h. während der Laufzeit einer Simulation nur selten Änderungen am Streckenverlauf erfolgen, kann die gesamte Berechnung der Streckeninformationen bereits vor Beginn der Simulation durchgeführt werden. Die ermittelten Daten werden für die jeweilige Strecke abgespeichert und können für jede Fahrspur separat abgerufen werden. Bei nur geringfügig ansteigendem Speicherbedarf der Strecken kann so ein deutlicher Performanzgewinn erzielt werden.

Die Erstellung der Strecken erfolgt automatisiert durch einen Routing-Algorithmus aus den vorhandenen Netzdaten. Die Klasse *Router* durchläuft dazu alle möglichen Kombinationen von Quellen und Senken, und ermittelt für jedes verbundene Quelle-Senke-Paar eine passende Strecke. Betrachtet werden nur Strecken, die den kürzesten Weg zwischen einer Quelle und einer Senke bilden. Zur Berechnung wird aus den Fahrspur-Informationen (Länge bzw. Verknüpfung mit anderen Fahrspuren) eine Adjazenzmatrix eines nach Entfernung gewichteten, gerichteten Graphen erstellt. Mit Hilfe dieser Matrix wird vom Routing-Algorithmus eine Liste von Strecken erstellt; implementiert wurden die Routing-Algorithmen von Floyd ([43]) und Dijkstra ([34]).

3.7 Hindernisse

Um die aktuelle Verkehrssituation korrekt einschätzen und stets sichere Fahraktionen durchführen zu können, ist die rechtzeitige Erkennung potentieller Hindernisse unerlässlich. Der menschliche Fahrer regelt dies durch ständige visuelle Beobachtung der Umgebung seines Fahrzeuges. Für die Simulation lässt sich dieses Verhalten jedoch nur mit großem Aufwand realisieren: Zu viele Umgebungsfahrzeuge müssten beobachtet und hinsichtlich ihrer Gefährlichkeit bewertet werden. Da dies gleichzeitig für Hunderte oder Tausende Fahrzeuge für jeden Zeitschritt durchzuführen wäre, würde der entstehende Zeitaufwand exponentiell anwachsen. Stattdessen wird ein vereinfachtes Verfahren zum Erkennen von Hindernissen angewendet.

Hierbei müssen zunächst die möglichen Interaktionspartner identifiziert werden, die potentiell als Hindernisse in Betracht kommen. Für Fahrzeuge sind dies in erster Linie die jeweiligen Vorderfahrzeuge. An ihnen muss sich der Fahrer (oder das Fahrverhalten) orientieren, um zu geringe Folgeabstände oder sogar Kollisionen auszuschließen. Solange sich ein Fahrzeug auf einer einstreifigen Fahrspur befindet, ist der Blick nach vorne die einzige relevante Blickrichtung. Erst auf mehrspurigen Strecken wird ein Blick nach hinten erforderlich, um gegebenenfalls Spurwechsel ohne Gefährdung des Hintermanns durchführen zu können.

Da ein Fahrzeug keine direkte Kenntnis über die übrigen Fahrzeuge in seiner Umgebung besitzt, fragt es zunächst seine eigene Fahrspur nach Hindernisse in Fahrtrichtung bzw. gegen die Fahrtrichtung. Die Fahrspur, die – wie zuvor beschrieben – über eine vollständige Liste aller auf ihr befindlichen Fahrzeuge verfügt, überprüft dann, ob eines dieser Fahrzeuge ein Hindernis für das nachfragende Fahrzeug darstellt. Sollten mehrere Fahrzeuge hierfür in Frage kommen, wird das nächste Fahrzeug zurückgeliefert. Sobald eine Fahrspur ein Hindernis gefunden hat, wird dies dem Fahrzeug gemeldet.

Wurde auf einer Fahrspur kein Hindernis entdeckt, heißt dies nicht, dass keines vorhanden ist. Vielmehr besteht immer noch die Möglichkeit, dass eine der sich anschließenden Fahrspuren ein Fahrzeug enthält, das für das betrachtete Fahrzeug zum Hindernis werden kann. Die Anfrage

an die Fahrspur muss also rekursiv an die nachfolgenden Fahrspuren weitergereicht werden – solange, bis entweder ein Hindernis gefunden wurde oder die maximale Sichtweite überschritten wurde. Diese Begrenzung der Suche ist notwendig, da es ansonsten unter Umständen – spätestens aber bei ringförmigen Streckennetzen – zu einem Überlauf des Methodenstapels kommt. Als Grenzwert wurde dabei eine maximale Sichtweite von einem Kilometer angesetzt, da das effektive Sichtvermögen eines menschlichen Fahrers diesen Wert auch bei idealen Bedingungen kaum überschreiten dürfte.

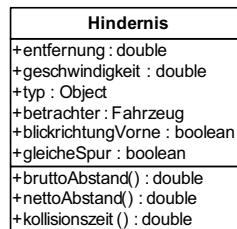


Bild 3.22: Klassendiagramm der Klasse Hindernis

Mit der Möglichkeit eines Spurwechsels ergeben sich für Mehrspurbereiche neue Interaktionen, die bei Spurwechselentscheidungen berücksichtigt werden müssen. Ein Spurwechsel kann immer nur dann durchgeführt werden, wenn sowohl zum potentiellen Vorder- als auch zum Hinterfahrzeug ein ausreichend großer Sicherheitsabstand bzw. eine ausreichend große Zeitlücke vorliegt. Hierdurch erhöht sich die Zahl der möglichen Interaktionspartner auf insgesamt sechs. Abbildung 3.23 zeigt ein Fahrzeug mit einer maximalen Anzahl an direkt benachbarten Fahrzeugen.

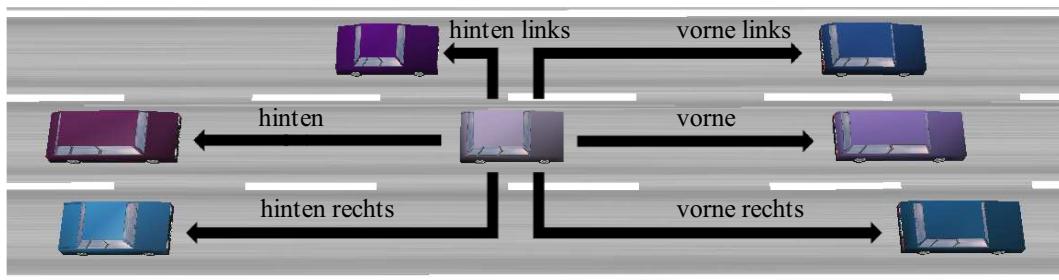


Bild 3.23: Die sechs direkten Interaktionspartner eines Fahrzeugs

Um die Interaktionspartner auf der Nachbarspur zu erkennen, kann nach dem gleichen Prinzip wie bei der beschriebenen einspurigen Suche verfahren werden: Es wird so lange auf den Fahrspuren in Fahrtrichtung gesucht, bis entweder ein Hindernis gefunden wurde oder eine vorgegebene Sichtweite überschritten wurde. Der hierfür notwendige Algorithmus wurde bereits bei der einspurigen Suche beschrieben. Es liegt also nahe, das obige Verfahren auch für die seitliche Hindernissuche zu verwenden.

Jedes Fahrzeug verfügt über die Fähigkeit, das nächste Hindernis in Fahrtrichtung und entgegen der Fahrtrichtung zu bestimmen. Um auf der Nebenspur ein Hindernis zu erkennen, müsste das Fahrzeug folglich kurzzeitig auf die linke Spur wechseln, dort die Distanz zum Hindernis bestimmen, um dann mit dieser Information auf seine ursprüngliche Fahrspur zurückzukehren.

Um dieses Versetzen des Fahrzeugs zu vermeiden wird jedem Fahrzeug ein so genanntes Dummy-Fahrzeuge zugeordnet. Dummy-Fahrzeuge sind virtuelle Fahrzeuge, die eine Kopie ihrer Besitzerfahrzeuge darstellen. Position, Geschwindigkeit, Typ und Abmessungen jedes Dummy-Fahrzeugs stimmen mit dem Referenzfahrzeug überein. Will ein Fahrzeug Informationen über

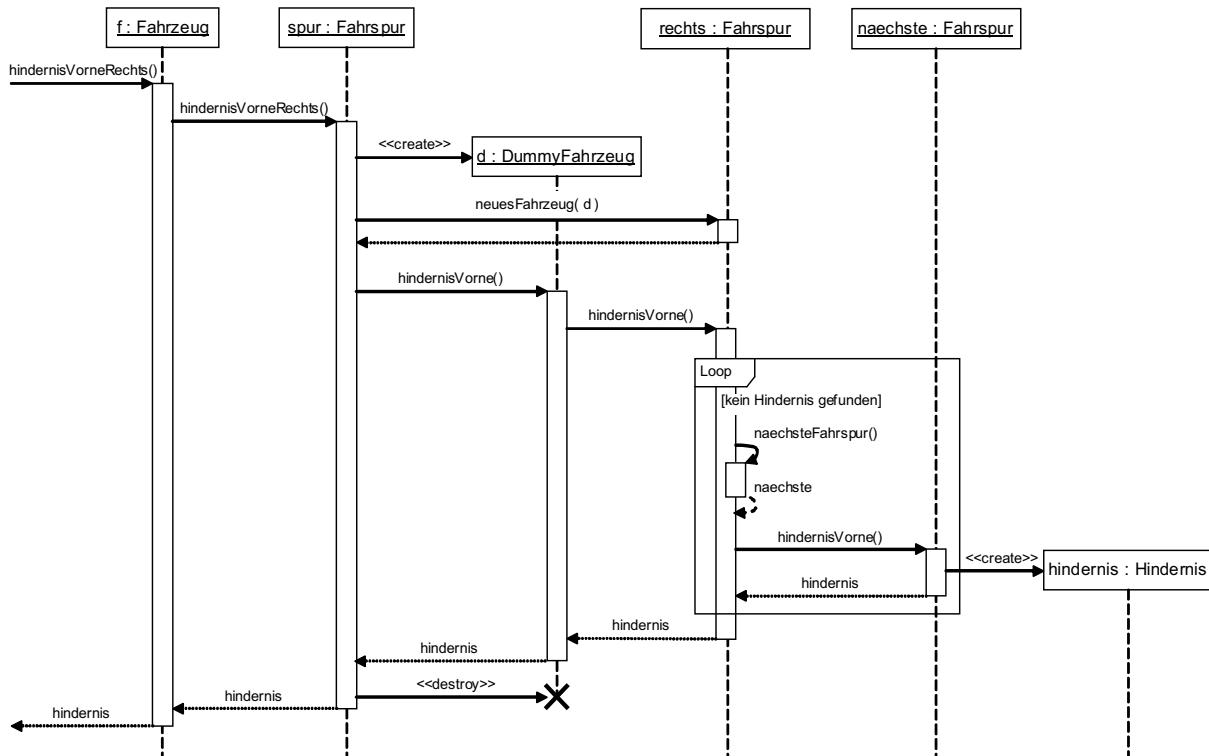


Bild 3.24: Sequenzdiagramm zur Hindernissuche vorne rechts

seine Nachbarspur erhalten, platziert es sein Dummy-Fahrzeug auf der zu betrachtenden Spur und ruft dessen Methode zur Hindernissuche auf. Nach erfolgreicher Bestimmung des Hindernisses wird das Dummy-Fahrzeug wieder von der Fahrspur entfernt, bevor es für die nächste Hindernisbestimmung verwendet wird.

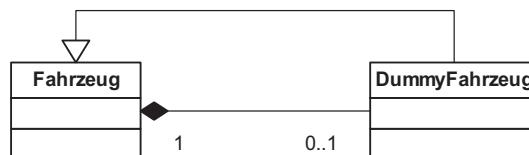


Bild 3.25: Klassendiagramm eines Fahrzeugs mit seinem Dummyfahrzeug

In der realen Welt stellen Fahrzeuge aber nicht die einzigen Hindernisse dar, denen ein Fahrer Beachtung schenken muss. Auch temporäre Hindernisse, wie Lichtsignalanlagen oder Haltepunkte an vorfahrtberechtigten Straßen, können zu einer Verlangsamung oder dem vollständigen Halt eines Fahrzeuges führen. Zur Umsetzung dieser Hindernisse in der Simulation kann die bereits beschriebene Hindernissuche für Fahrzeuge ebenfalls genutzt werden. Wie oben dargelegt, ist die Suche nach einem Hindernis Aufgabe der jeweiligen Fahrspuren. Dabei kann eine Fahrspur auch selbst als Hindernis aktiv werden und die Straße für das anfragende Fahrzeug blockieren. Tabelle 3.6 gibt eine Übersicht über Fahrspuren, die zu einem Hindernis werden können:

Die Information, ob und in welcher Entfernung ein Hindernis in den sechs Betrachtungsrichtungen (vgl. Abbildung 3.23) vorhanden ist, wird von den Abstands- und Spurwechselmodellen bestimmt und in die Entscheidungsfindung eingebaut. Welche Richtungen dabei betrachtet werden, ist vom jeweiligen Verhaltensmodell abhängig und kann den weiteren Beschreibungen in

Fahrspurtyp	Beschreibung
Fußgängerüberweg	blockiert die Straße so lange, bis alle eventuell vorhandenen Fußgänger einen vorgegebenen Fahrbahnbereich verlassen haben.
Haltepunkt	meldet sich selbst so lange als Hindernis, bis im betrachteten Kontrollbereich der bevorrechtigten Straße kein potentielles Hindernis existiert.
Parkbox	sperrt solange die Fahrspur, bis ein ein- oder ausparkendes Fahrzeug den Parkvorgang abgeschlossen hat.
Signalgeber	gibt ein Hindernis zurück, so lange die zugehörige Signalgruppe auf Rot steht.
Stopschild	gibt sich so lange als Hindernis aus, bis ein Fahrzeug knapp davor zum Stehen kommt. Schaltet danach so lange auf frei, bis das Fahrzeug die Haltelinie überfahren hat.

Tabelle 3.6: Fahrspuren, die potentiell ein Hindernis darstellen können

Kapitel 4 entnommen werden. Da die Verhaltensmodelle ggf. zu jedem Zeitschritt Informationen über die Hindernissituation benötigen und daher die entsprechenden Methoden sehr häufig aufzurufen sind, werden die Hindernisinformationen in einem separaten Berechnungsschritt vor der eigentlichen Fahrzeugbewegung ermittelt und zwischengespeichert. Dabei können Symmetrieffekte ausgenutzt werden, um die Laufzeit zu verringern (besitzt z. B. ein Fahrzeug A ein Fahrzeug B als Hindernis vorne, so stellt Fahrzeug A gleichzeitig das Hindernis hinten für Fahrzeug B dar). Durch Vermeidung redundanter Berechnungen lässt sich die Geschwindigkeit der Hindernissuche so deutlich erhöhen.

3.8 Simulationsauswertungen

Verkehrssimulation ist kein Selbstzweck, sondern sie dient stets der Untersuchung und Bewertung von Verkehrsszenarien. Hierfür ist es erforderlich, während des Simulationslaufes ausreichende Datensätze über Simulationsverlauf aufzunehmen und für eine spätere Auswertung bereitzustellen. Es wurden verschiedene Auswertungswerkzeuge in das Simulationsmodell integriert, die sich im Wesentlichen an den in der Verkehrstechnik üblichen Messverfahren orientieren. Im Folgenden wird eine kurze Übersicht über die Messmethoden und ihre Realisierung im Objektmodell gegeben.

3.8.1 Lokale Messungen

Induktionsschleifen, Videoaufzeichnungen und Verkehrszählungen sind übliche Methoden, um Daten über Verkehrsströme an einem Messquerschnitt zu erlangen. Da sich dieser Messquerschnitt während der Messdauer nicht bewegt, spricht man von lokaler Messung (vgl. Abbildung 2.1). Aus Messungen an lokalen Messquerschnitten lassen sich Informationen über die Verkehrsstärke, die Fahrstreifenverteilungen und die mittlere lokale Geschwindigkeit ermitteln (Kapitel 2.3.7).

Um lokale Messungen in der Simulation zu realisieren, wird eine Klasse implementiert, die jedes

Fahrzeug, das einen bestimmten Messquerschnitt passiert, automatisch registriert und seine Bewegungsdaten intern ablegt. Hierfür bietet es sich an, das Baukastenprinzip der Fahrspur-Klassen auszunutzen und den neuen Messquerschnitt zwischen zwei existierende Fahrspuren zu platzieren. Abbildung 3.26 zeigt, wie ein Messquerschnitt (*Zählpunkt*) zwischen eine Gerade und eine Kurve gelegt werden kann.

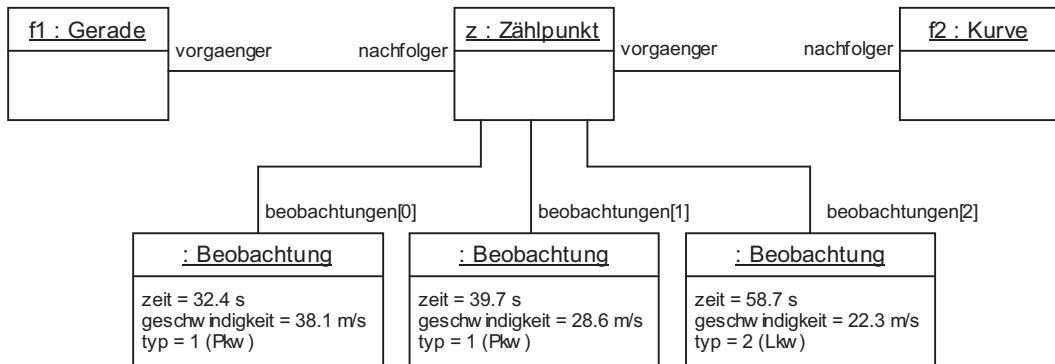


Bild 3.26: Objektdiagramm eines Zählpunktes mit drei registrierten Fahrzeugen

Die Einbindung von Zählpunkten als eigene Fahrspuren direkt in das Streckennetz hat den Vorteil, dass der Zählpunkt reaktiv auf die Vorbeifahrt eines Fahrzeugs warten kann, ohne zu jedem Zeitschritt eine Überprüfung der Positionen aller Fahrzeuge auf einer bestimmten Fahrspur durchführen zu müssen. Hierdurch ergibt sich ein weiterer Performanzgewinn, der jedoch erfordert, dass Zählpunkte nur an den Endpunkten von Fahrspuren platziert werden können. Messungen in einem beliebigen Punkt entlang einer Fahrspur sind nicht möglich.

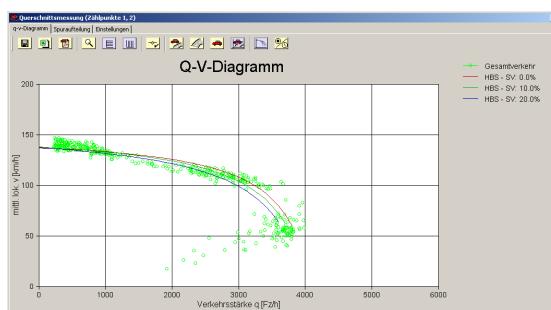


Bild 3.27: Beispiel für ein mit einem Zählpunkt aufgenommenes q-v-Diagramm

Gegenüber der realen Messung hat das Computermodell einen klaren Vorteil: durch Auslesen der Attribute eines Fahrzeugs kann dessen Geschwindigkeit am Zählpunkt exakt bestimmt werden, ohne – wie in der Realität – die Geschwindigkeit durch mehrere Messungen oder Analyse von Videoaufzeichnungen abschätzen zu müssen. So kann direkt bei der Vorbeifahrt des Fahrzeugs die momentane Geschwindigkeit, der Fahrzeugtyp (Pkw oder Lkw) und der genaue Zeitpunkt der Messung für jede Fahrspur getrennt erfasst werden. Die Beobachtungen, im Objektdiagramm in Abbildung 3.26 als Instanzen einer eigenen Klasse dargestellt, werden intern aus Effizienzgründen als einfaches Object-Array abgelegt. Aus den aufgenommenen Daten können dann, sowohl während der Simulation als auch danach, Auswertungen (z.B. in Form von q-v-Diagrammen wie in Abbildung 3.27) durchgeführt werden.

3.8.2 Momentane Messungen

Bei einer momentanen Messung wird für einen bestimmten Zeitpunkt die Position aller Fahrzeuge festgehalten, die sich in einem definierten räumlichen Kontrollgebiet aufhalten. Diese „schnappschussartigen“ Messungen erlauben Rückschlüsse auf die Verkehrsdichte, die momentanen Geschwindigkeiten und die Folgeabstände der Fahrzeuge zueinander. Durch einen Vergleich mehrerer, zeitlich versetzter momentaner Messungen kann das zeitliche Verhalten eines Verkehrsstroms, insbesondere die Ausbildung von Stauwellen, untersucht werden. Da eine momentane Messung sich über eine oder mehrere Fahrspuren erstrecken kann, wird sie im Simulationsmodell als *Streckenmessung* bezeichnet.

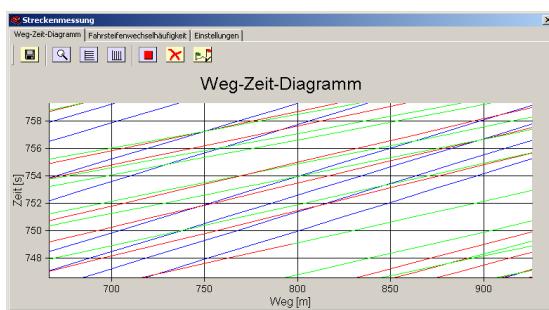


Bild 3.28: Beispiel für ein Weg-Zeit-Diagramm

Für die Durchführung von Streckenmessungen sind – im Gegensatz zur lokalen Messung – keine Änderungen am eigentlichen Streckennetz erforderlich. Stattdessen werden der Instanz der Klasse *Streckenmessung* lediglich die Fahrspuren mitgeteilt, die überwacht werden sollen. Die Daten, die eine Streckenmessung dann während der Simulation sammelt, stehen anschließend für Auswertungen zur Verfügung. Auf diese Weise können Weg-Zeit-Diagramme gewonnen werden (vgl. Abbildung 3.28) oder Rückschlüsse über die Häufigkeit von Spurwechselvorgängen ziehen.

3.8.3 Reisezeiten

Zur Beurteilung der Verkehrsqualität ist die Reisezeit – also die Zeit, die ein Fahrzeug für die Fahrt von einer Quelle zu einer Senke benötigt – ein wichtiges Kriterium. Deshalb wurde eine zentrale Klasse zur Verwaltung von Fahrzeug-Reisezeiten in die Simulation integriert. Jedes Fahrzeug verfügt schon zum Zeitpunkt seiner Erzeugung über eine eigene Instanz der Klasse Reisezeit (vgl. Abbildung 3.8). Sobald ein Fahrzeug auf das Simulationsnetz gesetzt wird, registriert das Reisezeit-Objekt die Aufsetzzeit; sobald das Fahrzeug das Ziel seiner Reise erreicht hat, kann die Reisezeit als Differenz zwischen der Ankunfts- und der Aufsetzzeit bestimmt werden.

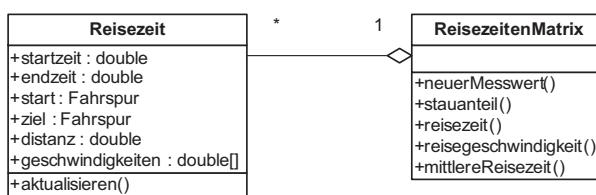


Bild 3.29: Klassendiagramm einer Reisezeitenmatrix mit Einzelreisezeiten

Für die Szenarioanalyse ist neben der Reisezeit – und damit, bei bekannter Strecke, der durchschnittlichen Reisegeschwindigkeit – auch das Geschwindigkeitsprofil entlang der Strecke von Interesse. Abbildung 3.30 zeigt hierzu ein Beispiel: Im linken Diagramm sind die Geschwindigkeiten zweier hypothetischer Fahrzeuge über die Zeit aufgetragen. Beide Fahrzeuge benötigen die gleiche Zeit für ihre Fahrt, und legen dabei die gleiche Distanz zurück (das Integral beider Kurven über die Zeit ist identisch). Die Durchschnittsgeschwindigkeit beider Fahrzeuge ist folglich ebenfalls gleich.

Bei alleiniger Betrachtung dieser Durchschnittsgeschwindigkeit könnte daher vermutet werden, dass die Fahrt beider Fahrzeuge ähnlich verläuft. Ein Blick auf die Geschwindigkeitskurven widerlegt diese Annahme jedoch deutlich: Während Fahrzeug 2 die Strecke mit annähernd konstanter Geschwindigkeit zurücklegt, muss Fahrzeug 2 nach anfänglich hoher Geschwindigkeit aufgrund einer Störung im Verkehrsfluss stark abbremsen. Erst nach einigen Minuten erholt sich der Verkehrsfluss und Fahrzeug 1 kann seine zügige Fahrt fortsetzen.

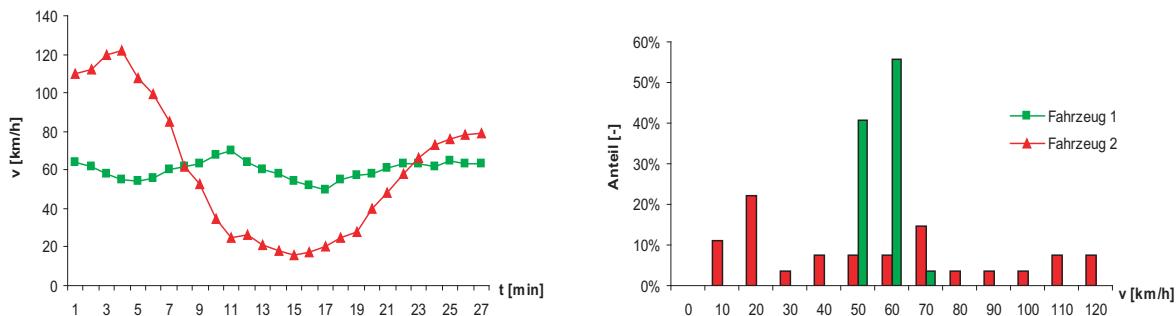
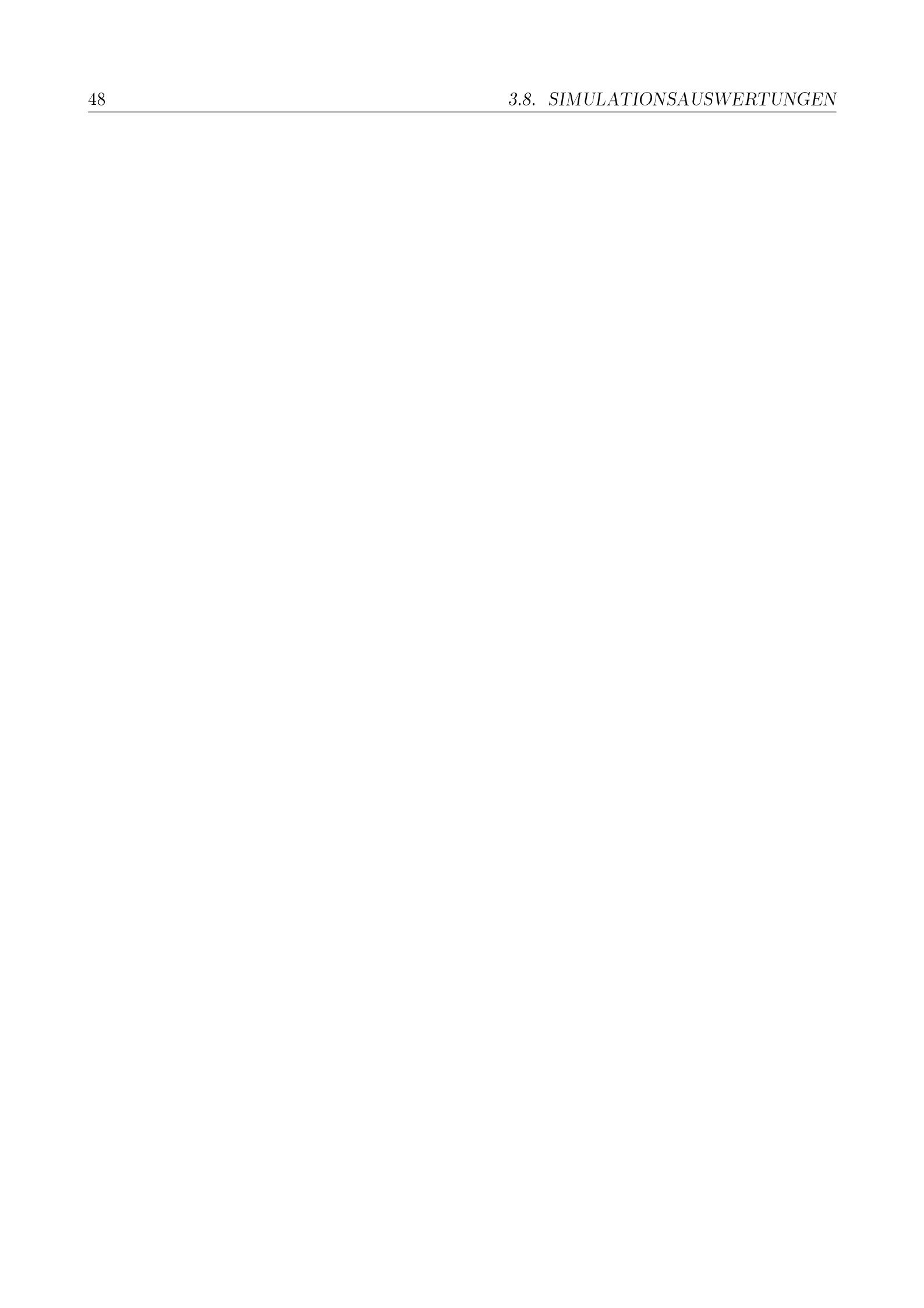


Bild 3.30: v-t-Diagramm zweier Fahrzeuge (links) mit Histogramm (rechts)

Um die Unterschiede in der Geschwindigkeitsverteilung erkennen und auswerten zu können, sind die Momentangeschwindigkeiten während der Fahrt für jedes Fahrzeug aufzuzeichnen. Eine vollständige Speicherung aller Werte – insbesondere bei großen Streckenlängen mit hohen Verkehrsstärken – würde einen sehr großen Speicherbedarf verursachen, der noch dazu mit zunehmender Fahrtzeit eines Fahrzeugs linear ansteigen würde. Um dies zu verhindern, wird anstelle des gesamten Geschwindigkeitsverlaufs lediglich das entsprechende Histogramm abgespeichert. Jeder einzelne Messwert wird dabei einer von 20 Geschwindigkeitsklassen zugeordnet; jede der 20 Geschwindigkeitsklassen entspricht jeweils einem Wertebereich von 10 km/h, beginnend bei 0 km/h. Dadurch, dass nur eine Häufigkeitsverteilung der Daten gespeichert wird, kann der Speicherbedarf pro Fahrzeug stark reduziert werden; zudem ist der Speicherbedarf unabhängig von der Simulationsdauer. Das Histogramm der Geschwindigkeitsverteilungen beider Fahrzeuge ist in der rechten Hälfte von Abbildung 3.30 abgebildet.

Hat ein Fahrzeug eine Senke erreicht, wird es unwiederbringlich aus der Simulation entfernt und alle zugehörigen Referenzen gelöscht. Seine Reisezeit hingegen wird der so genannten Reisezeitenmatrix hinzugefügt. Diese akkumuliert alle Reisezeiten und wertet sie aus. Für jedes Quelle-Senke-Paar kann dann u.a. die erreichte Reisegeschwindigkeit, die mittlere Reisezeit pro 100 km sowie der Stauanteil (Summe der Histogramm-Anteile für Geschwindigkeiten unter 20 km/h) errechnet werden.



Kapitel 4

Verhaltensmodelle

4.1 Einleitung

Die Aufgabe eines Verhaltensmodells in der mikroskopischen Verkehrssimulation ist es, die Fahrzeuge so durch das auf dem Computer simulierte Straßennetz zu steuern, dass alle Fahrzeugbewegungen sowohl auf makroskopischer als auch mikroskopischer Ebene möglichst gut mit der Realität übereinstimmen. Bereits seit den späten 1950er Jahren wurden erste mikroskopische Verhaltensmodelle aufgestellt und bis heute ständig weiterentwickelt. Um diese bestehenden Verhaltensmodelle in Objektmodellform zu bringen, sind erhebliche Änderungen und Anpassungsarbeiten erforderlich.

In diesem Kapitel werden die wesentlichen Verhaltensmodelle dargestellt, die während der Entwicklung der Simulationsumgebung untersucht und implementiert wurden. Neben einer Beschreibung der grundlegenden verkehrstechnischen Konzepte wird insbesondere auf die Umsetzung aus informatischer Sicht eingegangen. Die Kalibrierung der Modelle wird dagegen an dieser Stelle nicht weiter beschrieben. Eine ausführlichere Darstellung der verkehrstechnischen Aspekte der Verhaltensmodellierung findet sich in Arbeiten von Brilon und Harding ([18],[19]); dort wird auch insbesondere auf die Kalibrierung einer Verkehrssimulation aufgrund realer mikroskopischer und makroskopischer Messdaten eingegangen.

Die hier beschriebenen Verhaltensmodelle lassen sich wegen ihrer unterschiedlichen Zielsetzungen grob in die beiden Gruppen „Abstandsmodelle“ und „Spurwechselmodelle“ unterteilen. Abstandsmodelle dienen allein der Regelung axialer Fahrzeuggeschwindigkeiten in Abhängigkeit von Position und Geschwindigkeit der Vorderfahrzeuge. Spurwechselmodelle hingegen steuern den lateralen Wechsel von Fahrspuren, wobei der Wechsel nur dann erfolgen kann, wenn keine Gefährdung oder Behinderung durch seitlich benachbarte Fahrzeuge zu erwarten ist. Die allgemeine Einbindung der Verhaltensmodelle in das Klassenmodell wurde in Kapitel 3.4.2 erläutert, die konkrete Umsetzung der Verhaltensmodelle wird nachfolgend beschrieben.

Während der Einbindung der behandelten Verhaltensmodelle in die Simulationsumgebung wurde untersucht, inwieweit sich die Modelle für den Einsatz in einem objektorientierten Klassenmodell eignen, und wo die Einsatzgrenzen der bestehenden Modelle liegen. Aufbauend auf den gewonnenen Erkenntnissen wird in Kapitel 5 ein verbessertes, vollständig modulares und jederzeit erweiterbares Verhaltensmodell vorgestellt.

4.2 Abstandsmodelle

Neben der Steuerung eines Fahrzeugs besteht die Hauptaufgabe eines Fahrers in der Wahl einer geeigneten Geschwindigkeit bzw. einer geeigneten Beschleunigung in Fahrspurlängsrichtung. Ausschlaggebend sind dabei – neben der vom Fahrer abhängigen Wunschgeschwindigkeit – äußere Einflussfaktoren, wie die zulässige Höchstgeschwindigkeit, die Fahrspurgeometrie (Steigung, Kurvigkeits) sowie Abstände und Geschwindigkeiten potentieller Hindernisse (vornehmlich andere Fahrer-Fahrzeug-Einheiten). Durch das Abstandsmodell wird zu jedem Zeitschritt der Simulation aus den vom Klassenmodell zur Verfügung gestellten Informationen die Fahrzeuggeschwindigkeit bestimmt. Im Folgenden wird ein grober Überblick über alle implementierten Abstandsmodelle gegeben. Für eine detailliertere Analyse existierender mikroskopischer Fahrzeugfolgemodele sei u.a. auf die Arbeiten von Brackstone ([9]), Janson/Tapani ([65]) und Helbing ([58]) verwiesen.

4.2.1 Fahrzeugfolge-Theorie

Die Modelle der Fahrzeugfolge-Theorie, auch Follow-the-Leader-Modelle genannt, basieren auf der Annahme, dass die Geschwindigkeitswahl eines Fahrzeugs allein vom Fahrzustand (Geschwindigkeit, Abstand) des Vorderfahrzeugs abhängen. Das deterministische Grundmodell von Reuschel ([94]) und Pipes ([87]) wurde später von der Gruppe um Herman und Gazis erweitert ([47]) und wie folgt vereinheitlicht ([48]):

$$a_{n+1} = \alpha_0 \cdot \frac{(v_n + a_n \cdot T)^m}{(\Delta x)^l} \cdot \Delta v \quad (4.1)$$

Hiernach wird die neue Beschleunigung a_{n+1} des Fahrzeugs berechnet aus der aktuellen Geschwindigkeit v_n , der aktuellen Beschleunigung a_n , der Reaktionszeit T , der Relativgeschwindigkeit Δv und der Bruttoweglücke Δx . Der Sensitivitätsfaktor α_0 und die beiden Parametern l und m dienen zur Kalibrierung des Modells und sind von der Fahrsituation abhängig. Entsprechend den Untersuchungen von Hoefs ([61]) werden dabei vier verschiedene Fälle unterschieden: Öffnen (Vergrößerung der Weglücke), Schließen (Verkleinern der Weglücke, mit und ohne Bremsleuchten) und sonstige Fälle. Für jeden der vier Fälle gilt ein eigener Parametersatz, der vom Fahrverhalten in Abhängigkeit von der Fahrsituation ausgewählt wird.

Im Rahmen der Neu-Kalibrierung dieses Modells haben Harding und Seifarth auf der Basis von empirischen Daten angepasste Werte für die Parameter α_0 , m und l ermittelt ([18]). Die Ermittlung basiert auf einem in die Simulationsumgebung implementierten Optimierungsmodul, das auf über vollständige Enumeration aller möglichen Parameterkombinationen den optimalen Parametersatz ermittelt. Die Optimierungsläufe erfolgten quasi-parallel auf verschiedenen, physisch nicht verbundenen Rechnern, die jeweils Simulationsläufe für eine Untermenge des gesamten Suchraums durchführten. Durch Vergleiche mit Messdaten von realen Verfolgungsfahrten konnten die Simulationsparameter mit den geringstmöglichen Abweichungen ermittelt werden (vgl. Kapitel 6.4.2). Mit den gefundenen Parametern kann mit Gleichung 4.1 die neue Beschleunigung eines Fahrzeugs zu jedem Zeitschritt bestimmt werden. Die informatische Umsetzung des Verfahrens ist damit einfach: Über das objektorientierte Simulationsmodell müssen lediglich die Geschwindigkeits- und Beschleunigungswerte des Vorderfahrzeugs sowie die Weglücke zwischen beiden Fahrzeugen bestimmt und in Gleichung 4.1 eingesetzt werden. Falls das

Modell an eine spezielle Verkehrssituation angepasst werden muss, kann eine manuelle Nachkalibrierung der vier Parametergruppen über ein Einstellung-Menü erfolgen.

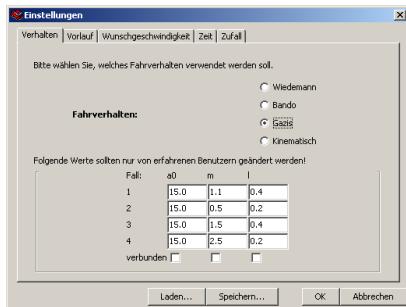


Bild 4.1: Menü zum Verändern der Fahrzeugfolge-Parameter

Eine verkehrstechnische Untersuchung der mit dem modifizierten Fahrzeugfolge-Modell erzielten Ergebnisse zeigt insgesamt eine unzureichende Übereinstimmung der Simulationsdaten mit den vom HBS vorgegebenen Vergleichsdaten ([57]). Schon bei vergleichsweise geringen Verkehrsstärken von ca. 2000 Fz/h fällt die Durchschnittsgeschwindigkeit stark ab, behält dann jedoch eine Geschwindigkeit von ca. 70 km/h – unabhängig von der aufgebrachten Verkehrsstärke – bei. Ein Verkehrszusammenbruch, wie er bei hohen Verkehrsstärken zu erwarten ist, kann mit dem Fahrzeugfolge-Modell nicht nachgebildet werden ([18]).

4.2.2 Optimal-Velocity-Model

Das von Bando et. al. entwickelte, in seiner Grundform ebenfalls deterministische Optimal-Velocity-Model (kurz: OVM), beruht auf der Annahme, dass ein Fahrzeug stets versucht, die Differenz zwischen seiner aktuellen Geschwindigkeit und seiner Wunschgeschwindigkeit zu minimieren ([4]). Dabei wird ein linearer Zusammenhang zwischen dieser Differenz und der resultierenden Beschleunigung des Fahrzeugs angenommen:

$$a_{n+1} = \alpha (V_{opt} (\Delta x) - v_n) \quad (4.2)$$

Die Beschleunigung zum nächsten Zeitschritt hängt somit von einem Proportionalitätsfaktor α , der aktuellen Geschwindigkeit v_n sowie einer von der Netto-Weglücke Δx abhängigen Wunschgeschwindigkeit V_{opt} ab. Diese optimale Geschwindigkeit (Optimal Velocity, OV) wurde von Bando als monoton wachsende Funktion mit einer oberen Schranke angegeben:

$$V_{opt} (\Delta x) = v_{max} \cdot (\tanh (0.0860 (\Delta x - 25)) + 0.913) \quad (4.3)$$

Neben dieser ursprünglichen OV-Funktion wurden seither zahlreiche alternative Ansatzfunktionen entwickelt. Die folgende Auswahl an OV-Funktionen wurde während der Entwicklung der Simulationsumgebung in die Fahrzeugfolgemodell-Implementierung aufgenommen:

- Bando et al. stellten 1998 eine überarbeitete OV-Funktion vor, die einen expliziten Verzögerungsterm τ in die Berechnung einbezieht ([5]).

- Tokumura und Tadaki unterteilten den Parameter α in zwei vom Vorzeichen des Restterms abhängige Parameter α_+ und α_- , die das asymmetrische Beschleunigungs- und Bremsvermögen der Fahrzeuge berücksichtigt sollen ([84]).
- Von Van Aerde stammt eine vier-parametrische Ansatzfunktion für eine dichteabhängige Geschwindigkeitsverteilung, die auch den Geschwindigkeitseinbruch bei gebundenem Verkehr erfasst ([113][114]).

Wie auch beim Fahrzeugfolge-Modell ist die informatische Umsetzung des Bando-Modells ohne großen Aufwand möglich. Zu jedem Zeitschritt wird, in Abhängigkeit von der gewählten OV-Funktion, eine der implementierten Methoden über einfache Fallunterscheidungen aufgerufen. Ein direkter Vergleich der mit dem Bando-Modell und seinen Variationen erzielbaren Simulationsergebnisse mit denen anderer Verhaltensmodelle zeigt, dass für zweispurige Autobahnen die Geschwindigkeits-Dichte-Verhältnisse bis zu einer Verkehrsstärke von ca. 3000 Fz/h gut mit den real beobachtbaren Werten übereinstimmen. Ein Zusammenbruch des Verkehrs, wie er eigentlich zu erwarten wäre, lässt sich mit diesem Modell jedoch ebenfalls nicht nachbilden ([18]).

4.2.3 Psycho-physisches Modell

Sowohl die Modelle der Fahrzeugfolgetheorie als auch das Optimal Velocity Modell gehen von der Annahme aus, dass Abstand und Geschwindigkeitsdifferenz zwischen einem Fahrzeug und seinem Interaktionspartner stets exakt bekannt sind und der Fahrer nach einem deterministischen Verfahren auf diese Eingangswerte reagieren kann. In der Realität hingegen zeigt sich, dass das Wahrnehmungsvermögen eines menschlichen Fahrers stets nur eine qualitative Einschätzung der Fahrsituation erlaubt. Ob und wann ein Fahrer auf eine Geschwindigkeitsdifferenz reagiert, wird in großem Maße von seinem optischen Wahrnehmungsvermögen beeinflusst.

Bereits 1963 führten Todosiev [110] und Michaels ([78]) Untersuchungen zum Wahrnehmungsverhalten von Autofahrern durch und stellten fest, dass ein Fahrer erst dann einen äußeren Einfluss reagiert, wenn eine gewisse visuelle Wahrnehmungsschwelle überschritten wurde. Wiedemann ([124]) entwickelte aus diesem Ansatz das sogenannte psycho-physische Fahrverhalten, das die Wahrnehmungseffekte berücksichtigt und eine realitätsnahe Nachbildung des Fahrverhaltens erlaubt.

Das psycho-physische Fahrverhalten ist bis heute eines der erfolgreichsten Fahrzeugfolgemodele, und wird, obwohl es bereits vor über 30 Jahren entwickelt wurde, auch heute noch vergleichsweise häufig verwendet. Einige kommerzielle Programme, darunter u.a. das in Deutschland weit verbreitete VISSIM, basieren noch immer im Wesentlichen auf dem von Wiedemann entwickelten Modell ([40]).

Zu jedem Zeitschritt befindet sich ein Fahrer nach dem Wiedemannschen Modell in genau einem diskreten Fahrzustand. Der Übergang zwischen zwei Zuständen findet statt, wenn die aktuellen Bewegungsgrößen des Fahrzeugs die erwähnten Wahrnehmungsschwellen überschreiten. Ein Beispiel für ein solches Übergangsverhalten ist die Reaktion eines Fahrers, der bemerkt, dass sein Abstand zum Vordermann einen für ihn als sicher empfundenen Mindestabstand unterschreitet: Der Fahrer verringert seine Geschwindigkeit, um so seinen Abstand zum Vordermann zu vergrößern. Dieser Zustand wird so lange aufrecht erhalten, bis wieder ein ausreichend großer Abstand erreicht ist. Bemerkt ein Fahrer, dass die Weglücke zum Vordermann zu groß wird,

beschleunigt er sein Fahrzeug, um die Lücke wieder kleiner werden zu lassen. Diese von Hoefs ([61]) beschriebenen Oszillationen um einen bevorzugten Folgeabstand lassen sich mit dem psycho-physischen Modell gut nachbilden (siehe auch Abbildung 4.2).

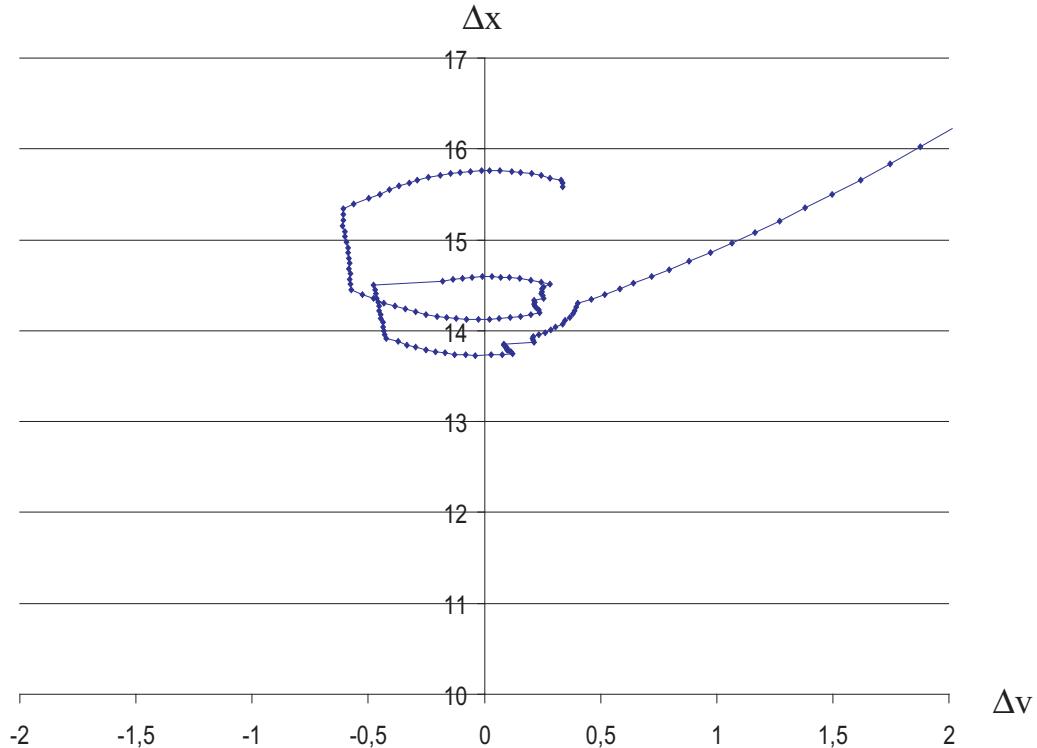


Bild 4.2: Ein Annäherungsvorgang nach dem psycho-physischen Folgemodell

Insgesamt unterscheidet Wiedemann vier verschiedene Fahrzustände:

- **WUNSCH:** Das Fahrzeug versucht – weitgehend unbeeinflusst durch andere Fahrzeuge – seine Wunschgeschwindigkeit zu erreichen.
- **FOLGEN:** Das Fahrzeug wird unbewusst durch sein Vorderfahrzeug beeinflusst und passt seine Geschwindigkeit nur über das Gaspedal entsprechend an.
- **BREMSBX:** Das Fahrzeug wird bewusst durch das Vorderfahrzeug beeinflusst und reagiert durch aktives Bremsen
- **BREMSAX:** Der Fahrer erkennt eine kritische Gefahrensituation und versucht, durch starkes Bremsen den Sicherheitsabstand zum Vordermann wieder herzustellen.

Um zu beurteilen, in welchem Fahrzustand sich ein Fahrzeug zu einem gegebenen Zeitpunkt befindet, definiert Wiedemann verschiedene Vergleichswerte für Geschwindigkeitsdifferenzen und Folgeabstände, die je nach Fahrereigenschaft (vgl. Kapitel 3.4.1) unterschiedlich ausfallen können. Über einen Entscheidungsbaum wird aus diesen Werte der aktuelle Fahrzustand abgeleitet (siehe Abbildung 4.3).

Das von Wiedemann erstellte Programm ist vollständig in Algol implementiert und steht als vollständigen Quelltext zur Verfügung ([124]), so dass eine direkte Übertragung in eine äquivalente Java-Klasse mit vergleichsweise geringem Aufwand möglich ist. Eine im Rahmen der

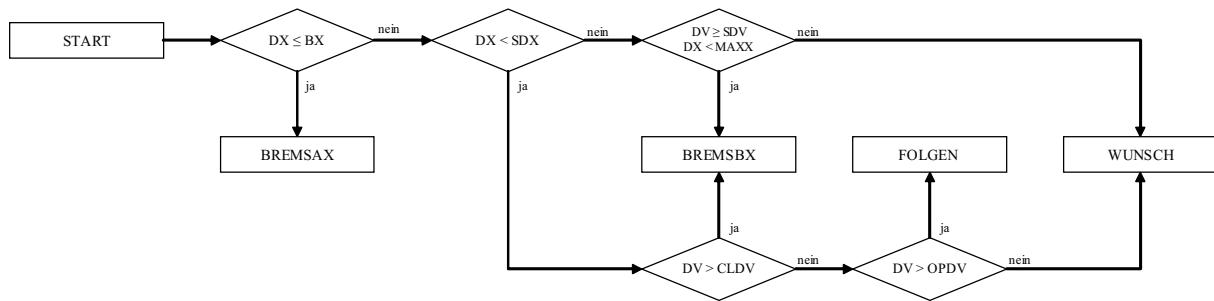


Bild 4.3: Entscheidungsbaum zur Ermittlung des Fahrzustands nach Wiedemann

Entwicklung der Simulationsumgebung eingeführte Modifikation des ursprünglichen psycho-physischen Modells wurde von Brilon und Harding vorgeschlagen ([18]) und basiert auf den Erfahrungen mit dem trotz seiner Einfachheit überraschend erfolgreichen Zellular-Automaten nach Nagel-Schreckenberg ([81]). Während die meisten bekannten mikroskopischen Verhaltensmodelle den Zusammenbruch des Verkehrs bei Erreichen der Sättigungsverkehrsstärke nur unzureichend abbilden, ist der Zellular-Automat in der Lage, dieses Phänomen (den sogenannten „Stau aus dem Nichts“) abzubilden. Erreicht wird dies durch einen künstlichen „Trödeleffekt“: Zu jedem Zeitschritt besteht eine gewisse Chance, dass sich ein Fahrzeug zurückfallen lässt und seine Geschwindigkeit reduziert. Da hierdurch ein eventuelles Hinterfahrzeug ebenfalls seine Geschwindigkeit verringern muss, kann es ab einer Mindestverkehrsstärke zu einem Aufschaukungseffekt kommen, der den Verkehr hinter dem trödelnden Fahrzeug zusammenbrechen lässt.

Überträgt man diesen stochastischen Mechanismus auf das Modell von Wiedemann (z. B. indem ein Fahrer mit einer bestimmten Wahrscheinlichkeit für einige Sekunden auf einen Beschleunigungsvorgang verzichtet), lassen sich ähnliche Verkehrszusammenbrüche auch hier nachbilden. Abbildung 4.4 zeigt exemplarisch das q-v-Diagramm eines Simulationslaufs für einen ebenen Autobahnabschnitt mit zwei Richtungsfahrstreifen. Ab einer Verkehrsstärke von ca. 3600 Fahrzeugen kommt es zu einem Verkehrszusammenbruch, Geschwindigkeit und erreichbare Verkehrsstärke sinken stark ab. Insgesamt ergibt sich hier eine gute Übereinstimmung zwischen den Simulationsdaten und den nach HBS zu erwartenden Werten ([57]).

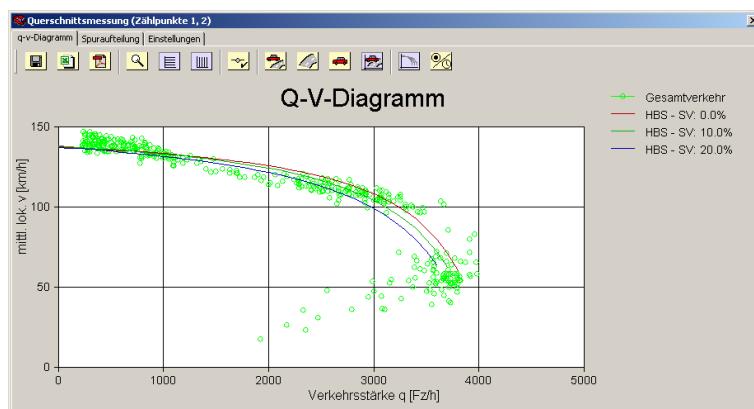


Bild 4.4: Mit dem modifizierten Wiedemann-Modell erstelltes q-v-Diagramm

Die Realitätsnähe der mit dem Wiedemannschen Modell durchgeföhrten Simulation ist insgesamt als sehr hoch einzustufen. Dies liegt sicherlich nicht zuletzt an der – verglichen mit anderen Fahrzeugfolgemodellen – hohen Komplexität des Verfahrens, das somit allerdings auch

mit einem höheren Berechnungsaufwand verbunden ist. Daher sind mit dem psycho-physische Verfahren auf einfachen Arbeitsplatzrechnern noch keine räumlichen derart umfassenden Netze realisierbar, wie sie z. B. mit dem Zellularautomaten erreicht werden können ([77][28]). Trotz seiner Komplexität hat sich das Wiedemann-Modell als sehr gut handhabbar und numerisch robust erwiesen, und liefert, insbesondere in der um einen Trödelfaktor erweiterten Form, sehr realitätsnahe Ergebnisse ([18]).

4.3 Spurwechselmodelle

Neben den axialen – auf die Fahrtrichtung eines Fahrzeugs bezogenen – Beschleunigungen ist die Nachbildung eines realitätsnahen lateralen Fahrverhaltens von entscheidender Bedeutung für die Güte der Simulation, insbesondere für korrekte Fahrstreifenverteilungen des Verkehrs und die Nachbildung strategisch-kooperativer Verhaltensmuster im Bereich von Ein- und Ausfahrten. Die Komplexität der erforderlichen Entscheidungen ist jedoch durch die größere Anzahl möglicher Interaktionspartner und Fahrerreaktionen wesentlich höher als bei den beschriebenen Fahrzeugfolgemodellen, und erfordert daher auch einen größeren Aufwand für Implementierung und Kalibrierung.

Spurwechsel sind per definitionem nur auf solchen Fahrspuren möglich, die mindestens eine lateral benachbarte Spur besitzen. Auf einspurigen Strecken ist kein Spurwechsel durchführbar, wodurch hier lediglich die Fahrzeugfolgeverhalten in Betracht zu ziehen sind. Ist eine Nachbarfahrspur vorhanden, wird durch das Spurwechselverhalten ermittelt, ob ein Spurwechsel gewünscht ist und ohne eine Gefährdung des Fahrers durchgeführt werden kann, und ob keine anderen Fahrzeuge durch den Spurwechsel behindert oder gefährdet werden. Zusätzlich können noch strategische Aspekte in die Entscheidungsfindung einfließen, wie z. B. das Unterstützen anderer Fahrzeuge zum Erreichen ihrer Fahrspur oder die Frage, wie weit vor einer Ausfahrt ein Vorderfahrzeug noch überholt werden kann.

Ob, wann und wie ein Spurwechsel stattfindet, ist also von zahlreichen Einflussfaktoren und damit stark von der jeweiligen Fahrsituation abhängig. Im Bereich einer Ein- oder Ausfahrt müssen andere Aspekte berücksichtigt werden als auf freier Strecke zwischen zwei Anschlussstellen. Das grundsätzliche Problem bei der Implementierung von Spurwechselmodellen ist also die Berücksichtigung des jeweiligen situativen Kontextes, in dem der Spurwechsel durchgeführt werden soll. Zusätzlich erschwerend kommt noch hinzu, dass neben der reinen Spurwechselentscheidung unter Umständen auch noch eine Beschleunigung in axialer Richtung berücksichtigt werden muss, und damit eine Interaktion mit dem (eigentlich in sich geschlossenen) Fahrzeugfolgeverhalten auftritt.

Die Kalibrierung von Spurwechselverhalten ist eine überaus anspruchsvolle Aufgabe, umso mehr, wenn das zu entwickelnde Modell für grundlegend verschiedene Fahrsituationen (z. B. Überholvorgänge, Verlassen einer Autobahn, Reißverschlussverfahren) anwendbar sein soll. Viele Forschungsarbeiten zu diesem Thema konzentrieren sich daher auf einzelne, klar definierte Fahrsituationen, die keine oder nur wenige Sonderfälle zulassen. Durch diese Spezialisierung ist es möglich, für die jeweils betrachtete Situation eine sehr gute Übereinstimmung zwischen Simulation und Realität zu erreichen. Im Folgenden werden zwei Spurwechselverhalten beschrieben, die sich jeweils auf den Spurwechsel auf der freien Strecke bzw. im Bereich von Ein- und Ausfahrten spezialisiert haben.

4.3.1 Sparmann

Aufbauend auf den Arbeiten von Wiedemann entwickelte Sparmann einen Spurwechselmechanismus für den Verkehr auf zweispurigen Straßenabschnitten ([105]). Auch für dreispurige Fahrbahnen lässt sich dieses Verfahren anwenden ([72]). Um die Fahrsituation der Fahrzeuge vor bzw. nach einem Spurwechsel zu beurteilen, verwendet Sparmann – in Anlehnung an die bei Wiedemann eingeführten Beeinflussungsbereiche – drei diskrete Zustände, die jeweils durch bestimmte Vergleichsgeschwindigkeiten oder -abstände beschrieben werden:

- **unbeeinflusst:** Es liegt keine Beeinflussung durch das Nachbarfahrzeug vor. Entspricht dem unbeeinflusstem Zustand des Wiedemannschen Modells.
- **potenziell beeinflusst:** Eine Beeinflussung zum aktuellen Zeitschritt liegt nicht vor, ist aber in den folgenden Zeitschritten wahrscheinlich.
- **aktuell beeinflusst:** Es liegt momentan eine (unbewusste, bewusste oder kritische) Beeinflussung vor.

Damit ein Fahrzeug seine Fahrspur wechseln kann, muss zunächst ein Bedarf für einen Spurwechsel entstehen. Sparmann definiert hierzu zwei Sätze von Regeln – jeweils getrennt für Spurwechsel nach links und rechts – die erfüllt sein müssen, damit ein Fahrzeug überhaupt einen Spurwechselwunsch aufbauen kann. So will ein Fahrzeug beispielsweise nur dann auf den linken Fahrstreifen wechseln, wenn eine Beeinflussung durch den Vordermann besteht oder eine potentielle Beeinflussung durch das linke Vorderfahrzeug vorliegt und somit die Gefahr des Rechtsüberholens entstehen könnte. Für eine vollständige Übersicht aller zu berücksichtigenden Situationen wird auf [105] oder [18] verwiesen.

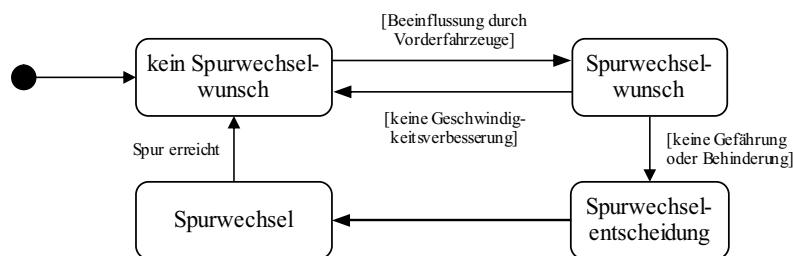


Bild 4.5: Zustandsdiagramm des Spurwechselvorgangs nach Sparmann

Das Vorhandensein eines Spurwechselwunsches allein stellt jedoch nur ein notwendiges und kein hinreichendes Kriterium für die Durchführung eines Spurwechsels dar. Zusätzlich müssen noch gewisse Sicherheitsbedingungen erfüllt sein, damit ein Fahrzeug durch den Spurwechsels weder sich noch die benachbarten Fahrzeuge in Gefahr bringt oder über Gebühr behindert. So wird z. B. ein Fahrzeug nur dann nach links wechseln, wenn der Hintermann entweder gar nicht beeinflusst wird, oder wenn nur eine potentielle Beeinflussung vorliegt und das wechselnde Fahrzeug seine Wunschgeschwindigkeit deutlich unterschreitet.

Damit ein Fahrzeug nach einem Spurwechsel zunächst seine Fahrspur beibehält, und nicht sofort einen erneuten Spurwechsel initiieren kann, wurde zusätzlich eine Entscheidungszeit vorgesehen, über die sich ein Spurwechselwunsch erst aufbauen muss, bevor tatsächlich der Spurwechsel eingeleitet wird. Verschwindet während dieser Zeit der Auslöser des Spurwechselwunsches (z. B.

durch ein Erreichen der Wunschgeschwindigkeit oder einen Spurwechsel des Vordermanns), baut sich der Spurwechselwunsch langsam wieder ab. Durch Variation der erforderlichen Zeitdauer eines Spurwechselwunsches kann die Spurwechselhäufigkeit damit direkt gesteuert werden.

Bei der Umsetzung des Sparmann-Modells in objektorientierte Form werden die historischen Wurzeln des Verfahrens deutlich: Wie das von Wiedemann entwickelte Modell stammt das Quelltext von Sparmann aus den späten siebziger Jahren, und verwendet die damals üblichen prozeduralen Programmiertechniken. Sparmann integrierte sein Spurwechselverhalten direkt als neues Modul in das in Algol programmierte Programmsystem MISSIS, das von Leutzbach, Wiedemann und Hubschneider entwickelt wurde([73]). Auf Details der Implementierung wird in Sparmanns Arbeit leider nicht konkret eingegangen, so dass der genaue Aufbau des Modells aus Sparmanns Beschreibung rekonstruiert werden musste.

Im Wesentlichen lässt sich der gesamte Spurwechselalgorithmus, wie auch das Fahrverhalten von Wiedemann, informatisch gesehen auf eine Abfolge von Alternativen zurückführen. Zunächst werden aus den vom Objektmodell bereitgestellten Informationen zu Geschwindigkeiten und Abständen des betrachteten Fahrzeugs sowie seiner Interaktionspartner die angestrebten Sicherheitsabstände und Relativabstände berechnet. Durch Vergleich mit den tatsächlichen Werten wird ermittelt, ob sich das Fahrzeug relativ zu seinem Interaktionspartner in einem unbeeinflussten, potentiell beeinflussten oder aktuell beeinflussten Fahrzustand befindet. Für jede Kombination von Fahrzuständen, die einen Spurwechsel auslösen kann, werden von Sparmann entsprechende *if*-Alternativen vorgesehen. Werden dabei sowohl die Bedingungen für einen Spurwechselwunsch als auch für eine Spurwechselentscheidung über einen ausreichend langen Zeitraum erfüllt, sendet das Spurwechselverhalten einen Spurwechselbefehl an das zugehörige Fahrzeug. Das Spurwechselverhalten wird anschließend wieder in seinen Ausgangszustand versetzt und der Zähler für die Entscheidungszeit zurückgesetzt.

Während das Modell von Sparmann auf mikroskopischer Ebene sehr realistisch wirkende Spurwechsel generieren kann, zeigen sich aus makroskopischer Sicht einige Mängel, wie Sparmann selbst anmerkt. So stimmen zwar die sich in der Simulation ergebenden Spurwechselhäufigkeiten relativ gut mit den von Sparmann in der Realität beobachteten überein, bei der Fahrstreifenverteilung jedoch ergibt sich eine deutliche Überschätzung der rechten Fahrspur gegenüber der Überholspur. Bei der Implementierung des Sparmann-Konzepts in die Simulationsumgebung konnte diesbezüglich eine Verbesserung der Verteilung erreicht werden. Ein weiterer Nachteil des Modells ist das Fehlen eines mehr antizipatorisch-strategischen Ansatzes. So betrachtet Sparmann ausschließlich die direkt benachbarten Fahrzeuge; andere Fahrzeuge, die potentiell zu Interaktionspartnern werden könnten – wie zum Beispiel auffahrende Fahrzeuge im Bereich einer Anschlussstelle – werden nicht berücksichtigt. Außerdem fehlt ein kooperativer Ansatz, der andere Fahrzeuge im Erreichen ihrer Fahrspur unterstützt. Für den Bereich der Ein- und Ausfahrten wurde daher ein zusätzliches Spurwechselmechanismus eingebaut, der die fehlenden Funktionalitäten des Sparmann-Modells kompensiert.

4.3.2 Theis

Speziell für das Verhalten an planfreien Anschlussstellen entwickelte erstmals Theis ein strategisches Spurwechselmodell, das die Ansätze von Wiedemann und Sparmann aufgreift und erweitert ([108]). Während in den bisher beschriebenen Modellen alle Fahrzeuge vorwiegend „egoistisch“ handeln und nur eigene Ziele verfolgen, werden beim Modell nach Theis bewusst kooperative Aspekte in Betracht gezogen: Fahrzeuge sind beim Erreichen einer bestimmten

Fahrspur nicht mehr auf sich selbst angewiesen, sondern können andere Fahrzeuge um Unterstützung beim Spurwechsel bitten. Hierdurch wird eine verbesserte Realitätsnähe in der Simulation erreicht werden, da auch vom realen Fahrer erwartet wird, dass er einfahrende oder ausfahrende Fahrzeuge soweit wie möglich unterstützt.

Das Modell von Theis ist konzipiert für den Einsatz im Bereich von planfreien Knotenpunkten auf Autobahnen, d.h. für Einfahrten, Ausfahrten und Verflechtungsbereiche. Grundsätzlich unterscheidet Theis drei Arten von Fahrzeugen:

- **Einfahrer:** Fahrzeuge, die an einer Einfahrt oder einem Verflechtungsbereich auf die linke Spur wechseln, um ihre Wunschspur zu erreichen.
- **Ausfahrer:** Fahrzeuge, die an der nächsten Ausfahrt oder dem nächsten Verflechtungsbereich ihre jetzige Fahrspur nach rechts verlassen.
- **Durchfahrer:** Fahrzeuge, die auf ihrer aktuellen Fahrspur bleiben können und lediglich unterstützend auf Einfahrer und Ausfahrer reagieren.

Primäres Ziel sowohl der einfahrenden als auch der von der linken Spur ausfahrenden Fahrzeuge ist das Erreichen einer geeigneten großen Lücke auf der rechten Spur der Hauptfahrbahn. Betrachtet werden dabei die dem Fahrzeug nächstliegende und die (in Fahrtrichtung gesehen) nachfolgende Lücke. Hinter dem Fahrzeug liegende Lücken werden nicht berücksichtigt, da sie ein Verzögern des Fahrzeugs erfordern und somit die Durchschnittsgeschwindigkeit unnötig senken würden. Hat sich ein Fahrzeug für eine geeignete Lücke entschieden, erhöht es – soweit erforderlich – die Geschwindigkeit, um direkt neben die Lücke zu gelangen. Anschließend passt sich das Fahrzeug durch „Gaswegnehmen“ an die Geschwindigkeit des Vorderfahrzeugs an, um sich schließlich, unter Beachtung der Sicherheitsabstände, in den fließenden Verkehr auf der gewünschten Zielspur einzureihen. Die Beurteilung der Sicherheitsabstände erfolgt erneut – wie bei Sparmann – nach dem Prinzip des psycho-physischen Modells von Wiedemann.

Reicht die vorhandene Lücke für das Einfädeln des Fahrzeugs (einschließlich der angestrebten Sicherheitsabstände) nicht aus, wird beim zukünftigen Hinterfahrzeug eine Vergrößerung der Lücke angefordert. Der Hintermann reagiert auf die Anforderung entweder durch eine Verlangsamung, oder – bei einem Einfahrvorgang – durch Spurwechsel auf die linke Fahrbahn. Diese aktive Unterstützung durch andere Fahrzeuge setzt dabei eine direkte Kommunikation zwischen zwei Fahrzeugobjekten voraus: Das unterstützende Fahrzeug hat so lange auf das die Spur wechselnde Fahrzeug Rücksicht zu nehmen, bis dieses entweder erfolgreich einen Spurwechsel durchführen konnte oder ein anderes Fahrzeug um Unterstützung gebeten wurde. Jedem Fahrzeug wird zu einem gegebenen Zeitpunkt maximal ein unterstützendes Fahrzeug zugeordnet.

Um einen vorausschauenden und damit flüssigen Einfahrvorgang zu gewährleisten, wird das Spurwechselverhalten bereits auf der Zufahrtrampe zur eigentlichen Auffahrt wirksam; schon dort wird mit der Suche nach einer geeigneten Lücke im Hauptverkehrsstrom begonnen. Hierfür ist es notwendig, dass ein Fahrzeug seine jeweils aktuelle Position auf die entsprechende Position auf der rechten Spur der Hauptfahrbahn projiziert. Erreicht das die Spur wechselnde Fahrzeug den Beginn des Einfädelungsstreifens, müssen sich Geschwindigkeit und Position relativ zur angestrebten Lücke soweit an den Verkehr auf der Hauptfahrbahn angeglichen haben, dass das Fahrzeug vor dem Ende der Auffahrt die gewünschte Spur erreichen kann.

Nähert sich ein ein- oder ausfahrendes Fahrzeug dem Ende der aktuellen Fahrspur, wird das Bedürfnis nach einem Spurwechsel immer dringender, so dass der Fahrer auch bereit ist, Risiken in

Kauf zu nehmen. In der Simulation wird das steigende Spurwechselbedürfnis des Fahrers durch eine Verringerung der Sicherheitsabstände realisiert, die sich ggf. bis zum Ende der Fahrspur immer weiter dem Stillstands-Sicherheitsabstand ax nach Wiedemann annähern. Zusammen mit der Unterstützung durch den Hintermann wird es dem Fahrzeug ermöglicht, seine jeweilige Wunschspur zu erreichen. Wird trotz dieser Regelung keine ausreichend große Lücke gefunden, muss das Fahrzeug kurz vor Spurende einen Bremsvorgang einleiten und auf eine geeignete Lücke warten.

Konnte ein Fahrer mit Hilfe eines verringerten Sicherheitsabstandes erfolgreich einen Spurwechsel durchführen, muss dieser Abstand mittelfristig wieder auf den gewohnten Wert (in Abhängigkeit seines persönlichen Sicherheitsbedürfnisses) gebracht werden. Diese Anpassung darf allerdings nicht zu abrupt geschehen, da sonst ein Ausbremsen des Hinterfahrzeugs erfolgen würde und damit – durch Rückkopplungseffekte – ein Zusammenbruch des Verkehrsstroms möglich wäre. Daher wird der Sicherheitsabstand nach dem Verlassen des Auffahrt- bzw. Ausfahrtbereichs wieder graduell an seinen ursprünglichen Wert angepasst.

Das Spurwechselverhalten wird von Theis für jeden der drei oben beschriebenen Fahrzeugtypen (Einfahrer, Ausfahrer, Durchfahrer) separat definiert und jeweils mit zusätzlichen Abstandsbedingungen versehen. So wird beispielsweise der Ausfahrvorgang in drei räumliche Bereiche unterteilt (1000 Meter vor Beginn der Ausfahrt, 500 Meter vor Beginn der Ausfahrt und der eigentliche Ausfahrtverflechtungsbereich). In jedem dieser Bereich sind dann jeweils nur bestimmte Aktionen zugelassen. So wird ein Fahrzeug, dass sich auf den letzten 500 Metern vor der Ausfahrt befindet, keinen Spurwechsel mehr nach links durchführen, um die gewünschte Ausfahrt nicht zu verpassen und den Ausfahrvorgang nicht unnötig zu erschweren. Ähnliche Regeln gelten auch für ein- und durchfahrende Fahrzeuge. Befindet sich ein Fahrzeug außerhalb des Einflussbereichs einer Einfahrt, wird das Spurwechselverhalten von Theis deaktiviert, und das für die freie Strecke gültige Modell (im Normalfall das Modell von Sparmann) wird gewählt.

Insgesamt weist das Theis-Modell einen hohen Detailierungsgrad auf, der weit über den der bisher beschriebenen Modelle hinausgeht. Die einzelnen, situationsspezifischen Verhaltensweisen der verschiedenen Fahrzeugarten werden differenziert abgebildet und ermöglichen so eine relativ realitätsnahe Simulation des Verhaltens im Bereich von Anschlussstellen mit Beschleunigungsbzw. Verzögerungsstreifen. Allerdings wird die Genauigkeitssteigerung durch eine starke Erhöhung der Komplexität erkauft, so dass der Aufwand bei der Umsetzung des Modells in die Simulationsumgebung höher ist als bei allen zuvor betrachteten Verhaltensmodellen.

Die ursprüngliche Implementierung des Modells von Theis erfolgte im Wiedemann und Reiter entwickelte MISSION-Modell ([125]). Als Programmiersprache wurde eine Pascal-Variante verwendet; der Quelltext der für das Spurwechselmodell relevanten Klassen liegt vor ([108]). Detaillierte Angaben über die Einbindung des Verhaltens in das MISSION-Modell (insbesondere über die Fahrspurgeometrien und die Simulationssteuerung) werden bei Theis hingegen nicht gemacht. Wie bei Wiedemann und Sparmann erfolgt die Simulation in konstanten Zeitschritten von einer Sekunde Dauer. Die verschiedenen Aktionen und Situationen (z. B. Einfahren, Fahren, Unterstützen) werden für die Umsetzung in das objektorientierte Modell jeweils in einer eigenen Methode gekapselt, innerhalb der die Entscheidungsfindung durch die Abarbeitung von Entscheidungsbäumen erfolgt.

Im Unterschied zu der im Objektmodell ursprünglich vorgesehenen Trennung zwischen Spurwechsel- und Abstandsmodell stellt das Theis-Modell eine Kombination zwischen beiden Verhaltenstypen dar: Neben den eigentlichen Spurwechselvorgängen (Einfahren, Ausfahren oder

Unterstützen) steuert das Verhalten auch die Geschwindigkeit – unter anderem zum Unterstützen anderer Fahrzeuge, zum Erreichen einer Lücke und zum Anhalten am Ende des Beschleunigungsstreifens. Die strikte Trennung zwischen Abstands- und Spurwechselverhalten, und damit auch deren modulare Austauschbarkeit, wurde durchbrochen. Zusätzlich zum eigentlichen Spurwechselverhalten musste ein hybrides Abstandsverhalten implementiert werden, das auf dem psycho-physischen Modell basiert und im Bereich von Anschlussstellen das voreingestellte Abstandsmodell ersetzt. Obwohl es dadurch nach einigen Anpassungsarbeiten möglich war, das Theis-Modell in die Simulationsumgebung zu integrieren, gingen die dafür erforderlichen Änderungen stark auf Kosten der Übersichtlichkeit, der Erweiterbarkeit und der Modularität des ursprünglichen Systems.

Für geringe Verkehrsstärken liefert das beschriebene Modell eine sehr realistische Simulation des Ein- bzw. Ausfahrverhaltens. Bei steigender Anzahl einfahrender Fahrzeuge führt jedoch die Unterstützung zu einer Verlangsamung des Verkehrs auf der rechten Hauptspur, die bis zum Stillstand führen kann. Durch die entstehende große Geschwindigkeitsdifferenz zwischen linker und rechter Hauptspur wird ein Spurwechsel der unterstützenden Fahrzeuge auf die Überholspur unmöglich gemacht, was die Stausituation auf der rechten Spur zusätzlich verschärft. Versuche, dieses Phänomen durch eine Mindestgeschwindigkeit auf der rechten Hauptspur zu umgehen, führten ebenso wie eine zusätzliche Unterstützung durch die Fahrzeuge der Überholspur nur teilweise zum Erfolg: Das Geschwindigkeitsprofil der einzelnen Fahrstreifen zeigt noch immer erhebliche Unterschiede zwischen den Spuren, so dass die auf einer Auffahrt erreichbaren Verkehrsstärken deutlich unter den erwarteten Werten bleiben [18].

4.4 Beschränkungen existierender Modelle

Die mikroskopische Verkehrssimulation kann mittlerweile auf eine fast fünf Jahrzehnte zählende Geschichte zurückblicken. Viele der zuvor präsentierten Verhaltensmodelle wurden ebenfalls – zumindest vom computertechnischen Standpunkt aus – vor vergleichsweise langer Zeit entwickelt und implementiert. Der damalige Stand der Computertechnik erforderte eine hocheffiziente Nutzung teurer Rechenkapazitäten und Speichermengen. Die Notwendigkeit zur Ressourcenschonung spiegelt sich noch heute in der einfachen Struktur der seinerzeit verwendeten Algorithmen wieder. Der Großteil in der Verkehrssimulation gebräuchlichen Programme verwendeten hauptsächlich das mittlerweile nicht mehr zeitgemäße prozedurale oder imperative Programmierparadigma. Bei der Anpassung der existierenden Verhaltensmodelle an das objektorientierte Modell wurden die Unzulänglichkeiten dieses Programmieransatzes deutlich.

Ein großes Problem, das sich bei allen komplexeren Modellen, insbesondere aber bei der Umsetzung des Modells von Theis zeigte, ist die häufige Verwendung von Alternativkonstruktionen (*if-then-else*) zur Fallunterscheidung. Eine Alternative dient dabei zur Beurteilung des aktuellen Fahrzustandes und der Auswahl einer zugehörigen Reaktion. Jede mögliche Fahrsituation, sei es das Fahren auf freier Strecke, das Einfahren von einem Beschleunigungsstreifen oder das Unterstützen eines ausfahrenden Fahrzeugs, muss dabei durch geeignete Alternativen erkannt und über eine angemessene Reaktion abgefangen werden. Tritt eine nicht vorhergesehene Situation auf (z. B. durch neue Fahrspurgeometrien, ungünstige Kombinationen äußerer Einflüsse), ist das Verhaltensmodell darauf nicht vorbereitet: die Konsequenz ist entweder ein unangemessenes Verhalten oder Inaktivität. Das Fahrmodell verhält sich dabei wie ein diskreter Zustandsautomat: Sobald eine passende Bedingung erfüllt wird, wechselt der Modell in einen neuen Zustand,

wobei immer nur ein Zustand zur gleichen Zeit erfüllt sein kann. Das gleichzeitige Erfüllen mehrerer Zustände (z. B. einfahrendes Fahrzeug unterstützen *und* gleichzeitig ein Überholmanöver durchführen) ist bei diesem Ansatz nicht möglich, oder zumindest nur mit unverhältnismäßig hohem Aufwand verbunden.

Soll ein solches bestehendes Modell erweitert werden, geschieht dies im Wesentlichen durch das Hinzufügen weiterer Fallunterscheidungen. So entsteht, ab einer gewissen Komplexität eines Verhaltensmodells, eine umfangreiche Liste aufeinander folgender *if*-Alternativen, die einen starren Verzweigungsbaum bilden. Abbildung 4.6 zeigt exemplarisch den vom Programm AIMSUN (vgl. Kapitel 2.5.4) für die Spurwechselentscheidung verwendeten Verzweigungsmechanismus ([7]).

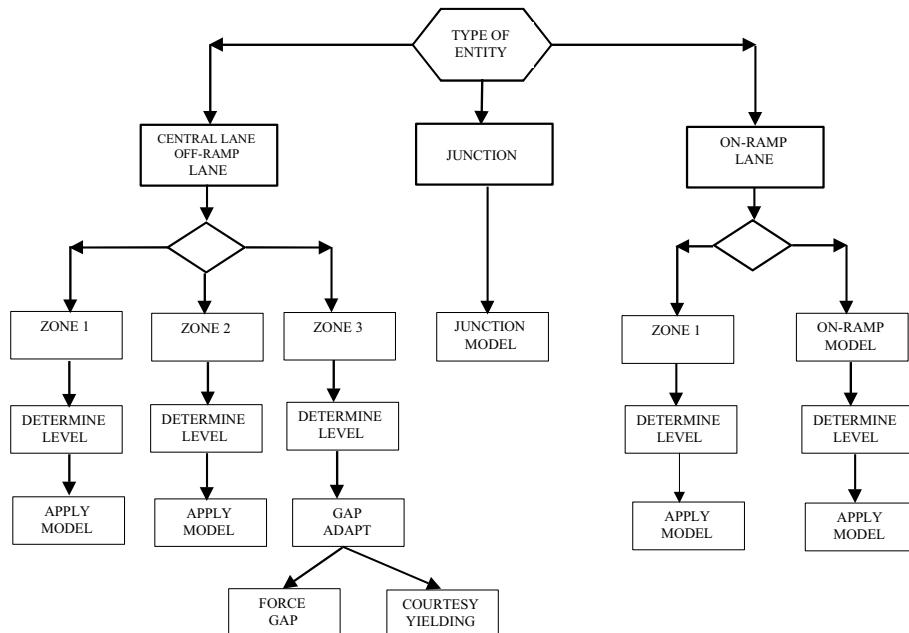


Bild 4.6: Ein starrer Spurwechsel-Entscheidungsbaum in AIMSUN

Der wesentliche Nachteil solcher Alternativ-Konstrukte besteht in der starren Struktur: Eine flexible Reaktion auf unerwartete Ereigniskombinationen ist nicht möglich. Jede Bedingung kann nur entweder „erfüllt“ oder „nicht erfüllt“ sein, unscharfe Zwischenzustände lassen sich mit diesem Ansatz nicht erfassen. Das Verkehrsverhalten realer Fahrer, die zu einem gegebenen Zeitpunkt durchaus mehrere Ziele verfolgen können, wird durch diskrete Entscheidungsbäume nur unzureichend abgebildet. Im Folgenden wird deshalb ein objektorientiertes Verhaltensmodell beschrieben, das in der Lage ist, einen Fahrer gleichzeitig mehrere Ziele oder Absichten verfolgen zu lassen und darauf eine zielgerichtete Reaktion abzuleiten.

Kapitel 5

Absichtsbasiertes Fahrverhalten

5.1 Grundidee

Beim Steuern seines Fahrzeuges muss ein menschlicher Fahrer im täglichen Verkehr verschiedene äußere Einflüsse berücksichtigen und geeignet darauf reagieren. Alle Entscheidungen, die er dabei fällt, zielen in letzter Konsequenz darauf ab, seine persönlichen Ziele zu erreichen. Hauptziel ist dabei das Erreichen des Zielortes seiner Fahrt; neben diesem übergeordneten Ziel existieren aber noch zahlreiche sekundäre Ziele, die ebenfalls so gut wie möglich erfüllt werden sollen, zum Beispiel die Maximierung der eigenen Sicherheit, das Fahren mit möglichst großer Geschwindigkeit oder das Unterstützen anderer Fahrzeuge. Jeder Fahrer muss bei seinen jeweiligen Entscheidungen die Vor- und Nachteile der ihm zur Verfügung stehenden Optionen bezüglich seiner individuellen Ziele abwägen, und sich schließlich für die für ihn günstigste Alternative entscheiden. Treten dabei Konflikte zwischen verschiedenen Interessen auf, müssen geeignete Kompromisse gefunden werden.

Bei Menschen finden derartige Überlegungen zum größten Teil intuitiv oder instinkтив statt. Von Kindheit an lernt man, den persönlichen Nutzen einer Verhaltensweise einzuschätzen und gegenüber anderen Verhaltensweisen abzuwegen. Das dabei im Laufe der Zeit angesammelte Erfahrungswissen erlaubt es, in Sekundenbruchteilen mögliche Aktionen zu bewerten, und aus der Fülle der parallel getroffenen Bewertungen eine möglichst vorteilhafte Aktion auszuwählen. Unter der Annahme, dass man als Mensch beim Verrichten einer speziellen, angelernten Tätigkeit (im vorliegenden Fall das Lenken eines Fahrzeugs) verschiedene vorgegebene Grundziele verfolgt und intern bewertet, lässt sich ein computergestütztes Modell entwickeln, das ein im Vergleich zu existierenden Modellen flexibles und erweiterbares Fahrverhalten aufweist.

Das grundlegende Konzept zu einem derartigen Modell stammt aus einer Arbeit von Hancock ([56]), der sich mit Systemen zum verteilten Schließen (*distributed reasoning*) für den Einsatz in Computerspielen befasste. Hancock wiederum wurde durch die Arbeiten von Rosenblatt inspiriert ([99]), der eine verteilte Architektur für die Steuerung autonomer Roboter entwickelte. Die Übertragung von Forschungsergebnissen aus diesen beiden grundlegend verschiedenen Einsatzgebieten in die Verkehrssimulation ist besonders deshalb interessant, weil alle drei Bereiche vergleichbare Ausgangssituationen aufweisen und gleichartige Probleme zu bewältigen sind: Sowohl die Verkehrssimulation als auch Computerspiele und Robotersteuerung erfordern die Lenkung einer autonomen Einheit (Fahrer-Fahrzeug-Element, Computergegner oder Roboter) in einer komplexen, aber überschaubaren Umwelt (Simulationsumgebung, Spielwelt oder Labor)

unter Ausnutzung sehr begrenzter Rechenkapazitäten und Einhaltung akzeptabler Antwortzeiten.

Beim verteilten Schließen wird die Entscheidungsfindung, die in klassischen System vollkommen zentralisiert von einer einzelnen Entscheidungskomponente getroffen wird, auf eine Menge von selbstständigen Beratern und einen Koordinator aufgeteilt. Jeder Berater arbeitet unabhängig von anderen Beratern und ist dabei auf die Verfolgung verschiedener Ziele spezialisiert. Unter Beachtung der für ihn geltenden Regeln und Bedingungen kann jeder Berater auf Anfrage eine individuelle Einschätzung der Situation vornehmen und seine persönlichen Handlungsempfehlungen abgeben. Alle Empfehlungen werden vom zentralen Koordinator gesammelt und ausgewertet. Aus der Gesamtmenge aller abgegebenen Empfehlungen kann der Koordinator dann in einem „demokratischen“ Abstimmungsprozess – unter Einhaltung gewisser Regeln – eine Handlungsanweisung ableiten.

Keiner der Berater hat dabei Kenntnis von den Entscheidungen der übrigen Berater; alle agieren vollkommen unabhängig voneinander. Kommunikation findet lediglich zwischen Berater und Koordinator statt, nicht aber direkt zwischen den verschiedenen Beratern. Sinn und Zweck dieser strikten Trennung ist es, die Modularität und Erweiterbarkeit des Systems sicherzustellen und unnötige Abhängigkeiten zwischen den einzelnen Berater-Modulen zu vermeiden. Ein weiterer Vorteil ist die Flexibilisierung des Systems: So steht es den einzelnen Beratern frei, ob und wann sie eine eigene Empfehlung abgeben wollen. Inaktive Berater, die sich nicht an der Abstimmung beteiligen wollen – etwa weil ihre Meinung zum gegenwärtigen Zeitpunkt nicht relevant ist – werden bei der Entscheidungsfindung auch nicht berücksichtigt. Wie die einzelnen Berater intern zu ihrer jeweiligen Entscheidung kommen, ist allein von der Art der Implementierung abhängig. So wäre es beispielsweise denkbar, einen Berater mit einem neuronalen Netz auszustatten, einen zweiten über ein Fuzzy-System Entscheidungen fällen zu lassen und einen dritten mit einem Bayes-Klassifikator zu versehen. Die reibungslose Kommunikation zwischen Berater und Kommunikator wird durch eine gemeinsame wohldefinierte Schnittstelle sichergestellt.

Um sicherzustellen, dass verschiedene Aktionen auch zeitlich parallel zueinander durchgeführt werden können (ein Pkw kann z.B. während eines Spurwechselvorgangs auch beschleunigen), wird für jede mögliche Aktion separat eine Abstimmung durchgeführt. Daher muss auch jeder Berater für jede betrachtete Aktion eine unabhängige Bewertung abgeben oder sich der Abstimmung für diese Aktion enthalten. Tritt nach einer Abstimmung ein Gleichstand zwischen zwei widersprüchlichen Aktionen auf (z.B. Spurwechsel nach links und gleichzeitig nach rechts), muss der Koordinator eine Entscheidung treffen und eine der konkurrierenden Aktionen auswählen. Wird zu einer Aktion überhaupt keine Bewertung abgegeben, wird die Aktion auch nicht durchgeführt.

Als Beispiel für die Anwendung des verteilten Schließens nennt Hancock das Lenken eines Fahrzeugs, das sich – mehr oder weniger frei – in einer simulierten Umgebung bewegen kann. Für eine endliche Menge vorgegebener Lenkrichtungen (z. B. stark nach links, geradeaus, schwach nach rechts) werden die Berater nach ihren subjektiven Einschätzungen des Nutzens jeder einzelnen Richtung befragt. Ein „hoher Nutzen“ liefert eine positive Bewertung, eine ablehnende Einschätzung („Veto“) wird durch einen hohen negativen Wert repräsentiert. Um aus der Gesamtmenge aller abgegebenen Bewertungen eine Handlungsempfehlung abzuleiten, empfiehlt Hancock eine gewichtete Summe (entspricht der größten Zustimmung durch die Berater) oder einen Minimum-Operator (entspricht der geringsten Ablehnung) zur Verrechnung der Einzelwerte. Aus der sich ergebenden Funktion wird der Maximalwert interpoliert und als empfohlene

Lenkrichtung zurückgegeben. Abbildung 5.1 verdeutlicht diesen Vorgang ([56]).

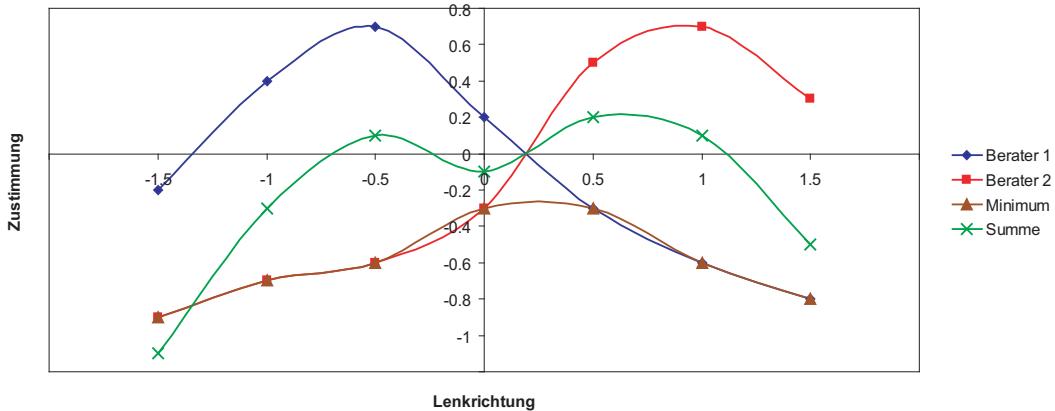


Bild 5.1: Abstimmung zweier Berater mit zwei Koordinatoren (nach Hancock)

Ein Nachteil dieses Verfahrens liegt darin, dass jeder Berater gegebenenfalls eine große Anzahl verschiedener Möglichkeiten überprüfen muss, was sich negativ auf die Performanz auswirkt. Außerdem müssen die verwendeten Algorithmen oder Heuristiken in der Lage sein, für viele verschiedene Aktionsmöglichkeiten differenzierte Bewertungen vornehmen zu können. Auch die Umsetzung einer diskreten Aktionsentscheidung (soll eine Aktion durchgeführt werden oder nicht?) ist mit dem beschriebenen Verfahren nicht möglich, da stets eine Menge an alternativen Ausprägungen der jeweiligen Aktionen vorhanden sein müssen. Für den Einsatz in der Verkehrssimulation wurde daher ein vereinfachter Ansatz verwendet, der die Vorteile des verteilten Schließens (modulare Struktur, leichte Erweiterbarkeit, transparente Entscheidungsfindung) nutzt, ohne die beschriebenen Nachteile aufzuweisen.

5.2 Lösungsansatz

Um die bisher betrachteten monolithischen Fahrverhaltensklassen in kleinere Module aufzuteilen, wurde der Begriff der *Absicht* eingeführt. Eine Absicht ist ein kleines, in sich abgeschlossenes Teilelement eines Fahrverhaltens, das für einen wohldefinierten Bereich der Verhaltensmodellierung zuständig ist. Absichten führen keine selbstständigen Aktionen aus, sondern geben stets nur Handlungsempfehlungen an die übergeordnete Verhaltensklasse ab, die dabei die Aufgabe des Koordinators übernimmt. Dieses sogenannte absichtsbasierte Fahrverhalten setzt sich also aus einer beliebigen Anzahl verschiedener Absichten zusammen, die bei Bedarf nach ihrer Meinung befragt werden können.

Unter dem Begriff „Absicht“ werden im Folgenden alle Instinkte, Ziele oder Wünsche eines menschlichen Fahrers zusammengefasst, die dessen Verhalten während der Fahrt bestimmen. Absichten können sowohl zielgerichtet (wie erreiche ich meine nächste Ausfahrt, wann überhole ich meinen Vordermann?) als auch präventiv wirken (Verhinderung gefährlicher Spurwechsel oder Folgeabstände). Dabei ist es bei komplexen Vorgängen auch durchaus möglich, dass sich ein reales Ziel (z. B. das Erreichen der nächsten Ausfahrt) aus mehreren modularen Absichten zusammensetzt (z. B. Folgeabstand sichern, Seitenabstand sichern und Route folgen). Da alle Absichten zur gleichen Zeit aktiv sein können und so beliebige Kombinationen von Absichten möglich sind, können komplexe Vorgänge aus vergleichsweise einfachen Teilmodulen zusammengestellt werden.

5.3 Mathematische Betrachtung

Gesucht ist ein deterministisches Verfahren, um aus einer Menge abgegebener Bewertungen oder Empfehlungen eine resultierende Gesamtbewertung zu ermitteln. Dabei sollen sowohl verstärkende (positive) als auch ablehnende (negative) Bewertungen möglich sein. Um ein größeres Spektrum möglicher Beurteilungen zu ermöglichen, sollen neben diskreten Bewertungen (*dafür*, *dagegen*) auch unscharfe Werte (z. B. *stark dafür*, *schwach dagegen*) zugelassen werden.

Sei $b_{n,m}$ die Bewertung einer Aktion m durch eine Absicht n . Der Wert der Bewertung b sei gegeben durch eine reelle Zahl aus dem Intervall $[-1; 1]$, wobei positive Werte eine Unterstützung und negative eine Ablehnung der Aktion m repräsentieren. Der Absolutbetrag $|b_{n,m}|$ gibt an, wie stark die Absicht n die Aktion m befürwortet bzw. ablehnt. Für $b_{m,n} \equiv 0$ ist die Absicht n bezüglich der Aktion m indifferent. Ist $|b_{m,n} \equiv 1|$, so liegt eine maximale Unterstützung bzw. Ablehnung vor.

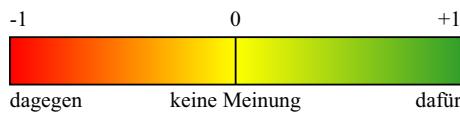


Bild 5.2: Unscharfes Bewertungsschema für das Absichtsverhalten

Die resultierende Bewertung a_m einer Aktion ergibt sich als Funktion der verschiedenen Einzelbewertungen: $a_m = f(b_{m,n})$. Diese Funktion muss dabei so gewählt werden, dass sie folgende Randbedingungen erfüllt:

1. Positive Bewertungen erhöhen die resultierende Bewertung.
2. Bewertungen mit dem Wert 0 (indifferent, *keine Meinung*) beeinflussen das Endergebnis nicht.
3. Negative Bewertungen senken die resultierende Bewertung.
4. Maximale negative Bewertungen ($b_{m,n} \equiv -1$) wirken prohibitiv und erzwingen automatisch eine negative Gesamtbewertung: $a_m = -1$.

Eine Funktion, die den genannten Anforderungen genügt und dabei nur geringen Rechenaufwand erfordert, basiert auf einem verschobenen geometrischen Mittel der Einzelwerte:

$$a_m = \sqrt[n]{\prod_i (b_{m,i} + 1)} - 1 \quad (5.1)$$

Das ursprüngliche Intervall $[-1; 1]$ wird für diese Berechnung in das Intervall $[0; 2]$ verschoben, und nach der Multiplikation der Einzelwerte wieder in das Ursprungsintervall zurückgerechnet. Es ist leicht ersichtlich, dass durch diese Verschiebung alle Bewertungen mit dem Wert 0 zum neutralen Element 1 der Multiplikation werden (vgl. Bedingung 2), und alle maximal negativen Bewertungen (-1) eine Multiplikation mit 0 hervorrufen, was zu einer insgesamt ablehnenden Bewertung von -1 führt (vgl. Bedingung 4). Vormals positive Bewertungen führen zu einer Multiplikation mit einer Zahl > 1 , also zu einer Verstärkung der Gesamtbewertung (vgl. Bedingung 1). Entsprechend rufen negative Bewertungen eine Multiplikation mit einer Zahl aus dem Intervall $[0; 1]$ hervor, die das Endergebnis folglich verringert (vgl. Bedingung 3).

Wird für die zu bewertende Aktion eine diskrete Entscheidung verlangt (z. B. ob ein Spurwechsel durchgeführt werden soll oder nicht), wird ein Schwellenwert $a_{m,min}$ festgelegt, der überschritten werden muss, damit eine Aktion durchgeführt werden kann:

$$a_m \geq a_{m,min} \quad (5.2)$$

Im Rahmen der Kalibrierung des absichtsbasierten Fahrverhaltens hat sich dabei ein Schwellenwert von $a_{m,min} = 0,9$ bewährt; dieser ist aber von der jeweiligen Anwendung abhängig und im Einzelfall neu zu bestimmen.

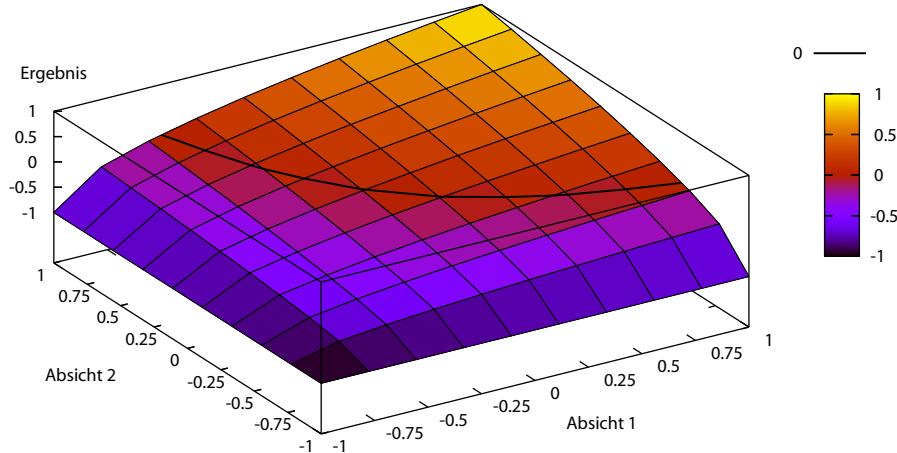


Bild 5.3: Verknüpfung zweier Einzelbewertungen zu einer Gesamtbewertung

Abbildung 5.3 illustriert den Zusammenhang zwischen zwei Absichtsbewertungen und der resultierenden Gesamtbewertung: Es zeigt sich, dass eine negative Bewertung einen deutlich stärkeren Einfluss auf das Gesamtergebnis besitzt als eine positive Bewertung. Diese „Schiefe“ der Auswertungsfunktion ist gewünscht, da sie dafür sorgt, dass gefährliche Situationen zuverlässig ausgeschlossen werden, und Aktionen nur dann ausgeführt werden können, wenn entweder keine ablehnenden Bewertungen vorhanden sind oder die wenigen (schwachen) Ablehnungen durch eine ausreichende Anzahl positiver Bewertungen kompensiert werden.

Der Wurzelterm der geometrischen Mittelung in Gleichung 5.1 sorgt für eine erneute Normierung des Abstimmungsergebnisses auf das Standard-Bewertungsintervall $[-1; 1]$. Dadurch ist sichergestellt, dass einzelne Bewertungen keinen zu großen Einfluss auf ein Abstimmungsergebnis erhalten - die Wurzel „demokratisiert“ die Ergebnisfindung, Einzelmeinung treten in den Hintergrund. Als Konsequenz wird es bei einer großen Anzahl von Absichten schwierig, genügend Stimmen für die Initiierung einer Aktion zu gewinnen – das Verhalten wird insgesamt „passiviert“. Für Anwendungsfälle, bei denen dies nicht wünschenswert ist, kann auf das Ziehen der Wurzel verzichtet werden, was zu der folgenden Formel führt:

$$a_m = \min \begin{cases} (\prod_i (b_{m,i} + 1)) - 1 \\ 1 \end{cases} \quad (5.3)$$

Jede beratende Absicht besitzt nun ein stärkeres Mitspracherecht, und kann dadurch das Abstimmungsergebnis stärker beeinflussen als bei der geometrisch gemittelten Variante (Formel 5.1). Selbst eine einzelne Absicht kann die Durchführung einer Aktion einleiten, solange keine andere Absicht einen Einspruch gegen diese Aktion einlegt. Tabelle 5.1 verdeutlicht das Abstimmungsverhalten an einigen exemplarischen Bewertungs-Tupeln. Hierbei wird aus jeweils

vier verschiedenen Absichtsbewertungen mit Hilfe von Formel 5.3 ein Abstimmungsergebnis berechnet.

Beispiel	Nr.1	Nr.2	Nr.3	Nr.4	Nr.5
Absicht 1	0.60	0.50	-0.50	0.00	0.20
Absicht 2	0.00	0.20	-0.20	-0.50	0.20
Absicht 3	0.00	0.10	-0.10	0.50	0.50
Absicht 4	0.00	0.00	0.00	0.00	-0.60
Ergebnis	0.60	0.98	-0.64	-0.14	-0.20

Tabelle 5.1: Beispiele für Abstimmungsergebnisse in Abhängigkeit der Einzelbewertungen

- **Beispiel 1:** Neutrale Bewertungen haben keinen Einfluss auf das Ergebnis, nur Werte ungleich Null werden berücksichtigt.
- **Beispiel 2:** Mehrere positive Bewertungen verstärken sich. Das Abstimmungsergebnis ist größer als die Summe seiner Einzelbewertungen.
- **Beispiel 3:** Negative Bewertungen führen ebenfalls zu einem stärker ablehnenden Ergebnis, allerdings ist der Verstärkungseffekt geringer als bei positiven Bewertungen.
- **Beispiel 4:** Negative Bewertungen gehen stärker in das Abstimmungsergebnis ein als betragsmäßig gleichstarke positive Bewertungen.
- **Beispiel 5:** Eine einzelne stark negative Bewertung ist ausreichend, um mehrere positive Bewertungen zu überstimmen.

Das beschriebene Verfahren bietet – ähnlich wie die von Zadeh entwickelte Fuzzy-Logik ([128]), die ebenfalls in der mikroskopischen Verkehrssimulation eingesetzt wird (vgl. [126][93]) – eine einfache Möglichkeit, aus einer Menge qualitativer bzw. unscharfer Werte über einen deterministischen Algorithmus einen scharfen Ausgangswert zu berechnen. Um die entwickelte Inferenzformel mit sinnvollen Ausgangsdaten zu versorgen, müssen die erforderlichen Aktionen und Absichten identifiziert und für die Simulation im Form einer Verhaltensklasse realisiert werden.

5.4 Implementierung

Lag die Aufgabe der Fahrzeugsteuerung bei den klassischen Konzepten (vgl. Kapitel 4) allein bei der Verhaltensklasse (und eventuell einigen wenigen unterstützenden Klassen, vgl. Kapitel 4.3.2), wird sie nun fast vollständig auf die einzelnen Absichten delegiert (vgl. *Delegate-Pattern*, [46]). Alle Absichtsklassen werden dabei von der abstrakten Oberklasse *Absicht* abgeleitet, die eine einheitliche Schnittstelle für die Kommunikation zwischen dem Absichtsverhalten und den Absichts-Unterklassen bereitstellt.

Jede Absicht kann eine Bewertung zu einer kleinen Menge an vordefinierten Aktionen abgeben. Die Aktionen wurden aufgrund der Erfahrungen, die während der Implementierung und Kalibrierung der klassischen Verhaltensmodelle gemacht wurden, festgelegt und decken alle Entscheidungsmöglichkeiten ab, die ein Fahrzeug während seiner Fahrt voraussichtlich zu treffen

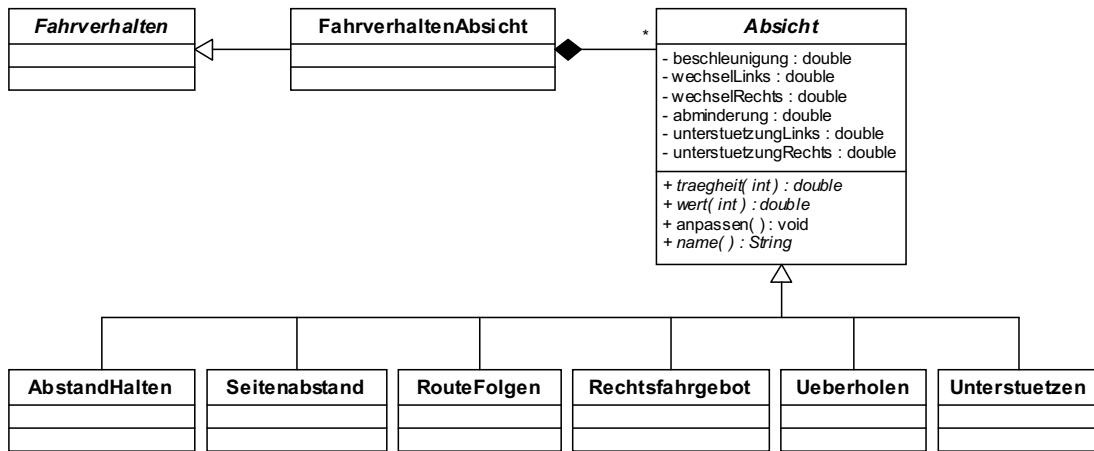


Bild 5.4: Klassendiagramm des absichtsbasierten Fahrverhaltens

hat. Jede mögliche Aktion wird ressourcenschonend über eine einzelne Fließkommavariablen abgebildet, die von der jeweiligen Absichtsklasse mit einem Wert belegt werden kann. Wie dieser Aktionswert nach der Abstimmung zu interpretieren ist (z. B. Einleitung eines Spurwechsels oder Beschleunigung des Fahrzeugs), wird in der Verhaltensklasse festgelegt. Diese zentrale Auswertung in der Abstimmungsklasse ist sinnvoll, da die Handlungsmöglichkeiten eines Fahrzeugs in der Simulation eingeschränkt sind und es nicht zu erwarten ist, dass eine große Anzahl weiterer Aktionen hinzukommen wird. Auf die Einführung einer eigenen Aktions-Klasse wurde deshalb aus Gründen der Ressourcenschonung verzichtet.

Insgesamt wurden sechs mögliche Aktionen bzw. Zustände identifiziert, die sich für eine Festlegung durch einen Abstimmungsprozess eignen. Obwohl sich diese Aktionen in ihren Ausprägungen grundlegend voneinander unterscheiden (diskrete/analoge Entscheidungen, unterschiedliche Wertebereiche), können alle nach dem gemeinsamen Bewertungsmechanismus nach Formel 5.3 ausgewertet werden. Im Einzelnen wurden die folgenden Aktionen definiert:

- **Beschleunigung:** Gibt die von dieser Absicht angestrebte Beschleunigung in m/s^2 zurück. Ein positiver Wert entspricht einer Beschleunigung, ein negativer einer Bremsung. Um eine einheitliche Handhabung der einzelnen Aktionen zu gewährleisten, werden die Beschleunigungen ebenfalls auf das Intervall $[-1; 1]$ normiert, wobei der Wert -1 dem Bremsvermögen und der Wert 1 dem maximalen Beschleunigungsvermögen des Fahrzeugs entspricht. Anders als die übrigen Aktionen wird die Beschleunigung *nicht* durch eine Abstimmung nach Formel 5.3, sondern durch das Minimum aller Bewertungen gebildet. Dadurch soll das voraussichtlich sicherste Verhalten gewählt und ungünstige Abstimmungsergebnisse vermieden werden (vgl. [56], Abbildung 4.15).
- **Wechsel links/rechts:** Gibt an, wie vorteilhaft ein Spurwechsel zum momentanen Zeitpunkt aus Sicht der jeweiligen Absicht wäre. Positive Werte befürworten einen Spurwechsel, negative weisen ihn ab. Gibt eine Absicht den Wert -1 zurück, wird ein Spurwechsel vollständig unterbunden. Spurwechselentscheidungen sind diskret, d. h. Spurwechsel werden nur dann eingeleitet, wenn das Abstimmungsergebnis einen bestimmten Mindestwert überschreitet.
- **Abminderung:** Ermöglicht, dass während und nach der Durchführung eines Spurwechsels auch kleinere Sicherheitsabstände akzeptiert werden. Eine Absicht, die eine Abmin-

derung des Sicherheitsabstands benötigt, wird für diese Aktion eine positive Bewertung aus dem Intervall $[0; 1]$ vornehmen. Ablehnende Bewertungen (< 0) sind zwar theoretisch möglich, werden im verwendeten Modell jedoch nie erforderlich. Eine Abminderung von 1 entspricht einer Halbierung des Sicherheitsabstandes bx (vgl. 4.2.3); bei einer Bewertung von 0 wird der ursprüngliche Wert beibehalten.

- **Unterstützung links/rechts:** Fordert bei einem seitlich benachbarten Fahrzeug eine Unterstützung zur Durchführung eines Spurwechselvorgangs an. Es können nur positive Werte aus dem Intervall $[0; 1]$ angenommen werden. Je größer der Wert, desto dringlicher wird der Unterstützungswunsch. Eine Unterbindung eines Unterstützungswunsches ist in der vorliegenden Implementierung zwar nicht vorgesehen, könnte aber durch Rückgabe eines negativen Wertes einfach realisiert werden. Es kann nach rechts bzw. links nur von jeweils einem Fahrzeug zur gleichen Zeit eine Unterstützung angefordert werden; fordern verschiedene Absichten Unterstützungen von verschiedenen Fahrzeugen an, entscheidet die Verhaltensklasse, welche Unterstützung-Anforderung tatsächlich umgesetzt wird.

Im Gegensatz zu den Aktionen wurde für die Absichten ein Vererbungsmodell vorgesehen, das den gleichzeitigen Einsatz beliebig vieler Absichten erlaubt. Um eine neue Absichts-Klasse zu erstellen, genügt es, eine neue Unterklasse von Absicht zu definieren, die erforderlichen Methoden zu implementieren, und im Konstruktor der Verhaltensklasse eine Instanz dieser Klasse anzumelden. So ist es in vergleichsweise kurzer Zeit möglich, besondere situationsabhängige Verhaltensweisen durch die Einbindung spezieller Absichtsklassen einzubauen.

Für den normalen Verkehr auf Autobahnen wurde eine Grundmenge an Absichtsklassen vorgegeben, die alle wesentlichen Verhaltensmuster auf freier Strecke und im Bereich von Anschlussstellen abdecken:

- **Abstand halten:** Sorgt für einen sicheren Folgeabstand zum Vorderfahrzeug oder einem sonstigen Hindernis im Bereich vor dem Fahrzeug. Verwendet intern das psycho-physische Modell von Wiedemann zur Abstandsbewertung. Spurwechsel werden von diesem Verhalten nicht eingeleitet. Berücksichtigt die im vorherigen Zeitschritt geforderte Abminderung des Sicherheitsabstands.
- **Seitenabstand sichern:** Verhindert Kollisionen oder gefährlich kleine Folgeabstände beim Spurwechsel. Durch diese Absicht kann ein Fahrzeug wirklich nur dann auf die Nachbarspur wechseln, wenn ein Spurwechsel erlaubt und ein ausreichender Abstand zu etwaigen Hindernissen gegeben ist. Die Geschwindigkeit wird durch diese Absicht nicht beeinflusst.
- **Route folgen:** Diese Absicht lässt alle Fahrzeuge Ihrer Wunschroute folgen. Je näher ein Fahrzeug einer angestrebten Ausfahrt kommt, desto größer wird der Wunsch, die Fahrspur zu wechseln, um nicht die Ausfahrt zu verpassen. Droht ein Fahrzeug seine Ausfahrt zu verpassen, weil ein Spurwechsel nicht möglich ist, wird zusätzlich seine Geschwindigkeit und der akzeptierte Sicherheitsabstand abgemindert. Kann ein Spurwechsel nicht ohne fremde Hilfe durchgeführt werden, fordert das Fahrzeug Unterstützung durch das seitliche Hinterfahrzeug an.
- **Rechtsfahrgebot:** Stellt sicher, dass ein Fahrzeug – soweit möglich – nach einer gewissen Zeit den Wunsch entwickelt, auf die rechte Spur zu wechseln. Außerdem wird das nach der

StVO verbotene Rechtsüberholen bei einer Geschwindigkeit über 60 km/h unterbunden. Diese Absicht basiert zum Teil auf dem von Sparmann entwickelten Spurwechselmodell.

- **Überholen:** Entwickelt bei Fahrzeugen den Wunsch, langsamere Fahrzeuge zu überholen, so lange die Fahrzeuge auf der linken Spur schneller sind als auf der aktuellen Spur. Basiert ebenfalls auf Teilen des Spurwechselmodells von Sparmann. Ist die linke Spur deutlich schneller als die aktuelle, wird auch ein verminderter Sicherheitsabstand beim Spurwechsel akzeptiert. Eine Änderung der Beschleunigung wird von dieser Absicht nicht angestrebt.
- **Unterstützen:** Unterstützt andere Fahrzeuge bei der Durchführung eines Spurwechsels. Jede Absicht kann, falls erforderlich, bei der koordinierenden Verhaltensklasse eine Unterstützung durch ein anderes Fahrzeug beantragen. Dieses wird dann – soweit möglich – versuchen, den Spurwechsel durch die Bildung einer Lücke zu erleichtern – entweder durch Verlangsamung oder durch einen eigenen Spurwechsel.

Nicht jede Absicht muss zwingend zu jeder Aktion eine Meinung abgeben. So nimmt beispielsweise das Abstandsmodell nur Einfluss auf die Beschleunigung des Fahrzeugs, die anderen Aktionen bleiben unbeeinflusst. Tabelle 5.2 liefert eine Übersicht über die Absichten und die von ihnen betrachteten Aktionsmöglichkeiten.

Aktion	Abstand	Seite	Route	Rechts	Überholen	Unterstützen
Beschleunigen	ja	nein	ja	ja	nein	ja
Wechsel links	nein	ja	ja	ja	ja	ja
Wechsel rechts	nein	ja	ja	ja	nein	ja
Abminderung	nein	nein	ja	nein	ja	ja
Unterst. links	nein	nein	ja	nein	nein	nein
Unterst. rechts	nein	nein	ja	nein	nein	nein

Tabelle 5.2: Übersicht über die Absichten und ihre Aktionen

Zu jedem Zeitschritt durchläuft das Absichtsverhalten alle ihm zugeordneten Absichten und befragt diese nach ihren Aktionsbewertungen. Aus allen Bewertungen wird dann, wie zuvor beschrieben, das Abstimmungsergebnis berechnet. Für diejenigen Aktionen, die direkt ausgeführt werden können und sollen, wird das Absichtsverhalten entsprechende Änderungen am Modellzustand einleiten – d. h. eine Beschleunigung des Fahrzeugs oder einen Wechsel der Fahrspur vornehmen. Die anderen Aktionen (Unterstützung, Abminderung) werden erst im nächsten Zeitschritt bei der Bewertung der Fahrsituation durch die betroffenen Absichten berücksichtigt.

Hierbei ergibt sich das Problem, dass das Fahrverhalten Entscheidungen in sehr kurzen Zeitabständen fällen kann. Die Zeit zwischen zwei Bewertungen entspricht der Länge eines Zeitschritts, liegt also zumeist bei einer Zehntelsekunde. Dies kann zu unrealistisch schnellen Aktionsfolgen führen, die nicht den realen Verhaltensmustern entsprechen. Bevor ein menschlicher Fahrer eine Aktion durchführt, muss sich der Wunsch zur Durchführung dieser Aktion zuvor über einen gewissen Zeitraum aufbauen. Dies kann zum einen von der Reaktionszeit eines Fahrers, zum anderen aber auch von Zeit abhängen, die der Fahrer benötigt, um eine Entscheidung zu treffen. Auch ist es sinnvoll, nach der Durchführung eines Spurwechsels erst eine gewisse Mindestzeit einzuplanen, bevor das Fahrzeug einen erneuten Spurwechsel angehen kann.

Um deshalb das Zeitverhalten der Aktionsbewertungen besser steuern zu können, wurde ein Verzögerungsterm, die sogenannte Trägheit, eingeführt. Die Trägheit $d_{m,n}$ gibt an, welchen Widerstand eine Absicht n gegen eine Änderung der Bewertung einer Aktion m aufbringt. Dabei wird von der Überlegung ausgegangen, dass eine radikale Änderung einer Situationseinschätzung (z. B. der Wechsel von starkem Beschleunigen auf eine starke Bremsung) eine längere Entscheidungszeit erfordert als ein kleinere Bewertungsänderung (z. B. schwaches Beschleunigen auf mittleres Beschleunigen). Der Wert $d_{m,n}$ beschreibt dabei die Anzahl an Zeitschritten, die erforderlich ist, um eine Bewertungsänderung um eine Bewertungseinheit (z. B. von 0 auf 1) herbeizuführen. Die effektive Bewertung $\tilde{b}_{m,n}$ entspricht dem Istwert dieser Bewertung, und bewegt sich immer mit konstanter Geschwindigkeit $\frac{1}{d_{m,n}}$ auf den angestrebten Sollwert $b_{m,n}$ zu, bis sie diesen erreicht hat. Mathematisch lässt sich der Einfluss der Trägheit auf die effektive Aktionsbewertung mit folgender Formel beschreiben:

$$\tilde{b}_{m,n}(t + \Delta t) = \begin{cases} \tilde{b}_{m,n}(t) + \frac{1}{d_{m,n}} & \text{wenn } \tilde{b}_{m,n}(t) + \frac{1}{d_{m,n}} < b_{m,n}(t + \Delta t) \\ \tilde{b}_{m,n}(t) - \frac{1}{d_{m,n}} & \text{wenn } \tilde{b}_{m,n}(t) - \frac{1}{d_{m,n}} > b_{m,n}(t + \Delta t) \\ \tilde{b}_{m,n}(t) & \text{sonst} \end{cases} \quad (5.4)$$

Die Trägheit wird in der jeweiligen Absichtsklasse festgelegt, und kann daher für jede Absicht und/oder jede Aktion frei gewählt werden. Aktionen, die eine sofortige Reaktion auf Bewertungsänderungen erfordern, können mit einer geringen Trägheit versehen werden. Andere Aktionen, die langsamer reagieren sollen, bekommen eine große Trägheit (d. h. eine geringe Anpassungsgeschwindigkeit) zugewiesen. Auf diese Weise ist es möglich, Bewertungsfunktionen, die starke Bewertungssprünge zwischen den einzelnen Zeitschritten aufweisen, zu glätten und so Bewertungsspitzen zu eliminieren.

Konkret wurden für alle Spurwechselaktionen hohe Trägheiten angesetzt, um zu schnelle Spurwechselfolgen zu vermeiden, während die Aktion Beschleunigen eine sehr geringe Trägheit vor sieht, um die mit dem Wiedemannschen Modell ermittelten Beschleunigungswerte nicht zu verfälschen. Generell ist die Trägheit ein Parameter, der für jede Absicht und – falls noch größere Feinkörnigkeit gefordert ist – für jede Aktion individuell kalibriert werden muss. Durch den jedoch vergleichsweise geringen Einfluss auf das Bewertungsergebnis sind Abschätzungen für die jeweiligen Trägheiten zumeist ausreichend.

5.5 Diskussion

Das in diesem Kapitel beschriebene Absichtsverhalten wurde entwickelt, weil sich die bisher existierenden Verhaltensmodelle für den Spurwechsel im Bereich von Autobahn-Anschlussstellen als nicht ausreichend herausgestellt hatten. Deshalb wurde nach einem alternativen Modell gesucht, das auch die starken Interaktionen zwischen ein- und ausfahrenden Fahrzeugen berücksichtigt ([19]). Das absichtsbasierte Verfahren bietet die folgenden Vorteile:

- **Modularität:** Statt eines einzelnen, vergleichsweise umfangreichen monolithischen Entscheidungsbaums werden einzelne Module definiert, die jeweils einen semantischen Teilaспект des Gesamtmodells abdecken. Aus der Interaktion der Module ergibt sich wieder das ursprüngliche Gesamtverhalten. Alle Module sind komplett gekapselt und nur durch eine

fest definierte Schnittstelle mit der zentralen Verhaltensklasse verbunden. Direkte Verbindungen zwischen den Modulen existieren nicht, wodurch die Fehlersuche stark vereinfacht wird.

- **Verringerung der Komplexität:** Durch die Zerlegung des Verhaltensmodells wird der Umfang, und damit die Komplexität, der einzelnen Module für den Programmierer merklich reduziert. Die Zusammenführung der Einzelbewertungen zu einem Gesamtergebnis erfolgt nach einem einfachen, für den Benutzer leicht nachvollziehbaren Verfahren, das nur geringe Programmierkenntnisse voraussetzt. Insgesamt werden komplexe Verhaltensmuster durch die Modularisierung leichter beherrschbar.
- **Wiederverwendbarkeit:** Existierende Code-Einheiten lassen sich mit geringem Programmieraufwand in das modularisierte Absichtsmodell einbauen, der erforderliche Anpassungsaufwand ist sehr gering. Die Wiederverwendbarkeit existierender Modelle ist gewährleistet: Im konkreten Beispiel konnte das zuvor entwickelte Abstandsmodell nach Wiedemann in wenigen Minuten in eine eigene Absicht eingebaut werden; die notwendigen Quelltextänderungen beschränken sich dabei auf wenige Zeilen.
- **Erweiterbarkeit:** Das Klassenmodell ist so angelegt, dass neue Absichtsklassen sehr schnell hinzugefügt werden können. Durch den Abstimmungsprozess ist der Einfluss der neuen Absicht auf das Gesamtergebnis überschaubar, eine Beeinflussung der bereits vorhandenen Absichten findet nicht statt. Die Anzahl aktiver Absichten ist prinzipiell unbegrenzt, sodass für jede Sondersituation eine eigene Absichtsklasse vorgesehen werden kann.
- **Übersichtlichkeit:** Ein entscheidender Vorteil ist die Erhöhung der Übersichtlichkeit des Modells. Die für den Entscheidungsprozess wesentlichen Bewertungsinformationen lassen sich zu jedem Zeitpunkt in einer grafischen Form darstellen. So kann auf einen Blick abgelesen werden, welche Absichten ein Fahrer gerade verfolgt, welche Interaktionen zwischen den Einzelbewertungen bestehen und welche Bewertung verantwortlich für die Durchführung bzw. Verhinderung einer Aktion war.

The screenshot shows the Pkw31 software interface with several tabs at the top: Info, Geschwindigkeit, Fahrer, Umgebung, Strecke, and Absichten. The Absichten tab is active, displaying six tables representing different driving intentions:

- Rechtsfahrgesbot:** Shows intentions like 'beschleunigen' (green), 'nach links' (yellow), 'nach rechts' (red), and 'Abminderung' (blue).
- Route folgen:** Shows intentions like 'beschleunigen' (green), 'nach links' (red), 'nach rechts' (green), and 'Abminderung' (blue).
- Abstand halten:** Shows intentions like 'beschleunigen' (red), 'nach links' (yellow), 'nach rechts' (blue), and 'Abminderung' (green). A note indicates a distance of -0,2.
- Seitenabstand sichern:** Shows intentions like 'beschleunigen' (green), 'nach links' (red), 'nach rechts' (red), and 'Abminderung' (blue).
- Überholen:** Shows intentions like 'beschleunigen' (green), 'nach links' (yellow), 'nach rechts' (blue), and 'Abminderung' (blue).
- Unterstützen:** Shows intentions like 'beschleunigen' (green), 'nach links' (yellow), 'nach rechts' (blue), and 'Abminderung' (blue).
- ERGEBNIS:** Shows the final outcome for each intention, with colors corresponding to the highest weighted intention: 'beschleunigen' (orange), 'nach links' (red), 'nach rechts' (red), and 'Abminderung' (blue).

At the bottom center of the interface, there is a button labeled "Folge Fahrzeug".

Bild 5.5: Momentanaufnahme der Absichten eines Fahrzeugs

Abbildung 5.5 zeigt exemplarisch den momentanen Fahrzustand eines Fahrzeugs in der Simulation. Jede Absicht wird dabei in Form einer kleinen Tabelle dargestellt, die wiederum die

aktuellen Bewertungen der einzelnen Aktionen enthält. Das Abstimmungsergebnis, das sich aus den Einzelbewertungen ergibt, wird in einer eigenen, farblich abgesetzten Tabelle dargestellt. Die vertikalen Striche innerhalb der Bewertungszeilen geben dabei den Sollwert, die farbigen Balken den Istwert der Bewertungen an. Eventuell vorhandene Unterstützungsanforderungen werden als Pfeilsymbole in der rechten Tabellenspalte angezeigt. Besonders bei laufender Simulation lässt sich so die Entscheidungsfindung des Fahrers gut beobachten und analysieren – ein Aspekt, der sich bei der Programmierung, Kalibrierung und Fehlersuche als wichtiges Werkzeug herausgestellt hat.

Insgesamt konnten mit dem Konzept der absichtsbasierten Steuerung vielversprechende Erfahrungen gesammelt werden. Die Fortentwicklung und Kalibrierung des Fahrmodells wird durch die Modularisierung erheblich vereinfacht, und die Möglichkeiten zur Erweiterbarkeit lassen erwarten, dass das Modell auch für zukünftige Einsatzgebiete gut geeignet ist. Da das beschriebene Absichtskonzept prinzipiell unabhängig vom Einsatzgebiet ist, ist es grundsätzlich möglich, den vorgeschlagenen Ansatz auch auf andere Gebiete als die Verkehrssimulation (z.B. Tunnelbau oder Baufortschrittsmodelle) zu übertragen. Insbesondere für die per definitionem kernetischen Multi-Agentensysteme bietet sich der Einsatz des Abstimmungsprozesses an.

Kapitel 6

Implementierung

6.1 Einleitung

Bei der bisherigen Beschreibung der Softwaremodellierung wurde ausschließlich die eigentliche Simulationswelt – bestehend aus Fahrzeugen, Fahrern, Fahrspuren, Streckennetzen etc. – betrachtet. Prinzipiell sind diese Klassen bereits ausreichend, um eine Simulation für ein Straßennetz durchzuführen. Die konkrete Instanziierung und Assoziierung der für die Simulation erforderlichen Objekte müsste dann jedoch direkt im Programm-Quelltext erfolgen. Dies setzt zum einen gute Kenntnisse des Klassenmodells und der verwendeten Programmiersprache voraus, und ist zum anderen umständlich und fehleranfällig (ein Beispiel für die programmgesteuerte Netz-Erzeugung ist dem Anhang A zu entnehmen). Um eine vollwertige Simulationsapplikation bereitzustellen, die von einem Benutzer auch ohne Programmierkenntnisse bedient werden kann, muss zusätzlich eine grafische Benutzungsoberfläche bereitgestellt werden, die eine komfortable Erstellung, Darstellung, Bearbeitung, Steuerung und Auswertung von Simulationsläufen ermöglicht.

In diesem Kapitel sollen die zur Unterstützung der Simulation erforderliche softwaretechnische Infrastruktur der Simulationsumgebung dargestellt, und die Implementierung einzelner Komponenten – wie die grafische Darstellung der Simulation und der Datenaustausch mit externen Programmen – genauer untersucht werden. Das Hauptaugenmerk liegt dabei eher auf der informatischen Umsetzung als auf verkehrstechnischen Betrachtungen. Außerdem wird das Laufzeitverhalten der Simulation im Rahmen einer parallelen Berechnung diskutiert.

6.2 Programmstruktur

Das Grundkonzept für die Simulationssoftware sieht eine grafische Benutzungsoberfläche vor, die gleichzeitig sowohl als Entwurfs-, Simulations- als auch Auswertungsumgebung genutzt werden kann. Eine Aufteilung dieser Grundaufgaben auf verschiedene Programmmodulen, wie sie in einigen älteren Simulationspakete Anwendung findet, wird als nicht zweckmäßig erachtet, da zum einen der Arbeitsfluss durch die verschiedenen Teilprogramme unterbrochen wird, und zum anderen jedes Teilprogramm seine eigene Bedienoberfläche erfordert, was das Erlernen der Bedienung unnötig erschwert. Aus diesen Überlegungen heraus entstand der sogenannte NetzEditor, der als zentrale Entwurfsoberfläche dient und eine konsistente Benutzerführung

gewährleistet.

Um das Programmsystem erweiterbar zu halten und zu große Abhängigkeiten zwischen der Benutzungsschnittstelle und dem Simulationsmodell zu vermeiden, wird das in der Informatik anerkannte *Model View Controller*-Paradigma (MVC) verwendet, das bereits Ende der 1970er Jahre von der Programmiersprache Smalltalk-80 verwendet wurde([92][22]). Das MVC-Architekturmuster unterteilt eine Anwendung in drei logische Komponenten: In ein grundlegendes Datenmodell, das den darzustellenden Sachverhalt beschreibt (*Model*), in eine Darstellungskomponente, die eine grafische Repräsentation des Datenmodells erzeugen kann (*View*), und in eine Steuerungskomponente, die das Datenmodell verändern und die Darstellungskomponente aktualisieren kann (*Controller*).

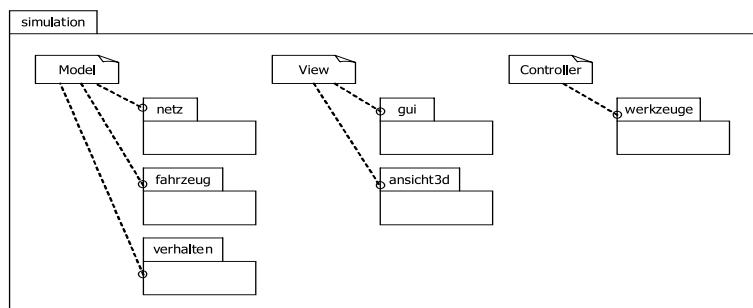


Bild 6.1: Paketdiagramm der Simulationsumgebung

Im Idealfall werden alle drei Komponenten so ausgelegt, dass sie leicht durch andere Komponenten ausgetauscht werden können (z. B. durch Einführung einer neuen Darstellungsform oder eine alternative Art der Steuerung). Besteht eine enge Bindung zwischen der View- und der Controller-Komponente, fasst man View und Controller zu einer Komponente zusammen; man spricht dann von einem *Document-View*-System. Bei der Implementierung der Simulationsumgebung wird dagegen das klassische MVC-Paradigma verwendet, wobei die Einzelkomponenten wie folgt ausgeprägt sind:

- **Model:** Umfasst das eigentliche Modell der Simulationswelt mit Fahrspuren, Fahrzeugen, Strecken, Verhaltensklassen etc. (vgl. Kapitel 3); es ist aufgeteilt auf die Pakete *simulation.netz*, *simulation.fahrzeug* und *simulation.verhalten*.
- **View:** Die bevorzugte Darstellungsform für die Simulation ist eine zweidimensionale Aufsicht auf das Verkehrsgeschehen; diese Sicht wird zum einen über das Paket *simulation.gui* und zum anderen über Visualisierungsmethoden innerhalb der dargestellten Objekte realisiert wird (vgl. Abschnitt 6.2.2). Darüber hinaus wird eine zweite Visualisierung über ein zusätzliches 3D-Paket erreicht, das alternativ aktiviert werden kann und im Paket *simulation.ansicht3D* definiert wurde (vgl. Abschnitt 6.2.3).
- **Controller:** Über den Controller steuert der Benutzer die Simulation mit einem vordefinierten Satz an Werkzeugen und Befehlen, die zum einen das Modell beeinflussen, zum anderen aber auch die Visualisierung steuern. Der Großteil der Werkzeuge ist speziell auf die zweidimensionale Visualisierung zugeschnitten, einige der Werkzeuge können aber auch direkt in der 3D-Darstellung eingesetzt werden. Alle Werkzeuge, Befehle und Optionen befinden sich im Paket *simulation.werkzeuge*. Die Steuerung der 3D-Ansicht geschieht direkt im Paket *simulation.ansicht3D*.

Daneben existieren noch zahlreiche weitere Komponenten, die auf in sich geschlossene Anwendungsbereiche spezialisiert sind. Hierzu gehören u. a. Pakete für die Auswertung, die Parallelisierung oder die Kalibrierung der Simulation. Durch die Aufteilung in Pakete wird die Modularisierbarkeit des Programmes gewährleistet, sofern sichergestellt ist, dass jedes Paket soweit wie möglich in sich geschlossen ist und zwischen den einzelnen Paketen nur wenige, wohldefinierte Interaktionspunkte bestehen. Auf diese Weise ist es mit nur geringem Aufwand möglich, Pakete für die dreidimensionale Darstellung oder die Videodarstellung zu entkoppeln und aus dem System zu entfernen, um so Programmversionen zu erzeugen, die spezifisch auf die Bedürfnisse eines bestimmten Anwendungsfalls ausgelegt sind, ohne unnötige Ressourcen zu belegen. In Tabelle 6.1 sind alle wesentlichen Pakete der Simulationsumgebung zusammengefasst.

simulation	Das Hauptpaket mit dem NetzGenerator und weiteren Hauptklassen
simulation.ansicht3D	Die dreidimensionale Darstellungskomponente
simulation.auswertung	Enthält Klassen zur Generierung von Auswertungsberichten
simulation.fahrzeug	Stellt Klassen für die Umsetzung von FFE bereit
simulation.gui	Klassen für die zweidimensionale Darstellung der Simulation
simulation.netz	Beinhaltet alle fahrspur- und streckenbezogenen Klassen
simulation.optimierung	Werkzeuge für die automatisierte Kalibrierung der Simulation
simulation.parallel	Ermöglicht die parallele Simulation auf mehreren Rechnern
simulation.tools	Diverse Hilfsklassen
simulation.verhalten	Enthält alle Spurwechsel- und Fahrverhaltensklassen
simulation.video	Werkzeuge für die Erstellung von Videosequenzen
simulation.werkzeuge	Befehle, Werkzeuge und Optionen zur Steuerung der Simulation

Tabelle 6.1: Übersicht über die Paketstruktur der Simulationsumgebung

6.2.1 Programmbedienung

Sobald die Simulationsumgebung vom Benutzer gestartet wurde, erscheint die Benutzungsoberfläche des NetzEditors (Abbildung 6.2). Diese besteht im Wesentlichen aus einer Konstruktionsfläche für die Erstellung und Darstellung des Streckennetzes sowie einer Menü- und Werkzeugzeile, die alle benötigten Befehle zur Verfügung stellt. Der Grundaufbau entspricht im Wesentlichen den etablierten Benutzungsstandards üblicher Zeichen- oder Konstruktionsprogramme. Die Konstruktion erfolgt in rein grafischer Form durch Benutzung der Computer-Maus; eine textuelle Befehlseingabe (wie sie z. B. in CAD-Programmen wie AutoCAD verwendet wird) ist nicht vorgesehen, da die Eingabe geodätsicher Koordinaten für die Straßenkonstruktion nicht praktikabel ist. Die Anforderungen an die Benutzungsoberfläche der Simulationsumgebung werden im Anwendungsfalldiagramm 6.3 zusammengefasst.

Die Durchführung eines Simulationslaufs umfasst in der Regel die folgenden Teilschritte:

- **Netz erstellen:** Der Anwender konstruiert – entweder ohne eine Vorlage oder auf Basis existierender Daten (z. B. Luftaufnahmen oder Kartenmaterial) – ein neues Streckennetz für den betrachteten Problemfall. Alternativ können auch bestehende Netze geladen oder aus externen Programmen importiert werden (vgl. Kapitel 6.3).
- **Einstellungen vornehmen:** Vor dem eigentlichen Start der Simulation werden alle erforderlichen Parameter eingestellt. Dazu gehören zum einen verkehrstechnische Parameter (Verkehrsstärken, Tages-Ganglinien, Lkw-Anteile, Streckensperrungen, Signalpläne),

zum anderen simulationsspezifische Werte (Zeitschrittweite, gewünschtes Verhaltensmodell, Verhaltensparameter). Während erstere fest dem Streckennetz zugeordnet werden und zusammen mit diesem abgespeichert werden können, sind die letzteren unabhängig von der Netzgeometrie und können in separaten Dateien abgelegt werden.

- **Simulation durchführen:** Zu einem Simulationslauf gehört die Aufzeichnung der bei der Simulation erzeugten Daten. Standardmäßig werden alle Reisezeiten jedes einzelnen Fahrzeugs erfasst und abgelegt. Sollen darüber hinaus strecken- oder querschnittsbezogene Daten aufgenommen werden, müssen vor dem Start der Simulation Beobachtungsgebiete mit festgelegt werden. Nachdem die Simulation gestartet wurde, läuft sie vollkommen autonom bis zum Erreichen einer Zielzeit oder bis zu einem manuellen Abbruch ab. Der Benutzer kann allerdings auch aktiv in den Simulationslauf eingreifen, die Simulation anhalten und bei Bedarf Einstellungsparameter ändern.
- **Simulation auswerten:** Nach Abschluß der Simulation können die aufgezeichneten Daten (Reisezeiten, lokale/momentane Messreihen, Spurwechselhäufigkeiten) analysiert, ausgewertet und in externe Programme (z. B. Microsoft Excel) exportiert werden. Bei Bedarf lässt sich die Simulation auch mit geänderten Anfangsbedingungen neu starten.

Primäres Entwurfsziel bei der Konzeption der grafischen Benutzungsoberfläche war die Gewährleistung guter Bedienbarkeit und hoher Produktivität. Hierzu wurde gezielt Erstellungs werkzeuge zur effizienten Konstruktion von befahrbaren Fahrspuren in Form von Geraden und (Bezier-)Kurven bereitgestellt. Kreisbögen oder Klothoiden wurden bewusst nicht implementiert, da der Einfluss der Fahrspurkrümmung auf die Simulationsergebnisse vernachlässigbar klein ist. Für den Fall, dass Bedarf nach präziseren Trassierungselementen besteht, erlauben die im Klassenmodell verwendeten Vererbungsmechanismen entsprechende Erweiterungen (vgl. Abbildung 3.14).

Soweit möglich wird von einer zweidimensionalen Darstellung des Streckennetzes ausgegangen; nur bei Bedarf werden zur Ergänzung von Informationen externe Dialoge eingesetzt. Als zentrales Bedienelement kommt der 2D-Benutzungsoberfläche daher eine entscheidende Bedeutung zu, was die Berücksichtigung von Performanzbetrachtungen erfordert.

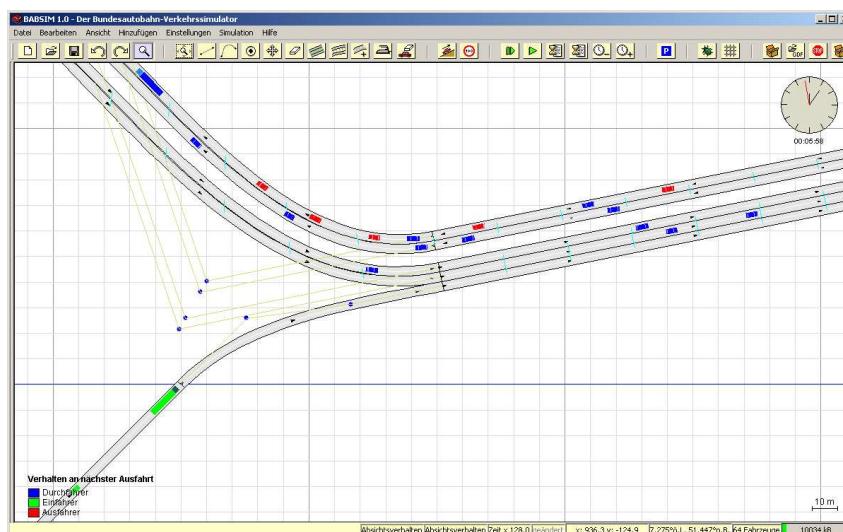


Bild 6.2: Benutzungsoberfläche des NetzGenerators

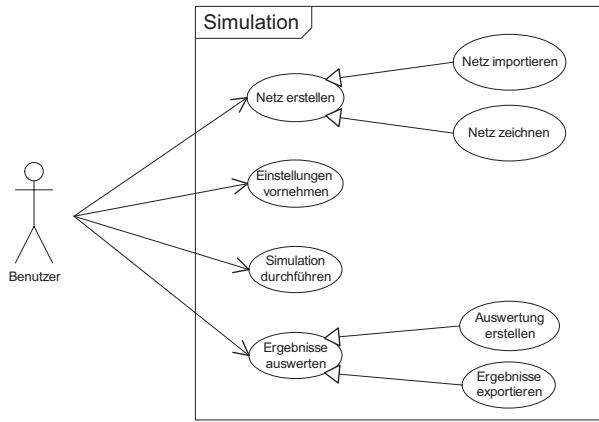


Bild 6.3: Anwendungsfalldiagramm der Simulationsumgebung

6.2.2 Zweidimensionale Darstellung des Streckennetzes

Die Erstellung und Darstellung der Simulationswelt geschieht auf einer zweidimensionalen Übersichtskarte, die eine Draufsicht des Streckennetzes und aller vorhandenen Fahrzeuge zeigt. Da Höheninformationen für die Simulation von vergleichsweise geringer Bedeutung sind (Steigungen beeinflussen lediglich das Beschleunigungs- und Bremsvermögen der Fahrzeuge), reicht eine Projektion der Geometrie auf eine horizontale Ebene für den Großteil der Anwendungen aus. Sollten Vorder- oder Seitenansichten der Szenerie erforderlich sein, wird hierfür die weiter unten beschriebene dreidimensionale Ansicht verwendet.

In der 2D-Ansicht werden alle Objekte dargestellt, die für die Netzerstellung und die Simulation relevant sind: Hierzu zählen alle real sichtbaren Objekte, wie z. B. Fahrspuren, Lichtsignalanlagen, Haltelinien, Gebäude, Fußgänger und Fahrzeuge, aber auch virtuelle Objekte wie Quellen, Senken, Zählpunkte oder Stützpunkte von Bézier-Kurven. Die Darstellung erfolgt auf einer Instanz der Klasse *Bild*, die ihrerseits eine Unterklasse der grafischen Java-Komponente *Canvas* bzw. *JPanel* repräsentiert [35].

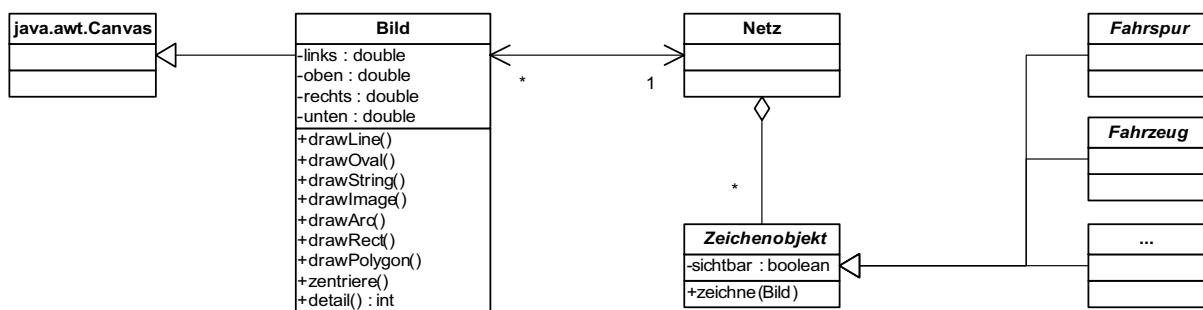


Bild 6.4: Klassenmodell für die zweidimensionale Darstellung

Ein „Bild“ stellt dabei einen rechteckigen Ausschnitt aus der gesamten Simulationswelt dar. Alle grafischen Grundelemente wie Punkte, Linien und Polygone können direkt über die Bild-Klasse gezeichnet werden. Dabei führt das Bild automatisch alle erforderlichen Transformationen aus, um die Zeichenoperationen (denen Weltkoordinaten mit der Einheit Meter zu Grunde liegen) auf das Grafikraster des Bildschirms zu übertragen. Für alle wichtigen von der Java-Klasse *Graphics* bekannten Zeichenbefehle mit ganzzahligen Pixelangaben stehen in der Klasse Bild entsprechende Zeichenbefehle mit reellwertigen Meterangaben zur Verfügung. Dies erleichtert

die grafische Darstellung erheblich, da alle Objekte in ihrem natürlichen Koordinatensystem gezeichnet werden und somit keine vorherigen manuellen Skalierungen oder Translationen mehr erforderlich sind. Welcher Teil der Simulationswelt dargestellt wird, kann durch entsprechende Werkzeuge vom Benutzer festgelegt oder automatisch durch das Programm bestimmt werden (z. B. um den dargestellten Ansichtsbereich auf einzelne Fahrzeuge zu zentrieren).

Beim Zeichnen eines Bildes wird zuerst der Hintergrund gelöscht und ein Gitterraster gezeichnet, dessen Rasterabstand sich dynamisch an den aktuellen Bildausschnitt anpasst. Anschließend werden alle Objekte eines Streckennetzes der Reihe nach durchlaufen und für jedes Zeichenobjekt die Methode `zeichne` aufgerufen (vgl. Abbildung 6.4). Jedes Zeichenobjekt ist selbst für seine Darstellung verantwortlich. Alle Zeichenoperationen werden über die entsprechenden Methoden der Klasse `Bild` durchgeführt. Sind für die Darstellung zusätzliche Informationen erforderlich (aktueller Bildausschnitt, gewünschter Detailgrad, Darstellungsoptionen), muss das Zeichenobjekt sich diese über das Bild oder die Hilfsklasse `Optionen` selbst besorgen. Besteht ein Zeichenobjekt aus mehreren anderen Zeichenobjekten (z. B. eine Kurve mit Stützpunkten), muss das aggregierte Zeichenobjekt selbst dafür Sorge tragen, dass seine Unterobjekte gezeichnet werden. Sobald alle Zeichenobjekte durchlaufen wurden, werden noch eventuell fehlende Bildelemente hinzugefügt (Legende, Uhrzeit, Textmitteilungen, Hilfslinien). Damit ist der Zeichenvorgang abgeschlossen. Abbildung 6.5 zeigt die einzelnen Phasen eines Zeichenvorgangs.

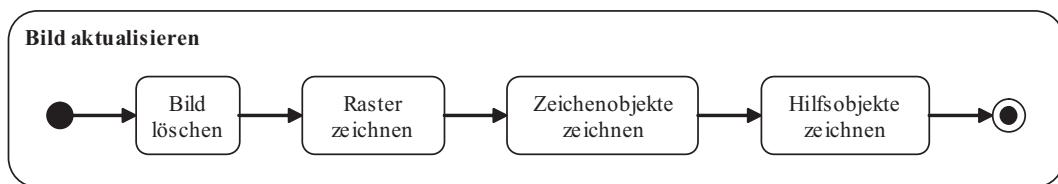


Bild 6.5: Aktivitätsdiagramm des Zeichenvorgangs

Bei der Entwicklung der Klasse `Bild` wurde vor allem auf eine hohe Effizienz der Grafikdarstellung geachtet. Hierfür wurde bewusst ein *low level*-Ansatz gewählt, der auf unnötige Grafikeffekte wie Transparenzen, Linienstile, Farbverläufe und Anti-Aliasing gezielt verzichtet. Durch ausschließliche Verwendung der vom Abstract Windowing Tool (AWT, [35]) bereitgestellten `Graphics`-Klasse ist es möglich, auf einem aktuellen Rechner Zehntausende von Objekten gleichzeitig darzustellen. Die vom AWT durchgeführten internen Optimierungen (Clipping, native Grafikkartenunterstützung) arbeiten derart effizient, dass keine weitere Optimierung der Grafikdarstellung erfolgen musste. Die hier beschriebene Grafik-Architektur konnte auch bereits erfolgreich in anderen Einsatzgebieten weiterverwendet werden ([37]).

6.2.3 Dreidimensionale Darstellung des Streckennetzes

Für den Großteil der Einsatzgebiete ist die oben beschriebene zweidimensionale Darstellung vollkommen ausreichend. Zur Konstruktion und Modifikation eines Streckennetzes bietet die Vogelperspektive eindeutig das größte Maß an Übersichtlichkeit und Bedienkomfort. Auch bei Auswertung der Simulationsabläufe lassen sich viele Phänomene in der Draufsicht schnell überschauen und identifizieren. Dennoch gibt es Fälle, für die eine zweidimensionale Projektion des Verkehrsgeschehens hinderlich sein kann. Daher wird auch eine dreidimensionale Darstellungsfläche vorgegeben, die Unzulänglichkeiten der 2D-Darstellung beseitigt und zusätzliche Vorteile bietet:

- **Plausibilitätskontrolle:** Während die 2D-Ansicht eine gute Übersicht des Verkehrsgeschehens liefert, bietet die 3D-Perspektive die Möglichkeit, sich direkt in die Situation eines Fahrers oder Fußgängers zu versetzen und die Verkehrslage aus dem Blickwinkel des Fahrers zu betrachten. Distanzen und Geschwindigkeiten lassen sich so besser einschätzen. Insbesondere für die Kalibrierung bietet die 3D-Ansicht zusätzliche Möglichkeiten zur optischen Plausibilitätprüfung des Simulationsverlaufs.
- **Erhöhung der Übersichtlichkeit:** Bei komplexen Fahrspurgeometrien, z. B. planfreien Knotenpunkten mit zahlreichen Brückbauwerken, kann eine räumliche Darstellung unterstützend wirken. Durch die Möglichkeit, die Straßenführung aus verschiedenen Blickwinkeln betrachten zu können, wird das räumliche Vorstellungsvermögen des Benutzers gefördert. Bei der Festlegung von Haltelinien an plangleichen Kreuzungen kann der Einfluss von Bauwerken oder sonstigen Hindernissen auf die Sichtweite des Fahrers berücksichtigt werden.
- **Visualisierung zu Präsentationszwecken:** Durch eine ansprechende Darstellung und Animation des Verkehrsgeschehens gewinnt die Simulation, insbesondere für außenstehende Betrachter, an Attraktivität. Zwar kann bei zu starken Detaillierungen der 3D-Visualisierung die Seriosität der Ergebnisse in Frage gestellt werden; bei gemäßigtem Einsatz der 3D-Visualisierung überwiegen jedoch die Vorteile.

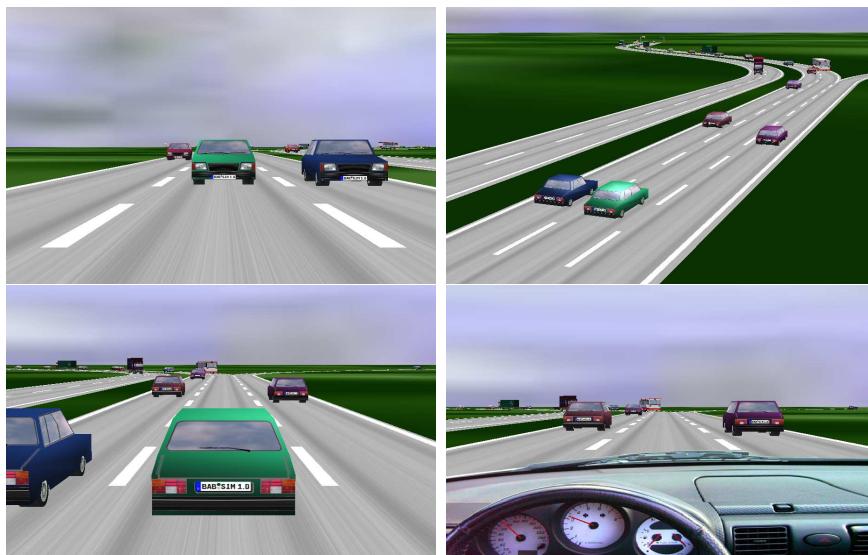


Bild 6.6: 3D-Visualisierung einer Verkehrssituation aus verschiedenen Kamerapositionen

Als Standardeinstellung wird grundsätzlich die zweidimensionale Darstellung gewählt, da diese für die Konstruktion und Modifikation der Fahrspuren verwendet wird. Im Normalfall wird der Benutzer erst nach der Modellierung des Streckennetzes und aller Netzeinstellungen auf die dreidimensionale Darstellung wechseln. Die Aktivierung der 3D-Ansicht geschieht über eine entsprechende Schaltfläche in der Befehlsleiste oder über einen Menüpunkt im Ansichts-Menü. Aus dem aktuellen Netzzustand wird eine dreidimensionale Repräsentation generiert und auf einer neuen Zeichenfläche dargestellt; die zweidimensionale Ansicht wird dabei ausgeblendet. Anschließend kann sich der Benutzer mit Tastatur oder Maus virtuell durch die 3D-Welt bewegen oder mit der virtuellen Kamera einem der Fahrzeuge folgen. Verschiedene automatische Kameraeinstellungen sind entsprechend vordefiniert (vgl. Abbildung 6.6).

Zur Realisierung einer dreidimensionalen Darstellung stehen in der Programmiersprache Java zahlreiche Grafikpakete zur Verfügung. Diese reichen von vergleichsweise einfachen, hardwarenahen Ansätzen (z. B. den Java-OpenGL-Implementierungen JOGL, LWJGL oder GL4Java [127]) über für schnelle Grafik ausgelegte Pakete, wie Xith3D oder jMonkey Engine, bis hin zu umfangreichen wissenschaftlichen Visualisierungswerkzeugen wie VTK ([102]) oder JVView ([80]). Die „naheliegendste“ Lösung ist die Verwendung des Java3D-Paketes, da dieses 1998 direkt vom Java-Entwickler Sun Microsystems entwickelt und veröffentlicht wurde. Seit 2004 ist Java3D im Rahmen des Java Community Process (JCP) als Open Source Projekt frei erhältlich und einsehbar. Die bei der Implementierung verwendete Version ist die Version 1.4 (März 2006, [66]).

Große Vorteile bei der Entwicklung einer objektorientierten Verkehrssimulation verbinden sich mit dem *Szenengraph*-basierten Ansatz von Java3D. Ein Szenengraph ist ein objektorientierter hierarchischer Baum, dessen Knoten die einzelnen Elemente der Simulationswelt repräsentieren (Abbildung 6.8). Der „Wurzelknoten“ umfasst das sogenannte *Virtuelle Universum*, alle anderen Knoten sind direkt oder indirekt abgeleitete Knoten („Kindknoten“) des Universums. Bis auf den Wurzelknoten haben alle Knoten des Szenengraphen genau einen Elternknoten. Durch diese hierarchische Struktur ist es möglich, einzelne Geometrie-Knoten durch gemeinsame Elternknoten zu gruppieren und so Veränderungen gezielt auf ganze Gruppen oder einzelne Knoten anzuwenden. Außerdem wird die Baumstruktur von Java3D für interne Optimierungen (z.B. Zusammenfassung von Transformationsmatrizen, Geometrien etc.) verwendet.

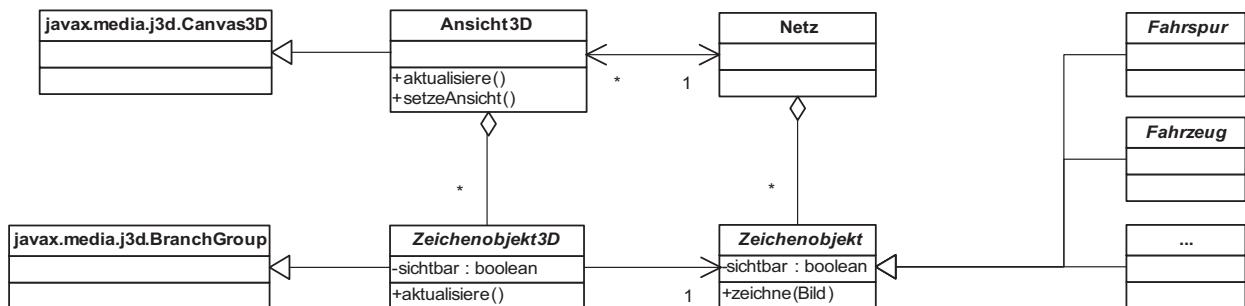


Bild 6.7: Klassenmodell für die dreidimensionale Darstellung

Um die aktuelle Verkehrssituation – einschließlich der Straßengeometrien – in eine dreidimensionale Darstellung zu übertragen, müssen entsprechende 3D-Objekte (sogenannte *Shape3D*-Objekte) erzeugt und in einen geeigneten Szenengraphen einordnet werden. Die Objektstruktur der zweidimensionalen Ansicht kann dabei größtenteils übernommen werden, d. h. für jedes zweidimensionale Zeichenobjekt kann eine zugehörige 3D-Entsprechung gefunden werden. Zuständig hierfür ist die Klasse *Zeichenobjekt3D*, die für jedes gegebene *Zeichenobjekt* einen neuen Knoten für den Szenengraphen erzeugt (vgl. Abbildung 6.7). Für jedes in der 3D-Ansicht sichtbare Zeichenobjekt wurde eine passende *Zeichenobjekt3D*-Klasse definiert (z. B. *Lkw*→*Lkw3D*, *Kurve*→*Kurve3D*).

Abbildung 6.8 zeigt eine einfache 3D-Szenerie, bestehend aus drei Fahrspuren und einem Fahrzeug; die verwendete Notation entspricht der von Sun vorgeschlagene Konvention zur Darstellung von Szenengraphen (siehe [79]).

Da die meisten 3D-Zeichenobjekte aus mehr als einem Geometrieobjekt bestehen – so setzt sich z. B. ein Lkw aus zwei Quadern, sechs Reifen, zwei Bremsleuchten und vier Blinkern zusammen – werden sie als Teilgraphen (*BranchGroups*) direkt der obersten Ebene der eigentlichen

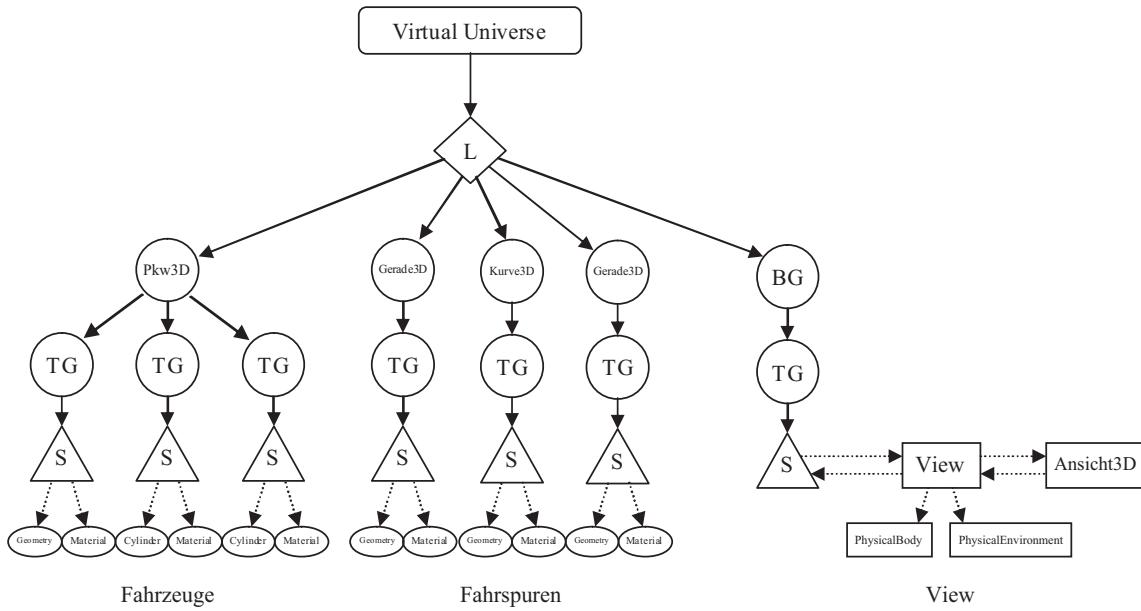


Bild 6.8: Beispiel für einen Java3D Scenegraph mit einem Pkw und drei Fahrspuren

Szene hinzugefügt. Über eine eigene *TransformGroup* kann jedes Zeichenobjekt jederzeit beliebig neu positioniert werden. Muss ein Fahrzeug aus der Simulation entfernt werden, wird die zugehörige BranchGroup bei der nächsten Aktualisierung der 3D-Ansicht automatisch aus dem Szenengraphen gelöscht.

Beim Starten der 3D-Ansicht werden zunächst alle sichtbaren Objekte in eine dreidimensionale Repräsentation konvertiert und dargestellt. Dieser Vorgang dauert beim ersten Wechsel in die 3D-Ansicht einige Sekunden, da zunächst die 3D-Klassen und die Texturen dynamisch nachgeladen werden. Da sich Straßen, Gebäude, Schilder etc. während des Simulationslaufs im Allgemeinen nicht ändern (und eine Manipulation der Objekte größtenteils nur in der 2D-Ansicht erlaubt ist), müssen nur die Fahrzeuge laufend aktualisiert werden. Hierfür werden nach Berechnung eines Simulationszeitschritts alle Transformationsmatrizen der Fahrzeuge aktualisiert, um sie auf die aktuelle Position und Ausrichtung zu bringen. Sollten zwischenzeitlich neue Fahrzeuge aufgesetzt worden sein, wird die benötigte Geometrie erzeugt und in den Szenengraphen eingebunden. Fahrzeuge, die eine Senke erreicht haben, werden entfernt.

Da die Zahl der neu erzeugten und zu entfernenden Fahrzeuge vergleichsweise gering ist und die Aktualisierung der Transformationsmatrizen einen nur sehr geringen Zeitbedarf erfordert, wird für die dreidimensionale Darstellung nur sehr wenig zusätzliche Rechenleistung benötigt. Der Großteil der Rechenlast wird von der 3D-Grafikkarte übernommen, die heute in den meisten Rechnersystemen vorhanden ist. Für Rechner ohne hardwaregestützte 3D-Grafikbeschleunigung oder entsprechende Grafiktreiber (Java3D setzt entweder einen OpenGL- oder DirectX-Treiber voraus) verlangsamt sich die Simulation allerdings deutlich. In diesem Fall kann auf die Nutzung der dreidimensionalen Darstellung verzichtet werden.

6.2.4 Animationen

Neben der interaktiven dreidimensionalen Darstellung der Simulation kann es auch sinnvoll sein, Simulationsläufe in Form von Videodateien persistent zu speichern. Derartige 3D-Animationen

können dann weitergegeben und präsentiert werden, ohne die eigentlich Simulationssoftware zu benötigen. Außerdem können verschiedene Kameraperspektiven und Kameraflüge genutzt werden, um die Anschaulichkeit und Übersichtlichkeit der Simulation zu erhöhen. Durch das Zusammenführen mehrerer Aufnahmen zu einer Videosequenz können zudem visuelle Szenarienvergleiche durchgeführt werden.

Um aus einer laufenden Simulation eine Videoanimation zu erstellen, müssen entsprechend getaktete Einzelaufnahmen der aktuellen Kameraansicht erstellt und hintereinander in einen Videostrom geschrieben werden. In Java steht hierfür das Java Media Framework (JMF) zur Verfügung, das die Erstellung, Bearbeitung und Wiedergabe von Audio- und Videoaufnahmen ermöglicht ([52]).

Als Ausgangsmaterial für die zu erstellenden Videodateien werden Bilder der aktuellen 3D-Ansicht benötigt. Diese werden auf einer zweiten Instanz der Klasse *Canvas3D* gezeichnet, die im sogenannten *Off-screen Rendering*-Modus betrieben wird. Die Maße der Zeichenfläche entsprechen dabei bereits der späteren Videogröße, um zusätzliche Qualitätsverluste durch Skalierungsoperationen zu vermeiden. Alle gerenderten Bilder werden nach und nach dem so-nannten *VideoGenerator* übergeben, der sie in einem separaten Thread mit einem gewählten Videocodec encodiert und in eine AVI-Datei schreibt (*Audio Video Interleaved*). Nach jedem Bild wird die Simulation um einen weiteren Zeitschritt weiterbewegt. Die Länge der Animation wird vom Benutzer vorgegeben. Wurde das letzte Bild der Animation bearbeitet, wird die Animationsdatei geschlossen und kann mit jedem AVI-fähigen Anzeigeprogramm abgespielt werden.

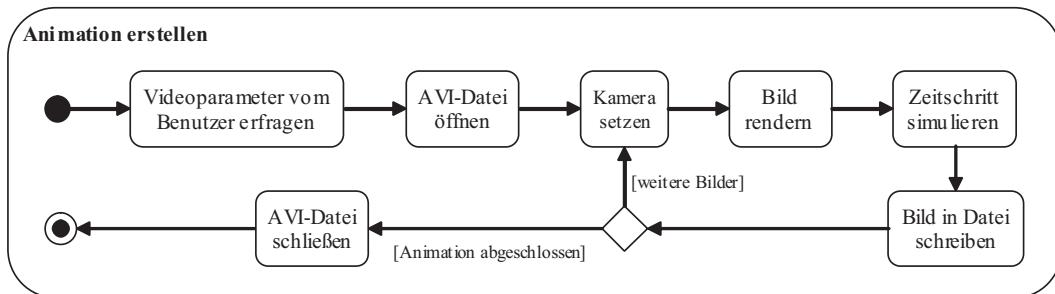


Bild 6.9: Aktivitätsdiagramm der Erstellung einer Animationsdatei

Intern besteht der *VideoGenerator* aus einem *VideoStream*, der die eigentlichen Bilddaten sammelt, und einer *VideoQuelle*, über die der *VideoStream* angefordert werden kann (Abbildung 6.10). Alle während der Simulation erzeugten Einzelbilder werden über den *Videogenerator* und das Java Media Framework in Videodateien gewandelt und in eine Videodatei geschrieben.



Bild 6.10: Klassendiagramm des Videogenerators und seiner Hilfsklassen

Während einer Animation ist es oft wünschenswert, den aktuellen Blickwinkel der virtuellen Kamera zu verändern, um so verschiedene Ansichten zu erstellen, z.B. eine Übersicht der Streckenführung oder einzelne Detailaufnahmen einzelner Fahrsituationen. Hierfür wird eine Zeitleiste eingeführt, welche die Festlegung von sogenannten *Keyframes* erlaubt. Jeder Keyframe repräsentiert dabei eine Position von Stand- und Blickpunkt der virtuellen Kamera zu einem festen Zeitpunkt. Aus der Abfolge von zwei oder mehr Keyframes ergibt sich dann ein Kameraflug durch die simulierte Welt. Für alle Bilder der Animation, denen kein Keyframe zugeordnet ist, wird die Kameraposition aus den beiden benachbarten Keyframes der Zeitleiste interpoliert. Abbildung 6.11 verdeutlicht exemplarisch das Prinzip des Kameraflugs über einen Autobahnabschnitt.

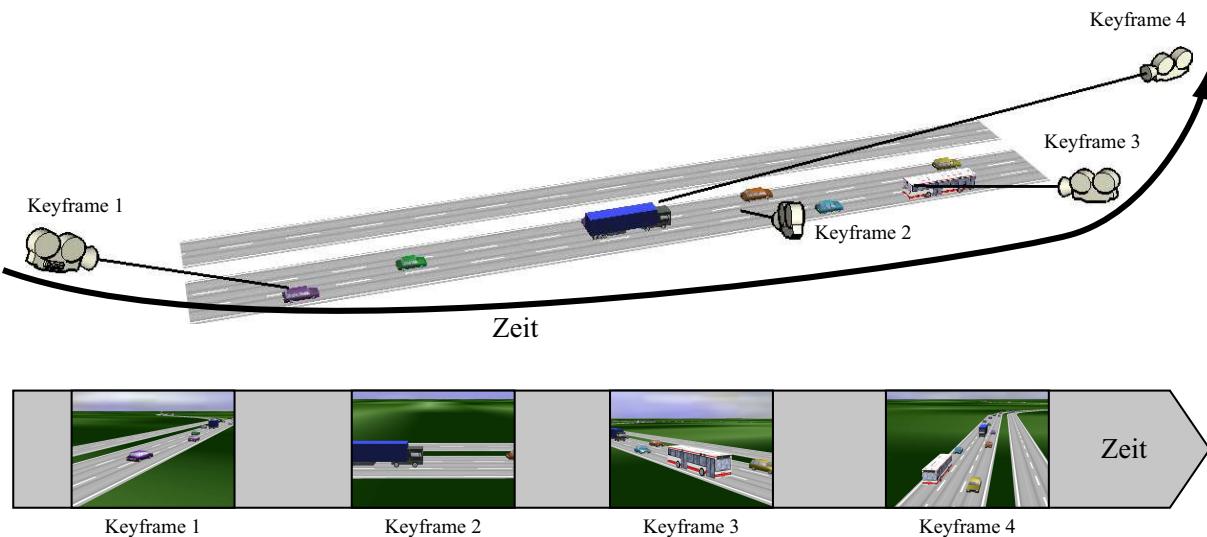


Bild 6.11: Übersicht über den Verlauf einer Animation mit vier Keyframes

6.3 Datenaustausch

Bei der Erstellung der Fahrspurgeometrie eines neuen Streckennetzes ist ein nicht unbeträchtlicher zeitlicher Aufwand erforderlich. Um diesen Aufwand möglichst gering zu halten, wird eine Anbindung an externe, bereits existierende Datenquellen vorgesehen, deren Daten als Grundlage bei der Netzerstellung verwendet werden können. Auch der Export von erstellten Netzen und von Simulationsergebnissen an bestehende Programme ist ein entscheidendes Kriterium für die Akzeptanz der Simulationsumgebung im praktischen Einsatz. Hierbei ist zu beachten, dass eine unüberschaubare Vielzahl verschiedener Formate existiert, die potentiell als Austauschformat für Verkehrssimulationen genutzt werden könnten. Im Folgenden soll daher ein kurzer Überblick über mögliche Standards gegeben und ihre Einsetzbarkeit für die Verkehrssimulation beurteilt werden.

6.3.1 Geographical Data Format (GDF)

GDF ist ein textbasiertes Standardformat zum Austausch von geographischen Datenmodellen, das speziell für den Einsatz in Fahrzeugnavigationssystemen entwickelt wurde und dort auch

heute noch eine breite Anwendung findet ([120]). Die ersten GDF-Versionen wurden bereits im Oktober 1982 im Rahmen des EUREKA-Projekts Demeter erarbeitet ([31]) und seitdem ständig weiterentwickelt. Seit 2004 wird das GDF-Format in der Version 4.0 als ISO-Standard 14825 eingesetzt [1], das einen international zertifizierten Standard für den Austausch von Straßendaten darstellt. Im Bereich der Verkehrssimulation können diese Daten verwendet werden, um ganze Straßennetze automatisiert zu importieren.

Die weltweit wichtigsten Anbieter für Fahrzeugnavigationsdaten sind momentan die Firmen Tele Atlas (Belgien, Niederlande) und NAVTEQ (USA). Auf Basis existierenden Kartenmaterials, Luft- und Satellitenaufnahmen sowie realen Messfahrten vertreiben beide Firmen in Datenbanken abgelegte geometrische, topologische und semantische Informationen über Straßen, Gebäude und das Gelände. Da sich die in Fahrzeugen eingesetzten Navigationssysteme (z. B. Blaupunkt TravelPilot DX, Becker, VDO Dayton, Siemens VDO, Clarion, Pioneer) im internen Datenformat stark voneinander unterscheiden, wird das GDF-Format als einheitliche Datenbasis für die Erstellung systemspezifischer Datensätze verwendet. Für den direkten Einsatz in Navigationssystemen ist GDF aufgrund seiner komplexen Datenstruktur und hohen Speicheranforderungen dagegen nicht geeignet.

GDF-Dateien bestehen aus einzelnen Datensätzen, den sogenannten *Records*, die über relationale Beziehungen miteinander verknüpft sind. Ein Record besteht dabei aus einer oder mehreren Textzeilen mit einer festen Länge von 80 Zeichen. Das letzte Zeichen einer Zeile gibt an, ob der Record beendet ist (0) oder in der nächsten Zeile fortgesetzt wird (1). Neben den eigentlichen Daten enthält eine GDF-Datei immer einen Kopf, in dem der Aufbau und die Syntax der folgenden Records beschrieben wird – die GDF-Datei liefert praktisch eine Beschreibung ihres eigenen Aufbaus mit. Listing 6.1 zeigt einen kurzen Ausschnitt aus einem *Record definition record*. Programme, die eine GDF-Datei einlesen wollen, müssen daher zunächst den Deklarationsteil analysieren, um nachfolgende Datensätze korrekt verarbeiten zu können.

```

0403 FIELDDEFREC11REC_DESCR FLD_NAME FIELD_SIZE DATA_TYPE DATA_UNIT UNIT_EXP 1
00NO_DATA MIN_VAL MAX_VAL DATA_USE FIELD_DESCField definition record 0
0404 RECDEFREC 7REC_DESCR REC_TYPE REC_CODE REC_NAME NUM_FIELD (FLD_NAME) 1
00COMMENT Record definition record 0
0405 ATDEFREC 10REC_DESCR ATT_TYPE FIELD_SIZE DATA_TYPE DATA_UNIT UNIT_EXP 1
00NO_DATA MIN_VAL MAX_VAL ATT_DESC Attribute definition record 0
0407 FEATDEFREC 5REC_DESCR FEAT_CODE FEAT_NAME LAN_CODE FEAT_SYN Feature def1
00init record 0
0415 DATTVALREC 3REC_DESCR ATT_TYPE DATT_VAL Default attribute record 0

```

Listing 6.1: Ausschnitt aus der Record Definition einer GDF-Datei

Jedes reale Objekt kann in einer GDF-Repräsentation in verschiedenen Genauigkeiten vorliegen. Die GDF-Spezifikation unterscheidet zwischen drei verschiedenen Abstraktionsstufen (vgl. Abbildung 6.12):

- **Level 0: Geometry.** Beschreibt den zweidimensionalen Aufbau der Kartenelemente in Form grafischer Primitive (*Nodes*, *Edges* und *Faces*). Linienzüge werden als Folge von Punkten beschrieben, Kurven müssen durch Geradensegmente approximiert werden.
- **Level 1: Simple Features.** Hier werden die grafischen Primitive des Levels 0 mit semantischen Informationen versehen – z. B. kann eine Linie als Straße oder ein Punkt als Kreuzung ausgezeichnet werden. Entsprechend den grafischen Primitiven unterscheidet man zwischen *Point Features*, *Line Features* und *Area Features*.
- **Level 2: Complex Features.** Ein Complex Feature fasst mehrere Simple Features zu einer Einheit zusammen (z. B. mehrere Straßen und Kreuzungen zu einem Knotenpunkt)

und versieht sie mit zusätzlichen Attributen. Zusätzlich können mehrere komplexe Features durch sogenannte *Relationships* einander zugeordnet werden.

Bei der Umwandlung eines realen Straßennetzes in eine GDF-Repräsentation gehen durch die verwendete Abstraktion zahlreiche Detailinformationen verloren; der Datenverlust muss aber in Kauf genommen werden, um Speicheranforderungen zu genügen. Durch die Unterscheidung zwischen verschiedenen Detailstufen können Daten für verschiedene Einsatzgebiete verwendet werden. So dienen die *Level 2*-Daten meist zur Routenberechnung (z.B. in Navigationssystemen), während *Level 1*-Daten den eigentlichen Fahrtrichtungsempfehlungen für den menschlichen zu Grunde gelegt werden.

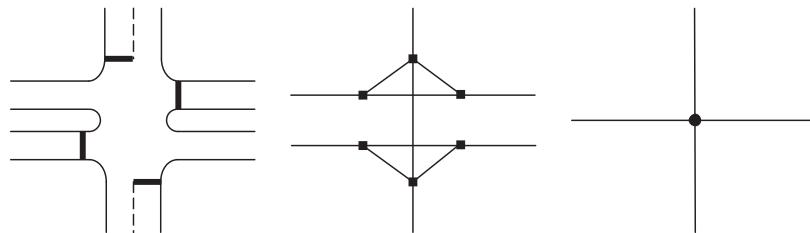


Bild 6.12: Eine reale Kreuzung (links) in GDF Level 1 (Mitte) und Level 2 (rechts)

Um eine gegebene Netzdatei im GDF-Format für die Verkehrssimulation nutzbar zu machen, müssen die einzelnen Records eingelesen, analysiert und in entsprechende Simulationsnetzelemente umgewandelt werden. Da sich die Nutzung des GDF-Formats weitgehend auf die Fahrzeug-Navigationsindustrie beschränkt, existieren bisher keine öffentlich zugänglichen Klassenbibliotheken zum Parsen einer GDF-Datei. Zwar existieren Pläne für eine Weiterentwicklung des GDF-Standards auf Basis des XML-Formats (X-GDF [115]), konkrete Lösungen fehlen jedoch bisher. Für den Import in die Verkehrssimulation ein eigener Java-basierter GDF-Parser entwickelt, der alle für die Netzerstellung benötigten Informationen extrahieren kann. Hierfür wird zunächst die komplette GDF-Datei eingelesen und anschließend in ein objektbasiertes Modell umgewandelt. Es wurde ein Klassenmodell entwickelt, das für jedes bei der Netzerstellung erforderliche GDF-Element eine entsprechende Java-Klasse vorsieht (vgl. Abbildung 6.13).

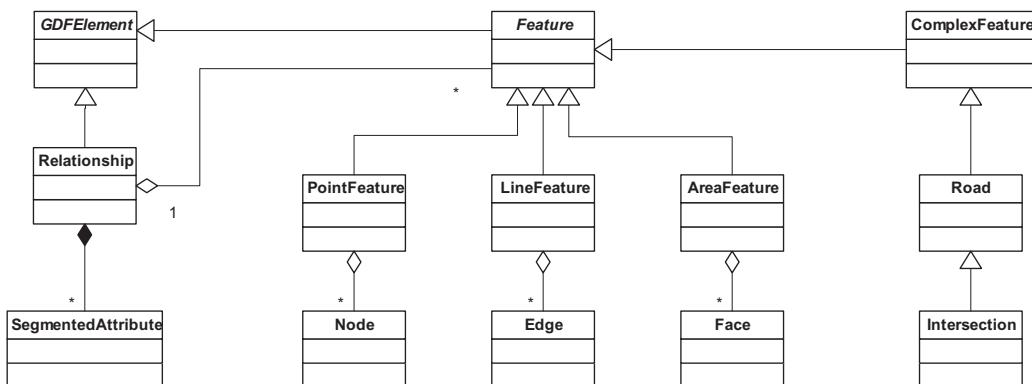


Bild 6.13: Klassen-Repräsentation der wesentlichen Elemente einer GDF-Datei

Sobald das Objektmodell vorliegt, kann die Umsetzung in die Netzstruktur der Verkehrssimulation erfolgen, indem alle gefundenen Straßen und Kreuzungen in entsprechende Fahrspur-Instanzen umgewandelt werden. Straßensegmente, die in beide Fahrtrichtungen befahren werden können, müssen dabei in zwei entgegengesetzte Geraden aufgeteilt werden; Kreuzungen

werden als Abzweige realisiert. Zusätzlich müssen alle benachbarten Fahrspuren ermittelt und miteinander verbunden werden. Abbildung 6.14 zeigt ein importiertes Streckennetz in einer Gesamt- und einer Detailansicht.

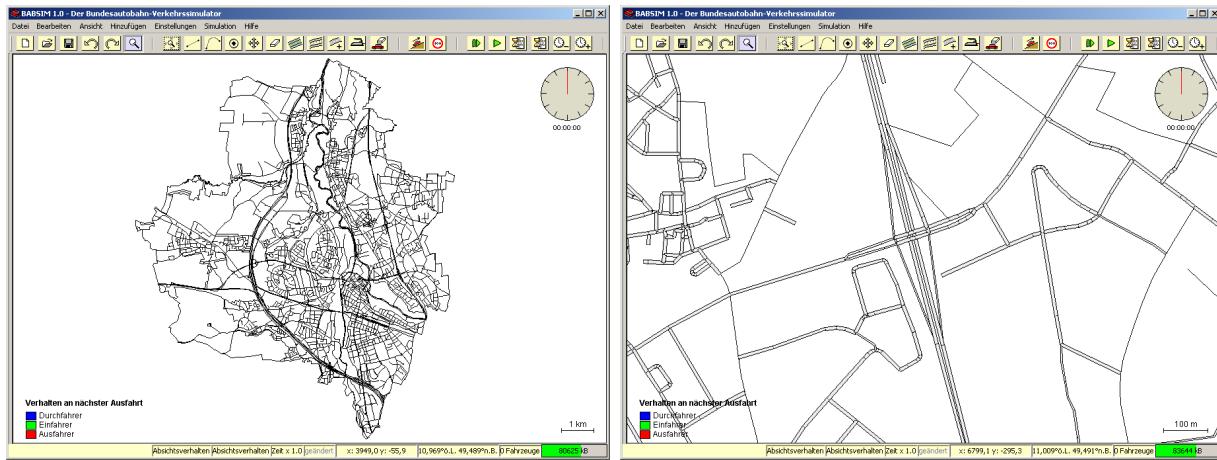


Bild 6.14: Eine importierte Beispiel-GDF-Datei in der Übersicht (links) und im Detail (rechts)

Für eine vollautomatische Importierung eines Straßennetzes ist der Detaillierungsgrad der untersuchten GDF-Dateien zu gering; wichtige Informationen, wie z.B. die Spuranzahl oder der Beginn von Ein- oder Ausfahrten, sind in den GDF-Daten nicht vorgeschrieben. Der in der Verkehrssimulation eingesetzte GDF-Filter erlaubt es nach dem derzeitigen Entwicklungsstand, eine grobe Netztopologie aus einer gegebenen GDF-Datei zu erzeugen. Diese kann als Ausgangspunkt für weitere manuelle Verfeinerungen dienen.

6.3.2 Web Map Service (WMS)

Für die Verbreitung von Geoinformationen stehen im Internet zahlreiche Kartendienste bereit, deren Daten sich potentiell für die Einbindung in eine Verkehrssimulation eignen. Einige weit verbreitete Standards für die Kommunikation mit den Kartenservern wurden vom Open Geospatial Consortium im Rahmen des OpenGIS-Projektes verabschiedet, darunter der *Web Feature Service* für die Einbindung von vektorbasiertem Kartenmaterial ([118]) und der *Web Map Service* zur Erzeugung von Bitmap-Karten ([33]). Da das Importieren von Rastergrafiken nur leichte Anpassungen des Grafiksystems der Verkehrssimulation erfordert, wurde eine Implementierung des WMS-Standards vorgenommen.

Um Luft- oder Satellitenaufnahmen in die Verkehrssimulation einzubinden, wurde ein eigenes Import-Werkzeug entwickelt (Abbildung 6.15, Quellen: NASA/JPL und Regionalverband Ruhr, Essen). Hiermit ist es möglich, aus einer Liste von vorgegebenen WMS-Diensten einen geeigneten Server auszuwählen, von dem die Bilddaten geladen werden sollen. Ausgewählte Kartenausschnitte können dabei stufenlos vergrößert, verschoben, in einer gewünschten Auflösung gespeichert und direkt in die Simulationsumgebung eingebunden werden. Dort kann sie als Konstruktionshilfe zur Netzerstellung, aber auch als Hintergrundbild für die dreidimensionale Darstellung verwendet werden.

Das Laden eines Kartenausschnitts von einem Web Map Service erfolgt über eine einfache HTTP-Anfrage (wahlweise in der POST oder GET-Variante); als Resultat wird ein Bild in

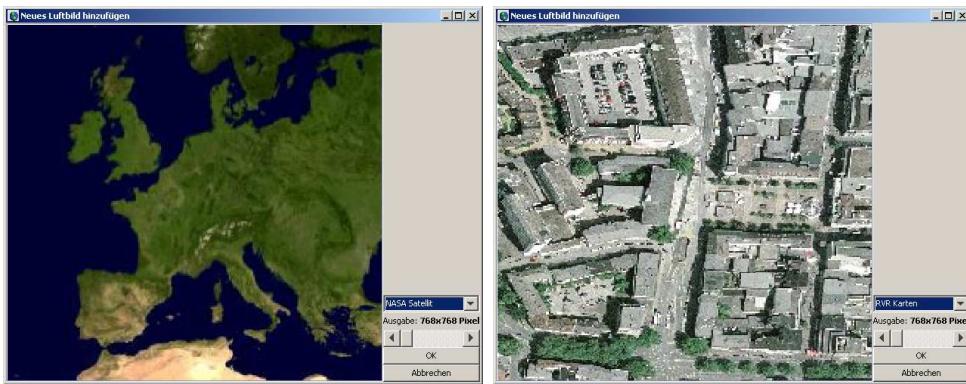


Bild 6.15: Import von Satelliten- (NASA, links) und Luftbildaufnahmen (PVR, rechts)

Form einer Bitmap-Datei (z. B. im JPEG, GIF oder PNG-Format) zurückgeliefert. Informationen über den Bildausschnitt und die Darstellungsform werden mittels mehrerer Parameter spezifiziert. Jeder WMS-Server muss mindestens die beiden Funktionen *getCapabilities* (zum Abfragen von Serverinformationen) und *getMap* (zum Erzeugen einer Kartenansicht) zur Verfügung stellen; optional kann die Methode *getFeatureInfo* zur Abfrage von semantischen Karteninformationen herangezogen werden. Um einen vom Benutzer gewählten Kartenausschnitt darzustellen, wird eine URL der folgenden Form generiert und über eine HTTP-GET-Anfrage an den WMS-Server (hier der Satellitenbildserver des NASA Jet Propulsion Laboratory in Pasadena) gesendet:

```
http://wms.jpl.nasa.gov/wms.cgi?request=GetMap&layers=modis&srs=EPSG:4326&width=512&
height=512&bbox=-15,36,30,61.5&format=image/jpeg&version=1.1.0&styles=
```

Listing 6.2: Beispiel-URL für den Zugriff auf einen Web Map Service

Damit die Anzahl der Anfragen an die Server möglichst gering ausfällt, werden die Bilder aus Teilelementen (*Tiles*) der Größe 256 Pixel mal 256 Pixel zusammengesetzt, die über einen Caching-Mechanismus zwischengespeichert werden, um unnötige Mehrfachanfragen an den Server zu vermeiden. Das resultierende Gesamtbild wird als Instanz der Klasse *HintergrundBild* in die Simulationswelt eingebunden. Die Mittelpunktskoordinaten des Kartenausschnitts werden verwendet, um das Weltkoordinatensystem der Simulationsumgebung an die gewählte Ansicht anzupassen (wird u.a. für den KML-Export benötigt, siehe Abschnitt 6.3.3).

Soll der Verkehrssimulation ein dreidimensionales Landschaftsmodell zu Grunde gelegt werden, sind neben Karten- und Satellitenbildern der Erdoberfläche zusätzliche Informationen über die Topografie des Geländes erforderlich ([107]). Das amerikanische *United States Geological Survey* (USGS) hat mehrere frei zugängliche Digitale Höhenmodelle (*Digital Elevation Model*, *DEM*) entwickelt, unter anderen GTOPO30 (*Global Topographic Data* [111]), SRTM (*Shuttle Radar Topography Mission* [123]) und DTED (*Digital Terrain Elevation Data* [117]). Einige WMS-Server bieten diese sogenannten Reliefdaten als zusätzliche Bildebenen an, die als Grauwerte kodiert sind; die geografische Höhe einer Koordinate kann so durch Interpolation von Grauwerten ermittelt werden.

Beim Import einer Luftbildaufnahme in die Verkehrssimulation hat der Benutzer die Möglichkeit, zusätzlich zum eigentlichen Hintergrundbild eine Relief-Bilddatei erzeugen zu lassen, die dann in der dreidimensionalen Ansicht als Höheninformation eingebunden werden kann. Das Hintergrundbild wird dafür auf ein Dreiecksgitter projiziert, dessen z-Koordinaten über Inter-

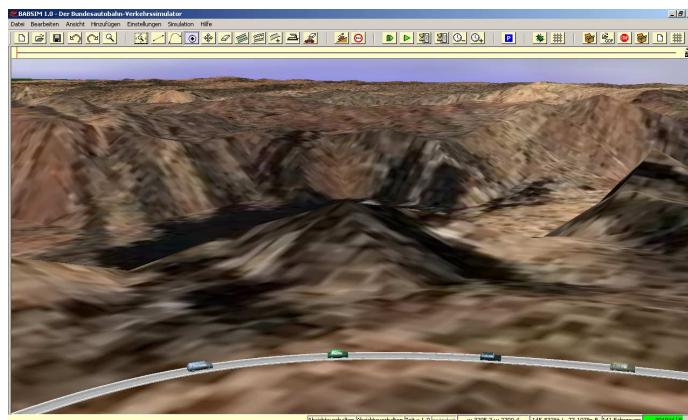


Bild 6.16: Verkehr am Grand Canyon: Einbindung von Luftbildern mit Höheninformationen

polation aus dem Reliefbild gewonnen werden. Abbildung 6.16 zeigt eine 3D-Ansicht des Grand Canyons, erstellt aus Kartenmaterial des bereits zuvor verwendeten NASA/JPL-Servers. Das so erzeugte Geländemodell kann dann dazu verwendet werden, den in der zweidimensionalen Ansicht erstellten Straßenverlauf an das Höhenprofil des realen Geländes anzupassen.

6.3.3 Keyhole Markup Language (KML)

Neben dem Import von bestehenden Daten zur Vereinfachung der Netzkonstruktion ist auch ein Export des Streckennetzes und der Simulationsdaten erforderlich, um den Datenaustausch mit anderen Anwendern zu ermöglichen und somit ein verteiltes Arbeiten zu unterstützen. Lösungsmöglichkeiten, wie Simulationsdaten exportiert und visualisiert werden können, werden von den modernen Geoinformationssystemen aufgezeigt. Für die Visualisierung von Geoinformationen hat sich in den letzten Jahren der sogenannte „virtuelle Globus“ (*virtual globe*) durchgesetzt, d. h. Programme, die die Erdoberfläche als frei drehbare Weltkugel darstellen, auf die sich Informationen aus verschiedenen Datenquellen projizieren lassen. Die beiden bekanntesten derartigen Programme sind momentan „Google Earth“ und „NASA WorldWind“ – daneben existieren aber noch zahlreiche weniger verbreitete Programme (siehe Tabelle 6.2).

ArcGIS Explorer	ESRI Geoinformatik GmbH	http://www.esri.com/software/arcgis/explorer
Dapple	Geosoft	http://dapple.geosoft.com
Earthsim	Earthsim Ltd.	http://www.earthsim.tv/
Gaia	Dmitry Marakasov	http://sourceforge.net/projects/gaia-clean
GeoFusion	Geofusion, Inc.	http://www.geofusion.com
Google Earth	Google, Inc.	http://earth.google.de
SkylineGlobe	Skyline Software	http://www.skylineglobe.com
Virtual Earth	Microsoft Corporation	http://virtualearth.msn.com
Virtual Globe	Norkart	http://www.virtual-globe.info
WorldWind	NASA	http://worldwind.arc.nasa.gov

Tabelle 6.2: Liste einiger virtueller Globus-Applikationen

Um die in der Verkehrssimulation berechneten Simulationsergebnisse in einer Virtual Globe-Umgebung darstellen zu können, müssen sie in einem geeigneten Dateiformat exportiert werden. Die Wahl fiel hierbei auf das ursprünglich für Google Earth entwickelte KML-Format (*Keyhole Markup Language*, [63]), das sich zu einer Art de-facto Standard entwickelt hat ([112]). KML

ist XML-basiert und orientiert sich stark an der Geographic Markup Language (GML, [64]). KML ermöglicht die Definition von geo-referenzierten Text- und Geometrieinformationen, einschließlich Angaben zur Darstellung und Funktionsweise. So lassen sich Ortsmarken, zwei- und dreidimensionale Polygone, Grafiken und extern definierte Inhalte definieren, die bei Bedarf automatisch aktualisiert werden können.

Die Darstellung des aktuellen Zustand einer Verkehrssimulation in KML-Form erfolgt über eine makroskopische Beschreibungsform; an Stelle von individuellen Fahrzeugen werden aktuelle Zustandsgrößen wie Verkehrsstärke, Verkehrsdichte oder Durchschnittsgeschwindigkeit grafisch repräsentiert. Alle Fahrspuren werden in Form von Polygonen als Textstrom in eine neue KML-Datei geschrieben. Das folgende XML-Fragment zeigt exemplarisch die KML-Datei einer einzelnen Fahrspur:

```
<kml xmlns="http://earth.google.com/kml/2.0">
<Document><Placemark><name>I-Nordstrasse</name>
  <Style><PolyStyle><color>FFFF0000</color></PolyStyle></Style>
  <Polygon><outerBoundaryIs><LinearRing>
    <coordinates>7.2608751291409135,51.446027842624270,0.5
      7.2660912293940780,51.447721730453594,0.5
      7.2660661675035950,51.447751610811680,0.5
      7.2608500672504310,51.446057697970650,0.5
    </coordinates></LinearRing></outerBoundaryIs>
  </Polygon></Placemark></Document></kml>
```

Listing 6.3: Eine einfache KML-Datei einer einzelnen Straße

In ähnlicher Weise kann das gesamte Straßennetz, einschließlich aller Geraden, Kurven, Mehrspurbereiche, Gebäude etc. in KML-fähige Applikationen exportiert werden. Zusätzlich können über Markierungen weitere Informationen ortsbezogen in die Darstellung eingebunden werden. Abbildung 6.17 zeigt eine direkt aus der Simulationsumgebung exportierte Auffahrt in Google Earth bzw. in Google Maps (Abbildung 6.17). Einblendbar sind auch Staumarkierungen, die beim Anwählen zusätzliche Informationen über die aktuelle Verkehrslage liefern.

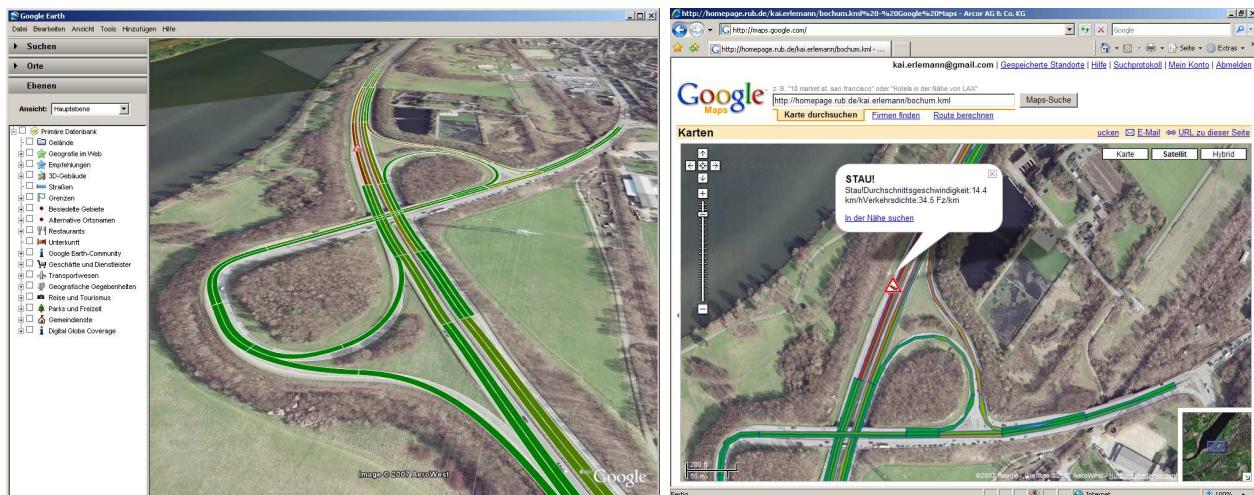


Bild 6.17: Ein simuliertes Netz in Google Earth (links) und Google Maps (rechts)

Neben statischen Informationen bieten KML-Dateien auch die Möglichkeit, dynamische Informationen über eine Internetverbindung einbinden zu können. So lassen sich Echtzeitdaten in re-

gelmäßigen, frei wählbaren Zeitabständen an den Benutzer senden, der so immer über die aktuelle Verkehrssituation informiert ist. Ein von einem Verkehrsingenieur generiertes Streckennetz wird an einen Berechnungsserver gesendet, der einen Simulationslauf startet. Durch Einbindung realer Verkehrs-Messdaten (z. B. aus Induktionsschleifen, Verkehrskameras oder über Floating Car Data) wird die Simulation mit den aktuellen Verkehrsstärken und -geschwindigkeiten versorgt, aus denen dann Prognosen über den weiteren Verkehrsfluss erstellt werden. Die Simulationsergebnisse werden als KML-Datei auf einem Webserver abgelegt, von dem aus sie dann vom Endanwender mit einer KML-fähigen Visualisierungssoftware betrachtet werden können. In regelmäßigen Zeitabständen werden die Ergebnisse neu vom Server angefragt und aktualisiert. Diese Echtzeitsimulation ist, mit Ausnahme der Einbindung realer Messdaten, in die Simulationsumgebung integriert worden.

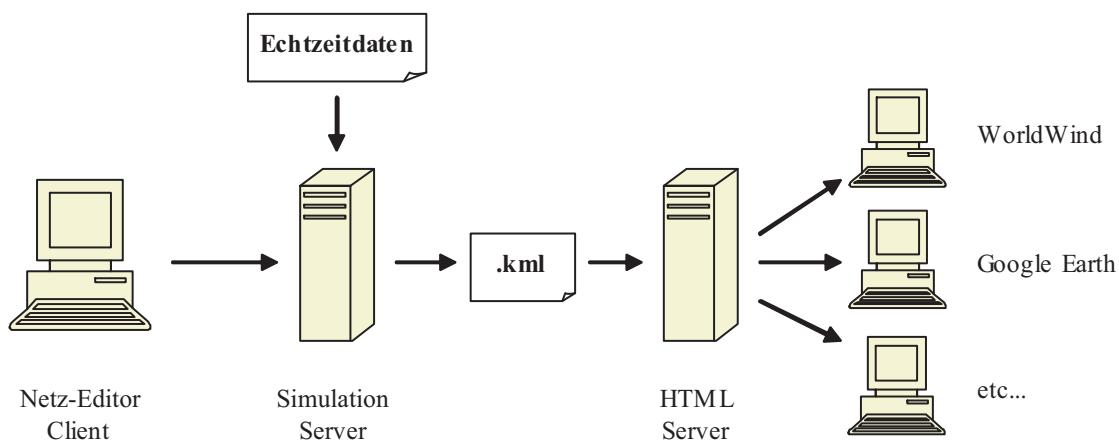


Bild 6.18: Mögliches Anwendungs-Szenario für die Echtzeiterzeugung von KML-Dateien

6.3.4 Scalable Vector Graphics (SVG)

Ein weiteres XML-basiertes Austauschformat ist das Scalable Vector Graphics Format (SVG, [41]) für das Erzeugen vektorbasierter Grafikdaten. Ähnlich zu Postscript- oder WMF-Dateien, lassen sich SVG-Bilder – im Gegensatz zu rasterbasierten Formaten wie GIF oder JPEG – verlustfrei beliebig vergrößern oder verkleinern, ohne die Grafikqualität zu beeinflussen. Für den Austausch von Streckennetz-Abbildungen, die mit der Verkehrssimulation erzeugt wurden, bietet sich ein vektorbasiertes Dateiformat an, da sich in einer exportierten Datei sowohl das gesamte Streckennetz als auch einzelne Detailansichten mit hoher Qualität betrachten lassen (vgl. Abbildung 6.19).

Die Erstellung von SVG-Grafiken des aktuellen Simulationszustandes basiert auf dem Zeichenmechanismus, der für die 2D-Darstellung entwickelt wurde (vgl. Abschnitt 6.2.2). Hierfür wurde eine Klasse *SVGBild* angelegt, welche die von der Oberklasse *Bild* ererbten Zeichenmethoden mit eigenen Methoden überschreibt (Abbildung 6.20). Innerhalb dieser Methoden wird für jeder Aufruf einer Zeichenoperation ein entsprechendes SVG-Codefragment generiert. Um eine neue SVG-Abbildung eines Straßennetzes zu erzeugen, muss eine neue Instanz der Klasse *SVGBild* angelegt und mit einem bestehenden *Bild* verknüpft werden. Beim Auslösen eines Zeichenvorgangs wird eine neue SVG-Datei geöffnet und über den normalen Zeichenmechanismus der Klasse *Bild* alle überladenen Zeichenmethoden aufgerufen. Nachdem alle Zeichenobjekte ge-

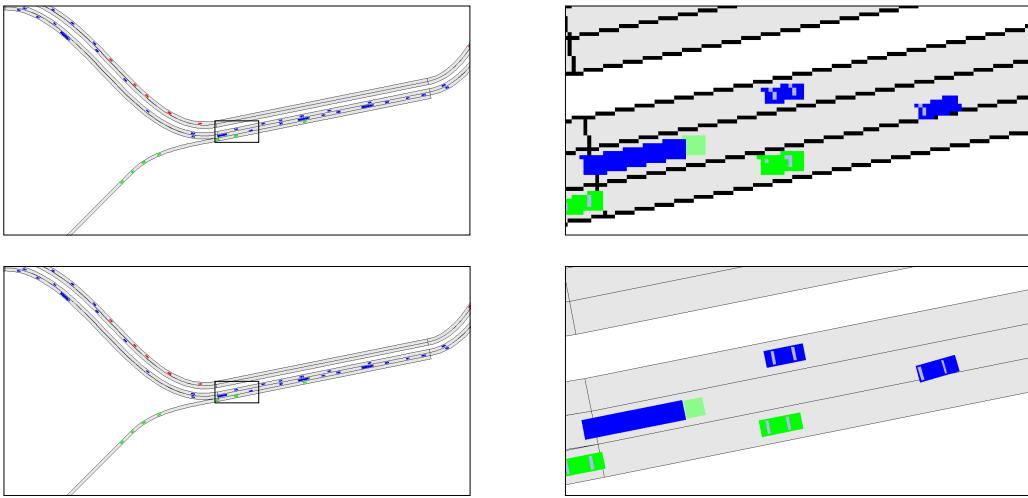


Bild 6.19: Vergleich zwischen raster- (oben) und vektorbasiertem (unten) Vergrößerung

zeichnet worden sind, wird der SVG-Quelltext abgeschlossen. Die Datei kann anschließend von externen SVG-fähigen Programmen betrachtet und bearbeitet werden.

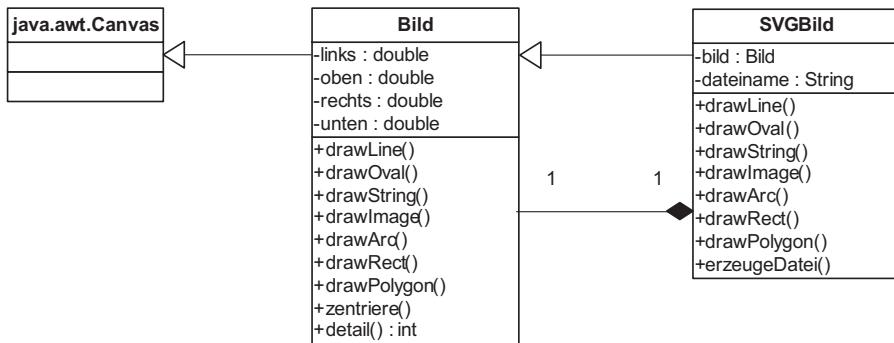


Bild 6.20: Klassendiagramm der SVG-erzeugenden Bildklasse

6.3.5 Sonstige Formate

Neben den hier beschriebenen Datenformaten existieren weitere Formate, die sich für eine Nutzung in der Verkehrssimulation anbieten und ebenfalls vollständig oder teilweise implementiert wurden. Durch die weite Verbreitung dieser Formate ist der Datenaustausch zwischen der Simulation und anderen Software-Produkten sichergestellt.

- **JPG/PNG/GIF:** Grafiken in diesen Formaten (z. B. Planzeichnungen, Karten oder Fotos) können als Hintergrundbilder in die zweidimensionale und dreidimensionale Ansicht eingebunden werden und dienen als Referenzbilder für die Netzkonstruktion.
- **DXF:** Für das Laden von Vektorgrafiken, die mit CAD-Programmen wie AutoCAD erstellt wurden, wurde ein rudimentärer Importfilter für einfache Zeichenelemente (Polylinien, Kreise, Rechtecke) erstellt. Die importierten Objekte dienen als Hilfsobjekte in der 2D-Ansicht und werden in der dreidimensionalen Ansicht nicht dargestellt.

- **3DS:** Über das 3DS-Format lassen sich dreidimensionale Modelle, die z. B. mit 3D Studio Max erstellt wurden, direkt in die Simulation importieren. Über einen existierenden Importfilter der Firma Starfire Research werden 3DS-Dateien in Java3D-BranchGroups umgewandelt und können frei in der 3D-Ansicht platziert werden.
- **CSV:** Bei der Simulation aufgezeichnete Messdaten (Reisezeiten, Querschnittsmessungen, Streckenmessungen) können als sogenannte *Comma Separated Value* (CSV)-Dateien exportiert und von Tabellenkalkulationsprogrammen wie Microsoft Excel oder OpenOffice wieder eingelesen werden.
- **PDF:** Ebenso können die Ergebnisse kompletter Auswertungsläufe in einer Datei im *Portable Document Format* gespeichert werden, die dann in jedem PDF-Betrachter (z. B. Adobe Acrobat Reader) angezeigt oder ausgedruckt werden können.

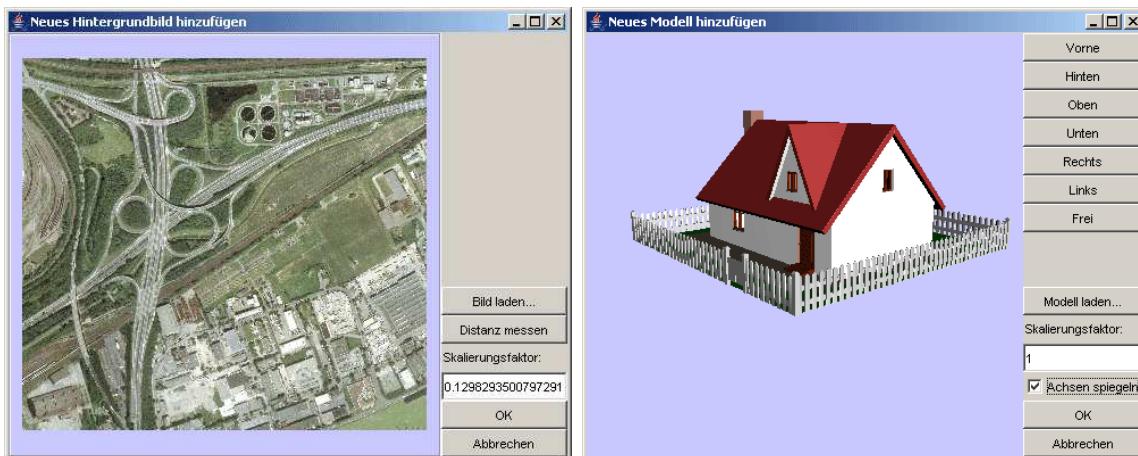


Bild 6.21: Importwerkzeuge für das Einbinden von JPG- (links) und 3DS-Dateien (rechts)

6.4 Parallelisierung der Verkehrssimulation

Je detaillierter und feinkörniger das Verhalten von Fahrern im Straßenverkehr nachgebildet werden soll, desto höher werden die Anforderungen an die Rechenleistung des eingesetzten Computersystems. Trotz zahlreicher Vereinfachungen gegenüber dem realen Fahrverhalten und Optimierung der verwendeten Algorithmen bezüglich Speicher- und CPU-Bedarf, ist insbesondere die mikroskopische Verkehrssimulation immer dann rechenintensiv, wenn großflächige Streckennetze auf einem einzelnen Rechner simuliert werden. Ab einer gewissen Netzgröße und -komplexität ist die Simulation auf einem einzelnen Rechner zu langwierig, so dass der Einsatz eines parallelen Simulationsverfahrens erforderlich wird.

Parallelisierung einer Verkehrssimulation bedeutet die Aufteilung der gesamten Rechenlast auf eine Gruppe von Rechnern bzw. Rechenknoten (*Nodes*) oder Prozessoren. Ziel dieser Arbeitsverteilung ist entweder die Minimierung der Gesamtrechenzeit (d.h. eine Beschleunigung des Simulationsvorganges zwecks Reduzierung der Antwortzeit) oder die Simulation von sehr großräumigen Streckennetzen, für die sowohl die Rechenleistung als auch der Speicherbedarf für einen einzelnen Rechner zu groß wäre. Im Idealfall sollte ein auf einem Einzelplatzsystem simulierbares Streckennetz mit nur geringem Mehraufwand auch auf einem parallelen (d. h. verteilten) Simulationssystem berechnet werden können.

Das Entwickeln einer Multiprozessor-Anwendung ist mit einem erheblichen Arbeitsaufwand verbunden. Die Kommunikation zwischen den beteiligten Knoten bzw. Prozessoren erfordert genau definierte Kommunikationsprotokolle und zahlreiche Synchronisationsmechanismen, um den korrekten Ablauf der Simulation sicherzustellen. Bei der Umwandlung einer existierenden Einzelplatz-Applikation in eine parallelisierbare Version muss deshalb die Programmstruktur an die Anforderungen paralleler Programmierung angepasst werden.

6.4.1 Parallelisierungsarten

Die Komplexität paralleler Software ist stark abhängig von den Anforderungen, die an das unterlegte Parallelisierungsmodell gestellt werden. Grundsätzlich wird unterschieden zwischen asynchroner und synchroner Parallelisierung. Während bei asynchronem Ablauf der Simulation die Einzelprozesse autonom voneinander ablaufen und nur wenige Informationen zwischen den Prozessen ausgetauscht werden müssen, erfordert eine synchronisierte Parallelisierung die Interaktion zwischen den Prozessen und somit eine intensive Kommunikation. Ob ein synchroner oder asynchroner Ansatz verwendet wird, hängt von der geplanten Einsatzart ab. Mögliche Anwendungen im Bereich der Verkehrssimulation sind insbesondere:

- **Kalibrierung:** Mehrere Instanzen derselben Simulation werden mit verschiedenen Parametersätzen gleichzeitig auf mehreren Rechnern ausgeführt, um so zutreffende Parameter für das Erzielen eines gewünschten Soll-Verhalten zu ermitteln.
- **Szenario-Vergleich:** Jeder Rechnerknoten wird für die Simulation eines dedizierten Szenarios (z. B. für verschiedene Straßenentwürfe) mit jeweils den selben Parametersätzen eingesetzt.
- **Optimierung:** Verschiedene Kalibrierungs- oder Szenario-Rechnungen werden parallel nebeneinander durchgeführt, die einzelnen Simulationsergebnisse miteinander verglichen,

um dann die Parameter- oder Netzgeometrien proaktiv in Richtung eines Optimums zu verbessern.

- **Partitionierung:** Ein großes, räumlich umfangreiches Straßennetz wird in mehrere Teilnetze zerlegt, die dann jedes für sich auf einem synchronisierten Rechnerknoten simuliert werden, wobei auch Wechselwirkungen zu berücksichtigen sind.

6.4.2 Asynchrone Parallelisierung

Die beiden erstgenannten Anwendungsarten sind programmiertechnisch vergleichsweise einfach zu realisieren: Die durchgeföhrten Simulationen sind voneinander vollständig unabhängig, Interaktionen zwischen den simulierten Netzen treten somit nicht auf. Für diese Ausprägung der asynchronen Parallelisierung ist es nicht einmal erforderlich, dass die beteiligten Rechner in einer direkten physischen Verbindung zueinander stehen. Jeder einzelne Simulationslauf kann „atomar“ auf einem autarken Rechnersystem durchgeführt werden.

Bei der Entwicklung der Verkehrssimulation fand dieses Verfahren unter anderem bei der Neukalibrierung der Parameter für das Fahrzeugfolgemodell nach Gazis et al. Anwendung (vgl. Kapitel 4.2.1). Hierfür wurde von Seifarth und Harding ([18]) über mehrere Tage hinweg auf etwa 12, teilweise bis zu 20 Einzelrechnern eine Serie von Simulationsläufen durchgeföhr. Jede Simulation erhielt einen eigenen Parametersatz, der durch vollständige Enumeration aller möglichen Parameterkombinationen innerhalb eines vorgegebenen Wertebereichs ausgewählt wurde. Das Starten der Simulation, das Eingeben der Parameterbereiche, das Sammeln der Simulationsprotokolle und deren Auswertung geschah dabei rein manuell – eine computergestützte Lastverteilung fand nicht statt. Diese Vorgehensweise ist zwar für einfache Probleme anwendbar, erfordert aber einen hohen Zeit- und Personalaufwand. Zudem ist die manuelle Aufgabenverteilung stark fehleranfällig, da fehlende Simulationsläufe oder falsche Parametereinstellungen erst spät bemerkt werden. Bei komplexeren Problemen müssen daher automatisierte Verfahren eingesetzt werden.

Sind alle für die Berechnung zur Verfügung stehenden Rechner per Netzwerk physikalisch miteinander verbunden, kann die asynchrone Simulation auch automatisch durch einen einfachen Parallelisierungsserver erfolgen; dieser verteilt die anstehenden Rechenaufträge im Batch-Betrieb an die zur Verfügung stehenden Rechner. Hierfür werden zunächst durch den Anwender oder das Simulationsprogramm alle Aufträge (Straßennetze inklusive aller Simulationsparameter) in einer Liste gesammelt und der Reihe nach auf freie Rechnerknoten verteilt. Hat ein Knoten seine Berechnung abgeschlossen, sendet er seine Ergebnisse an den Parallelisierungsserver zurück und wartet auf neue Aufträge. Der Server hat die Aufgabe, Aufträge und Rechenlasten so zu verteilen, dass die gesamte zur Verfügung stehende Prozessorleistung möglichst optimal ausgenutzt wird, um so optimale Performanz zu erzielen. Auch hier läuft die gesamte Simulation vollständig asynchron ab, d. h. während des Simulationslaufes werden keine Informationen zwischen Server und Knoten bzw. zwischen Knoten und Knoten ausgetauscht. Nur die Zeitbereiche in denen abzuarbeitenden Aufträge bzw. Simulationsergebnisse (nach Beendigung eines Simulationslaufs) zu übertragen sind, müssen synchronisiert werden. Aufgrund der einfachen Kommunikationsstruktur reichen einfache Techniken wie der direkte Dateiaustausch über FTP-Verbindungen, Socketverbindungen oder *Remote Procedure Calls* (RPC) aus, um die parallele Abarbeitung zu realisieren.

6.4.3 Synchrone Parallelisierung

Teilweise oder vollständig synchronisierte Parallelisierung stellt ungleich höhere Anforderungen an die verwendeten Kommunikationsmethoden und die informative Umsetzung. Synchronisierung bedeutet, dass einzelne Rechenknoten und/oder der zentrale Organisationsknoten (*Master*), der für die Aufgabenverteilung zuständig ist, sich während des Rechenablaufs untereinander austauschen und dabei ggf. Wartepausen in Kauf nehmen müssen, um gewünschte Informationen zu erhalten. Mögliche Gründe für solche Wartezeiten sind unterschiedliche Rechenzeiten oder Latenzzeiten beim Nachrichtenaustausch. Benötigt ein Knoten zwingend Informationen von einem anderen Knoten, ohne die er seine Bearbeitung nicht fortsetzen kann, versendet er eine entsprechende Anforderungsnachricht und wartet auf die angeforderte Information. Erst wenn der Nachrichtenaustausch abgeschlossen wurde, wird der Programmablauf auf beiden beteiligten Rechnern fortgesetzt – beide sind synchronisiert.

Eine synchrone Parallelisierung bietet sich in der Verkehrssimulation dann an, wenn der Verkehr auf einem Streckennetz simuliert werden soll, das aufgrund seiner räumlichen Ausdehnung und der großen Zahl gleichzeitig bewegter Fahrzeuge die Rechenkapazität eines Einzelrechners übersteigt. Um die Arbeitslast auf mehrere Rechner aufzuteilen, muss das Netz in kleinere Teilnetze zerlegt und auf die verfügbaren Prozessoren verteilt werden. Um eine gleichmäßige Lastverteilung auf allen Rechenknoten zu erzielen, sollten dabei die einzelnen Teilnetze in Größe und Verkehrsbelastung in etwa übereinstimmen.

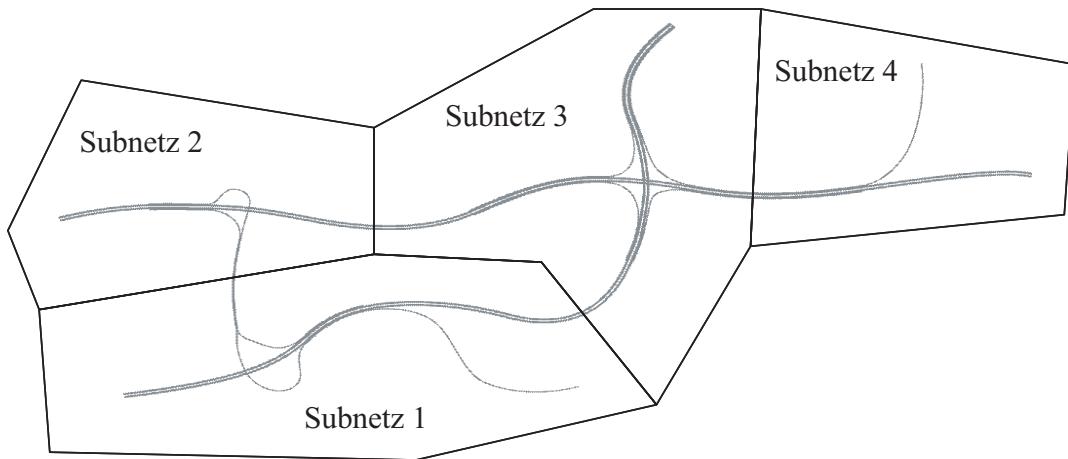


Bild 6.22: Partitionierung eines Streckennetzes in vier Subnetze

Bei der Partitionierung des Streckennetzes muss darauf geachtet werden, dass die Unterteilung nur an solchen Streckenabschnitten stattfindet, wo keine starken Interaktionen zwischen den Teilnetzen vorliegen, um zu starke Wechselwirkungen in der Simulation zu vermeiden. Beispielsweise sollte ein Autobahnnetz immer so zerlegt werden, dass alle Schnitte auf freier Strecke liegen (geringe Interaktion) und nicht im Bereich von Anschlussstellen bzw. Autobahnkreuzen (hohe Interaktion). Zwar werden Informationen über benachbarte Fahrzeuge, Geschwindigkeiten und Ausfahrtentfernungen auch über die Grenzen der Teilnetze hinweg übertragen, strategische Verhaltensmuster – wie die Unterstützung beim Einfahren – können aber nur schwer über Netzgrenzen hinweg erfolgen. Schließlich sollte darauf geachtet werden, dass bei der Partitionierung so wenig Trennstellen wie möglich entstehen, um die Kommunikation zwischen den Netzen so gering wie möglich zu halten. Abbildung 6.22 zeigt eine sinnvolle Unterteilung eines Streckennetzes in vier Subnetze vergleichbarer Komplexität.

An den Schnittstellen, an denen das Netz in Teilnetze unterteilt wird, müssen die Fahrspuren, die zu unterschiedlichen Teilnetzen gehören, getrennt werden. Somit entstehen neben den ohnehin schon existierenden Quellen und Senken weitere Fahrspuren, an denen Fahrzeuge in einem Teilnetz erzeugt bzw. aus einem Teilnetz entfernt werden. Das Aufsetzen bzw. Entfernen muss dabei so erfolgen, dass der Simulationsablauf so wenig wie möglich gestört und die makroskopischen Kenngrößen des simulierten Verkehrs nicht verfälscht werden.

Um bei der Parallelisierung möglichst reibungslose Netzübergänge zu gewährleisten, stehen im Wesentlichen zwei mögliche Lösungsansätze zur Verfügung:

- **Teilsynchronisation:** Ausgehend von der Annahme, dass sich der Verkehrsfluss an Netzgrenzen insgesamt träge verhält, kann ein statistischer Abgleich zwischen Teilnetzen erfolgen: In festgeschriebenen Zeitintervallen werden die wesentlichen Eigenschaften des aus einem Teilnetz herausfliessenden Verkehrs (Verkehrsstärke, Geschwindigkeit, Fahrstreifenverteilung und Lkw-Anteil) gemessen und zwischengespeichert. Nach Ablauf einer Intervalldauer wird die Simulation in allen an der Simulation beteiligten Teilnetzen angehalten (synchronisiert), und die ermittelten Messdaten der Netzausgänge an die entsprechenden Netzeingänge der Nachbarnetze übersandt. Anschließend wird die Simulation fortgesetzt. Aus den übertragenen Daten werden an den Verbindungsstellen der Teilnetze neue Fahrzeuge generiert, in Anlehnung an die herkömmliche Fahrzeugerzeugung an Quellen (vgl. Kapitel 3.5.2). Da die Simulation nur in festen Zeitintervallen synchronisiert wird und ansonsten innerhalb der Intervalle asynchron abläuft, spricht man von Teilsynchronisierung.
- **Vollsynchronisation:** Obwohl jeder Rechnerknoten für ein eigenes Teilnetz zuständig ist, finden alle Simulationsschritte in einer vollsynchronisierten Zeitdomäne statt, d. h. jeder Knoten muss sich nach jedem Simulationsschritt mit allen anderen Knoten (auch dem verwaltenden Hauptknoten) abgleichen und benötigte Informationen austauschen. Erst wenn sich alle Knoten beim Hauptknoten zurückgemeldet haben, wird die Simulation fortgesetzt. So wird sichergestellt, dass alle Netze zu jedem Zeitpunkt zueinander konsistent sind. Der Rechner mit der längsten Rechenzeit bestimmt die Simulationszeit des Gesamtsystems.

Beide Parallelisierungs-Strategien haben ihre spezifischen Vorteile und Nachteile: Die asynchrone Simulation erfordert nur einen geringen Kommunikationsaufwand, die Knoten arbeiten weitgehend autark und müssen sich nur selten abstimmen. Gleichzeitig müssen aber enorme Anstrengungen unternommen werden, um statistische Verfälschungen des Verkehrsflusses beim Übergang zwischen den Teilnetzen gering zu halten. So müssen beispielsweise Vorlaufstrecken (vgl. Abschnitt 3.5.3) vorgesehen werden, um die neu erzeugten Fahrzeuge an den Verkehrsfluss im neuen Netz anzupassen. Fahrzeuge in vollsynchronisierte Netze hingegen benötigen keinen Vorlauf, da sie direkt über die Netzgrenzen übertragen werden und ihre Fahrt danach unverändert fortsetzen können. Jedes Fahrzeug verhält sich so, als wäre keine Zerlegung in Teilnetze vorgenommen worden. Durch Vollsynchronisation erhöht sich allerdings der Kommunikations-Overhead; das gesamte Netz muss stets auf den langsamsten Knoten warten, so dass in der Regel die Rechenzeit gegenüber der teilsynchronisierten Variante erhöht wird.

Insgesamt überwiegen aber klar die Vorteile der Vollsynchronisation, da Zeitverluste bei hohen Netzwerk-Datendurchsätzen im Vergleich zum erhaltenen Genauigkeitsgewinn gering sind. Ferner ist die vollsynchronisierte Variante, wie nachfolgend dargestellt wird, einfacher zu implementieren.

6.4.4 Message Passing Interface

Die Effizienz der Nachrichtenübermittlung hat einen großen Einfluss auf die Anzahl der effektiv vernetzbaren Rechnerknoten. Aus diesem Grund wird für Parallelrechner und Rechnercluster (Zusammenschluß aus mehreren Rechner zu einem Rechnersystem) häufig das *Message Passing Interface* (MPI [119]) verwendet, ein *low-level*-Kommunikationsprotokoll, das eine effiziente und ressourceschonende Kommunikation zwischen den beteiligten Rechnern gewährleistet. MPI wurde 1994 entwickelt und hat sich als ein De-facto-Standard in der Parallelisierung etabliert.

MPI definiert eine ganze Reihe von Methoden zum Senden und Empfangen von Nachrichten (wahlweise synchronisiert oder unsynchronisiert), die zwischen Paaren von Knoten, allen vorhandenen Knoten oder definierten Knotengruppen verschickt werden können. Auch mehrere gleichzeitige Nachrichten von mehreren Knoten zu mehreren anderen Knoten sind möglich. Jede Nachricht besteht dabei aus einem Array eines wählbaren Basis-Datentyps. Komplexere Objekte müssen zuvor in Arrayform konvertiert werden. Es existieren zahlreiche Implementierungen von MPI; hier wurde das Paket *LAM/MPI* ([24]) bzw. dessen Nachfolger *OpenMPI* ([53]) gewählt. Um MPI in Java-Programmen einsetzen zu können, steht das Paket *MPIJava* zur Verfügung, das an der Universität von Indiana entwickelt wurde ([27]) und über das Java Native Interface (JNI) die Methodenaufrufe der zugrunde liegenden MPI-Implementierung kapselt.

6.4.5 Implementierung der verteilten Simulation

Ziel der Parallisierung ist es, eine einfache Möglichkeit bereitzustellen, um verteilte Simulationsläufe für selbst erstellten Streckennetze auf einer dedizierten Rechnerplattform durchführen lassen zu können. Dabei wird davon ausgegangen, dass der Benutzer nur einen herkömmlichen Arbeitsplatzrechner zur Verfügung hat, und die Berechnung (Simulation) von einem externen Dienstleister durchgeführt wird, der die Parallelrechner-Architektur anbietet. Die folgende Beschreibung der Referenzimplementierung orientiert sich am Ablauf eines typischen Bearbeitungsvorgangs:

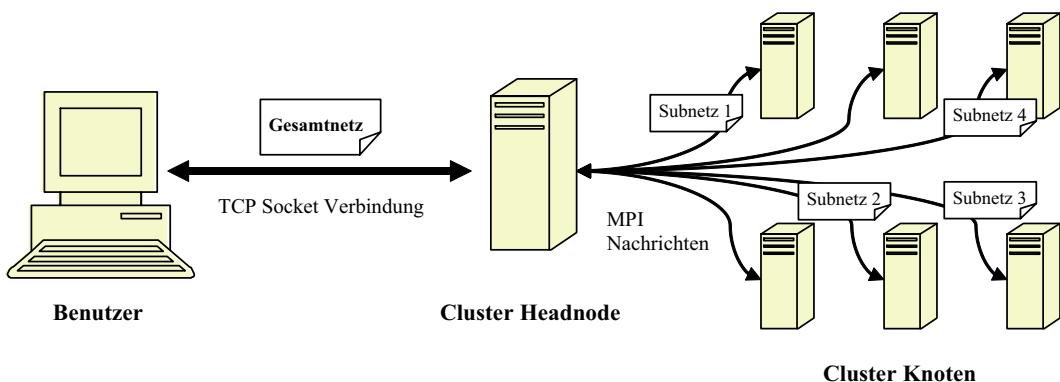


Bild 6.23: Kommunikation zwischen Benutzer, Headnode und Rechenknoten

- **Anmeldung am Rechenserver**

Innerhalb der Simulationsumgebung wurde ein großräumiges Streckennetz erstellt, das auf mehrere Berechnungsknoten verteilt werden soll. Dazu wird zunächst eine Netzwerksverbindung zum gewünschten Rechenserver aufgebaut. Da diese Verbindung unabhängig

vom eingesetzten Rechnersystem, Betriebssystem oder installierter Software erfolgen soll, ist sie als TCP/IP-Socket realisiert. Für die Kommunikation der Rechnerknoten untereinander wird hingegen MPI als Übertragungsschnittstelle verwendet. Der Rechenserver (der sogenannte *Headnode*) dient als Vermittlungsstelle zwischen beiden Kommunikationsprotokollen.

Im praktischen Einsatz ist damit zu rechnen, dass gleichzeitig mehrere Arbeitsaufträge an den Parallelisierungsdienst ergehen, d. h. mehrere Nutzer versuchen gleichzeitig sich mit dem Headnode zu verbinden. Für jede Instanz der Simulationssoftware, die eine Verbindung aufzubauen versucht, wird auf Serverseite jeweils eine eigene Instanz der Klasse *SIMClient* angelegt, die für die Kommunikation zwischen Server und Client zuständig ist.

- **Reservierung von Rechenknoten**

Zu Parallelisierungszwecken steht am Lehrstuhl für Ingenieurinformatik seit Mitte 2005 ein leistungsfähiger PC-Cluster zur Verfügung, auf dem die Implementierung getestet werden konnte (Abbildung 6.24). Die technischen Daten des Systems sind Tabelle 6.3 zu entnehmen.

Betriebssystem:	Suse Linux 10.1
Anschaffung:	Juni 2005
Anzahl Knoten:	56
Prozessoren:	je 2 x 2,2 GHz AMD Opteron 64bit
Hauptspeicher:	2 GB RAM
Festplatte:	80 GB SCSI
Netzwerk (intern):	2x1 GB/s Ethernet
Netzwerk (extern):	100 MB/s Ethernet
Parallelisierungs-API:	LAM/MPI bzw. OpenMPI

Tabelle 6.3: Technische Daten des PC-Clusters

Beim Starten der MPI-Umgebung wird auf jedem reservierten Cluster-Knoten eine Instanz der Simulationssoftware erzeugt, die zentral auf dem Headnode bereitgehalten wird. Da die parallele Berechnung keine grafische Oberfläche benötigt, wird dabei eine Version der Simulationsumgebung ohne den grafischen Netzeditor verwendet. Jeder Rechnerknoten wartet nach dem Start auf eine MPI-Nachricht des Headnodes, die ihn über den Beginn eines neuen Simulationslaufs verständigt.

Dem Benutzer wird nun eine Übersicht über alle zur Verfügung stehenden Rechnerknoten angezeigt. Um einen Arbeitsauftrag zu versenden, müssen Knoten reserviert werden. Auf jedem Rechnerknoten kann stets nur maximal eine Simulation gleichzeitig durchgeführt werden; bereits von anderen Nutzern reservierte Knoten stehen für weitere Arbeitsaufträge nicht zur Verfügung. Die Anzahl der reservierten Knoten sollte stets der Anzahl der Teilnetze entsprechen. Ist eine Reservierung erfolgt, werden die für den Benutzer reservierten Knoten vom Headnode zu einer *Prozessgruppe* zusammengefasst. Über ein sogenanntes *Kommunikator*-Objekt kann eine prozessgruppenweite Kommunikation angestoßen werden, welche für die Synchronisation der Rechenschritte benötigt wird.

- **Partitionierung des Netzes**

Die Partitionierung von Graphen ist ein NP-vollständiges Problem, zu dem bereits zahlreiche Forschungsarbeiten existieren (siehe u. a. [100]). Wie zuvor beschrieben, müssen bei der Partitionierung eines Streckennetzes viele Faktoren wie Teilnetzgröße, zu erwartende



Bild 6.24: Der Linux-Cluster des Lehrstuhls für Ingenieurinformatik im Bauwesen

Fahrzeugmenge, Interaktionsdichte an Netzgrenzen, Anzahl der Netzverbindungsstellen etc. berücksichtigt werden. Die Erstellung eines Algorithmus, der diese Einflüsse – zusätzlich zur eigentlichen Netztopologie – zuverlässig berücksichtigt, ist anspruchsvoll und damit zeitintensiv. Daher wurde zunächst nur eine manuelle Partitionierung des Streckennetzes implementiert, die ein geschulter Benutzer relativ schnell und intuitiv korrekt vornehmen kann. Insbesondere die häufig benötigte Partitionierung in eine geringe Anzahl von Teilnetzen kann von Hand schnell und effektiv durchgeführt werden.

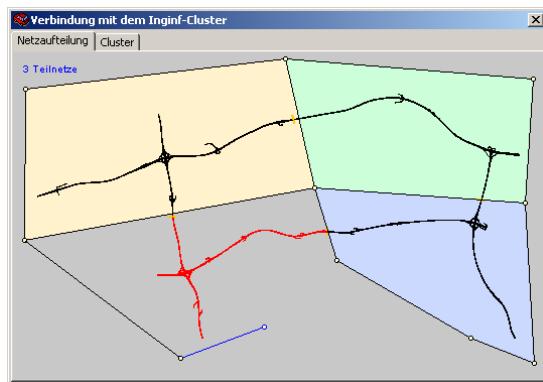


Bild 6.25: Manuelle Partitionierung eines Streckennetzes

Für jeden reservierten Rechenknoten markiert der Benutzer dazu per Maus eine Reihe von Stützpunkten und Verbindungslinien. Aus dem so gebildeten Graphen werden über einen Umlauf-Algorithmus geschlossene Polygone ermittelt, die jeweils die Umgrenzung für ein Teilnetz darstellen. Das Gesamtnetz wird automatisch entlang der Polygonumgrenzungen in Teilnetze zerlegt; dabei erfolgen die Schnitte jeweils an den nächstliegenden Verbindungspunkten der Fahrspuren. Nachdem auf diese Weise alle Teilnetze erzeugt wurden, werden sie an den Headnode gesendet, der sie dann – zusammen mit den vom Benutzer gewählten Simulationsparametern – auf die reservierten Rechnernote verteilt.

- **Durchführung der Simulation**

Sobald einem Rechenknoten sein Teilnetz zugeteilt worden ist, wird mit der Bearbeitung des Simulationsauftrags begonnen. Durch Aufruf der Methode `zeitschritt` wird – wie bei der parallelen Simulation – für alle Fahrspuren und Fahrzeuge ein diskreter Zeitschritt mit der vom Benutzer gewählten Zeitschrittlänge durchgeführt. Anschließend wird der Knoten mit allen anderen Knoten seiner Prozessgruppe synchronisiert, d. h. er meldet der Prozessgruppe, dass sein Berechnungsschritt durchgeführt wurde. Der Knoten wartet dann, bis sich alle anderen Rechner ebenfalls zurückgemeldet haben. Anschließend wird wieder ein Zeitschritt mit anschließender Synchronisierung durchgeführt, bis schließlich die vorgegebene Simulationsdauer erreicht ist.

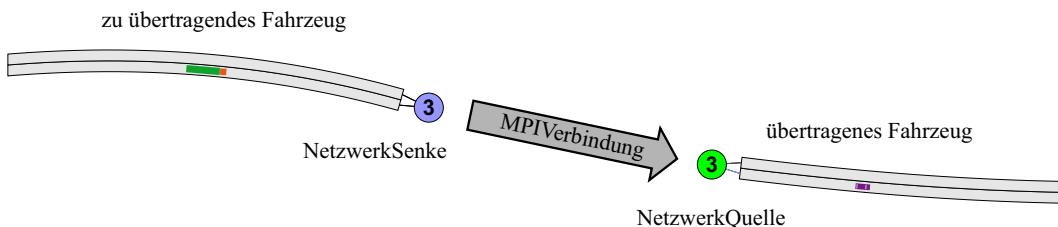


Bild 6.26: Versenden eines Fahrzeugs von einer Senke (links) zu einer Quelle (rechts)

Zwischen den bei der Partitionierung getrennten Fahrspuren wird automatisch ein *NetzwerkSenke-NetzwerkQuelle*-Paar eingefügt. Jede dieser Netzwerk-Senken besitzt eine korrespondierende Netzwerk-Quelle in einem benachbarten Teilnetz. Sobald ein Fahrzeug während der Zeitschrittberechnung eine Netzwerk-Senke erreicht, wird es von dieser zwischengespeichert und aus der Simulation entfernt. Während der Synchronisierung werden von jedem Teilnetz alle zwischengespeicherten Fahrzeuge gesammelt an diejenigen Rechenknoten versendet, die eine zur Netzwerk-Senke gehörende Netzwerk-Quelle enthalten. Die Übertragung geschieht über den Java-Serialisierungsmechanismus, der die Fahrzeuge samt aller Referenzen in Form eines Byte-Arrays versendet und auf Empfängerseite wieder zusammensetzt. Hat ein Knoten Fahrzeuge von anderen Knoten erhalten, werden diese auf die jeweilige Fahrspur gesetzt und die Simulation wird fortgesetzt. Abbildung 6.26 illustriert den verwendeten Übertragungsmechanismus, Abbildung 6.27 zeigt die alle an der Parallelisierung beteiligten Klassen.

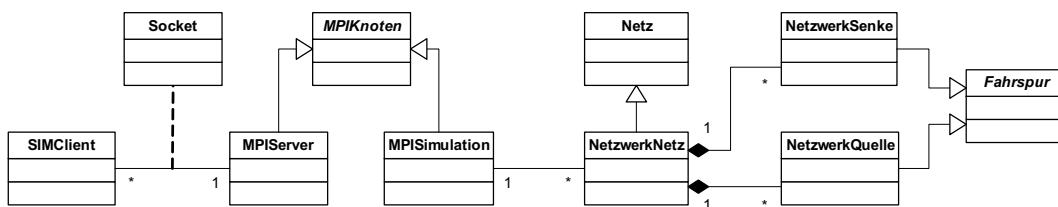


Bild 6.27: Klassendiagramm der wichtigsten Parallelisierungsklassen

- **Sammlung der Ergebnisse**

Sobald alle Rechenknoten einer Prozessgruppe ihre Simulationen erfolgreich beendet haben, senden sie ihre Teilnetze wieder zurück an den Headnode, der diese sammelt und zu einem Gesamtnetz zusammenfügt. Das Gesamtnetz wird dann per Socketverbindung zurück an den Client-Rechner geschickt, auf dem das Ergebnis direkt im Netzeditor der

Simulationsumgebung begutachten kann. Dabei bleiben alle zu den Teilnetzen zugehörigen Simulationsdaten erhalten – eventuell vorhandene Zählpunkte oder Streckenmessungen sind also weiterhin vorhanden und können ausgewertet werden. Bei erfolgreicher Durchführung der Simulation und Übertragung der Daten stehen dem Benutzer somit die gleichen Ergebnisse zur Verfügung, wie er sie auch bei einem lokalen Simulationslauf erhalten würde.

Das hier beschriebene Verfahren wurden vollständig umgesetzt und auf dem Lehrstuhl-Cluster getestet. Die praktische Anwendung der Parallelisierung samt Beurteilung des Laufzeitverhaltens der parallelen Berechnung im Vergleich zur Einzelplatzsimulation erfolgt in Kapitel 8, in dem ein umfassendes Streckennetz untersucht wird.

Kapitel 7

Erweiterungen des Objektmodells

In diesem Kapitel soll anhand einiger praxisnaher Beispiele die Vielseitigkeit der entwickelten Software in der Simulation und Analyse verschiedener Probleme aus dem Alltag des Verkehringenieurs demonstriert werden. Mit Hilfe vergleichsweise einfacher Erweiterungen des Klassenmodells können auch neue Problemstellungen effizient im Computer abgebildet werden. Während in den vorherigen Kapiteln vornehmlich der Verkehr auf Autobahnen beschrieben wurde, wird hier insbesondere auf die Abbildung innerstädtischen Verkehrs eingegangen. Alle im Folgenden dargestellten Beispiele wurden direkt in die Simulationsumgebung eingebunden.

7.1 Kreisverkehr

Ein Anwendungsgebiet der Verkehrssimulation im innerstädtischen Bereich ist die Simulation von Kreisverkehren. Aufgrund unzureichender Bemessungsverfahren wurden in Deutschland einige Jahrzehnte lang nur wenige neue Kreisverkehre gebaut, bereits bestehende wurden häufig durch lichtsignalgesteuerte Kreuzungen ersetzt. Spätestens seit Mitte der 1990er Jahre kommen wieder verstärkt Kreisverkehre zum Einsatz. Es existieren einige neuere Untersuchungen zur Leistungsfähigkeit von Kreisverkehrsanlagen ([15][16]) sowie ausgereifte Bemessungsmethoden ([42][57]). Diese analytischen Verfahren sind aber nicht ausreichend, um die starken Interaktionen zwischen den Ein- und Ausfahrten auf hochbelasteten Kreisverkehren zu erfassen, die Kapazität und Verkehrsqualität des Kreisverkehrs stark beeinflussen. Diese Phänomene können nur durch mikroskopische Simulationen abgebildet werden.

7.1.1 Verhalten am Kreisverkehr

Ein Kreisverkehr besteht aus einer ein- oder mehrspurigen Kreisfahrbahn, in die radial mehrere Zufahrtsstraßen (mindestens drei) einmünden. Bei einem Kreisverkehr muss der einfahrende Verkehr dem Verkehrsstrom auf der Kreisfahrbahn gemäß StVO Vorfahrt gewähren ([106]). Zwar gibt es auch vereinzelte Kreisverkehrsanlagen mit Lichtsignalsteuerung, diese unterscheiden sich jedoch in der Netzerstellung und in der Simulation im Wesentlichen nicht von herkömmlichen Kreuzungen mit Lichtsignalanlagen. Im Folgenden sollen jedoch daher nur vorfahrtsgeregelte einspurige Kreisverkehre betrachtet werden.

Die Fahrt eines Fahrzeugs, dass einen Kreisverkehr passiert, lässt sich in drei Bereiche einteilen:

- **Einfahrt:** Das Fahrzeug nähert sich dem Kreisverkehr. Durch das Verkehrsschild „Vorfahrt gewähren“ wird es gezwungen, sich dem durchfahrenden Verkehr im Kreisverkehr unterzuordnen. Hierzu fährt es langsam an die vorgegebene Haltelinie vor dem Kreisverkehr heran, und sucht eine ausreichend große Lücke zum Einfädeln in den Hauptverkehr auf der Kreisbahn. Erst wenn diese Lücke vorhanden ist, fährt das Fahrzeug ebenfalls in den Kreisverkehr ein.
- **Kreisverkehr:** Das Fahrzeug durchfährt die Kreisbahn so lange, bis die gewünschte Ausfahrt erreicht wird. Welche Ausfahrt angesteuert wird, wird von der Wunschstrecke des Fahrzeugs vorgegeben, die ihm bereits zur Erzeugungszeit zugewiesen wurde. Durch die Vorfahrtregelung müssen keine Fahrzeuge auf den übrigen Zufahrtstraßen beachtet werden; allerdings ist gegebenenfalls die eigene Geschwindigkeit an die Geschwindigkeit der gerade eingefahrenen Fahrzeuge anzupassen, um einen Auffahrunfall im Kreisverkehr zu vermeiden.
- **Ausfahrt:** Hat das Fahrzeug die angestrebte Ausfahrt erreicht, wird es den Kreisverkehr verlassen. Für jede Ausfahrt ist hierzu auf der Kreisfahrbahn eine Abzweig-Fahrspur vorgesehen, die quasi als „Weiche“ für die Fahrzeuge dient und sie auf ihre gewünschte Anschlussfahrbahn lenkt. Sobald das Fahrzeug über den Abzweig die Kreisfahrbahn verlassen hat, fährt es mit normalem Fahrverhalten weiter.

7.1.2 Approximation von Kreisbögen

Neben den – im Normalfall als Geraden ausgelegten – Zu- und Ausfahrten besteht ein Kreisverkehr aus ringförmig aneinandergefügten Kreissegmenten, aus denen die eigentliche Kreisfahrspur gebildet wird, und aus kurzen Kurvenstücken, die Kreisbahn mit den Zu- bzw. Ausfahrten verbinden. Die runden Segmente des Kreisverkehrs werden durch geeignete Kurven approximiert. Verwendet werden kubische Bézierkurven der folgenden Form:

$$C(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3, \quad t \in [0, 1] \quad (7.1)$$

Für die Erstellung der Software-Implementierung wird ein Verfahren verwendet, das einen gegebenen Kreisabschnitt durch eine möglichst gut approximierte Kurve ersetzt: Gegeben seien Mittelpunkt P_m , Radius r und Start- und Endwinkel α und β des Kreisbogens. Gesucht sind die vier Punkte P_0 bis P_3 des Stützpolygons der Bézierkurve (vgl. Abbildung 7.1).

Die beiden Randpunkte P_0 und P_3 können direkt aus den Bogenparameter bestimmt werden:

$$P_0 = P_m + \begin{pmatrix} r \cdot \cos \alpha \\ r \cdot \sin \alpha \end{pmatrix} \quad (7.2)$$

$$P_3 = P_m + \begin{pmatrix} r \cdot \cos \beta \\ r \cdot \sin \beta \end{pmatrix} \quad (7.3)$$

Die Positionen der beiden Hilfspunkte P_1 und P_2 lassen sich allerdings nicht direkt ermitteln, da sie nicht auf dem Kreisbogen liegen. Es ist lediglich bekannt, dass sie sich auf den Tangenten befinden, die durch die Punkte P_0 bzw. P_1 verlaufen. Die Länge l zwischen den Rand- und Hilfspunkten kann durch Gleichsetzen von Kreis- und Bézierfunktion in der Mitte des Bogens

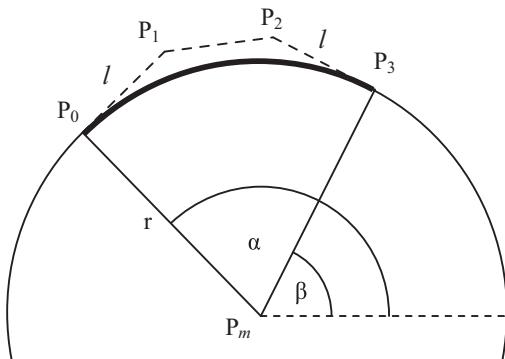


Bild 7.1: Approximation eines Kreissegments mit Hilfe einer kubischen Bézier-Spline

bestimmt werden:

$$C(t = 0,5) \equiv P_m \left(\begin{array}{l} r \cdot \cos \frac{\alpha+\beta}{2} \\ r \cdot \sin \frac{\alpha+\beta}{2} \end{array} \right) \quad (7.4)$$

Für die Länge l ergibt sich nach Vereinfachung die Formel:

$$l = \frac{\frac{4}{3} \cdot r \left(1 - \cos \frac{\beta-\alpha}{2}\right)}{\sin \frac{\beta-\alpha}{2}} \quad (7.5)$$

Mit bekanntem l , das durch die Symmetrieeigenschaften des Kreisbogens für beide Stützlinien identisch ist, können die beiden fehlenden Stützpunkte berechnet werden:

$$P_1 = P_0 + \left(\begin{array}{l} l \cdot \cos(\alpha + 90^\circ) \\ l \cdot \sin(\alpha + 90^\circ) \end{array} \right) \quad (7.6)$$

$$P_2 = P_3 + \left(\begin{array}{l} l \cdot \cos(\beta - 90^\circ) \\ l \cdot \sin(\beta - 90^\circ) \end{array} \right) \quad (7.7)$$

Die so gewonnene Kreissegment-Approximation ist für Bogenwinkel unter 90° hinreichend genau, um eine annähernd kreisförmige Bewegung der Fahrzeuge zu beschreiben. Da Richtungsänderungen der Fahrbahn nicht in das Fahrverhaltensmodell einfließen, ist durch die Bézier-Approximation der Kreisbahn keine Einschränkung des Fahrverhaltens zu erwarten. Bézierkurven kommen im Gegensatz zu Kreisfunktionen vollständig ohne den Einsatz trigonometrischer Funktionen aus, und senken dadurch den Berechnungsaufwand für die Darstellung der Kurve und die Positionsbestimmung entlang der Fahrspur. Außerdem ermöglicht die durchgehende Verwendung von Béziersplines eine konsistenter Benutzerführung.

7.1.3 Modellierung eines Kreisverkehrs

Die Generierung eines Kreisverkehrs erfolgt über einen eigenen Eingabedialog, in dem die Anzahl und Orientierung der einzelnen Zu- und Ausfahrten festgelegt werden. Der Radius der Kreisfahrbahn und die Länge der Zufahrten kann ebenso vom Benutzer vorgegeben werden. Der Mittelpunkt des Kreisels wird mit der Maus direkt im vorhandenen Streckennetz ausgewählt. Sind alle Parameter korrekt eingegeben, wird der Kreisverkehr automatisch erzeugt und

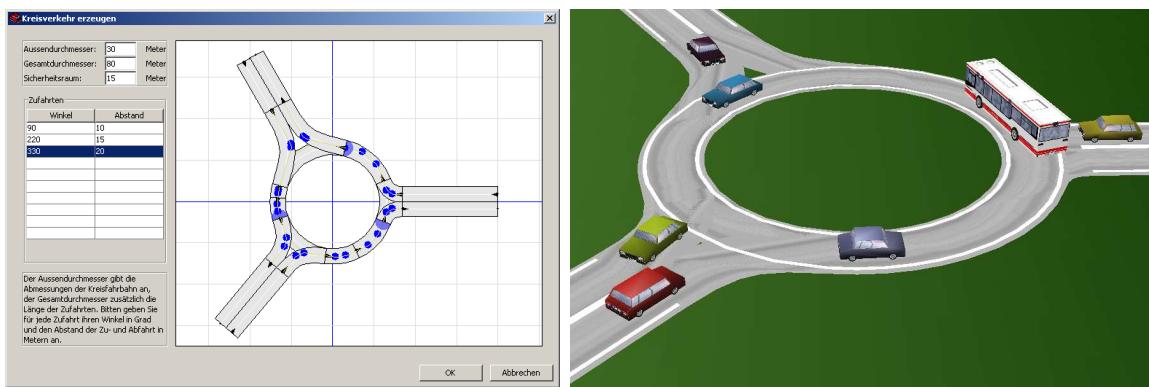


Bild 7.2: Ein Kreisverkehr bei der Eingabe (links) und in der 3D-Ansicht (rechts)

in das Streckennetz eingefügt. Der Benutzer muss nun nur noch die Zu- und Ausfahrten mit den benachbarten Fahrspuren verbinden.

Für jeden Zufahrtsarm des Kreisverkehrs wird jeweils eine Zufahrtsgerade mit einer Zufahrtskurve verbunden, die direkt in die Kreisbahn mündet. Zwischen Gerade und Kurve wird dabei zusätzlich ein sogenannter Haltepunkt eingefügt, der das Fahrzeug bei vorhandenem vorfahrtberechtigen Fahrzeugen auf der Kreisfahrbahn veranlasst, auf eine ausreichend große Lücke im Verkehrsstrom zu warten. Dazu überwacht der Haltepunkt einen festlegbaren Bereich der Kreisfahrbahn und überprüft, ob in diesem Beobachtungsgebiet Fahrzeuge vorhanden sind. Ist dies nicht der Fall, kann sich das Fahrzeug wie bei normalen Fahrspuren einfädeln; sind hingegen Fahrzeuge vorhanden, sperrt der Haltepunkt – vergleichbar mit einer roten Lichtsignalanlage oder einem Stoppschild – die Fahrspur und hindert das Fahrzeug an der Einfahrt in den Kreisverkehr. Erst wenn das letzte Fahrzeug den Beobachtungsbereich verlassen hat, wird der Haltepunkt freigegeben. Abbildung 7.3 verdeutlicht den internen Aufbau des Kreisverkehr-Modells.

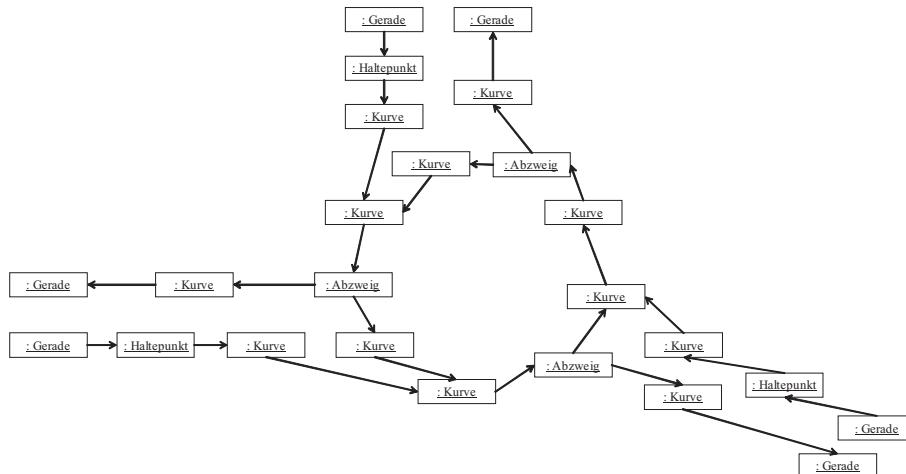


Bild 7.3: UML-Objektdiagramm der Fahrspuren eines dreiarmigen Kreisverkehrs

Die Geschwindigkeit auf der Kreisfahrbahn wird durch eine wählbare Höchstgeschwindigkeit begrenzt; für den implementierten Prototypen wurde eine Höchstgeschwindigkeit von 20 km/h angesetzt. Ebenso wurde die Länge der erforderlichen Mindestweglücke zu 20 Metern angenommen. Das Verfahren der festen Weglücken wurde gewählt, da die Überprüfung einer festen

Anzahl an Fahrspuren auf das Vorhandensein von Fahrzeugen nur wenig Rechenzeit benötigt und einfach zu realisieren ist. Nachteilig ist, dass die Länge der Weglücke direkt in der Geometrie des Kreisverkehrs festgelegt wird und nachträglich nicht mehr geändert werden kann. Eine mögliche Verbesserung des Kreisverkehrs-Modells wäre die Verwendung einer Zeitlücke oder einer kombinierten Weg-Zeit-Lücke als Zufahrtskriterium.

7.2 Parkplätze

Neben der Simulation des fließenden Verkehrs kann es in bestimmten Fällen von Interesse sein, auch ruhenden Verkehr – also geparkte Fahrzeuge – in einer Simulation nachzubilden: Insbesondere private PKW befinden sich zu einem erheblichen Teil ihrer Lebenszeit im ruhenden Zustand, und die Suche nach einem geeigneten Parkplatz bzw. die An- und Abfahrt zu geeigneten Stellplätzen machen einen nicht unerheblichen Teil des täglichen Verkehrsgeschehens aus. Der größte Teil des Parkplatzverkehrs findet dabei im innerstädtischen Bereich statt. Aber auch für Autobahn-Simulationen kann das Abstellen von Fahrzeugen auf Park- oder Rastplätzen von Bedeutung sein, insbesondere dann, wenn die Gefahr einer Rückstauung in den Bereich der Ausfahrt besteht.

Nachfolgend soll deshalb demonstriert werden, wie mit wenig Aufwand die entwickelte Simulationssoftware um eine neue Klasse zur Modellierung von Parkplätzen erweitert werden kann.

7.2.1 Modellierung von Stellplätzen

Grundsätzlich existieren drei verschiedene Typen von Stellplätzen, die anhand ihres Aufstellwinkels relativ zur Fahrbahn klassifiziert werden können. Man unterscheidet Stellplätze für Längsaufstellung (Aufstellwinkel 0°), Schrägaufstellung (45° - 90°) und Senkrechtaufstellung (90° , vgl. Abbildung 7.4). Bei Längs- und Senkrechtaufstellung muss zuätzlich unterschieden werden, ob die Parklücke für Vorwärts- oder Rückwärtseinparken geeignet ist ([11]).

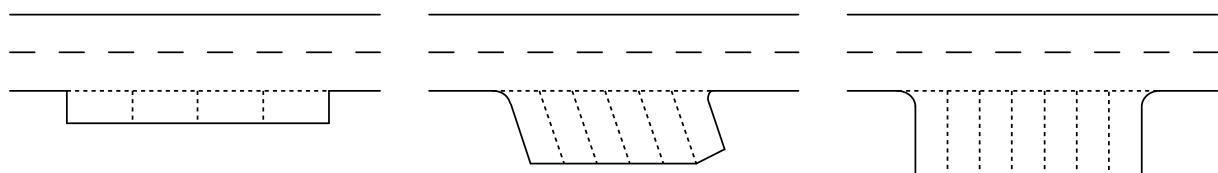


Bild 7.4: Stellplätze für Längs-, Schräg- und Senkrechtaufstellung

Vereinfachend wird im Folgenden lediglich die Umsetzung von Stellplätzen für Schräg- und Senkrechtaufstellung für vorwärts einparkende Fahrzeuge beschrieben, da die vom Fahrzeug beim Einparken beschriebene Kurve für beide Stellplatztypen eine vergleichbare geometrische Form aufweist. Eine Erweiterung auf Längsaufstellung und/oder Rückwärtseinparken nach dem gleichen Prinzip ist möglich, wird hier aber nicht weiter verfolgt.

7.2.2 Auswahl eines Stellplatzes

Jeder Stellplatz bietet stets Platz für maximal ein Fahrzeug. Die Breite und Länge eines Stellplatzes bestimmen, welche Fahrzeuge tatsächlich geparken werden können. Ein Fahrer entschei-

det sich nur dann für einen Stellplatz, wenn dieser ausreichenden Platz für sein Fahrzeug bietet. Zusätzlich können Stellplätze auch über weitere Attribute gekennzeichnet und damit nur für bestimmte Fahrzeuge zugänglich gemacht werden, z. B. gesondert ausgeschilderte Stellplätze für bestimmte Fahrzeugtypen (Taxis, Busse oder Lkw) oder Personengruppen (Frauen, Kurzzeitparker, Behinderte).

Die Wahl eines Parkplatzes für ein gegebenes Fahrzeug kann nach verschiedenen Strategien erfolgen. Eine mögliche Vorgehensweise ist die selbstständige Auswahl des Stellplatzes durch Fahrverhalten: nähert sich ein Fahrzeug einem Stellplatz, entscheidet das Verhaltensmodell, ob diese geeignet ist, und leitet gegebenenfalls einen Einparkvorgang ein. Dieses autonome Verhalten entspricht weitgehend dem realen Verkehrsgeschehen: Der Fahrer fährt so lange über den Parkplatz, bis er einen Stellplatz gefunden hat, der seinen Anforderungen genügt (ausreichende Abmessungen, günstige Lage zum Zielort). Dabei nutzt der menschliche Fahrer zahlreiche Fähigkeiten, die ihm bei der Parkplatzsuche helfen: Erfahrungswerte sagen ihm, ob ein Parkplatz bereits zu großem Anteil mit Fahrzeugen gefüllt ist und in welcher Richtung eventuell noch freie Plätze zu finden sind. Über seine visuelle Wahrnehmung kann er den günstigsten Bereich eines Parkplatzes abschätzen und freie Stellplätze auch noch in einiger Entfernung ausmachen.

Leider ist es schwierig, die beschriebenen menschlichen Fähigkeiten auf ein Computermodell zu übertragen. Intuition und Erfahrungswissen sind einem künstlichen Fahrer nur schwer zu vermitteln, und die Nachbildung von Lerneffekten, wie sie bei menschlichen Fahrern auftreten, verkomplizieren das Modell zusätzlich. Daher wurde bei der Implementierung der Parkplatzsuche ein anderer, vereinfachender Ansatz gewählt.

Eine mögliche Lösung ist eine dezentralisierte Herangehensweise. Hierbei werden alle auf einem Parkplatz verfügbaren Stellplätze von einer zentralen Verwaltungsinstanz, im Folgenden Parkleitsystem genannt, kontrolliert. Jedes Fahrzeug, das auf einen Parkplatz fährt, wird dem Parkleitsystem gemeldet. Dieses beginnt daraufhin, eine für das Fahrzeug geeignete Lücke zu suchen. Hierbei können alle beschriebenen Kriterien – wie Art, Abmessungen, Lage und Attraktivität des Stellplatzes – berücksichtigt werden. Wurde eine geeignete Parkbox gefunden, wird diese für das Fahrzeug reserviert. Für alle anderen Fahrzeuge ist der Stellplatz nicht mehr sichtbar. Erst nachdem das Fahrzeug die Parkbox wieder verlassen hat, wird der Reservierungsmarker entfernt und die Parkbox steht nachfolgenden Fahrzeug zur Verfügung.

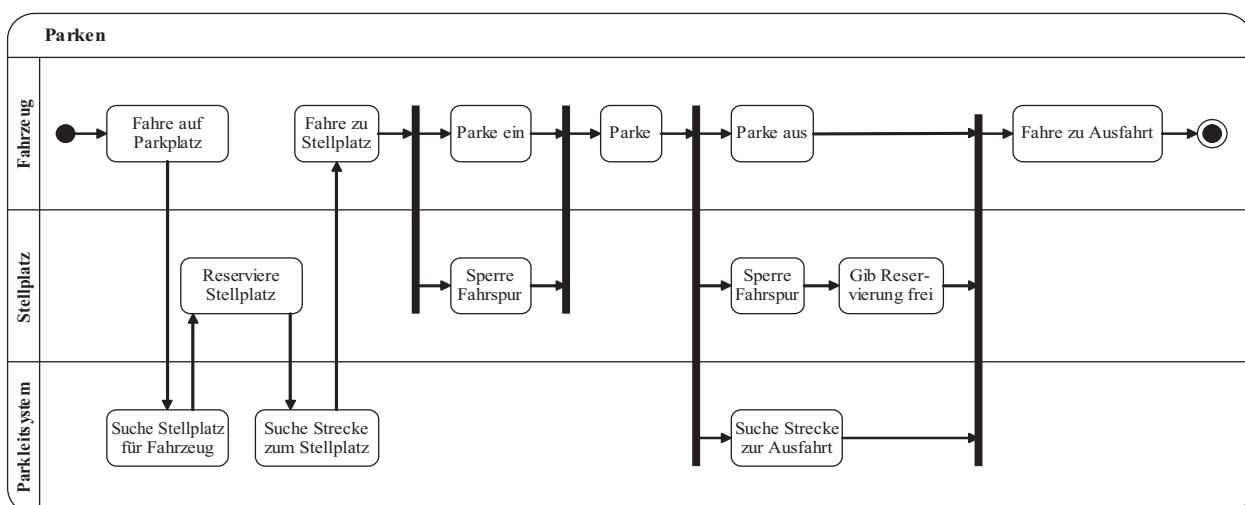


Bild 7.5: Aktivitätsdiagramm des gesamten Parkvorgangs

Sobald für ein Fahrzeug eine Parklücke reserviert wurde, muss sichergestellt werden, dass das Fahrzeug den ihm zugewiesenen Platz auch tatsächlich findet und erreichen kann. Hierzu muss das Parkleitsystem zu jedem ihm zugehörigen Stellplatz den optimalen Zufahrtsweg von der jeweiligen Parkplatzzufahrt ermitteln und sie dem Fahrzeug mitteilen. Hierzu kann der Routing-Algorithmus aus Kapitel 3.6 wiederverwendet werden, der eine Liste aller möglichen Strecken innerhalb des Parkplatzes erstellt und während der Simulation bereithält. Auf diese Weise wird jedem Fahrzeug bei der Einfahrt in den Parkplatz mit sehr geringem Aufwand eine Strecke zu seiner Parkbox zugewiesen.

7.2.3 Nachbildung des Einparkvorgangs

Ein Vorwärts-Einparkvorgang lässt sich vereinfachend in drei Abschnitte zerlegen (vgl. Abb. 7.6):

- **Abschnitt 1:** Vorwärtsbewegung zur Annäherung an den Beginn der Parklücke
- **Abschnitt 2:** Abbiegevorgang zum Einfahren in den Stellplatz
- **Abschnitt 3:** Erneute Vorwärtsbewegung bis zum Ende des Stellplatzes

Dabei wird idealisierend angenommen, dass ein Fahrzeug stets „in einem Zug“, ohne zusätzliche Rangierbewegungen, in einen Stellplatz fährt. Alle Fahrer werden also als „perfekte Einparker“ abgebildet. Da die Einparkzeit durch die vernachlässigten Rangierzeiten unterschätzt wird, muss zum Ausgleich die Einparkgeschwindigkeit reduziert werden, um so realistischere Zeiten für das Einparken zu erreichen.

Die Entscheidung, ob ein Fahrzeug in eine Parklücke einbiegt oder an ihr vorbeifährt, wird am Berührungszeitpunkt der Abschnitte 1 und 2 (vgl. Abbildung 7.6) gefällt, also vor Beginn eines möglichen Abbiegevorgangs. Es können nur Fahrzeuge in einen Stellplatz einfahren, die sich zuvor bei diesem registriert und einen Stellplatz reserviert haben. Parklücken stellen Abzweige dar, die nur angemeldete Fahrzeuge einfahren lassen; alle anderen Fahrzeuge werden geradeaus auf die Fahrgasse geleitet.

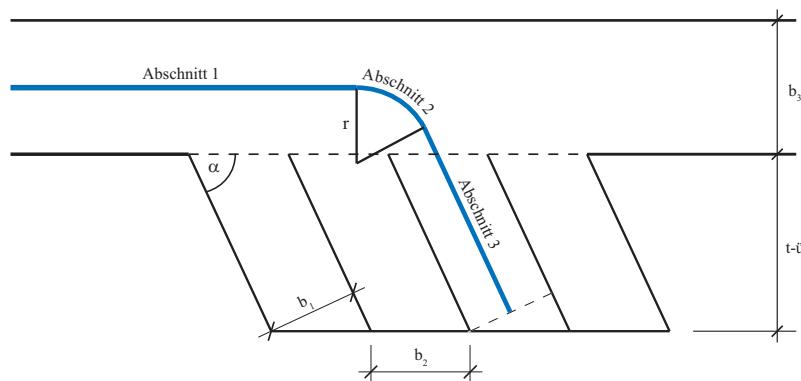


Bild 7.6: Geometrie eines Stellplatzes in der Simulation

Befindet sich ein Fahrzeug auf einer Fahrspur vom Typ Stellplatz, wird es fortan bei jedem Zeitschritt mit einer angenommenen Einparkgeschwindigkeit von 0,2 m/s in Richtung seiner

eigentlichen Parkposition bewegt. Position und Ausrichtung des Fahrzeugs wird vom Stellplatz entsprechend Abbildung 7.6 festgelegt. Hat die Fahrzeugfront das Ende der Stellfläche erreicht, wird das Fahrzeug abgestellt und bleibt bis zum Ende seiner Parkzeit dort stehen. Die Dauer der Parkzeit wird zufällig gewählt: Zujedem Zeitschritt besteht eine gewisse Wahrscheinlichkeit dafür, dass das Fahrzeug die Parkbox wieder verlässt; es wird eine gleichverteilte Wahrscheinlichkeit von 0,001 pro Zeitschritt vorgegeben. Entscheidet sich das Fahrzeug zum Ausparken, wird es mit einer negativen Geschwindigkeit – analog zum Einparkvorgang – aus der Parkbox herausbewegt.

Während des Ein- oder Ausparkens muss der Bereich vor dem Stellplatz für den nachfolgenden Verkehr gesperrt werden, um Auffahrunfälle zu verhindern. Dabei stellt der Stellplatz während der Dauer des Parkmanövers ein Hindernis für alle Fahrzeuge dar, die sich auf die Parkbox zufahren. Erst nach Beendigung des Parkvorgangs wird die Fahrspur wieder freigegeben. Abbildung 7.7 zeigt eine Stellplatzanlage mit einem einparkenden und einem wartenden Fahrzeug.

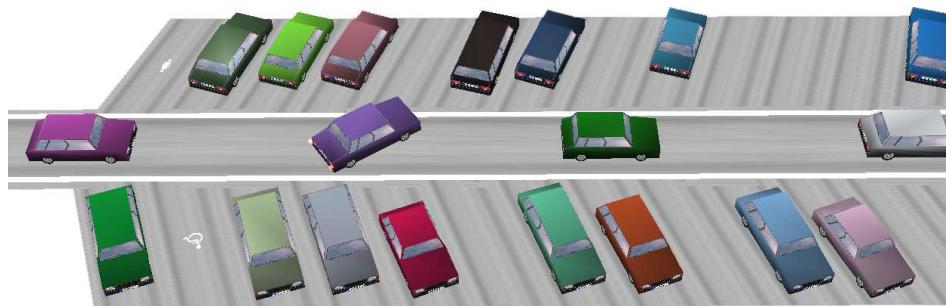


Bild 7.7: Eine simulierte Stellplatzanlage mit einem einparkenden Fahrzeug

Für die Erzeugung von Stellplätzen steht in Simulationssoftware ein eigenes Werkzeug zur Verfügung (s. Abb. 7.8). Aufstellwinkel, Stellplatzlänge, -breite und -abstände können frei gewählt werden. Die Erzeugung erfolgt stets entlang einer vorhandenen Geraden – Stellplätze an kurvigen Straßen sind nicht vorgesehen, lassen sich aber bei Bedarf ergänzen.

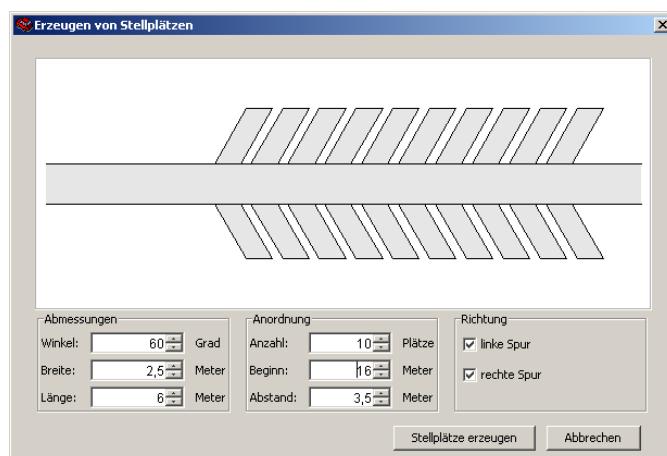


Bild 7.8: Dialog zur interaktiven Erstellung von Parkplätzen

7.3 Lichtsignalanlagen

Neben planfreien Knotenpunkten, wie sie mit dem in Kapitel 3 vorgestellten Grundmodell für Autobahnen nachgebildet werden können, sind auch plangleiche Knotenpunkte (Kreuzungen und Einmündungen), von Interesse. Zum einen lassen sich zu- und abfliessenden Verkehrsströme an Autobahnanschlussstellen abbilden (z. B. Kolonnenbildung als Folge der Grünphasen von Lichtsignalanlagen), und zum anderen die Simulation innerstädtischen Verkehrs ermöglicht.

Bei einem mit Lichtsignalanlage (LSA) geregelten Knotenpunkt werden die Zufahrten durch Lichtzeichen so gesteuert, dass alle unverträglichen (potentiell kollisionsgefährdeten) Verkehrsströme zeitlich getrennt die Konfliktflächen durchfahren und vollständig räumen. Bedingt verträgliche Ströme (z. B. Abbiegen durch den Gegenverkehr oder über Fußgängerfurten), können auch eine gleichzeitige Freigabe erhalten; Konfliktsituationen müssen in der Simulation durch entsprechende Haltepunkte abgesichert werden. Der Stand der Technik für den Entwurf von Lichtsignalanlagen ist in Deutschland in den *Richtlinien für Lichtsignalanlagen* (RiLSA, [97]) dargestellt.

Lichtsignalanlagen können entweder von einem festen Signalzeitenplan (*Signalprogramm*) oder dynamisch in Abhängigkeit von der aktuellen Verkehrssituation gesteuert werden. Vereinfachend wird hier nur das feste Signalprogramm behandelt, da die dynamisierte Verkehrssteuerung ungleich komplexer ist und viele unterschiedliche Verfahren existieren (z. B. [62][44]). Jedes Signalprogramm im Simulationsmodell ist für die Steuerung eines Knotenpunktes zuständig und besitzt eine konstante Umlaufzeit, die zwar vom Benutzer während der Entwurfsphase, nicht jedoch während der Simulation geändert werden darf. Alle Signalgeber mit synchroner Signalisierung werden zu einer *Signalgruppe* zusammengefasst und von dieser gesteuert. Jedes Signalprogramm kann beliebig viele Signalgruppen aufnehmen (vgl. Abbildung 7.9).

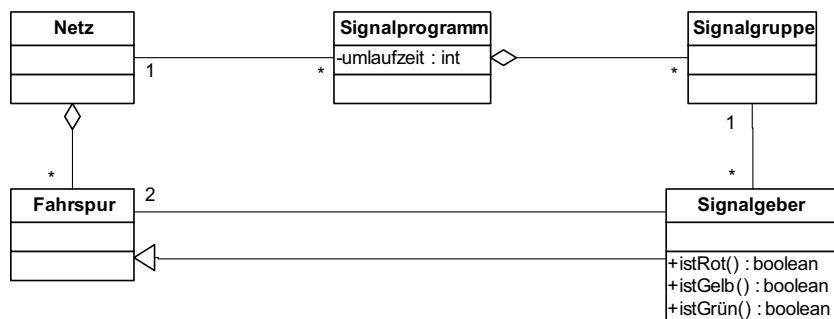


Bild 7.9: UML-Klassendiagramm der LSA-Steuerungslogik

Um eine beliebige Fahrspur lichtsignalgesteuert gegenüber Konfliktsituationen zu sichern, wird am gewünschten Haltepunkt eine Instanz der punktförmigen Fahrspur *Signalgeber* eingefügt und mit der Vorgänger- und Nachfolgerspur verbunden. In der Simulationsumgebung wird jeder Signalgeber durch ein grafisches Symbol repräsentiert, das den aktuellen Signalisierungszustand darstellt. Die eigentliche Verkehrssteuerung erfolgt durch die Signalgeber-Fahrspur. Analog zum Verhalten eines Haltepunktes am Kreisverkehr (Abschnitt 7.1.3) oder an einem Fußgängerüberweg (Abschnitt 7.4.2) kann eine Signalgeber-Fahrspur dem sich nähерnden Verkehr entweder als Hindernis oder als freie Strecke erscheinen. Freigegeben wird die Strecke nur dann, wenn sich die zugehörige Signalgruppe in einer Grünphase befindet. In allen anderen Signalisierungszuständen wird die Strecke gesperrt und der Signalgeber meldet sich selbst als Hindernis zurück. Durch Einbeziehung der Rot-Gelb- bzw. der Gelb-Phase in die Sperrstellung

wird näherungsweise die menschliche Reaktionszeit bzw. das rechtzeitige Abbremsen vor Beginn der Rotphase nachgebildet.



Bild 7.10: 3D-Ansicht einer lichtsignalgeregelten Kreuzung

Grundsätzlich muss jede Fahrspur einer Kreuzung mit einem eigenen Signalgeber und damit einer eigenen Haltelinie versehen werden. Bei mehrspurigen Straßen ist es empfehlenswert, mehrere benachbarte Signalgeber der selben Signalgruppe zu einem eigenen Mehrspurbereich zu gruppieren, um sicherzustellen, dass alle Fahrspuren bei der Streckenerstellung korrekt berücksichtigt werden (vgl. Abschnitt 3.5.5). Teilt sich ein mehrspuriger Bereich an einer Lichtsignalanlage in Fahrspuren mit unterschiedlicher Fahrtrichtung auf, ist es sinnvoll, diese separat anzulegen und mit jeweils einem eigenen Signalgeber auszustatten. In der dreidimensionale Visualisierung ist es möglich, die Signalgeber durch Nachbildung realer Richtungspfeile auf den Signallampen optisch voneinander zu unterscheiden, wie die folgende Abbildung zeigt.

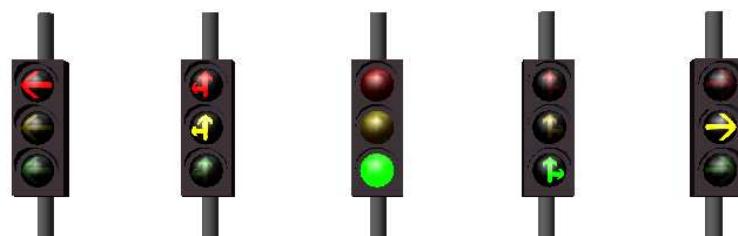


Bild 7.11: Verschiedene Instanzen der Klasse Signalgeber3D

Für die Eingabe des Signalzeitenplans steht ein grafischer Editor zur Verfügung, der nach der Auswahl eines Signalprogramms alle zugehörigen Signalgruppen darstellt (Abbildung 7.12). Jede Zelle des Diagramms repräsentiert den Zustand einer Signalgruppe während eines Zeitintervalls von einer Sekunde. Die fünf möglichen Zustände einer Signalgruppe (*rot*, *rot-gelb*, *grün* und *aus* bzw. *gelb-blinkend*) werden vom Benutzer mit der Maus direkt in das Diagramm eingezeichnet. Die Berechnung von Grünzeiten, Räumwegen, Zwischenzeiten etc. obliegt der Verantwortung des Anwenders.

Sobald das Signalprogramm vollständig erstellt ist, kann jedem Signalgeber eine Signalgruppe zugewiesen werden. Signalprogramme und -gruppen sind dabei nicht räumlich begrenzt, d. h. es ist theoretisch möglich, die Signalsteuerung räumlich getrennter Knotenpunkte durch ein gemeinsames Signalprogramm zu synchronisieren und damit z. B. „grüne Wellen“ nachzubilden, ohne für jede Kreuzung eigene Signalprogramme erstellen zu müssen.

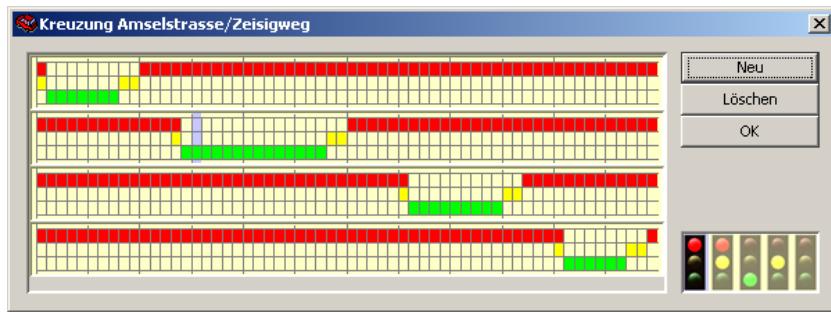


Bild 7.12: Eingabe eines Signalplans mit vier Signalgruppen

Die grundlegende Nachbildung der Funktionalität lichtsignalgeregelter Knotenpunkte ist, wie beschrieben, mit nur geringem Aufwand verbunden und wurde lauffähig umgesetzt. Insbesondere Kreuzungen ohne bedingt verträgliche Fahrzeugströme können vollständig modelliert und simuliert werden. Sind allerdings zusätzlich bedingt verträgliche Fahrzeugströme vorhanden, insbesondere beim Abbiegen durch den Gegenverkehr, muss zusätzlich eine Fahrspur zur Nachbildung einer Konfliktfläche, ähnlich zum Haltepunkt des Kreisverkehrs, in die Abbiegespur eingebunden werden. Die Einfahrt in diese Konfliktfläche ist demnach nur erlaubt, wenn andere Fahrzeuge vorrangiger Ströme die Konfliktfläche vollständig verlassen haben oder eine ausreichend große Zeitlücke vorhanden ist, um die Konfliktfläche vollständig zu überfahren.

7.4 Fußgänger

Um die Realitätsnähe der mikroskopische Verkehrssimulation weiter zu steigern können auch Fußgängermengen in die Simulation einbezogen werden. Ähnlich zur Fahrzeugsimulation werden dabei einzelne menschliche Verkehrsteilnehmer als Individuen im Softwaremodell nachgebildet und durch eine virtuelle Umgebung bewegt. Aufgrund ihrer geringen Größe, hohen Bewegungsfreiheit, starken Interaktionsabhängigkeiten und direkter Reaktion auf Umwelteinflüsse unterscheidet sich die Fußgängersimulation von der Fahrzeugsimulation. Sie stellt daher auch andere Anforderungen an das Simulationsmodell.

Einige herkömmliche Modellierungsansätze für die Beschreibung von Fußgängerströmen verwenden Regressionsrechnungen zur Mengenprognose ([82][104]) oder Warteschlangenmodelle zur Fluchtsimulation ([3]). Andere Verfahren nutzen strömungsmechanische Konzepte zur makroskopischen Bestimmung der Fußgängerströme ([60]). Aufgrund des stark variablen Verhaltens einzelner Personen in einer Menschenmenge bietet es sich allerdings an, ein mikroskopisches Modell zur Nachbildung des Bewegungsverhalten eines individuellen Fußgängers zu entwickeln, aus dem sich dann das makroskopische Verhalten durch Emergenzeffekte ableiten lässt ([58]). Bei den mikroskopischen Modellen muss zwischen räumlich diskreten und kontinuierlichen Modellen unterschieden werden.

Räumlich diskrete Modelle basieren auf dem Prinzip des Zellularautomaten, wie er auch für die Fahrzeugsimulation eingesetzt wird ([81]). Das Untersuchungsgebiet wird dabei in ein Raster gleichgroßer Zellen aufgeteilt, die jeweils entweder keinen oder einen (bei manchen Ansätzen auch mehrere) Fußgänger aufnehmen. Die Bewegung findet durch vordefinierte Regeln statt und erfolgt in ganzzahligen Schritten entlang des verwendeten Rasters, das meist quadratisch oder hexagonal ausgelegt ist. Das erste Modell dieser Art wurde von Gipps und Markjö entwickelt

([51]) und seitdem ständig verbessert ([101], [25], [68]). Wie auch bei der Verkehrssimulation können raumdiskrete Verfahren zwar makroskopische Phänomene gut nachgebilden, das mikroskopisches Verhalten einzelner Simulationselemente lassen sich aber nur grob abstrahieren. Soll auch die Bewegung einzelner Fußgänger mit höchstmöglicher Realitätsnähe simuliert werden, kommen raumkontinuierliche Strategien zur Anwendung.

Für die nahtlose Interaktion zwischen Fahrzeugen und Fußgängern in der Simulation wurde bei der Modellierung für beide Verkehrsträger ein vergleichbar hoher Detailierungsgrad angestrebt. Dazu wurde ein Fußgänger-Modell mit kontinuierlichen 2D-Koordinaten gewählt, das ausreichend realitätsnah und robust ist und gleichzeitig einen akzeptablen Rechen- und Kalibrierungsaufwand benötigt. Existierende raumkontinuierliche Simulationsverfahren, z. B. das soziale Kräftemodell nach Helbing ([59]) oder das Simulex-Modell von Thompson ([109]), lassen sich hinsichtlich ihrer Granularität nicht an das vorhandene Fahrzeugmodell anpassen und bleiben deshalb unberücksichtigt. Stattdessen wurde ein neues Fußgängermodell entwickelt, das auf dem bekannten, bereits 1986 entwickelten *Boids*-Modell von Reynolds ([96], [95]).

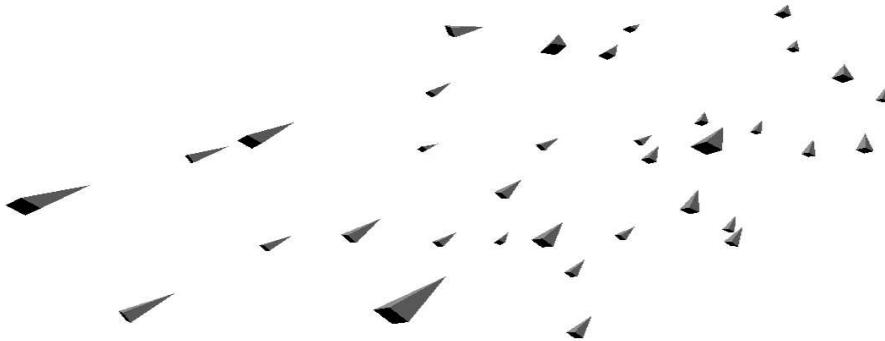


Bild 7.13: Ein Schwarm dreidimensionaler Boids (OpenSteer Project, Sourceforge)

Boids (als Abkürzung für *Birdoids* oder *Vogelähnliche*) sind Software-Agenten, die sich schwarmartig in einer virtuellen Umgebung nach einem Satz vorgegebener Verhaltensmuster bewegen. Jeder Boid wird durch eine punktförmige Masse repräsentiert, die sich mit einer Geschwindigkeit durch die zwei- oder dreidimensionale Simulationswelt bewegt. Beschleunigt werden Boids durch Verhaltenregeln. In der ursprünglichen Version der Boids-Software werden drei grundlegende Verhaltensmodelle unterschieden: Abstand halten (*separation*), Ausrichtung (*alignment*) und Zusammenhalt (*cohesion*). Die sich aus diesen einfachen lokalen Regeln entwickelnden globalen Verhaltensmuster erinnern stark an das reale Schwarmverhalten von Vögeln oder Fischen. Im Laufe der Zeit entwickelten Reynolds und andere zahlreiche weitere einfache Verhaltensmuster, zum Beispiel für das Verfolgen, Fliehen, Vermeiden von Hindernissen oder Ansteuern eines Zielpunktes. Ein großer Vorteil des Boids-Ansatzes ist die Modularität der einzelnen Verhaltensweisen, die je nach Anwendungsfall beliebig kombiniert und zu neuen, komplexeren Verhaltensstrategien zusammengesetzt werden können.

7.4.1 Fußgängermodell

Entsprechend dem in Kapitel 3 beschriebenen Modell für Fahrzeuge werden *Fußgänger* als Instanzen einer Klasse *Fussgaenger* zu jedem Zeitschritt der Simulation durch die Simulationswelt bewegt; die grafische Darstellung wird durch Vererbung von der Klasse *Zeichenobjekt* realisiert (vgl. Abbildung 7.14). Jeder Fußgänger hat dabei eine Position im dreidimensionalen Raum

und eine aktuelle Geschwindigkeit, mit der er sich fortbewegt. Im Gegensatz zum Fahrzeug ist ein Fußgänger nicht an eine feste Fahrbahn gebunden, sondern kann sich prinzipiell frei in der Simulationswelt bewegen.

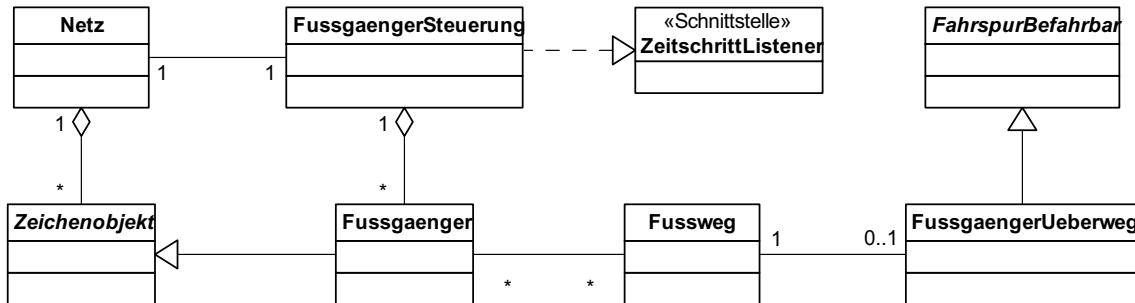


Bild 7.14: UML-Diagramm aller relevanten Klassen des Fußgängermodells

Jeder Fußgänger folgt bei seiner Bewegung einer Serie von Wegpunkten, die ihm zur Erzeugungszeit zugewiesen werden. Jeder Wegpunkt markiert einen Berührungspunkt zwischen zwei Fußwegen. Erreicht ein Fußgänger einen Wegpunkt, wird er auf den benachbarten Fußweg gesetzt. Jeder Fußgänger versucht stets, auf einem Fußweg zu bleiben. Befindet sich ein Fußgänger nicht auf einem Fußweg (z.B. weil er auf einem Parkplatz aus einem Fahrzeug ausgestiegen ist), versucht er den nächstgelegenen Fußweg zu erreichen. Sobald ein Fußgänger am letzten Wegpunkt der ihm zugewiesenen Strecke ankommt, hat er sein Ziel erreicht und wird aus der Simulation entfernt.

Insgesamt verfolgt ein Fußgänger im implementierten Modell drei verschiedene Ziele, die aus den entsprechenden Steuerungsverhalten des Boid-Modells abgeleitet wurden (vgl. Illustration 7.15).

- **Seek:** Der Fußgänger wird immer mit einer konstanten Wert in Richtung des nächsten Zielpunktes beschleunigt. Nur wenn der Zielpunkt gleichzeitig ein Fußgänger-Haltepunkt ist, wird die Beschleunigung ab einer Mindest-Entfernung linear verkleinert, um den Fußgänger vor dem Punkt zum Halten zu bringen.
- **Separation:** Benachbarte Fußgänger stoßen sich gegenseitig ab, um nicht aufeinander gedrängt zu werden. Die Abstoßungskraft steigt antiproportional zum Absolutabstand beider Fußgänger.
- **Containment:** Über zwei virtuelle Messfühler, die jeder Fußgänger in einem 30° Winkel rechts und links zu seiner aktuellen Ausrichtung vor sich herträgt, wird überprüft, ob sich das Ende eines Gehweges nähert. Verlässt ein Messfänger den Fußweg, wird die letzte Position innerhalb des erlaubten Gebietes ermittelt und eine orthogonale Beschleunigung antiproportional zur freien Weglänge aufgebracht.

Aus allen ermittelten Teilbeschleunigungen wird nach dem Superpositionsprinzip die resultierende Beschleunigung ermittelt. Die Geschwindigkeit des Fußgängers wird anschließend für die Dauer eines Zeitschritts um diese Beschleunigung erhöht. Übersteigt die neue Geschwindigkeit die Wunschgeschwindigkeit des Fußgängers, wird der Geschwindigkeitsvektor so skaliert, dass sein Betrag der gewünschten Geschwindigkeit entspricht. Erst wenn für alle Fußgänger die neue

Geschwindigkeit ermittelt wurde, werden sie entsprechend weiterbewegt. Durch diese zweistufige Berechnung ist sichergestellt, dass die Abstände zwischen den Fußgängern während der Berechnung nicht verfälscht werden.

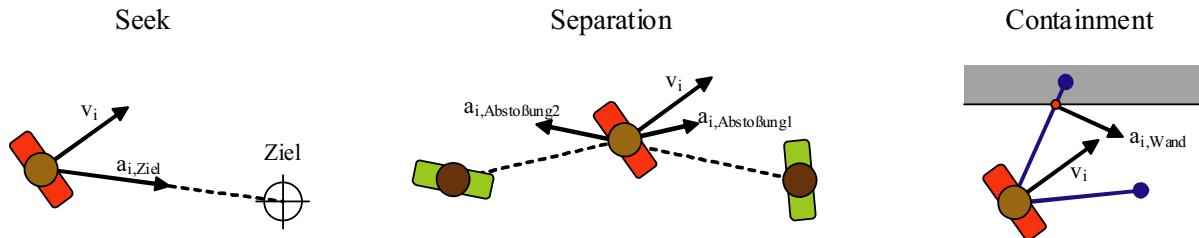


Bild 7.15: Die drei implementierten Verhaltensmuster des Fußgängermodells

Für die Wunschgeschwindigkeit wurde die Studie von Weidmann ([121]) herangezogen, der 1992 verschiedene Untersuchungen zur Durchschnittsgeschwindigkeit von Fußgängern durchgeführt hat. Danach kann die mittlere Geschwindigkeit eines einzelnen Fußgängers mit 1,34 m/s bei einer Standardabweichung von 0,26 m/s angenommen werden. Mit diesen Werten wird die Wunschgeschwindigkeit eines Fußgängers bei der Erzeugung festgelegt und während der Lebenszeit des Objektes konstant gehalten. Der in der Wirklichkeit zu beobachtende Effekt, dass die tatsächliche Geschwindigkeit mit steigender Fußgägerdichte abnimmt, ist durch gegenseitige Abstoßung dicht benachbarter Personen gewährleistet.

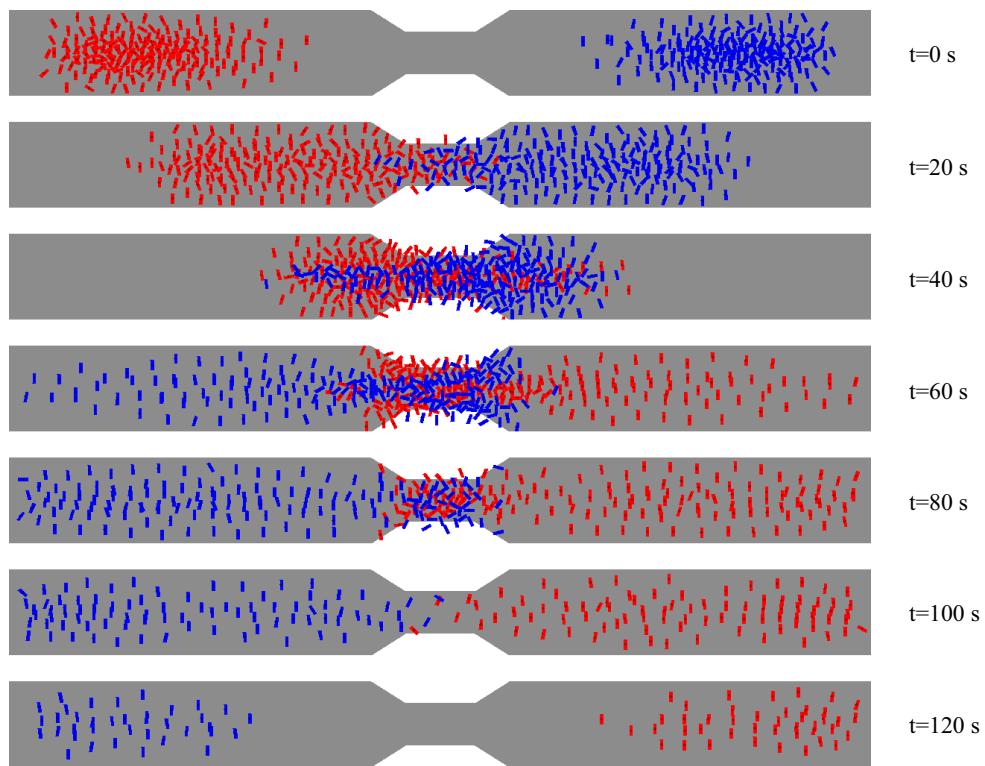


Bild 7.16: Bildfolge zweier aufeinander treffender Fußgängergruppen

Abbildung 7.16 zeigt zwei aufeinander zugehende Gruppen von je 200 Fußgängern, die eine Engstelle passieren müssen. Durch die Abstoßung an Wänden und benachbarten Fußgängern entsteht schnell ein Stau im Bereich vor der Engstelle. Nach und nach bilden sich einzelne Gassen heraus, durch die der Personenstrom zur jeweils anderen Seite abfließen kann. Je schneller

sich der Stau auflöst und damit der Druck durch die von hinten nachströmenden Personen nachlässt, desto schneller kann der Fußgängerstrom die Engstelle passieren. Nach etwa 100 Sekunden haben alle Fußgänger den Bereich um die Engstelle verlassen und bewegen sich mit Wunschgeschwindigkeit auf ihren Zielpunkt zu.

7.4.2 Kopplung mit dem Straßenverkehr

Um den Rechenaufwand für Interaktionen zwischen Fahrzeugen und Fußgängern so gering wie möglich zu halten, werden beide grundsätzlich getrennt voneinander simuliert. Nur an vordefinierten Schnittstellen haben sie direkten Kontakt. Die Bewegung von Fahrzeugen ist begrenzt auf ihre Fahrspuren; Fußgänger hingegen bewegen sich, mit wenigen Ausnahmen, ausschließlich auf Fußwegen. Schnittstellen treten somit dort auf, wo sich die beiden Verkehrsinfrastruktur-Elemente „Fußweg“ und „Fahrspur“ überschneiden. Die folgenden Berührungspunkte können auftreten:

- **Fußgängerüberweg bzw. Fußgängerfurt:** Vordefinierte Flächen, an denen Fußgänger einzelne Fahrspuren des Straßennetzes plangleich überqueren können. Um Unfälle für alle Beteiligten auszuschließen, muss jeweils der nach der StVO bevorrechtigte Verkehrsteilnehmer die Konfliktfläche passiert und verlassen haben, bevor der nicht-bevorrechtigte Verkehr diese betreten bzw. befahren darf. An Fußgängerüberwegen („Zebrastreifen“) und lichtsignalgeregelten Fußgängerfurten ist der Fußgängerverkehr bevorrechtigt, an allen anderen Fußgängerfurten der nicht-abbiegende Fahrzeugverkehr. Die Vorgabe, welcher Verkehrsstrom bevorrechtigt ist, wird in der Simulation durch geeignete Attribute und Assoziationen festgelegt.
- **Parkplatz bzw. Stellplatz:** An Stellplätzen können Fahrer ihre Fahrzeuge verlassen und werden zu Fußgängern. Sobald ein Fahrer sein Fahrzeug verlässt, muss er als Fußgänger auf direktem Weg den nächsten Fußweg ansteuern und sich entlang einer Folge von Wegpunkten einem Ziel nähern. Umgekehrt kann ein Fußgänger auch wieder zum Fahrer eines Fahrzeuges werden, wenn der Endpunkt seiner Wegpunktliste zu einem Stellplatz gehört. Ob ein Fußgänger einen Stellplatz ansteuert, wird vom Wegfindungssystem der Fußgänger festgelegt.
- **ÖPNV-Haltestelle:** An Haltestellen von Busen, Straßenbahnen und ähnlichen ÖPNV-Fahrzeugen können in der Simulation neue Fußgänger generiert bzw. bestehende Fußgänger aus der Simulationswelt entfernt werden. Die Schnittstelle zwischen Fahrzeug und Fußgänger stellt hierbei die Haltestelle dar, die über eine Assoziation mit der Fahrspur des ÖPNV-Fahrzeugs verbunden ist. Fußgänger, die in ein ÖPNV-Fahrzeug einsteigen wollen, werden über das Wegpunktesystem bis zum Haltestellen-Fußweg bewegt und warten dort auf das Eintreffen eines Fahrzeugs, in das sie anschließend einsteigen. Gleichzeitig werden aussteigende Passagiere zu neuen Fußgänger und auf dem Haltestellen-Fußweg mit einer gewünschten Zielroute gesetzt.

ÖPNV-Verkehr und Stellplätze werden in den Abschnitten 7.5 und 7.2 näher beschrieben, daher wird hier nur auf die Modellierung von Fußgängerüberwegen eingegangen. Als Beispiel wird ein bevorrechtigter Fußgängerüberweg mit Zebrastreifen (Zeichen 293, StVO) betrachtet, der sich über zwei einstreifige Richtungsfahrbahnen erstreckt (Abbildung 7.17). Auf beiden Fahrbahnen

müssen die Fahrzeuge vor dem Zebrastreifen anhalten, wenn sich Fußgänger auf oder kurz vor dem Zebrastreifen befinden. Können die Fahrzeuge nicht mehr rechtzeitig anhalten, oder befinden sie sich bereits auf dem Zebrastreifen, müssen sie den Überweg so schnell wie möglich verlassen.

Das Anhalten der Fahrzeuge wird über eine unsichtbare Haltelinie, ca. einen Meter vor dem eigentlichen Zebrastreifen, realisiert und über die punktförmige Fahrspur vom Typ *FussgaengerUeberweg* modelliert. Zu jedem Zeitschritt wird überprüft, ob sich im Einzugsbereich des Überwegs (der sich vom Zebrastreifen bis zum eigentlichen Bürgersteig erstreckt) Fußgänger befinden. Falls ja, wird die Fahrspur für den Verkehr gesperrt, der Fußgängerüberweg wird den sich nähernden Fahrzeugen als Hindernis gemeldet. Erst wenn der letzte Fußgänger den Kontrollbereich verlassen hat, wird der Überweg wieder für Fahrzeuge freigegeben.



Bild 7.17: Haltegebot für Fahrzeuge an einem Fußgängerüberweg

Für den Fußgänger stellt ein Fußgängerüberweg in der Simulationswelt einen normalen Fußweg dar, dem er mit seinem normalen Laufverhalten folgt. Da der Fahrzeugverkehr Fußgängern an einem Fußgängerüberweg Vorrang gewähren muss, ist von Seiten der Fußgänger keine Berücksichtigung des Fahrzeugverkehrs nötig. Lediglich bei nicht oder nur bedingt bevorrechtigten Überquerungsmöglichkeiten (z.B. Fußgängerfurten an Lichtsignalanlagen) sind die Wegpunkte am Fahrbahnrand zusätzlich als Haltepunkte auszulegen. Haltepunkte erlauben es den Fußgängern erst dann die Fahrbahn zu betreten, wenn keine Gefährdung durch den Straßenverkehr besteht bzw. eine Lichtsignalanlage die Überquerung gestattet.

Das Objektdiagramm in Abb. 7.18 verdeutlicht die Wechselwirkung der Objekte, die zur Modellierung eines einfachen Fußgängerwegs (Abb. 7.17) eingesetzt werden.

Für die Ablaufsteuerung wird angenommen, dass stets der gesamte Zebrastreifen für den Fahrzeugverkehr gesperrt ist, selbst wenn alle Fußgänger den für ein Fahrzeug relevanten Fahrstreifen bereits verlassen haben. Um die „Vollsperrung“ zu vermeiden und eine Trennung zwischen beiden Fahrspuren zu erreichen, kann der Fußweg in der Fahrbahnmitte in zwei miteinander verbundene Fußwege aufgeteilt werden, die jeweils mit einem der beiden Fußgängerüberwege assoziiert sind. So lassen sich auch Übergänge an Verkehrsinseln realisieren.

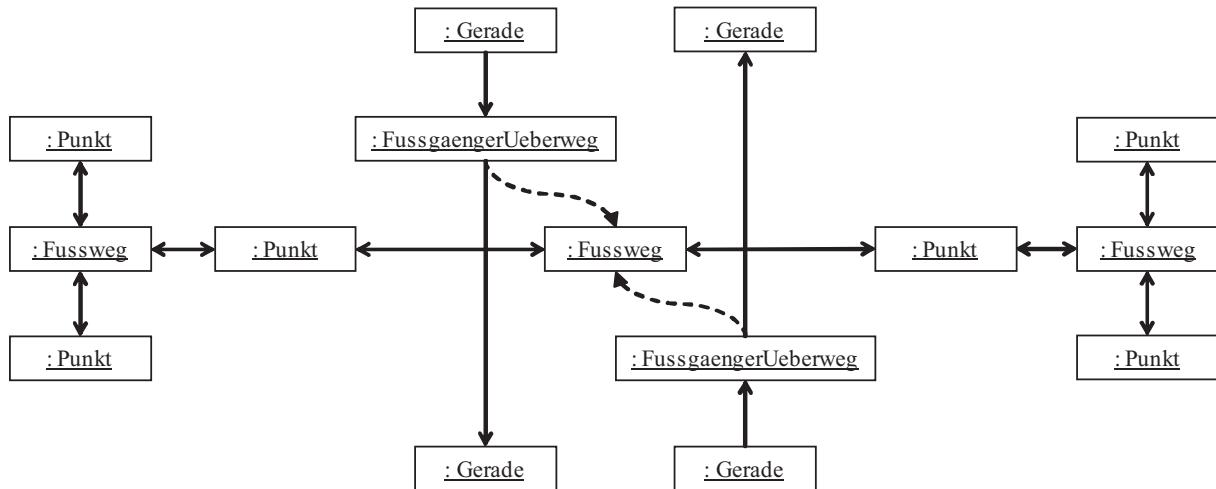


Bild 7.18: Objektdiagramm eines Fußgängerüberwegs über zwei Fahrspuren

7.5 ÖPNV

Ein weiteres mögliches Einsatzgebiet für innerstädtische Verkehrssimulation ist die Nachbildung und Planung des Öffentlichen Personennahverkehrs (ÖPNV) für den schienen- oder straßengestützten Transport von Passagieren. Da der größte Teil dieser Verkehrsform auf festen Taktfahrplänen beruht, sind die ÖPNV-Verkehrsunternehmen auf eine hohe zeitliche Zuverlässigkeit ihrer Fahrzeuge angewiesen. Verlustzeiten durch Staus oder Streckensperrungen wirken sich sehr nachteilig auf Verkehrsqualität und Zuverlässigkeit des ÖPNV aus, da insbesondere Anschlußverbindungen nicht mehr erreicht werden können. Es bietet sich also an, den reibungslosen Verkehrsablauf auch bei hohen Verkehrsstärken mit einer geeigneten Simulation zu überprüfen. Weiterhin kann die Simulation helfen, die Effekte straßenplanerischer Eingriffe in das Streckennetz zugunsten des ÖPNV-Verkehrs zu untersuchen: Separate Busspuren, „Grüne Wellen“ für Busse oder Straßenbahnen, die Änderung von Fahrbahngeometrien im Bereich von ÖPNV-Haltestellen oder die Effekte einer Taktverdichtung sind einige der Untersuchungsmöglichkeiten.

Zu den im ÖPNV eingesetzten Transportmitteln zählen nach dem Personenbeförderungsgesetz Busse, Straßenbahnen, U-Bahnen und – je nach Rechtsform – andere Schienenbahnen. Ungewöhnliche Verkehrsmittel, wie zum Beispiel Fähr- und Schiffsverkehr, Liftanlagen, Hängebahnen (wie die Wuppertaler Schwebebahn), oder ÖPNV-Sonderformen wie Anrufsammeltaxis werden aufgrund ihrer vergleichsweise geringen Verbreitung im Weiteren nicht behandelt. Bei den schienengebundenen Systemen gibt es nur wenige modellierungsspezifische Unterschiede; aus diesem Grund wird exemplarisch die Umsetzung von Straßenbahnlinien betrachtet.

7.5.1 Linienbusse

Das ursprünglich in die Simulationssoftware eingebundene Fahrzeug-Klassenmodell umfasst lediglich zwei Arten von Fahrzeugen: Pkw und Lkw. Die Gruppe der Lkw umfasst dabei auch Lkw-ähnliche Fahrzeuge, wie Lastzüge, Linienbusse oder Reisebusse. Die Ähnlichkeit bezieht sich dabei auf vergleichbare Abmessungen, maximale Geschwindigkeitsniveaus und Beschleunigungs- bzw. Bremsvermögen. Um Linienbusse als dritte Fahrzeugart in die Simulation zu integrie-

ren, werden die fahrzeugspezifischen Eigenschaften der Lkw so weit wie möglich übernommen, um eine Neukalibrierung des Fahrverhaltens für Busse zu vermeiden und den makroskopischen Einfluss des Lkw-Verkehrs nicht zu ändern. Im Klassenmodell der Simulationssoftware wird dies dadurch realisiert, dass jeder Bus zunächst grundsätzlich alle Fahreigenschaften eines Lkws erbt; nur bei Bedarf werden Änderungen an einzelnen Komponenten des Fahrzeugmodells vorgenommen.

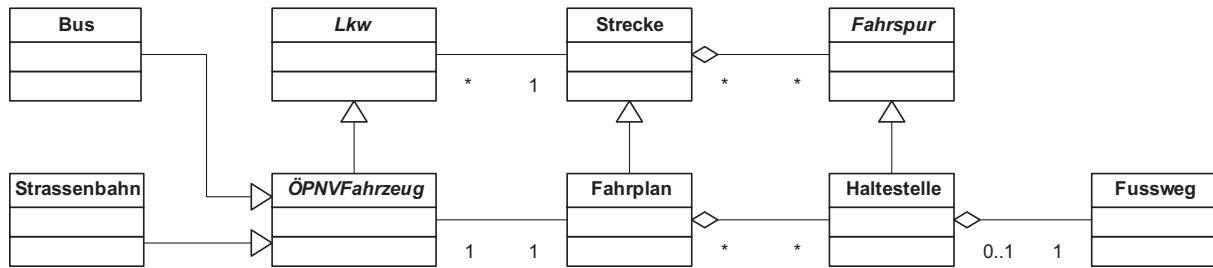


Bild 7.19: Klassendiagramm für die Modellierung von ÖPNV

Jedem Bus wird ein ihm eigener Fahrplan zugewiesen, den er – Haltestelle für Haltestelle – abzufahren hat. Ein Fahrplan ist dabei eine normale Strecke (wie sie auch von allen anderen Fahrzeugen befahren wird), die jedoch um zusätzliche Informationen über Haltestellen und Abfahrtzeiten ergänzt wird. Im Gegensatz zu normalen Strecken besitzt ein Fahrplan keine Quelle oder Senke als Zielpunkt, sondern verhält sich zyklisch, um die Wiederholungen der Busfahrten (z. B. im Halbstunden- oder Stundentakt) umsetzen zu können. Vereinfachend fahren alle Busse ihren Fahrplan in einer unendlichen Schleife immer wieder ab - bei Bedarf kann aber auch bei Erreichen einer Endzeit (etwa bei Betriebsschluß der Buslinie) auf eine normale Strecke umgeschaltet werden, um den Bus zurück ins Depot zurückzufahren. Wird auf die automatische Wiederholung der Fahrten verzichtet (z. B. bei häufigen Taktänderungen im Laufe eines Betriebstages), kann auch ein Tagesfahrplan spezifiziert werden, der nur einmal durchlaufen wird. Die Angabe der Abfahrtzeiten an jeder Haltestelle erfolgt in ganzen Minuten, es ist jedoch auch eine sekundengenaue Taktung möglich.

Jedes Fahrzeug besitzt eine maximale Fahrgäste-Kapazität. Diese ist abhängig vom jeweiligen Fahrzeugtyp und wird von den Unterklassen der Klasse *ÖPNVFahrzeug* individuell bestimmt. Das Aussteigen der Fahrgäste erfolgt nach dem Zufallsprinzip. Haltestellen fungieren als Haltepunkte für alle ÖPNV-Fahrzeuge, die an dieser Haltestelle anhalten müssen. Damit ein ÖPNV-Fahrzeug eine Haltestelle als Hindernis erkennt, muss mindestens eine der folgenden Bedingungen erfüllt sein:

1. im Fahrzeug befinden sich Fahrgäste, die an der Haltestelle aussteigen wollen
2. an der Haltestelle warten Fußgänger, die in das Fahrzeug einsteigen wollen
3. die fahrplanmäßige Abfahrtszeit ist noch nicht erreicht

Ist eine dieser Bedingungen erfüllt, bleibt das Fahrzeug (der Bus) vor der imaginären Haltestelle stehen. Sobald dabei die Geschwindigkeit einen Grenzwert von 0,1 m/s unterschreitet, werden die Türen des Fahrzeugs geöffnet, und die im bzw. vor dem Fahrzeug wartenden Fahrgäste können aus- bzw. einsteigen. Jeder Fußgänger, der in ein ÖPNV-Fahrzeug einsteigt, wird aus der Simulationswelt entfernt und der internen Passagierliste des Fahrzeugs

hinzugefügt. Fußgänger, die aussteigen wollen, werden dagegen aus der Passagierliste entfernt und wieder in die Simulation eingefügt. Die Ausstiegszeit pro Fahrgäst ist frei wählbar und entspricht in der Referenzimplementierung der Dauer eines Zeitschritts (0,1 Sekunde). Jede Haltestellen-Fahrspur ist mit einem Fußweg assoziiert, der von den aussteigenden Fußgänger genutzt wird. Auf diese Weise ist sichergestellt, dass jeder Fußgänger schnellstmöglich den Fahrbahnbereich in Richtung Fußweg verlässt. Neu erzeugten Fußgänger wird beim Verlassen des ÖPNV-Fahrzeugs eine neue Folge von Wegpunkten als Strecke zugewiesen. Das ÖPNV-Fahrzeug setzt seine Fahrt fort, wenn alle Passagiere ein- bzw. ausgestiegen sind und die Abfahrtzeit erreicht wurde.



Bild 7.20: Erzeugung von Fußgängern durch ein ÖPNV-Fahrzeug

7.5.2 Straßenbahnen

Mit dem entwickelten Klassenmodell für ÖPNV-Fahrzeuge ist die Umsetzung von Straßenbahnen (stellvertretend für andere Schienenfahrzeuge des ÖPNV) allein durch Ausnutzung von Vererbungseffekten möglich: Die Klasse *Strassenbahn* wird direkt als Unterklasse von *ÖPNVFahrzeug* definiert und erbt automatisch alle Eigenschaften, die das Aufnehmen von Passagieren, das Halten an Haltestellen und das Ein- und Aussteigen von Passagieren erlauben. Das normale Fahrverhalten auf freier Strecke hingegen wird von der Oberklasse Lkw übernommen. Lediglich Straßenbahn-spezifische Werte, wie Beschleunigungs- und Bremsvermögen, Passagierkapazität, Fahrzeugabmessungen und Wunschgeschwindigkeitsverteilung, müssen dabei angepasst werden. Im Folgenden werden nur Straßenbahnen betrachtet, die über ein separates Gleisbett verfügen und damit nicht direkt mit dem Straßenverkehr interagieren.

Bezüglich der Netzgeometrie bestehen einige Unterschiede zwischen Straßen- und Schienenverkehr: Schienenverkehr mit separatem Gleisbett ist aufgrund der langen Bremswege zum größten Teil signalgesteuert, d. h. das Abstandsmodell muss vorwiegend auf Haltepunkte und weniger auf andere Fahrzeuge reagieren. Mehrspurbereiche sind aufgrund der Schienengebundenheit der Straßenbahn nicht erforderlich. Gleiswechsel finden ausschließlich an Weichen statt, denen somit die Rolle der Abzweige im Straßenverkehr zukommt. Geraden und Kurven repräsentieren gerade und gebogene Schienensegmente. Alle wesentlichen Elemente des Straßenverkehrs können somit direkt auf den ÖPNV übertragen werden.

Innerhalb des Netzeditors der Simulationssoftware erfolgt die Erstellung von Straßen bzw. Schienen völlig gleichartig. Alle Werkzeuge zur Erstellung von Straßen können ebenso für die Erstellung von Schienen verwendet werden. Welcher Typ von Trasse (Schiene/Straße) erzeugt wird,

hängt allein davon ab, ob die *OptionSchiene* aktiviert oder deaktiviert ist. Bei aktiverter Option wird jede neu erzeugte Fahrspur als Schiene markiert. Intern werden alle Schienen als Fahrspuren mit aktiver Schienenmarkierung in Form eines *boolean*-Attributs abgelegt. Abbildung 7.21 zeigt zwei Gruppen von Fahrspuren, die sich allein durch den Wert ihrer Schienen-Attributs unterscheiden.

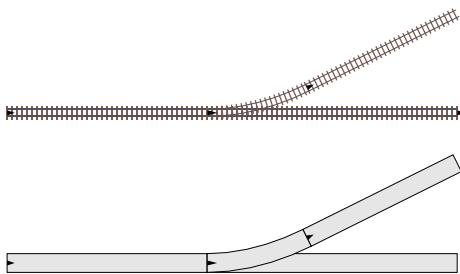


Bild 7.21: Straßen und Schienen werden auf die gleiche Weise erstellt

Bezogen auf das bei der Implementierung verwendete MVC-Schema (vgl. Kapitel 6.2) besteht das wesentliche Unterscheidungsmerkmal zwischen einem Gleis und einer Straße allein in der *View-Komponente*, also der grafischen Darstellung; *Model* und *Controller* des MVC-Schemas bleiben unverändert. Um die Visualisierung der Schiene ohne große Änderungen an den existierenden Zeichenmethoden zu ermöglichen, wird vor dem Aufruf der eigentlichen Zeichenmethoden (die eine Fahrspur zeichnen würden) abgefragt, ob die Fahrspur eine Schiene ist. In diesem Fall wird das Zeichnen an die separate Zeichenklasse *Gleis* (für die zweidimensionale Ansicht) bzw. *Gleis3D* (für die dreidimensionale Darstellung) verwiesen. Abbildung 7.22 zeigt eine Straßenbahn in der dreidimensionalen Ansicht.



Bild 7.22: Eine Straßenbahn an einer Weiche

Kapitel 8

Simulation großflächiger Straßennetze

8.1 Einordnung des Anwendungsspektrums

Auf der Basis des in den vorherigen Kapiteln beschriebenen grundsätzlichen Aufbaus und der Funktionsweise der Simulationssoftware kann eine Abschätzung der Einsatzfähigkeit der Simulationssoftware für die Abbildung des Verkehrsflusses auf räumlich ausgedehnten Streckennetze erfolgen. Gestützt auf ein theoretisch fundiertes, informatisch konsistent umgesetztes objektorientiertes Modell lassen sich mit den Werkzeugen der Software Streckennetze hinreichend detailliert aufbauen, gezielt Simulationsläufe – auch mit variierenden Parametern – durchführen und sowohl qualitativ als auch quantitativ auswerten. Es soll auch gezeigt werden, dass die konsequente Einhaltung und Anwendung objektorientierter Programmierprinzipien die Wiederverwendbarkeit der Software bei neuen Aufgabenstellungen unterstützt und ihre Erweiterbarkeit ermöglicht.

Der Kern der Software wurde für die Simulation des Verkehrs auf kurzen Autobahn-Teilstücken mit Längen von maximal 50 Kilometern und einer überschaubaren Anzahl an Ein- und Ausfahrten ausgelegt. Auch während der Entwicklung und Kalibrierung beschränkten sich die Abmessungen der untersuchten Beispielnetze auf wenige Kilometer. Für das gewählte Anwendungsszenario soll nun die Verwendbarkeit des Netzeditors unter praxisnahen Einsatzbedingungen untersucht und die Erstellung eines umfassenden Autobahnnetzes – einschließlich der Detailmodellierung aller Anschlussstellen – unter Verwendung der entwickelten Modellierwerkzeuge gezeigt werden. Untersucht wird hierbei insbesondere, bis zu welcher Netzkomplexität der Editor noch sinnvollerweise verwendet werden kann. Es werden dabei auch Lösungsansätze diskutiert, die die Handhabung größerer Netze erleichtern.

Für die Simulation des Gesamtnetzes wird zunächst ein einzelner Rechner eingesetzt und sein Laufzeitverhalten in Abhängigkeit von den simulierten Fahrzeugmengen untersucht. Wichtig ist insbesondere, wie der Speicherbedarf mit der Fahrzeughichte korreliert ist. Zum Vergleich wird dasselbe Netz anschließend an den in Kapitel 6.4 beschriebenen Parallelrechner gesendet, dort ein verteilter Simulationslauf durchgeführt und die Ergebnisse den Einzelplatzrechnungen gegenübergestellt. Das Kapitel schließt ab mit einer generellen Beurteilung der Einsatztauglichkeit der entwickelten Simulationssoftware, insbesondere in Hinblick auf die Simulation großer Netze.

8.2 Netzerstellung

Der Simulation wird ein Ausschnitt aus dem realen deutschen Bundesautobahnnetz einschließlich aller Zu- und Ausfahrten zu Grunde gelegt. Insbesondere Bereiche mit hoher Fahrzeug-Interaktionsdichte, etwa Autobahnkreuze oder dicht aufeinander folgende Anschlussstellen, werden dabei möglichst detailgetreu erfasst, um den Korrektheit der Verhaltensmodelle überprüfen zu können. Die Gesamtlänge der Autobahnteilstücke ist so zu bemessen, dass eine Unterteilung in Subnetze von geeigneter Größe möglich ist, und damit die Parallelisierungseigenschaften der Simulation untersucht werden können.

Ausgewählt wurde ein Gebiet im mittleren Ruhrgebiet, das sich in Ost-West-Richtung von Dortmund bis Essen über ca. 23 km erstreckt, und in Nord-Süd-Richtung vom Autobahnkreuz Recklinghausen bis zum Autobahnkreuz Wuppertal-Nord eine Länge von ca. 32 Kilometern aufweist (Abbildung 8.1). Das Beobachtungsgebiet hat eine hohe Autobahndichte: Hier verlaufen die Bundesautobahnen A2, A40, A42, A43, A44 und A45; allein 9 Autobahnkreuze liegen in diesem Bereich. Dabei finden sich im ausgesuchten Verkehrsgebiet sowohl sehr dicht befahrene Autobahnabschnitte mit zahlreichen dicht aufeinander folgenden Anschlussstellen (z.B. die A40 bei Bochum mit einer durchschnittlichen täglichen Verkehrsstärke (DTV) von 110.864 Fahrzeugen) als auch gering belastete Teilstücke mit nur wenigen Anschlussstellen (A43 bei Sprockhövel mit einer DTV von 58.544 Fahrzeugen).

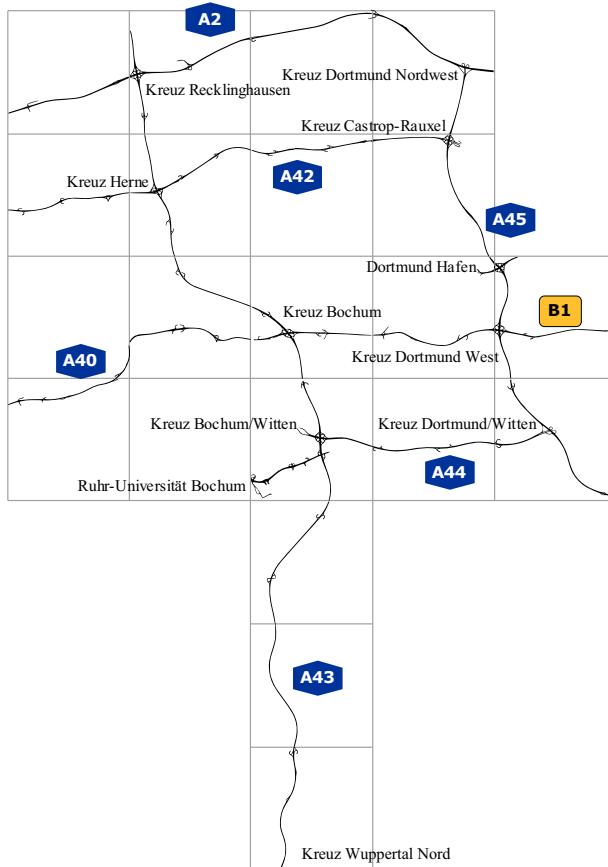


Bild 8.1: Die in ein Streckenmodell umzusetzenden Autobahnen im östlichen Ruhrgebiet

Das modellierte Streckennetz enthält in erster Linie Bundesautobahnabschnitte; die Nachbildung beschränkt sich auf die autobahnähnlich ausgebauten Universitätsstraße in Bochum zwischen

der Ruhr-Universität und der Anschlussstelle Bochum Querenburg. Da bei der Netzerstellung nicht abzusehen war, bis zu welcher Netzgröße die Konstruktion und Simulation ohne Performanzprobleme durchgeführt werden kann, wurde ein modularartiges Vorgehen gewählt: Der betrachtete Gesamtbereich wird über ein Gitterraster in einzelne quadratische Segmente zerlegt. Jedes Segment wird dabei als abgeschlossener Bereich betrachtet, der jeweils separat modelliert und gespeichert wird. Erst im Anschluss daran werden die Einzelmodule zu einem Gesamtnetz zusammengesetzt. Hierdurch ist sichergestellt, dass die Netzgröße bei Bedarf durch Hinzufügen oder Entfernen einzelner Segmente variiert werden kann. Außerdem lassen sich auf diese Weise gezielt nur einzelne Bereiche des Gesamtnetzes ausgewählen, um Detailuntersuchungen durchzuführen. Insgesamt ergeben sich so 21 Teilnetzen (vgl. Abbildung 8.1); Gebiete, in denen sich keine Autobahnabschnitte befinden, werden nicht berücksichtigt.

Als Vorlage für die Modellierung werden Referenz-Luftaufnahmen verwendet, die über das in Kapitel 6.3.2 beschriebene Luftbild-Werkzeug direkt in den Netz-Editor importiert werden (Abbildung 8.2). Datenquelle ist hierbei der WMS-Server des Regionalverbands Ruhr, der das betrachtete Gebiet flächig abdeckt und in den Orthofotoaufnahmen eine optische Auflösung von bis zu 0,5 m pro Pixel erreicht. Bei dieser Detailtiefe sind Autobahnfahrspuren, Verzögerungs- und Beschleunigungsstreifen bereits deutlich zu erkennen und können während der Konstruktion als Orientierungspunkte dienen.

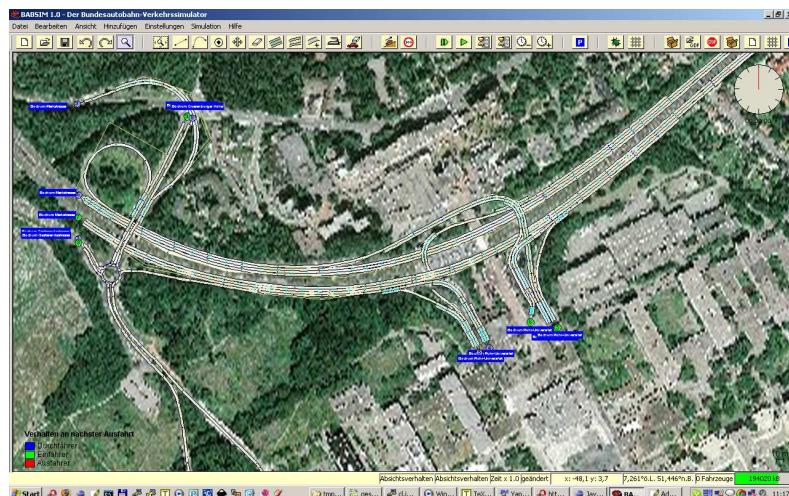


Bild 8.2: Fahrspur-Konstruktion auf Luftbildern (Regionalverband Ruhr, Essen)

Die Hintergrundbilder werden beim Importieren automatisch so skaliert, dass ihre Größe im Koordinatensystem der Simulationswelt der Größe des von ihnen in der Realität abgedeckten Gebiets entspricht. Der Referenzpunkt des Koordinatenstroms wird auf den Einfügepunkt des Hintergrundbildes gesetzt und stellt so die deckungsgleiche Lage des realen und des simulierten Koordinatensystems sicher. Werden einem Teilnetz weitere Teilnetze hinzugefügt, werden diese im lokalen Koordinatensystem der Simulation so verschoben, dass die Referenzpunkte aller Teilnetze übereinstimmen. Bei dieser Projektion der Weltkoordinaten (verwendet wird das Referenzellipsoid WGS 84) auf eine zweidimensionale Ebene kommt es zwangsläufig zu Verzerrungen; diese sind jedoch für die betrachteten Netzgrößen vernachlässigbar klein und liegen unterhalb der angestrebten Konstruktionsgenauigkeit.

8.2.1 Hauptfahrspuren

Im Netz-Editor der Simulationsumgebung sind verschiedene Möglichkeiten vorgesehen, um neue Fahrspuren zu erzeugen. Für einzelne befahrbare Fahrspuren existieren Werkzeuge zur Erstellung von Geraden und Kurven. Liegen mehrere gleichartige Fahrspuren seitlich benachbart nebeneinander, können sie durch das *Verbinden*-Werkzeug zu einem gemeinsamen Mehrspurbereich zusammengefasst werden. Über das Werkzeug *Mehrspur erweitern* werden neue lateral angrenzende Fahrspuren erzeugt, wodurch Spuraufweiterungen oder -einziehungen modelliert werden können.

Speziell für mehrstreifige Fahrbahnquerschnitte wird ein Werkzeug eingesetzt, das die Erzeugung von Regelquerschnitten nach den Richtlinien für die Anlage von Straßen (RAS-Q, [91]) ermöglicht. Hierzu wird zunächst mit den beiden Einzelfahrspur-Werkzeugen *Gerade* und *Kurve* die Mittelachse des zu erstellenden Mehrspurbereichs festgelegt (Abbildung 8.4 oben), wobei die importierten Luftbilder als Referenz verwendet dienen. Als hilfreich erweist sich dabei die Flexibilität der Kurven, die als kubische Bézier-Splines umgesetzt wurden und daher durch Verschieben der Stützpunkte leicht an den tatsächlichen Straßenverlauf angepasst werden können. Mit dem Werkzeug *Übergänge glätten* wird sichergestellt, dass die Tangenten der Kurven in allen Berührungs punkten stetig an die Nachbarfahrspuren anschließen.

Bei der Erstellung der Fahrbahnmittelachse ist darauf zu achten, an welchen Stellen der Straße später eine Querschnittsänderung erfolgt. Am Beginn und am Ende eines Verzögerungs- oder Beschleunigungsstreifens sind entsprechende Zwischenpunkte vorzusehen, da ein späteres Unterteilen der Mehrspurbereiche erheblichen Mehraufwand erfordert.

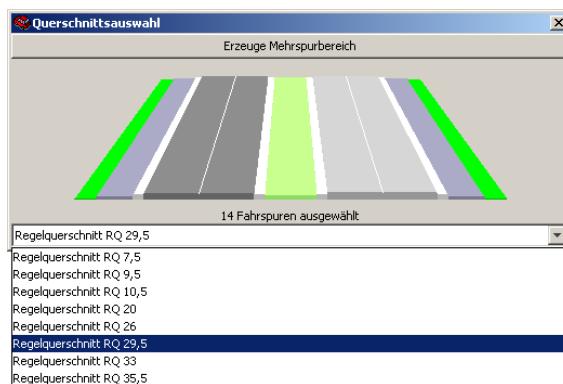


Bild 8.3: Werkzeug zur Erzeugung von Regelquerschnitten nach RAS-Q

Sobald alle Fahrspuren der Fahrbahnmittelachse erstellt sind, werden über das sogenannte *Autobahn*-Werkzeug (Abbildung 8.3) die umzuwandelnden Fahrspuren ausgewählt. Unterstützt werden alle nach RAS-Q definierten Regelquerschnitte – mit Ausnahme des asymmetrischen Querschnitts RQ 15,5, der nicht automatisch erzeugt, aber bei Bedarf manuell modelliert werden kann.

Für die Modellierung des Beispielnetzes wurden für alle Autobahnabschnitte mit zwei Fahrstreifen pro Fahrtrichtung der Regelquerschnitt RQ 29,5 gewählt, für alle dreistreifigen RQ 35,5. Durch Bestätigung der Auswahl wird die Hilfsspur entlang der Fahrbahnmittelachse entfernt und durch neue Mehrspurbereiche mit dem gewählten Querschnitt ersetzt (vgl. Abbildung 8.4 Mitte).

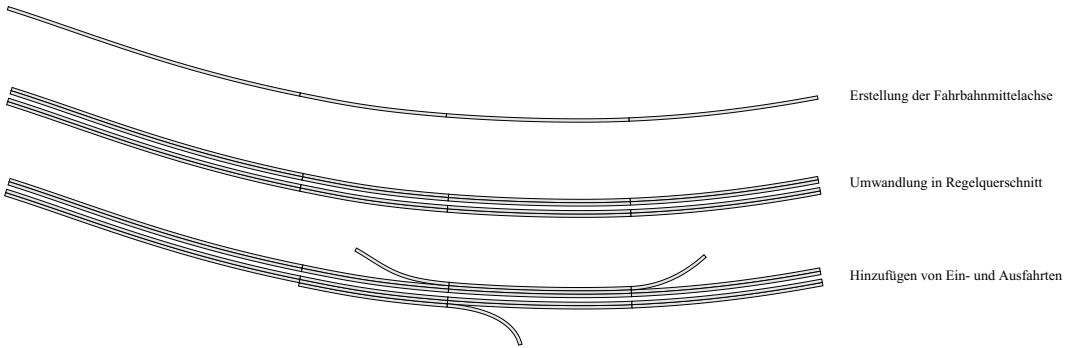


Bild 8.4: Vorgehensweise bei der Erstellung einer Autobahnanschlussstelle

8.2.2 Zu- und Abfahrten

Nachdem alle Hauptfahrbahnen erstellt wurden, erfolgt die Detailmodellierung der Anschlussstellen. Wichtigstes Werkzeug für das Hinzufügen einer Beschleunigungs- oder Verzögerungsspur ist das bereits beschriebene *Mehrspur erweitern*-Werkzeug, mit dem zusätzliche Fahrspuren rechts von bestehenden Mehrspurbereichen angefügt werden können. Durch das Einfügen von Kurven in oder entgegen zur Fahrtrichtung entstehen so Ausfahrten, Zufahrten oder Verflechtungsbereiche. Abbildung 8.4 (unten) zeigt die Erweiterung der zuvor erzeugten Autobahn um eine Ausfahrt und eine Verflechtungsspur.

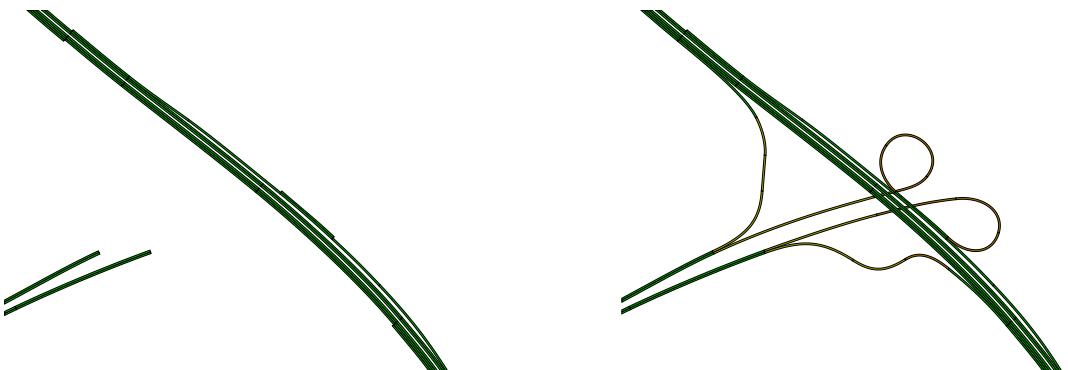


Bild 8.5: Rampen-Modellierung am Beispiel des Autobahnkreuzes Dortmund/Witten

Nach der gleichen Vorgehensweise können neben Anschlussstellen auch Autobahnkreuze und -dreiecke modelliert werden: Zunächst werden beide Hauptfahrsäulen der Autobahnen erstellt, anschließend die Verteilerfahrbahnen einschließlich Verzögerungs- und Beschleunigungsstreifen hinzugefügt, und schließlich über Kurven als Verbindungsrampen miteinander verbunden. Bild 8.5 zeigt die Verbindung zweier Autobahnen am Beispiel des Autobahnkreuzes Dortmund-Witten, das als halbes Kleeblatt ausgeführt ist.

Um ein realitätsnahe Geschwindigkeitsprofil der ein- und ausfahrenden Fahrzeuge zu erhalten, werden auf allen Beschleunigung- und Verzögerungsstreifen geeignete Geschwindigkeitsbegrenzungen vorgesehen. Da diese nicht aus dem vorhandenen Kartenmaterial abgeleitet werden können, sind geeignete Schätzungen vorzunehmen. Die Geschwindigkeitsbegrenzung wird den Fahrsäulen über das Werkzeug *Höchstgeschwindigkeit* direkt zugewiesen, und kann durch Wahl der entsprechenden Farbpalette auch direkt im Netz-Editor optisch kontrolliert werden (Abbildung 8.6).

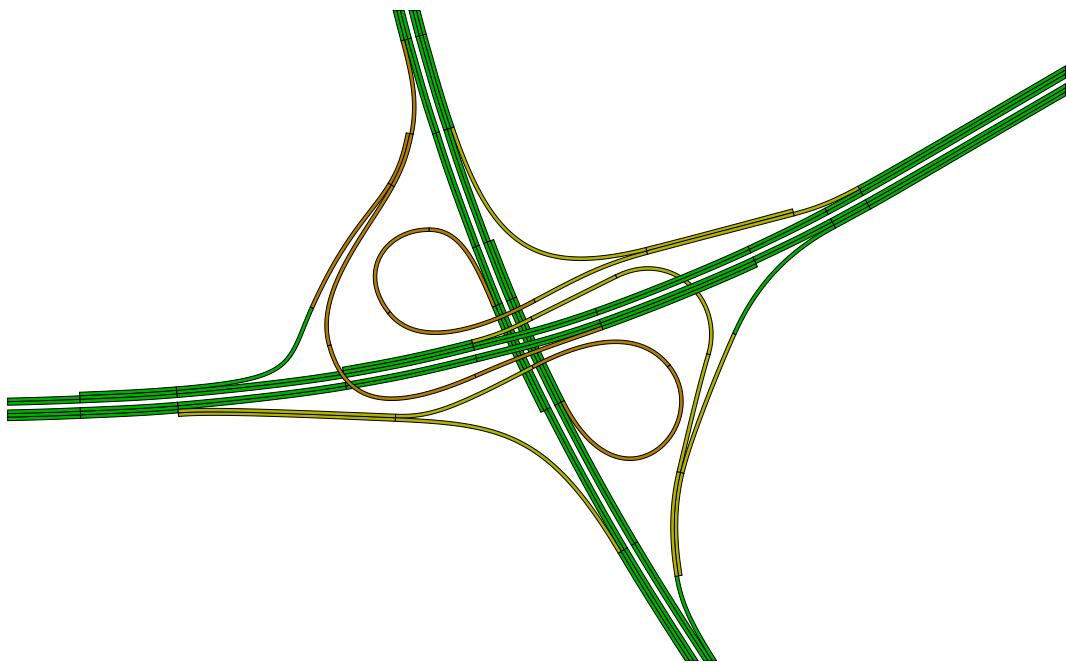


Bild 8.6: Vollständiges Modell des Autobahnkreuzes Herne

8.2.3 Verbinden der Teilnetze

Für jedes Teilnetz werden auf nach dem oben beschriebenen Verfahren Autobahnabschnitte erzeugt und miteinander verbunden. Die Erstellungszeit je Netz liegt dabei – in Abhängigkeit von der Straßendichte und -komplexität – zwischen einer halben Stunde und zwei Stunden. Hierin eingeschlossen ist der Aufwand für das Erstellen von Quellen und Senken am Ende der Fahrspuren. An denjenigen Stellen, wo zwei benachbarte Teilnetze aneinander stoßen, werden keine Quellen oder Senken vorgesehen, da dort später beide Netze über zusätzliche Fahrspuren miteinander verbunden werden. Jedes fertig gestellte Teilnetz wird schließlich in einer separaten Datei abgespeichert (Dateigröße etwa 1 MB).

Um aus den einzelnen Teilnetzen das Gesamtnetz zu erzeugen, wurde eine erweiterte Ladefunktion implementiert, die nicht – wie sonst üblich – die bestehenden Netzdaten mit dem geladenen Netz überschreibt, sondern beide logisch zu einem gemeinsamen Netz verbindet. Die Verbindungsstücke zwischen beiden Teilen müssen anschließend vom Benutzer erstellt werden, analog zum Vorgehen bei der Teilnetzerstellung.

Tabelle 8.1 liefert eine kurze Zusammenfassung der wichtigsten Netzdaten. Insgesamt umfasst das Netz 4324 Fahrspuren mit einer Gesamtlänge von fast 200 km (pro Fahrtrichtung) verbunden. Der erforderliche Hauptspeicherbedarf für das Laden des Gesamtnetzes liegt ca. 150 MB.

8.3 Festlegung der Verkehrsstärken

Bevor auf dem Straßennetz die ersten Fahrzeuge aufgesetzt werden können, müssen zunächst die Strecken generiert und die zugehörigen Verkehrsstärken festgelegt werden. Für kleine Netze ist diese Aufgabe relativ einfach, da nur eine geringe Anzahl an Strecken vorhanden ist. Da jedoch

Anzahl Fahrspuren:	4324
Gesamtlänge aller Einzelpuren:	778,6 km
Gesamtlänge aller Mehrspurbereiche:	381,5 km
Anzahl Einfahrten:	105
Anzahl Ausfahrten:	106
Anzahl Strecken:	6425
Anzahl Autobahnkreuze:	9
Anzahl Beschleunigungsstreifen:	129
Anzahl Verzögerungsstreifen:	132
Anzahl Verflechtungsstreifen:	46
Anzahl Teilnetze:	21

Tabelle 8.1: Daten des erstellten Streckennetzes

mit steigender Netzgröße auch die Anzahl der Quellen und Senken ansteigt, erhöht sich die Anzahl potentieller Strecken – und somit die Menge der zu ermittelnden Verkehrsstärken. Bei der Streckengenerierung werden nur die kürzeste Strecken berücksichtigt; die maximal mögliche Streckenanzahl entspricht somit dem Produkt aus den Anzahlen von Quellen und Senken.

Für das vorliegende Netz mit 105 Quellen und 106 Senken ergeben sich theoretisch mehr als 10000 mögliche kürzeste Strecken, für die eine Verkehrsstärke zu ermitteln ist. Da jedoch nicht jede Senke von jeder Quelle aus erreichbar ist, verringert sich die Anzahl der vorhandenen Strecken zwar auf 6425, diese Zahl ist jedoch immer noch zu hoch, um die F_{ij} -Matrix manuell mit Werten zu befüllen. Im Folgenden wird daher beschrieben, wie die Strecken effizient generiert, redundante Strecken eliminiert und Verkehrsstärken festgelegt werden.

8.3.1 Streckenerstellung

Für die automatische Generierung der Strecken wurden in der Klasse *Router* verschiedene Algorithmen zur Bestweg-Suche (d.h. zur Ermittlung der kürzeste Verbindung zwischen einem Quelle-Senke-Paar) implementiert. Ein einfach zu implementierender Algorithmus stammt von Floyd ([43]), der die kürzesten Strecken zwischen allen möglichen Fahrspur-Paaren durch vollständige Enumeration ermittelt. Ein entscheidender Nachteil des Floyd-Algorithmus, dass er zur Gruppe der $\mathcal{O}(x^3)$ -komplexen Algorithmen gehört, d. h. seine Laufzeit nimmt kubisch mit der Anzahl der betrachteten Fahrspuren zu. Während dieses Zeitverhalten für kleine Netzgrößen noch vernachlässigbar ist, steigt die Rechenzeit (und der Speicherverbrauch) für große Netze stark an. Für das betrachtete Netz mit seinen 4324 Fahrspuren ergeben sich mehr als 80 Milliarden Vergleichsoperationen, was selbst auf schnellen Prozessoren zu einer nicht praktikablen Laufzeit führt.

Der Algorithmus von Dijkstra ([34]) ermittelt für einen gegebenen Knoten eines Graphen alle kürzesten Strecken zu allen verbundenen Knoten. Dabei werden zwei Listen mit abzuarbeitenden bzw. abgearbeiteten Knoten vorgehalten. Werden diese Listen intern in Form eines *Fibonacci-Heap* oder einer vergleichbaren Datenstruktur abgelegt (in Java steht dafür die Klasse *TreeSet* zur Verfügung), kann die Laufzeit gegenüber dem Algorithmus von Floyd von $\mathcal{O}(n^3)$ auf $\mathcal{O}(n \cdot \log n + m)$ verringert werden, wobei n die Anzahl der Fahrspuren und m die Anzahl der Kanten repräsentiert.

Eine weitere Effizienzsteigerung wird durch Optimierung der vom Algorithmus verarbeiteten Datenstrukturen erreicht: Alle direkt zusammenhängenden Fahrspuren, bei denen keine Ver-

zweigungen (und somit auch keine Fahrtrichtungsentscheidungen) auftreten, werden zu einem einzelnen Hilfsobjekt zusammengefasst. Insbesondere bei mehreren direkt aufeinander folgenden Mehrspurbereichen, die keine Anschlussstellen enthalten, kann so die Anzahl zu untersuchenden Elemente entscheidend reduziert werden. Für das betrachtete Streckennetz ergibt sich eine Komplexitätsreduktion von 4324 Fahrspuren auf 724 Fahrspuren bzw. Hilfsobjekte. Bezogen auf den Floyd-Algorithmus ergibt sich so eine ca 200fache Geschwindigkeitssteigerung.

8.3.2 Streckenelimination

Um die Anzahl der zu berücksichtigenden Strecken auf eine handhabbare Größe zu verringern, wurde eine Plausibilitätskontrolle in den Router integriert, die dafür sorgt, dass Strecken automatisch aus der Streckenliste entfernt werden, wenn sie eine der folgenden Eigenschaften haben:

1. Strecken, die an einer Quelle beginnen und wieder bei einer Senke gleichen Namens enden (z. B. Fahrt von „Witten Herbede“ nach „Witten Herbede“)
2. Strecken von einer Quelle zu einer Senke, für die bereits eine kürzere Strecke mit gleichnamigen Endpunkten existiert (z.B. das Auffahren in „Witten Herbede“ in südlicher Richtung, obwohl die nördliche Richtung schneller zum Ziel führen würde)

Für die automatische Streckenelimination werden die Namen der Quellen und Senken verwendet, die der Benutzer bei der Netzerstellung vergeben hat. Gleichnamige Quellen oder Senken werden vom Eliminationsalgorithmus als Gruppe behandelt. Durch dieses Verfahren lässt sich die Anzahl der Strecken von 6425 auf 2970 verringern.

8.3.3 Setzen der Verkehrsstärken

Ausgangspunkt zur Festlegung der Verkehrsstärken sind Verkehrserhebungsdaten aus dem Jahre 2001, die im Auftrag der Bundesanstalt für Straßenwesen zusammengestellt wurden ([70]) und auf Jahresauswertungen von automatischen Dauerzählstellen beruhen. Die für den betrachteten Bereich des Autobahnnetzes relevanten Zählstellen sind Tabelle 8.2 zu entnehmen. Gewählt wurde jeweils die richtungsbezogene *Maßgebende Stündliche Verkehrsstärke* (MSV_R) der stärker belasteten Fahrspur an den betrachteten Messstellen. Die MSV_R stellt jeweils den Wert der dreiflöhöchsten Stunde dar und wurde gewählt, um eine ausreichend hohe Belastung des Streckennetzes zur Evaluation der Simulationskomponente aufzubringen.

Die betrachteten sieben Messstellen (Tabelle 8.2) reichen nicht aus, um die Verkehrsstärken der fast 3000 Strecken eindeutig festzulegen. Daher wurden nur die Verkehrsstärken der Hauptstrecken rechnerisch ermittelt; allen übrigen Fahrspuren wurde eine einheitliche Grundverkehrsstärke von 10 Fz/h aufgeprägt. Da von einigen Quellen mehr als 80 Strecken ausgehen können, wird dort eine Grundverkehrsstärke von ca. 800 Fahrzeugen erreicht, die im Bereich von Beschleunigungsstreifen bereits der Kapazitätsgrenze der Fahrspur entspricht.

Nachdem auf alle Strecken die Grundverkehrsstärke aufgebracht wurde, können die zu erwartenden Verkehrsstärken auf den einzelnen Fahrspuren ermittelt werden. Die Verkehrsstärke einer Fahrspur entspricht der Summe der fahrstreifenbezogenen Verkehrsstärken derjenigen Strecken, die über die betrachtete Fahrspur verlaufen. Für die Referenz-Messquerschnitte können die

Straße	Zählstelle	DTV	MSV_R
A 2	Oberhausen	100.125	4.850
A 40	Bochum	110.864	4.390
A 42	Wanne-Eickel	84.802	3.857
A 43	Herne	96.872	4.276
A 43	Sprockhövel	58.544	3122
A 44	Soest	53.036	2.730
A 45	Dortmund-Süd	78.472	3.510

Tabelle 8.2: Verwendete Referenzverkehrsstärken nach BASt, Heft V110 (Stand: 2001)

vorhergesagten Verkehrsstärken mit den Sollwerten nach Tabelle 8.2 verglichen werden. Die Differenz entspricht der Verkehrsstärke, die zusätzlich aufgebracht werden muss, um die Soll-Verkehrsstärke zu erreichen.

Zusätzlich zu dieser Grundverkehrsstärke wurden 14 Hauptverkehrsströme identifiziert, die dem Durchgangsverkehr entlang der einzelnen Autobahnen entsprechen. Durch Variation der Verkehrsstärken dieser 14 Ströme kann die Verkehrsstärke an den Messquerschnitten so angepasst werden, dass sie mit den real gemessenen MSV_R -Werten übereinsimmen. Das sich ergebende Gleichungssystem ist unterbestimmt, d.h. es existieren mehrere Sätze von Verkehrsstärken, welche die Randbedingungen der Messquerschnitte erfüllen. Zur Ermittlung eines geeigneten Parametersatzes wurde ein Monte-Carlo-Algorithmus implementiert, der durch zufällige Variation der Hauptverkehrsstärken die Summe der Abweichungsquadrate der Verkehrsstärken an den Messquerschnitten von ihren Sollwerten minimiert. Die so berechneten Verkehrsstärken liegen in Form einer F_{ij} -Matrix vor und können im Streckeneditor der Simulationsumgebung eingesehen und bearbeitet werden (Abbildung 8.7).

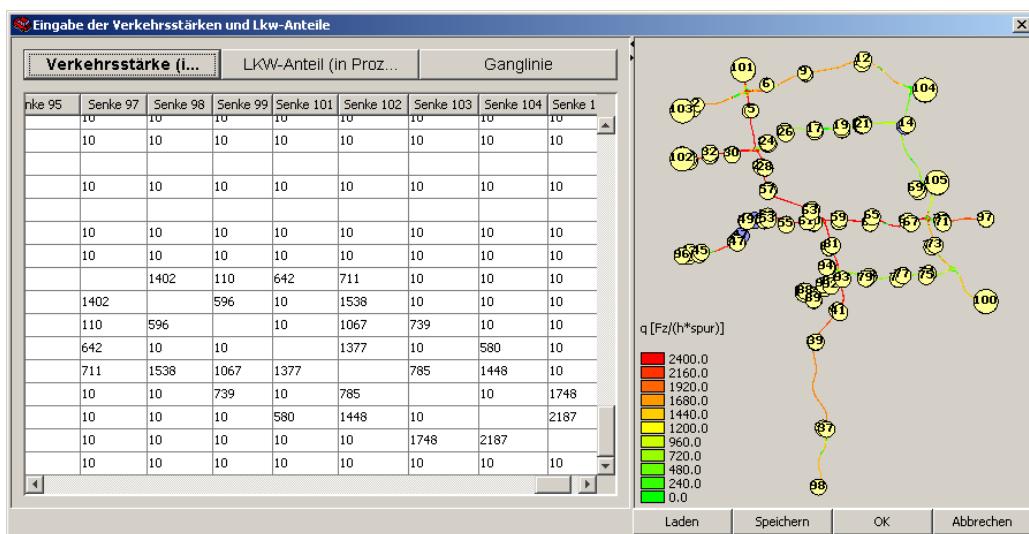


Bild 8.7: Eingabe der Verkehrsstärken für den Hauptstrecken

Das beschriebene Verfahren liefert eine qualitativ ausreichende Näherung der Verkehrsstärke für die Haupt-Transitstrecken des Netzes. Für eine darüber hinausgehende Anpassung des simulierten Verkehrsflusses an die realen Gegebenheiten ist die Implementierung alternativer Umlegungs-Verfahren und die Verwendung einer größeren Eingangsdatenbasis mit zusätzlichen Messquerschnitten erforderlich.

8.4 Einzelplatz-Simulation

8.4.1 Hardwareanforderungen

Bevor der Ablauf einer parallelisierten, also auf mehrere Rechner verteilten Simulation untersucht wird, ist es sinnvoll, zunächst das Laufzeitverhalten der Simulation auf einem einzelnen Rechner zu betrachten. Eine Parallelisierung der Simulation ist nur dann sinnvoll, wenn eine Einzelplatzberechnung aufgrund fehlender Leistungsfähigkeit des Rechners nicht möglich oder nur unter erheblichem zeitlichen Mehraufwand durchführbar ist. Die Anforderungen an das verwendete Rechnersystem sind vom Anwendungsfall abhängig: Während bei der Netzerstellung in erster Linie die Leistungsfähigkeit der CPU und der Grafikkarte eine Rolle spielt, zählen während der Simulation (bei ausgeschalteter grafischer Darstellung) vor allem die CPU-Leistung und der zur Verfügung stehende Hauptspeicher. Tabelle 8.3 zeigt die technischen Daten der für die Einzelplatzsimulation eingesetzten Rechnersysteme:

	System 1	System 2
Prozessor:	AMD Turion 64 Mobile MT-34	Intel Pentium DualCore D950
Taktfrequenz:	1.8 Ghz	3.4 Ghz
RAM:	1.0 GB	2.0 GB
Betriebssystem:	Windows XP Home	Windows 2000

Tabelle 8.3: Eingesetzte Rechnerkonfigurationen

Vor allem die Größe des Hauptspeichers hat einen sehr großen Einfluß auf die erreichbare Simulationsgeschwindigkeit: Auf Rechnersystemen, die über weniger als 2 Gigabyte Hauptspeicher verfügen, steigt die für das Laden und Routen des Gesamtnetzes benötigte Rechenzeit stark an, da der fehlende Hauptspeicher durch Festplattenzugriffe kompensiert werden muss. Sind alle Streckenoptimierungen aktiviert (siehe oben), steigt die für das Routing aller Strecken benötigte Zeit auf Rechner 2 (Tabelle 8.3) auf etwa 23 Minuten an. Auf dem mit 1 GB Hauptspeicher ausgestatteten System 1 kann das Routing nicht abgeschlossen werden, da die Rechenzeit zu stark ansteigt. Die im Folgenden beschriebenen Einzelplatz-Simulationsläufe wurden daher auf dem System mit 2 GB Hauptspeicher durchgeführt.

8.4.2 Simulationablauf

Bei der für die Simulation aufgebrachten Verkehrsstärke sind bereits nach kurzer Zeit erste Verkehrszusammenbrüche zu beobachten. Die hohe Verkehrsichte führt bereits bei kleinen Störungen des Verkehrsablaufs zu einem großen Kapazitätseinbruch der Verkehrsflusses, der sich aufgrund der konstant hohen stationären Quellströme auch nicht wieder erholen kann. Abbildung 8.8 zeigt exemplarisch einen Stau, der sich auf der A43 gebildet hat und eine Rückstau über die Auffahrt Bochum-Querenburg auf die Universitätsstraße verursacht.

Insgesamt konnten in den durchgeföhrten Simulationsläufen drei Phänomene beobachtet werden, durch die Verkehrszusammenbrüche an verschiedenen Stellen des Gesamtnetzes verursacht werden:

- **Konservatives Spurwechselverhalten:** Aus Angst, eine in mittelbarer Entfernung gelegene Ausfahrt zu verpassen, tendiert ein Großteil der ausfahrenden Fahrzeuge dazu, die

rechte Fahrspur zu bevorzugen, was dort zu einem Kapazitätseinbruch des Verkehrsstroms führt.

- **Überhöhte Verkehrsstärken auf Einfädelspuren:** Aufgrund des verwendeten Verfahrens zur Festlegung der F_{ij} -Matrix werden die Verkehrsstärken einzelner Zufahrts- oder Verflechtungsspuren stark überschätzt, so dass es zu einem Stau auf der Zufahrtsrampe kommt.
- **Ungünstige Streckenwahl:** durch die Verwendung einer Bestweg-Streckensuche werden an einigen Autobahnenkreuzen Strecken statt über die Hauptfahrspuren über die kürzeren Verteilerfahrspuren geleitet. Während die Hauptfahrbahn frei von Fahrzeugen bleibt, kommt aus auf der benachbarten Verteilerfahrbahn zum Stau.

Das Auftreten dieser Stauphänomene kann durch manuelle Anpassungen der Verkehrsstärken-Matrix bzw. des Streckenverlaufs verhindert werden. Alternativ ist die Implementierung eines alternativen Verfahrens zur Verkehrsumlegung vorzusehen, das auch eine dynamische Streckenbelegung unterstützt.



Bild 8.8: Ein Rückstau von der A43 bis zur Ruhr-Universität in Google Earth

8.4.3 Zeitverhalten

Zwischen der Berechnungszeit für einen einzelnen Zeitschritt und der Anzahl der im Netz befindlichen Fahrzeuge besteht eine enge Korrelation, da die Steuerung der Fahrzeuge – insbesondere die Ermittlung von Hindernissen – den Großteil der Rechenlast ausmacht. Für die Einzelplatzberechnung wurde daher untersucht, bis zu welcher Verkehrsichte die Simulation in akzeptabler Zeit auf einem einzelnen Prozessor berechnet werden kann. Der durch den oben beschriebenen Verkehrszusammenbruch entstehende Rückstau der Fahrzeuge ermöglicht es, den Rechenzeitverlauf bei stetig steigender Fahrzeuganzahl zu beobachten.

Diagramm 8.9 macht deutlich, dass zwischen der Fahrzeuganzahl und der benötigten Rechenzeit ein parabelförmiger Zusammenhang besteht; die Rechenzeit nimmt mit steigender Fahrzeuganzahl quadratisch zu. Bei gegebener Netzgeometrie und Verwendung des Rechnersystems

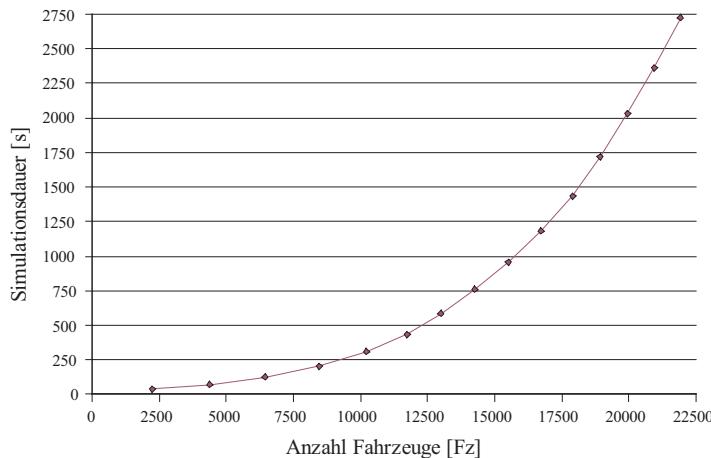


Bild 8.9: Abhängigkeit der Simulationsdauer von der Fahrzeuganzahl

2 (Tabelle 8.3) ist – unter Annahme einer Zeitschrittweite von 10 Schritten pro Sekunde – bis zu einer Fahrzeugmenge von etwa 5500 Fahrzeugen im Gesamtnetz eine Simulation in Echtzeit möglich; bei größeren Verkehrsdichten jedoch ist eine Echtzeitsimulation nur noch möglich, wenn eine Lastverteilung auf mehrere Rechner vorgenommen wird.

8.5 Parallele Simulation

Für die parallele Simulation auf mehreren Rechnern ist eine geeignete Partitionierung des Gesamtstraßennetzes in Teilnetze von vergleichbarer Größe erforderlich. Im betrachteten Anwendungsfall wurde die Partitionierung dabei so angelegt, dass die Grenzen der Teilnetze einen möglichst großen Abstand zu den neun Autobahnkreuzen einhalten, gleichzeitig aber auch keine normalen Anschlussstellen kreuzen. Zusätzlich wurden längere Straßenabschnitte ohne Autobahnkreuze (die südliche A43 und der Ruhrschnellweg) jeweils in ein eigenes Teilnetz zerlegt. Eine Partitionierung in noch kleinere Teilnetze ist nicht zielführend, da dazu Schnitte in den Einzugsbereichen der Anschlussstellen erfolgen, die das Spurwechselverhalten negativ beeinflussen.

Im Partitionierungs-Editor werden nun interaktiv per Maus die umschreibenden Polygone erzeugt, entlang derer die Fahrspuren getrennt und zu Teilnetzen zusammengefasst werden. Insgesamt entstehen so 10 Teilnetze, die jeweils an einen Rechenknoten des eingesetzten PC-Clusters gesendet werden. Zusammen mit dem für die Netzverteilung zuständigen Hauptknoten werden folglich 11 Rechner benötigt. Für jedes *NetzwerkNetz* wird ein eigener Cluster-Knoten reserviert. Sobald der Benutzer die Partitionierung und Knotenauswahl durchgeführt hat, wird der Partitionierungsvorgang gestartet.

Für die Verteilung des Gesamtnetzes an die Netzwerkknoten werden nacheinander Teilnetze erzeugt und mit den zugehörigen Fahrspuren gefüllt. An den Schnittstellen zu benachbarten Teilnetzen werden Netzwerk-Quellen bzw. -Senken hinzugefügt. Die Strecken der Teilnetze werden, basierend auf den Strecken des Gesamtnetzes neu berechnet. Das Teilnetz (samt Strecken) kann nun an den zuständigen Netzwerkknoten versendet werden. Dieser Vorgang wird so lange wiederholt, bis alle Rechenknoten ein Teilnetz erhalten haben und die Simulation gestartet werden kann.

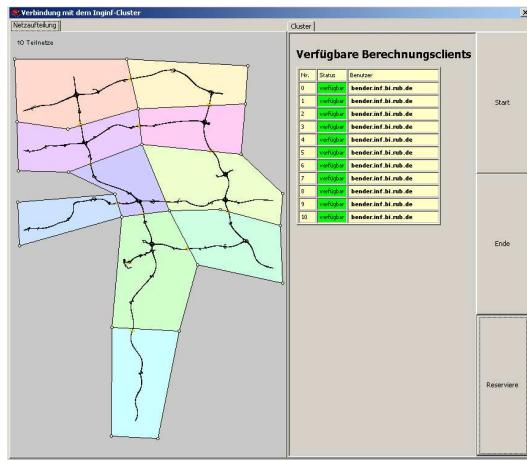


Bild 8.10: Partitionierung des Streckennetzes in 10 Teilnetze

Während dieses Partitionierungsvorgangs müssen für jedes Teilnetz Kopien der Fahrspuren und Strecken angelegt werden. Diese benötigen zusätzlichen Speicherplatz und erhöhen somit die Speicheranforderungen an die Simulation. Je nach Komplexität und Größe des Teilnetzes kann der Speicherbereich so um bis zu 50 Prozent ansteigen. Bei der Auswahl des für die Verteilung vorgesehenen Rechnersystems ist der gegenüber der Einzelplatzsimulation erhöhte Speicherbedarf zu berücksichtigen.

Sobald alle Teilnetze übertragen wurden, beginnt der eigentliche Simulationsvorgang. Alle Rechenknoten führen nun gleichzeitig ihre Berechnungen durch, koordiniert durch den Headnode, der für die Synchronisierung der Zeitschritte verantwortlich ist. Der Rechner, auf dem der Simulationsauftrag gestartet wurde, ist am eigentlichen Rechenvorgang nicht beteiligt und wartet auf die Übermittlung der Simulationsergebnisse. Maßgebend für die Simulationsgeschwindigkeit ist primär die Prozessorleistung der Rechenknoten. Der Speicherbedarf ist während des Simulationsablaufs nur noch von nachrangiger Bedeutung, da die Teilnetze deutlich weniger Speicher benötigen als das Gesamtnetz vor der Partitionierung. Gegenüber einer lokalen Berechnung auf einem Einzelplatzrechner bleibt der Speicherbedarf für die parallele Berechnung weitgehend unverändert.

Auch die Zeit für die Datenübertragung der Fahrzeuge zwischen den Teilnetzen ist gegenüber der Berechnungszeit für die einzelnen Teilschritte vernachlässigbar klein. Jedes Teilnetz des betrachteten Straßennetzes ist mit maximal drei bzw. vier benachbarten Teilnetzen verbunden (vgl. Bild 8.10). Jede Verbindung entspricht (gemäß der gewählten Partitionierung) jeweils einem *NetzwerkQuelle-NetzwerkSenke*-Paar pro Fahrtrichtung. Bei maximaler Auslastung einer Fahrspur beträgt die Zeitlücke zwischen zwei Fahrzeugen noch mindestens 15-20 Zeitschritte (bei einer Zeitschrittdauer von 0,1 s), d.h. die Anzahl der zu übertragenden Fahrzeuge zwischen den Netzen ist sehr gering. Da auch die zu übertragende Datenmenge pro Fahrzeug sehr gering ausfällt, kann die Synchronisationsphase in ausreichend kurzer Zeit durchgeführt werden.

Insgesamt ergeben sich für das betrachtete Streckennetz die in Tabelle 8.4 angegebenen Laufzeiten. Gegenüber der sequentiellen Simulationsberechnung auf einem einzelnen Rechner kann durch die Parallelisierung die durchschnittliche Simulationsgeschwindigkeit etwa um den Faktor sechs gesteigert werden. Die effektive Geschwindigkeitssteigerung ist dabei von zahlreichen Faktoren abhängig, wie der Netzgröße, der Verkehrsstärken oder der Anzahl der Verknüpfungspunkte zwischen den Teilnetzen. Eine homogene Teilnetzkomplexitätsverteilung ist hierbei von

	Einzelrechner	Rechencluster
Routing:	1:25 min	1:25 min
Optimierung:	1:10 min	1:10 min
Verteilung der Teilnetze:	—	ca. 30 s pro Teilnetz
Simulation (600 Zeitschritte):	32 s	5 s

Tabelle 8.4: Vergleich des Laufzeitverhaltens

besonderer Bedeutung, da ein einzelner, mit einem komplexen Teilnetz belasteter Rechnerknoten aufgrund der Vollsynchroneisation der Simulation alle anderen Rechenknoten ausbremsen kann. Insgesamt haben die Untersuchungen des Zeitverhaltens jedoch gezeigt, dass bei geeigneter Lastverteilung die Simulation durch den Einsatz einer parallelen Rechnerarchitektur signifikant beschleunigt werden kann.

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Die Nachbildung des Straßenverkehrs in Form einer computergestützten stochastischen Simulation erfordert, neben einer eingehenden Untersuchung der verkehrstechnischen Aspekte, vor allem einen geeigneten informatischen Lösungsansatz zur umfassenden und realitätsnahen Modellierung der Verkehrsabläufe. In dieser Arbeit wurde die Entwicklung eines Klassenmodells beschrieben, das die wesentlichen Komponenten zur Nachbildung des Verkehrsgeschehens auf Bundesautobahnen und – in Ansätzen – im innerstädtischen Bereich zur Verfügung stellt. Dabei wurden sowohl die grundlegenden theoretischen Modellierungsansätze als auch deren konkrete Umsetzung in ein praktisch einsetzbares Softwarepaket beschrieben. Entwickelt wurde ein vollständig in der Programmiersprache Java geschriebenes objektorientiertes Simulationsprogramm, das sowohl die Erstellung von Streckennetzen als auch die Durchführung von Simulationen und Auswertungen in einer einheitlichen Arbeitsumgebung erlaubt. Dabei wurde sowohl auf einen hohen praktischen Nutzwert für den Einsatz als praxisgerechtes Verkehrsplanungswerkzeug als auch auf eine gute Erweiterbarkeit im Dienste der verkehrswissenschaftliche Forschung Wert gelegt.

Als Ausgangspunkt für die Modellierung wurde eine allgemeine Beschreibung des Verkehrs begriffs und seiner Ausprägungen gegeben. Es wurden alle wesentliche am Verkehrsfluss beteiligten Komponenten identifiziert und definiert. Insbesondere wurde dabei die wesentlichen Beschreibungsformen des physikalischen Bewegungszustands der am Straßenverkehr teilnehmenden Fahrzeuge dargelegt und ihre Repräsentation im späteren Objektmodell angedeutet. Aufbauend auf diesen Grundgrößen wurde das grundsätzliche Vorgehen bei der Diskretisierung des ursprünglichen kontinuierlichen Verkehrsablaufs in eine zeitschrittisierte stochastische Simulation beschrieben, und gegenüber anderen existierenden Simulationsstrategien abgegrenzt. Der Betrachtungshorizont einer mikroskopischen Simulation wurde mit demjenigen der gröber auflösenden makroskopischen und mesoskopischen Simulation, und der noch detaillierteren, aber rechenintensiveren submikroskopischen Simulation verglichen. Die Vor- und Nachteile des mikroskopischen Ansatzes werden erläutert.

Da bereits eine breite Basis an existierenden Programmen aus dem Bereich der mikroskopischen Simulation existiert, wurde zunächst untersucht, welche Eigenschaften und Fähigkeiten die bestehenden Lösungen auszeichnen, und an welchen Stellen noch ein potentieller Verbesserungsbedarf besteht, den ein neu zu entwickelndes Programmsystem erfüllen muss. Aufbauend

darauf wurde ein allgemeines Anforderungsprofil definiert, das die konkreten Ziele für die Entwicklung der Simulationsumgebung auflistet.

Hauptarbeit bei der informatischen Umsetzung eines gegebenen Themenkomplexes ist die Überführung der realen Interaktionseinheiten in ein computergestütztes Objektmodell, das jedem real existierenden, für den Simulationsablauf relevanten Objekt eine Entsprechung in Form einer Modell-Instanz zuordnet. Grundlegendes Element der Verkehrssimulation ist das Straßennetz, das sich aus verschiedensten Ausprägungen der Klasse Fahrspur zusammensetzt, die wiederum von Instanzen der abstrakten Oberklasse Fahrzeug befahren werden. Durch geeignete Ausnutzung von Vererbungsmechanismen ist es möglich, wesentliche Modellierungsbestandteile in die Oberklassen auszulagern und nur in den spezialisierten Unterklassen weiter zu präzisieren. Es wurde beschrieben, wie einzelne Fahrer und Fahrzeuge per Assoziation zu logischen Fahrer-Fahrzeug-Einheiten verbunden werden und mit individuellen Fahrverhalten versehen werden können. Die konkrete Implementierung der unterschiedlichen Fahreigenschaften der Fahrzeugtypen Pkw und Lkw wurde ebenso beschrieben wie die assoziative Verbindung zwischen Fahrzeugen und ihren Fahrspuren. Der streng objektorientierte Ansatz erlaubt nach dem „Baukastenprinzip“ das einfache Zusammensetzen komplexer Straßennetze aus einer überschaubaren Grundmenge an Fahrspuren. Die wesentlichen Unterschiede der Fahrspur-Unterklassen wurden an Beispielen erläutert. Damit sich ein Fahrzeug zielgerichtet durch das Streckennetz bewegt, müssen ihm in geeigneter Form Informationen über seine direkte Umgebung mitgeteilt werden. Diese werden dann vom Fahrzeug-Steuerungsmodell bei Bedarf in Änderungen des Fahrzeug-Bewegungszustands umgesetzt. Zu diesem Zweck wurde ein flexibles Hindernis-Erkennungssystem entwickelt, das alle für den Fahrer wichtigen potentiellen Kollisionspartner aufzeigt.

Die Modellierung der verschiedenen Fahr- und Spurwechselverhalten beruht auf der Analyse bestehender Modelle, deren jeweilige Konzepte und Implementierungsansätze ausführlich dargestellt wurden. Gleichzeitig wurden aber auch die Unzulänglichkeiten dieser Systeme aufgezeigt, wie z. B. die fehlende Erweiterbarkeit, zu starke Abhängigkeit von strikten Fallunterscheidungen und die Nachteile prozeduraler Programmierkonzepte. Als vielseitig einsetzbare Alternative wurde daher ein neues Fahrverhaltensmodell entwickelt, das Entscheidungen über das Fahrverhalten in einem verteilten Abstimmungsprozess trifft und dadurch gleichzeitig multiple Ziele verfolgen kann. Dieses innovative, streng objektorientierte Konzept ermöglicht – vor allem im Bereich der strategischen Verhaltensplanung – die Umsetzung weitaus stärker vorausschauender Verhaltensmuster als dies bisher mit herkömmlichen Verfahren erreicht werden konnte. Das mathematische Konzept des Absichtsverhaltens und dessen Implementierung wurde detailliert hergeleitet. Ein weiterer, nicht unwesentlicher Vorteil ist die Möglichkeit der grafischen Darstellung des aktuellen Entscheidungsfindungsprozesses, wodurch insbesondere die Fehlersuche vereinfacht wird.

Zusätzlich zum verkehrstechnischen Kernmodell muss eine Software-Infrastruktur vorhanden sein, über die der Benutzer auf das Objektmodell zugreifen und es modifizieren kann. Es wurde je eine zwei- und eine dreidimensionale Visualisierungskomponente vorgestellt, die eine direkte Darstellung und Beeinflussung des internen Objektmodells auf Basis des *Model-View-Controller*-Konzepts erlauben. Zahlreiche Werkzeuge und Befehle wurden implementiert, darunter zahlreiche Funktionen zum Datenaustausch mit externen Programmen. Besonderes Augenmerk wurde dabei auf den Import von bestehenden Daten aus Bereich der Geoinformationssysteme gelegt, da diese die Konstruktion von neuen Streckennetzen wesentlich vereinfachen können. Außerdem wurde beschrieben, wie bestehende virtuelle Globusanwendungen als zu-

sätzliche Visualisierungskomponenten für die in der Verkehrssimulation erhaltenen Simulationsergebnisse verwendet werden können.

Die starke Kapselung der Simulationsfunktionalität in einzelne Klassen stellt sicher, dass das Simulationsmodell auch außerhalb des Bundesautobahn-Kontextes sinnvoll weiterverwendet werden kann. So ist es mit vergleichsweise geringem Aufwand möglich, den Einsatzbereich der Simulation auch auf innerstädtische Verkehrsanlagen wie Kreisverkehre, Lichtsignalanlagen und Fußgängerüberwege sowie ÖPNV-Fahrzeuge auszuweiten. Diese Beispiele können auch als Vorlage für weitere mögliche Zusatzfunktionen angesehen werden, von denen einige bereits skizzenhaft angedeutet wurden und noch auf eine spätere vollständige Implementierung warten.

Um einen Simulationslauf mit der Verkehrsssoftware durchführen zu können, stehen dem Benutzer prinzipiell zwei unterschiedliche Verfahren zur Auswahl: Zum einen kann die Simulation auf einem einzelnen Rechner vollständig durchgeführt werden, solange dieser die nötigen Hardwareanforderungen ausreichend erfüllt, und zum anderen kann ein verteilter Simulationslauf gestartet werden, der mehrere Instanzen der Simulationssoftware parallel zueinander auf einem Rechnerverbund ausführen kann. Die besonderen Herausforderungen, die bei der Parallelisierung einer bestehenden Simulationssoftware auftreten, wurden beschrieben. Schließlich wurde anhand eines umfangreichen Beispielnetzes die prinzipielle Leistungsfähigkeit der Software in den Bereichen Netzerstellung und Simulation unter Beweis gestellt, und sowohl im parallelen wie sequentiellen Simulationsmodus getestet. Obwohl noch umfangreicher Verbesserungsbedarf in zahlreichen Detailfunktionen besteht, ist die grundsätzliche Einsatzfähigkeit der Simulationssoftware für große und kleine Streckennetze im innerörtlichen wie autbahngestützten Verkehr gezeigt worden.

9.2 Ausblick

Die Bedeutung der Verkehrssimulation für den planerischen Bereich wird in naher Zukunft sicherlich noch stark ansteigen, ermöglicht sie doch die Untersuchung zahlreicher Phänomene, die durch herkömmliche Bemessungsverfahren nur unzureichend abgedeckt werden können. Mit den immer leistungsfähigeren Rechnersystemen und der ständigen Weiterentwicklung der Simulationsprogramme können immer größere Streckennetze auf handelsüblichen Rechnersystemen simuliert werden. Außerdem sinkt durch die zunehmende Vertrautheit des Ingenieurs mit dem Werkzeug Computer die Hemmschwelle zur Anwendung computergestützter Planungsverfahren. Insbesondere die gleichzeitige verteilte Anwendung der Simulationssoftware durch mehrere Benutzer über eine internetgestützte Datenverbindung stellt einen großen potentiellen Wachstumsmarkt für zukünftige Simulationspakete dar.

Im Laufe der nunmehr mehr als sechsjährigen Entwicklungszeit der Simulationssoftware ist ein robustes und leistungsfähiges Programmpaket entstanden, das sich bereits an einigen der kommerziell vertriebenen Simulationssysteme – die oftmals mit weitaus größerem Personalaufwand entwickelt wurden – messen lassen kann. In diesem Zeitraum wurde die erstellte Software bereits in mehreren Forschungsprojekten am Lehrstuhl für Verkehrswesen der Ruhr-Universität Bochum erfolgreich eingesetzt ([12][17]). Es konnte dabei direkt verwendet werden, ohne dass große Anpassungsgesarbeiten erforderlich gewesen wären. Einem weiteren Einsatz der Software steht, aufgrund der bisherigen guten Erfahrungen, prinzipiell nichts im Wege. Insbesondere eine mögliche Erweiterung des bisher nur für wenige Einzelfälle kalibrierten Absichtsfahrverhaltens könnte zeigen, ob das Konzept der modularen Verhaltensmodelle auch für weitere Fahrsituati-

tionen angewendet werden kann. Ebenso ist eine Übertragung des absichtsbasierten Verhaltens auf Anwendungsgebiete außerhalb der Verkehrssimulation denkbar.

In seiner jetzigen Form lässt der Netz-Editor eindeutig erkennen, dass sein ursprünglich vorgesehenes Einsatzgebiet die Simulation des Autobahnverkehrs war. Werkzeuge für die Erstellung von innerstädtischen Knotenpunkten mit Vorfahrtregelungen und komplexen Lichtsignalsteuerungen sind noch nicht oder nur in rudimentären Ansätzen vorhanden; eine komfortable Erstellung plangleicher Kreuzungen ist somit noch nicht möglich. Auch sonst besitzen die Konstruktionswerkzeuge im Netz-Editor noch starken Verbesserungsbedarf in Hinblick auf die Benutzungsergonomie. Im Rahmen weiterer Anwendertests muss ermittelt werden, inwieweit die Bedienelemente der Simulationsumgebung an die Anforderungen der Benutzer angepasst werden können und sollen. Weiterhin sollte der quelloffene Java-Ansatz weiter verfolgt werden, um auch andere Forscher und Softwareentwickler für die Weiterentwicklung der generischen Simulationsumgebung gewinnen zu können. Es sollte eine Weiterentwicklung auch dahingehend erfolgen, dass die Netzgenerierung aus den Datenbeständen von straßenspezifischer CAD-Software erfolgen kann.

Die Liste der für eine zukünftige Weiterentwicklung denkbaren neuen Einsatzgebiete ist lang. So wäre es ohne große Änderungen am grundlegenden Objektmodell möglich, die Simulation auf aktuelle Forschungsbereiche – wie Lärm- und Schadstoffemission, Maut-Systeme, Straßenabnutzung durch Schwerlastverkehr, intelligente Verkehrsleittechnik – auszuweiten.

Anhang A

Programmgesteuerte Erzeugung eines Streckennetzes

A.1 Erstellung einer Einfahrt

Neben der manuellen Erstellung von Fahrspuren und Streckennetzen über den grafischen Netzeditor kann es unter Umständen hilfreich sein, auch direkt über eine Java-Klasse die benötigten Objekte zu erzeugen und miteinander zu verbinden. Insbesondere zu Testzwecken oder für die Entwicklung eigener Importfilter kann diese Vorgehensweise genutzt werden. Im Folgenden soll als Beispielnetz eine einfache Auffahrtrampe vom Standardtyp E 1 (nach HBS 2001 [57]) erzeugt und in das Simulationssystem eingefügt werden.

Die Auffahrt soll aus einer zweistreifigen geraden Hauptfahrspur und einer einspurigen Zufahrtsrampe, bestehend aus einer Geraden und einer Kurve, zusammengesetzt sein. Die Länge der Einfädelspur betrage 200 Meter, vor und nach dem dieser Spur schließen sich Geradensegmente mit einer Länge von jeweils 100 Metern an. An die freien Enden des Netzes schließen sich zwei Quellen und eine Senke an. Abbildung A.1 illustriert die Netzgeometrie.



Bild A.1: Aufsicht der zu erstellenden Einfahrt

Für die Umsetzung müssen zunächst einzelne Fahrspuren erzeugt, miteinander verbunden und zu einem neuen Streckennetz hinzugefügt werden. Weiterhin sollen für die beiden sich ergebenden Strecke Verkehrsstärken vorgegeben werden. Damit das erzeugte Netz direkt im Netz-Editor getestet werden kann, wird ein vollständiges Beispielprogramm angegeben, das eine neue Instanz der Bedienoberfläche erzeugt und die Auffahrt dort anzeigt.

Naturgemäß kann das hier beschriebene kurze Beispiel nur eine grobe Übersicht über die Vorgehensweise bei der programmierten Erstellung eines eigenen Streckennetzes geben. Es soll aber verdeutlicht werden, wie einfach die Manipulation der Straßenelemente innerhalb des objektorientierten Klassenmodells sein kann. Für eine vollständige Übersicht über die Simulations-API wird auf das dem Programm beiliegende JavaDoc-Verzeichnis verwiesen.

A.2 Erzeugung der Fahrspuren

Vor der eigentlichen Geometrieerzeugung müssen zunächst alle im Weiteren benötigten Klassen bzw. ihre Pakete importiert werden. Die Hauptklassen, insbesondere der NetzGenerator, befinden sich im Paket simulation, alle Fahrspur- und Strecken-bezogenen Klassen im Paket simulation.netz. Für die Erstellung von Punkten wird das Paket simulation.gui benötigt. In simulation.werkzeuge und simulation.tools befinden sich weitere Hilfsklassen für die Erstellung von Fahrspuren und Strecken.

```
import simulation.*;
import simulation.netz.*;
import simulation.gui.*;
import simulation.werkzeuge.*;
import simulation.tools.*;
```

Um ein vollständig lauffähiges Programm zu erhalten, wird eine neue Klasse EinfahrtBeispiel definiert, die eine ausführbare main-Methode enthält:

```
public class EinfahrtBeispiel
{
    public static void main( String[] args )
    {
```

Wird diese Methode aufgerufen, soll zunächst eine vollständige neue Instanz der eigentlichen Simulationsumgebung erzeugt und gestartet werden. Die Hauptklasse der Bedienungsoberfläche heißt NetzGenerator und kann über eine eigene Factory-Methode instanziert werden. Es kann stets nur eine einzelne Instanz von NetzGenerator ausgeführt werden (vgl. *Singleton- und Factory Method-Pattern*, [46]).

```
NetzGenerator ng = NetzGenerator.erzeugeInstanz();
```

Nach dem Aufruf der Factory-Methode steht eine vollständige Bedienoberfläche zur Verfügung, die auch direkt angezeigt wird. Über die Variable ng kann im Folgenden bei Bedarf direkt auf den NetzGenerator und sein Simulationsmodell zugegriffen werden.

Bei seiner Erzeugung besitzt ein NetzGenerator bereits ein leeres Netz-Objekt, das noch keine Fahrspur-Elemente enthält. Dieses Netz kann entweder weiterverwendet oder durch ein neues Netz überschrieben werden. Hier wird der letztere Weg gewählt, um auch später bestehende Streckennetze überschreiben zu können. Die Erzeugung eines Netzes erfolgt über den Standardkonstruktor:

```
Netz n = new Netz();
```

In das Netz sollen nun nach und nach die benötigten Fahrspuren eingefügt werden. Wir beginnen mit den drei linken Fahrspur-Segmenten der Hauptfahrbahn. Der Einfachheit halber definieren wir, dass alle drei Fahrspuren jeweils in östlicher Richtung verlaufen und ihr Ursprung im Koordinaten-Nullpunkt liegt. Zunächst werden daher vier Punkte definiert, die später die Endpunkte der drei Geraden darstellen:

```
Punkt p1 = new Punkt(0,0);
Punkt p2 = new Punkt(100,0);
Punkt p3 = new Punkt(300,0);
```

```
Punkt p4 = new Punkt(400,0);
```

Zwischen den Punkten können dann die Geraden eingefügt werden. Bei der Erzeugung muss die korrekte Reihenfolge der Punkte (*Startpunkt*, *Zielpunkt*) beachtet werden, um die richtige Fahrtrichtung auf der Fahrspur zu gewährleisten.

```
Gerade g1 = new Gerade(p1,p2);
Gerade g2 = new Gerade(p2,p3);
Gerade g3 = new Gerade(p3,p4);
```

Obwohl die beiden Punkte p2 und p3 geometrische Berührungspunkte zwischen den Geraden darstellen, herrscht noch keine logische Verbindung zwischen den Fahrspuren. Diese muss explizit über den Aufruf einer statischen Methode der Klasse Fahrspur hergestellt werden:

```
Fahrspur.verbinde(g1,g2);
Fahrspur.verbinde(g2,g3);
```

Nun sind alle drei Geraden miteinander verbunden, und könnten von einem manuell aufgesetzten Fahrzeug bereits befahren werden. Leider bilden die Spuren bisher nur eine einspurige Straße, auf der kein Spurwechsel möglich ist. Um die Straße auf eine zweispurige Strecke zu erweitern, müssten – ebenso wie zuvor – drei weitere Geraden erzeugt werden, die jeweils um eine Straßenbreite südlicher verlaufen. Um sie mit ihren Nachbarn zu verbinden, müssten zusätzlich drei MehrspurBereiche definiert und mit den Fahrspuren gefüllt werden.

Glücklicherweise existieren bereits Werkzeuge, die dem Programmierer diese umständlichen Arbeiten zumindest teilweise abnehmen. So kann das Werkzeug „MehrspurErweitern“ verwendet werden, um neben einer Fahrspur eine benachbarte zweite Fahrspur hinzuzufügen. Alle lateralen Assoziationen und MehrspurBereiche werden automatisch miterzeugt. Mit den folgenden Anweisungen wird aus der einspurigen eine zweispurige Straße:

```
WerkzeugMehrspurErweitern we = new WerkzeugMehrspurErweitern();
we.fahrspurHinzufuegen(g1, false);
we.fahrspurHinzufuegen(g2, false);
we.fahrspurHinzufuegen(g3, false);
```

Der boolean-Parameter gibt dabei an, ob eine Fahrspur rechts (=false) oder links (=true) von der gegebenen Fahrspur erzeugt werden soll. Auf die gleiche Weise kann nun auch die eigentliche Einfahrt, also der 200 Meter lange dreispurige Bereich, erzeugt werden. Dazu wird einfach eine neue Fahrspur rechts neben der soeben erzeugten rechten Fahrspur von g2 erzeugt:

```
we.fahrspurHinzufuegen(g2.rechteSpur, false);
```

Damit existieren nun alle Fahrspuren der Hauptfahrbahn, und es kann mit der Erzeugung der Auffahrtrampe begonnen werden. Vereinfachend wird die Rampe zunächst aus zwei Geraden gebildet, die an den Startpunkt der soeben erzeugten Auffahrt-Fahrspur (die Spur, die zwei Spuren rechts von g2 liegt) angeschlossen werden:

```
Gerade auffahrt = (Gerade)g2.rechteSpur.rechteSpur;
Gerade g5 = new Gerade(0,40,50,20);
Gerade g6 = new Gerade(g5.p2,auffahrt.p1);
Fahrspur.verbinde(g5,g6);
Fahrspur.verbinde(g6,auffahrt);
```

Damit ist die Erstellung der Basis-Fahrspurgeometrie abgeschlossen. Um die Fahrspuren allerdings innerhalb des Streckennetzes anzeigen und simulieren zu können, müssen sie noch zum Streckennetz `n` hinzugefügt werden:

```
n . neuesZeichenobjekt (g1) ;
n . neuesZeichenobjekt (g2) ;
n . neuesZeichenobjekt (g3) ;
n . neuesZeichenobjekt (g1 . rechteSpur) ;
n . neuesZeichenobjekt (g2 . rechteSpur) ;
n . neuesZeichenobjekt (g3 . rechteSpur) ;
n . neuesZeichenobjekt (g2 . rechteSpur . rechteSpur) ;
n . neuesZeichenobjekt (g5) ;
n . neuesZeichenobjekt (g6) ;
```

Gerade `g6` stellt eigentlich die Verbindungsklothoide zwischen Auffahrtrampe `g5` und dem Einfädelbereich dar. Da für die manuelle Erzeugung einer Kurve allerdings die Koordinaten der Beziér-Stützpunkte bekannt sein müssen, wurde hier stattdessen eine Gerade verwendet. Diese kann nun nachträglich in eine Kurve umgewandelt werden, wobei die Krümmung der Kurve automatisch an die von den Nachbarfahrspuren geforderten Richtung angepasst wird:

```
g6 . abrunden () ;
```

A.3 Quellen und Senken

Die Einfahrt ist im Prinzip fertiggestellt und einsatzbereit; sie kann aber erst dann tatsächlich von Fahrzeugen befahren werden, wenn an ihren Enden Quelle und Senke zur Erzeugung bzw. Entfernung der Fahrzeuge eingefügt werden. Zunächst wird die Quelle der Hauptfahrspur erzeugt und mit den ersten beiden Fahrspuren verbunden.

```
Quelle q1 = new Quelle () ;
q1 . setzeNachfolger (g1) ;
q1 . setzeNachfolger (g1 . rechteSpur) ;
```

Auf die gleiche Weise werden auch die Quelle der Auffahrtrampe und die einzelne Senke am Ende der Hauptfahrspur erzeugt und verknüpft.

```
Quelle q2 = new Quelle () ;
q2 . setzeNachfolger (g5) ;

Senke s = new Senke () ;
Fahrspur . verbinde (g3 , s) ;
Fahrspur . verbinde (g3 . rechteSpur , s) ;
```

Auch Quellen und Senken stellen grafische Objekte dar, die in der Visualisierung des Netzes auftauchen müssen. Daher werden sie, wie zuvor die Fahrspuren, zum Streckennetz hinzugefügt:

```
n . neuesZeichenobjekt (q1) ;
n . neuesZeichenobjekt (q2) ;
n . neuesZeichenobjekt (s) ;
```

A.4 Verkehrsstärken

Bei der Erzeugung eines Streckennetzes werden beim ersten Start der Simulation vom Router automatisch Strecken erzeugt und mit einer vorgegebenen Verkehrsstärke belegt. Meist ist es sinnvoll, diese Vorgabewerte mit eigenen Verkehrsstärken zu überschreiben.

Zunächst wird das zuvor erzeugte Streckennetz dem NetzGenerator übergeben, der es direkt in seine grafische Darstellung und den Simulationsthread übernimmt:

```
ng.bild.netz(n);
```

Nun kann die eigentliche Erzeugung der Strecken angestoßen werden. Zuständig hierfür ist die Klasse Router, die als separater Thread im Hintergrund alle möglichen Strecken durch das Streckennetz bestimmt. Um sicherzustellen, dass alle Strecken gefunden wurden, wird eine Synchronisations-Methode aufgerufen, die erst nach Beendigung des Routing-Algorithmus wieder verlassen wird.

```
Router.berechneStrecken(n);
Router.warteAufErgebnis();
```

Nun liegen alle Strecken innerhalb des Netzes vor und können vom Programmierer direkt verändert werden. Im vorliegenden Fall existieren nur zwei Strecken, auf die über die jeweiligen Quellen zugegriffen werden kann:

```
Strecke s1 = (Strecke)q1.strecken.elementAt(0);
Strecke s2 = (Strecke)q2.strecken.elementAt(0);
```

Jede Strecke zugeordnet ist eine Instanz der Klasse Verkehrsstärke, die Informationen über die Spitzenverkehrsstärke, die Ganglinie und den Lkw-Anteil festlegt. Im vorliegenden Beispiel wollen wir eine Verkehrsstärke von 2200 Fz/h auf der Hauptfahrspur und 550 Fz/h auf der Auffahrt verwenden. Außerdem soll die Auffahrt einen Lkw-Anteil von 20 Prozent aufweisen:

```
s1.verkehrsstaerke.q(2200);
s2.verkehrsstaerke.q(550);
s2.verkehrsstaerke.lkwAnteil(.2);
```

Schließlich muss nur noch das Bild so verschoben werden, dass das gesamte Netz korrekt angezeigt wird. Nun kann die Simulation gestartet werden:

```
ng.bild.zoomGrenzen();
Optionen.angehalten=false;
}
```


Literaturverzeichnis

- [1] ISO 14825:2004. Intelligent transport systems - Geographic Data Files (GDF) - Overall data specification. *Internationale Organisation für Normung, Genf.*, Oktober 2004.
- [2] acatech. Mobilität 2020. Perspektiven für den Verkehr von Morgen. *acatech - Konvent für Technikwissenschaften der Union der deutschen Akademien der Wissenschaften e.V.*, 2006.
- [3] N. Ashford, M. O'Leary, and P. McGinity. Stochastic modelling of passenger and baggage flows through an airport terminal. *Traffic Engineering and Control*, 17, 1976.
- [4] M. Bando, H. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical Review*, E51, 1995.
- [5] M. Bando, K. Hasebe, K. Nakanishi, and A. Nakayama. Analysis of optimal velocity model with explicit delay. *Physical Review E*, 58, 1998.
- [6] J. Barceló. AIMSUN Microscopic Traffic Simulator: A Tool for the Analysis and Assessment of ITS Systems. *HCC Simulation Meeting*, 2001.
- [7] J. Barceló and J. Casas. Dynamic Network Simulation with AIMSUN. *Proceedings of the International Symposium on Traffic Simulation, Yokohama, Kluwer*, 2003.
- [8] J. Barceló, J. Casas, JL Ferrer, and D. García. Modeling Advanced Transport Telematic Applications with Microscopic Simulators: The case of AIMSUN. *Traffic and Mobility, Simulation*, 1999.
- [9] M. Brackstone and M. McDonald. Car-following: a historical review. *Transportation Research* F, 2, 1999.
- [10] A. Breßler. Verkehrssicherheit und Verkehrsablauf an Steigungsstrecken - Kriterien für Zusatzfahrstreifen. *Dissertationsschrift*, Ruhr-Universität Bochum, 2001.
- [11] W. Brilon. *Vorlesungsunterlagen Verkehrsplanung I*. Lehrstuhl für Verkehrswesen, Ruhr-Universität Bochum, 1999.
- [12] W. Brilon and C. Betz. Entwurf und Bemessung von Autobahnknotenpunkten unter Berücksichtigung der Wechselwirkung zwischen den Elementen. *Entwurf des Schlussberichts FE 02.219/2002/GGB*, 2006.
- [13] W. Brilon and U. Brannolte. Simulationsmodell für den Verkehrsablauf auf zweispurigen Straßen mit Gegenverkehr. *Forschung Straßenbau und Straßenverkehrstechnik*, Heft 239, 1977.
- [14] W. Brilon and A. Bressler. Traffic Flow on Freeway Upgrades. *Transportation research record*, 2004.
- [15] W. Brilon and H. Bäumer. Überprüfung von kreisverkehren mit zweistreifig markierter oder einstreifig markierter, aber zweistreifig befahrbarer kreisfahrbahn. *Schriftenreihe Forschung Straßenbau und Straßenverkehrstechnik*, 876, 2004.

- [16] W. Brilon and H. Bäumer. Verkehrsablauf, Kapazität und Verkehrssicherheit an Kreisverkehren in Hessen. *Schlussbericht zum Forschungsauftrag des Hessischen Landesamtes für Straßen- und Verkehrswesen. Lehrstuhl für Verkehrswesen, Ruhr-Universität Bochum*, 2004.
- [17] W. Brilon and J. Geistefeldt. Autobahnen und Autobahnknotenpunkte mit vierstreifigen Richtungsfahrbahnen - Gestaltung und Bemessung. *Schlussbericht FE 02.249/2004/FGB*, Lehrstuhl für Verkehrswesen, Ruhr-Universität Bochum.
- [18] W. Brilon, D. Hartmann, D. Harding, K. Erleemann, and S. Seifarth. Fortentwicklung und Bereitstellung eines bundeseinheitlichen Simulationsmodells für Bundesautobahnen. *Forschung Straßenbau und Straßenverkehrstechnik*, Heft 918, 2005.
- [19] W. Brilon, D. Hartmann, J. Harding, and K. Erleemann. Erweiterung des Softwareprogramms BABSIM um ein Verhaltensmodell zur Abbildung der in den RAA dargestellten Typen von Ein- und Ausfahrten. *Nachtrag zum Forschungsantrag FE 01/157/2001/IRB der Bundesanstalt für Straßenwesen*, Dezember 2006.
- [20] W. Brilon, A. Weinert, and N. Wu. Verkehrstechnische Untersuchung der Verkehrsbeeinflussungsanlage A 44. Schlussbericht zum Forschungsprojekt, Landschaftsverband Westfalen-Lippe, 2000.
- [21] Bundesministerium für Verkehr, Bau- und Wohnungswesen. Straßenbaubericht 2004. *Bundesministerium für Verkehr, Bau- und Wohnungswesen*, 2004.
- [22] S. Burbeck. Applications Programming in Smalltalk-80: How to use Model-View-Controller. 1992.
- [23] W. Burghout. *Hybrid microscopic-mesoscopic traffic simulation*. Royal Institute of Technology, Stockholm, 2004.
- [24] G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. *Proceedings of Supercomputing Symposium*, 94, 1994.
- [25] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295, 2001.
- [26] G. Cameron, B.J.N. Wylie, and D. McArthur. Paramics: moving vehicles on the connection machine. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*. ACM Press, New York, 1994.
- [27] B. Carpenter, V. Getov, G. Judd, T. Skjellum, and G. Fox. MPJ: MPI-like Message Passing for Java. *Concurrency: Practice and Experience*, Volume 12, Number 11, September 2000.
- [28] N. Cetin and K. Nagel. A large-scale agent-based traffic microsimulation based on queue model. *Swiss Transport Research Conference (STRC)*, 2003.
- [29] D. Charypar, K. Axhausen, and K. Nagel. An Event-Driven Queue-Based queue-based traffic flow microsimulation. *86th Annual Meeting of the Transportation Research Board*, 2007.
- [30] H.J. Cho and Y.T. Wu. Modeling and simulation of motorcycle traffic flow. In *IEEE International Conference on Systems, Man and Cybernetics*, 2004.
- [31] H. Claussen, W. Lichtner, L. Heres, P. Lahaije, and J. Siebold. GDF - A proposed standard for digital road maps to be used in carnavigation systems. *Vehicle Navigation and Information Systems Conference*, 1989.
- [32] R. Dawson. The hypererlang probability distribution - a generalized traffic headway model. *Schriftenreihe Straßenbau und Straßenverkehrstechnik*, 86, 1969.

- [33] J. de la Beaujardiere. OpenGIS Web Map Server Implementation Specification 1.3.0. *Open Geospatial Consortium Inc.*, März 2006.
- [34] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1959.
- [35] Java 2 Standard Edition. API Specification. <http://java.sun.com>, Version 1.5.0, 2004.
- [36] Emmerich and Courbey. Visualization of traffic simulation in urban planning. *Technische Universität Helsinki*, 2004.
- [37] K. Erlemann. Eine grafische Programmier-Umgebung zur Verhaltensmodellierung in der Verkehrssimulation. In *Forum Bauinformatik 2005*, volume Progress in Bauinformatik. Brandenburg University of Technology at Cottbus, 2005.
- [38] M. Fellendorf. VISSIM: A microscopic Simulation Tool to Evaluate Actuated Signal Control including Bus Priority. In *64th ITE Annual Meeting*, Dallas, Texas, 1994.
- [39] M. Fellendorf and P. Vortisch. Integrated Modeling of Transport Demand, Route Choice, Traffic Flow and Traffic Emissions. *Preprint CD-ROM of the 79 thAnnual Meeting of the Transportation Research Board*, 2000.
- [40] M. Fellendorf and P. Vortisch. Validation of the Microscopic Traffic Flow Model VISSIM in Different Real-World Situations. *National Research Council. 80th Meeting of the Transportation Research Board*, 2001.
- [41] J. Ferraiolo, F. Jun, and D. Jackson. Scalable Vector Graphics (SVG) 1.1 Specification. *The World Wide Web Consortium (W3C)*, 2003.
- [42] FGSV. Merkblatt für die Anlage von kleinen Kreisverkehren. *Forschungsgesellschaft für Straßen- und Verkehrswesen*, 1998.
- [43] R. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5, 1962.
- [44] B. Friedrich. Adaptive Signal Control BALANCE Traffic and Control Models. *8th International Conference of the European Working Group on Transportation*, 2000.
- [45] H. Fritzsche. A model for traffic simulation. *Traffic Engineering and Control*, Mai 1994.
- [46] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1995.
- [47] D.C. Gazis, R. Herman, and R.B. Potts. Car-Following Theory of Steady-State Traffic Flow. *Operations Research*, 7(4), 1959.
- [48] D.C. Gazis, R. Herman, and R.W. Rothery. Non-linear follow-the-leader models of traffic flow. *Operations Research* 9, No. 4, 1961.
- [49] P. Gipps. Behavioral Car-Following Model for Computer Simulation. *Transport Research*, 15(2), 1981.
- [50] P. Gipps. A model for the structure of lane-changing decisions. *Transport Research*, 20(5), 1986.
- [51] PG Gipps and B. Marksjö. Micro-simulation model for pedestrian flows. *Mathematics and Computers in Simulation*, 27, 1985.
- [52] R. Gordon and S. Talley. *Essential JMF: Java Media Framework*. Prentice Hall, 1999.

- [53] R. Graham, G. Shipman, B. Barrett, R. Castain, G. Bosilca, and A. Lumsdaine. Open MPI: A High-Performance, Heterogeneous MPI. In *Fifth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, Barcelona, 2006.
- [54] M. Grossmann. Methoden zur Berechnung und Beurteilung von Leistungsfähigkeit und Verkehrsqualität an Knotenpunkten ohne Lichtsignalanlagen. *Schriftenreihe des Lehrstuhls für Verkehrswesen*, 9, 1991.
- [55] Object Management Group. Unified Modeling Language 2.0 Superstructure. *Specification*, 2004.
- [56] J. Hancock. A distributed-reasoning voting architecture. In *Game Programming Gems 4*. Charles River Media, 2004.
- [57] HBS. *Handbuch für die Bemessung von Straßenverkehrsanlagen*. Forschungsgesellschaft für Straßen- und Verkehrswesen, 2001.
- [58] D. Helbing. *Verkehrsphysik - Neue physikalische Modellierungskonzepte*. Springer Verlag, 1997.
- [59] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51, 1995.
- [60] L. Henderson. On the fluid mechanics of human crowd motion. *Transportation Research*, 8, 1974.
- [61] D.H. Hoefs. Untersuchung des Fahrverhaltens in Fahrzeugkolonnen. *Forschung Straßenbau und Straßenverkehrstechnik*, Heft 8, 1972.
- [62] P. Hunt. *SCOOT-a traffic responsive method of coordinating signals*. Transport and Road Research Laboratory, 1981.
- [63] Google Inc. KML Specification 2.1. <http://earth.google.com/kml>, 2006.
- [64] ISO 19136. Geographic information - geography markup language (gml). *ISO/TC 211/WG 4*, 2003.
- [65] J. Janson and A. Tapani. Comparison of car-following models. *Swedish National Road and Transport Research Institute*, 2003.
- [66] Java3D. API Specification. <https://java3d.dev.java.net/>, Version 1.4.0, März 2006.
- [67] R.D. Kühne and M. Rödiger. Macroscopic simulation model for freeway traffic with jams and stop-start waves. *Proceedings of the 23rd conference on Winter simulation*, 1991.
- [68] C. Kinkeldey and M. Rose. Fußgängersimulation auf der Basis sechseckiger zellulärer Automaten. *Forum Bauinformatik 2003 - Junge Wissenschaftler forschen*, 2003.
- [69] I. Kosonen. Hutsim - urban traffic simulation and control model: Principles and applications. *Dissertationsschrift*, Helsinki University of Technology, 1999.
- [70] S. Laffont, G. Nierhoff, G. Schmidt, and T. Kathmann. Verkehrsentwicklung auf Bundesfernstraßen 2001 - Jahresauswertung der automatischen Dauerzählstellen. *Berichte der Bundesanstalt für Straßenwesen*, Heft V 110, Januar 2004.
- [71] W. Leutzbach. Zeitlückenverteilungen. *Straßenverkehrstechnik*, 1, 1957.
- [72] W. Leutzbach and F. Busch. Spurwechselvorgänge auf dreispurigen BAB-Richtungsfahrbahnen. *Institut für Verkehrswesen, Universität Karlsruhe*, 1984.

- [73] W. Leutzbach, W. Wiedemann, and H. Hubschneider. Simulation des Verkehrsablaufs auf Autobahnen mit zweispurigen Richtungsfahrbahnen im Hinblick auf empirisch nicht ausreichend verifizierbare Situationen. *Forschungsauftrag der Bundesanstalt für Straßenwesen, Institut für Verkehrswesen, Universität Karlsruhe*, 1977.
- [74] M.J. Lighthill and G.B. Whitham. On Kinematic Waves. I: Flood Movement in Long Rivers. *Proceedings of the Royal Society of London*, 229, 1955.
- [75] M.J. Lighthill and G.B. Whitham. On Kinematic Waves. II: A Theory of Traffic Flow on Long Crowded Roads. *Proceedings of the Royal Society of London*, 229, 1955.
- [76] J. Ludmann. Beeinflussung des Verkehrsablauf auf Straßen, Analyse mit PELOPS. *Dissertationsschrift*, RWTH Aachen, 1998.
- [77] F. Mazur, D. Weber, R. Chrobok, S.F. Hafstein, A. Pottmeier, and M. Schreckenberg. Basics of the online traffic information system autobahn.nrw. In *Hannovermesse 2005*, Hannover, April 2005.
- [78] R. Michaels. Perceptual factors in car following. *Proceedings of the 2nd International Symposium on the Theory of Road Traffic Flow*, 1963.
- [79] Sun Microsystems. The Java3D Tutorial. Version 1.6, 2001.
- [80] J. Moore. JView: an information visualization paradigm. 2002.
- [81] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I France*, 2, 1992.
- [82] D. Oeding. Verkehrsbelastung und Dimensionierung von Gehwegen und anderen Anlagen des Fußgängerverkehrs. *Straßenbau und Straßenverkehrstechnik*, 22, 1963.
- [83] T.G. Oketch. New Modeling Approach for Mixed-Traffic Streams with Nonmotorized Vehicles. *Transportation Research Record*, 2000.
- [84] A. Okumura and S. Tadaki. Asymmetric Optimal Velocity Model. *Journal of the Physical Society of Japan*, 72, 2003.
- [85] LE Owen, Y. Zhang, L. Rao, and G. McHale. Traffic flow simulation using CORSIM. In *Proceedings of the Winter Simulation Conference*, Orlando, Florida, 2000.
- [86] H.J. Payne. Models of freeway traffic and control. *Mathematical Models of Public Systems*, 1, 1971.
- [87] L.A. Pipes. An operation analysis of traffic dynamics. *Journal of Applied Physics*, 24, 1953.
- [88] A. Pottmeier, R. Chrobok, S.F. Hafstein, F. Mazur, and M. Schreckenberg. Olsim: Up-to-date traffic information on the web. In *Proceedings of the Third IASTED International Conference Communication, Internet, and Information Technology.*, St. Thomas, US Virgin Islands, November 2004.
- [89] I. Prigogine and R. Herman. *Kinetic theory of vehicular traffic*. American Elsevier, 1971.
- [90] Quadstone. Quadstone paramics v5.0 technical notes. *Quadstone Limited*, Oktober 2004.
- [91] RAS-Q. *Richtlinien für die Anlage von Straßen - Querschnitt*. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), Köln, 1996.
- [92] T. Reenskaug. Models-Views-Controllers. *Xerox PARC*, 1979.
- [93] A. Rekersbrink. Verkehrsflusssimulation mit Hilfe der Fuzzy-Logic und einem Konzept potentieller Kollisionszeiten. *Dissertationsschrift*, Universität Karlsruhe, 1994.

- [94] A. Reuschel. Fahrzeugbewegung in der Kolonne bei gleichförmig beschleunigtem oder verzögertem Leitfahrzeug. *Zeitschrift des österreichischen Ingenieur und Architektenvereins*, 7(8), 1950.
- [95] C. Reynolds. Flocks, herds and schools: A distributed behavioral model. *14th Annual Conference on Computer Graphics and Interactive Techniques*, 1987.
- [96] C. Reynolds. Steering behaviors for autonomous characters. *Game Developers Conference*, 1999.
- [97] RiLSA. *Richtlinien für Lichtsignalanlagen (RiLSA)*. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), Köln, 1992, Teilstrechreibung 2003.
- [98] M. Rose. Modellbildung und Simulation von Autobahnverkehr. *Dissertationsschrift*, Universität Hannover, 2003.
- [99] J. Rosenblatt. DAMN: A Distributed Architecture for Mobile Navigation. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2), 1997.
- [100] K. Schloegel, G. Karypis, and V. Kumar. Graph partitioning for high performance scientific simulations. In *CRPC Parallel Computing Handbook*. Morgan Kaufmann Pub., 2000.
- [101] M. Schreckenberg and S. Sharma. *Pedestrian and Evacuation Dynamics*. Springer Verlag, 2002.
- [102] W.J. Schroeder, L.S. Avila, and W. Hoffman. Visualizing with VTK: A Tutorial. 2000.
- [103] A. Schuhl. The Probability Theory Applied to the Distribution of Vehicles on Two-Lane Highways. *Poisson and Traffic*, 1955.
- [104] A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes. The fundamental diagram of pedestrian movement revisited. *Journal of Statistical Mechanics*, 2005.
- [105] U. Sparmann. Spurwechselvorgänge auf zweispurigen BAB-Richtungsfahrbahnen. *Straßenbau und Straßenverkehrstechnik*, 263, 1978.
- [106] StVO. Straßenverkehrs-Ordnung. *Bundesministerium für Verkehr, Bau und Stadtentwicklung*, Mai 2006.
- [107] A. Szofran. Global terrain technology for flight simulation. *Game Developers Conference*, 2006.
- [108] C. Theis. Modellierung des Fahrverhaltens an Autobahnanschlussstellen. *Dissertationsschrift*, Universität Karlsruhe, 1997.
- [109] P. Thompson and E. Marchant. A Computer Model for the Evacuation of Large Building Populations. *Fire Safety Journal*, 24, 1995.
- [110] E.P. Todosiev. The action point model of the driver-vehicle system. *Dissertationsschrift*, Ohio State University, 1963.
- [111] USGS. GTOPO 30 Höhenmodell. *United States Geological Survey, EROS Data Center*, 1996.
- [112] L. Valcic, J. Bailey, and J. Dehn. Visualizing Scientific Data Using Keyhole Markup Language (KML). *American Geophysical Union, Fall Meeting*, Dezember 2006.
- [113] M. Van Aerde. Single regime speed-flow-density relationship for congested and uncongested highways. *Transportation Research Board 74th Annual Meeting, Washington, DC*, 1995.
- [114] M. Van Aerde and H. Rakha. Multivariate calibration of single regime speed-flow-density relationships. In *Proceedings of the Vehicle Navigation and Information Systems (VNIS) Conference*, Seattle, Washington, 1995.

- [115] R. van Essen and V. Hiestermann. X-GDF - The ISO Model of Geographic Information for ITS. *ISPRS Workshop on Service and Application of Spatial Data Infrastructure, Hangzhou, China*, 2005.
- [116] VDI. *VDI-Richtlinie 3633: Simulation von Logistik-, Materialfluss- und Produktionssystemen: Begriffsdefinitionen*. Beuth Verlag, Berlin, 1996.
- [117] K.L. Verdin and S. Jenson. Development of continental scale DEMs and extraction of hydrographic features. *Third International Conference on Integrating GIS and Environmental Modeling, Santa Fe*, 1996.
- [118] P. Vretanos. Web Feature Service Implementation Specification 1.1.0. *Open Geospatial Consortium Inc.*, Mai 2005.
- [119] D. Walker and J. Dongarra. MPI: a standard Message Passing Interface. *Supercomputer*, 12, 1996.
- [120] V. Walter. Zuordnung von raumbezogenen Daten-am Beispiel der Datenmodelle ATKIS und GDF. *Dissertationsschrift*, Universität Stuttgart, 1997.
- [121] U. Weidmann. Transporttechnik der Fussgänger. *Strasse und Verkehr*, 78, 1992.
- [122] F. Weiser. Die Häufigkeit von Begegnungen zwischen Fahrzeugen auf zweistreifigen Straßen und ihr Einfluss auf den Verkehrsablauf. *Schriftenreihe des Lehrstuhl für Verkehrswesen*, 17, 1996.
- [123] M. Werner. Shuttle Radar Topography Mission(SRTM) - Mission overview. *EUSAR 2000, München*, 2000.
- [124] R. Wiedemann. Simulation des Straßenverkehrsflusses. *Habilitationsschrift*, Universität Karlsruhe, 1974.
- [125] R. Wiedemann and U. Reiter. Microscopic Traffic Simulation - The Simulation System MISSICON - Background and Actual State. Technical report, Project ICARUS (V1052), Brüssel, CEC, 1992.
- [126] J. Wu, M. Brackstone, and M. McDonald. Fuzzy sets and systems for a motorway microscopic simulation model. *Fuzzy Sets and Systems*, 116(1), 2000.
- [127] Z. Xu, Y. Yan, JX Chen, and C. Sichuan. OpenGL programming in Java. *Computing in Science & Engineering*, 7, 2005.
- [128] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3), 1965.