



# **Eine Simulationssoftware für die ereignisorientierte, mikroskopische Modellierung innerstädtischer Verkehrsnetze**

**Christian Gruhler**

**Konstanz, 14.01.2009**

**DIPLOMARBEIT**



# **Diplomarbeit**

**zur Erlangung des akademischen Grades**

**Diplom-Informatiker (FH)**

**an der**

**Hochschule Konstanz**

Technik, Wirtschaft und Gestaltung

Fakultät Informatik

Studiengang WI

Thema:

**Eine Simulationssoftware für die  
ereignisorientierte, mikroskopische  
Modellierung innerstädtischer Verkehrsnetze**

Diplomand:

Christian Gruhler, Wilhelmstraße 8, 78532 Tuttlingen

1. Prüfer:

Prof. Dr. Ulrich Hedtstück

2. Prüfer:

Prof. Dr. Michael Grütz

Abgabedatum:

14.01.2009



## **Ehrenwörtliche Erklärung**

Hiermit erkläre ich, Christian Gruhler, geboren am 28.11.1982 in Tuttlingen,

- (1) dass ich meine Diplomarbeit mit dem Titel:

**„Eine Simulationssoftware für die ereignisorientierte,  
mikroskopische Modellierung innerstädtischer Verkehrsnetze“**

im Institut unter Anleitung von Professor Dr. Ulrich Hedtstück selbständig und ohne fremde Hilfe angefertigt habe und keine anderen als die angeführten Hilfen benutzt habe;

- (2) dass ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 14.01.2009

[ausgeschriebene Unterschrift]



## **Zusammenfassung (Abstract)**

**Thema:** Eine Simulationssoftware für die ereignisorientierte, mikroskopische Modellierung innerstädtischer Verkehrsnetze

**Diplomand:** Christian Gruhler

**Firma:** HTWG

**Betreuer:** Prof. Dr. Ulrich Hedtstück

**Abgabedatum:** 14.01.2009

**Schlagworte:** Verkehrssimulation, Ereignisorientierung, Kreuzungen, Kreisverkehr, Ampel, Straße, Fahrzeug, Verkehrsnetz, Mikroskopisch

Die meisten modernen Verkehrssimulationen sind nach dem Prinzip der periodenorientierten Simulation aufgebaut. Neben diesem Prinzip gibt es noch das Prinzip der ereignisorientierten Simulation. In dieser Arbeit wird untersucht, ob es möglich ist, mit vergleichsweise geringem Aufwand auch eine ereignisorientierte Verkehrssimulation zu erstellen. Dafür wurden Konzepte und Modelle entwickelt um das Verhalten von Fahrzeugen und deren Interaktion im Straßennetz einer Stadt zu simulieren. Für die Abbildung der Fahrzeuge gibt es ein einfaches physikalisches/psychologisches Modell, welches die grundlegenden Verhaltensweisen von Fahrzeugen erklären kann. Die Kreuzungen werden je nach Kreuzungsart mit einem individuellen Kreuzungsmodell beschrieben, um den grundlegenden Unterschieden gerecht zu werden. Weiterhin werden auch weitere Komponenten eines Straßennetzes modelliert. Um auch komponentenübergreifende Phänomene wie Stau zu erklären, wurden auch dafür Methoden zur Simulation entwickelt. Dies alles wurde schließlich in Form einer Simulationssoftware implementiert und die Funktionsweise an Hand zweier Fallstudien demonstriert.



# Inhaltsverzeichnis

<b>1 EINLEITUNG.....</b>	<b>1</b>
<b>1.1 Vergleich mikroskopische und makroskopische Simulationen.....</b>	<b>1</b>
1.1.1 Die mikroskopische Simulation.....	2
1.1.2 Die makroskopische Simulation.....	2
1.1.3 Die mesoskopische Simulation.....	2
1.1.4 Einordnung meiner Verkehrssimulation.....	3
<b>1.2 Stand der Technik.....</b>	<b>4</b>
1.2.1 Kommerzielle Verkehrssimulationen.....	4
1.2.2 Die Theorie der Verkehrssimulation.....	4
1.2.3 VISSIM.....	4
<b>2 DIE PHYSIKALISCHEN GRUNDLAGEN.....</b>	<b>7</b>
<b>2.1 Die maximale Geschwindigkeit eines Fahrzeugs in einer Kurve.....</b>	<b>7</b>
2.1.1 Die grundlegende Idee.....	7
2.1.2 Die physikalische Formel.....	8
<b>2.2 Der Geschwindigkeitsverlust bei kurzfristiger Blockade eines Fahrzeugs.....</b>	<b>9</b>
2.2.1 Die physikalischen Formeln.....	9
<b>2.3 Die Geschwindigkeit des nachfolgenden Fahrzeugs beim Anfahren in der Gruppe.....</b>	<b>10</b>
2.3.1 Die physikalischen Formeln.....	11
2.3.2 Einschränkung.....	12
<b>2.4 Berechnung des Zeitbedarfs zum Zurücklegen einer Strecke.....</b>	<b>12</b>
2.4.1 Die grundlegende Idee.....	12
2.4.2 Die physikalischen Formeln.....	14
2.4.2.1 Herleiten der Formel für den 2. Fall.....	14
2.4.3 Verwendete Hilfsfunktionen.....	16
2.4.3.1 Die Methode „zeitZumKonstantFahren(double v, double s)“.....	16
2.4.3.2 Die Methode „zeitZumBeschleunigen(double v1, double v0, double a)“.....	16
2.4.3.3 Die Methode „geschwNachBeschleunigenUeberStrecke(double v0, double s, double a)“.....	17
2.4.3.4 Die Methode „streckeZumBeschleunigen(double v1, double v0, double a)“.....	17
2.4.3.5 Die Methode „beschleunigenUeberStrecke(double v1, double v0, double s)“.....	17
2.4.4 Die Implementierung.....	18
<b>3 DIE SIMULATION IM DETAIL.....</b>	<b>19</b>
<b>3.1 Allgemeiner Simulationsaufbau.....</b>	<b>19</b>
3.1.1 Grundprinzipien der ereignisorientierten Simulation.....	19
3.1.1.1 Das Ereignis.....	19
3.1.1.2 Die Ereignisliste.....	19
3.1.2 Die grundlegende Komponentendefinition als Simulationsabschnitt.....	20
3.1.3 Das globale Blockadesystem.....	21
3.1.3.1 Beispiel für einen Rückstau.....	21
3.1.3.2 Funktionsweise des Blockadesystems.....	22
3.1.3.3 Einschränkung des Blockadesystems.....	22
<b>3.2 Die grundlegenden Simulationskomponenten.....</b>	<b>22</b>
3.2.1 Die Senke.....	22
3.2.2 Die Quelle.....	23
3.2.2.1 Die Ankunftsverteilung.....	24
3.2.2.2 Die Erlang-Verteilung.....	25

3.2.3 Die Straße.....	26
3.2.3.1 Das Modell.....	27
<b>3.3 Die Fahrzeuge.....</b>	<b>28</b>
3.3.1 Was macht ein reales Fahrzeug aus?.....	28
3.3.2 Welche dieser Punkte sind für die Simulation wichtig?.....	29
3.3.3 Welchen Einschränkungen unterliegt ein Fahrzeug innerhalb des Modells?.....	29
3.3.4 Wie wird der Unterschied zwischen einem PKW und einem LKW im Modell realisiert?.....	30
3.3.5 Die Standardeinstellungen der Simulation.....	30
3.3.6 Wie wurde dies Implementiert?.....	31
3.3.6.1 Die Schnittstelle „Fahrzeug“.....	32
3.3.6.2 Die Schnittstelle „KreuzungFahrzeugContainer“.....	32
<b>3.4 Allgemeine Kreuzungskonzepte.....</b>	<b>33</b>
3.4.1 Das Warteschlangenkonzept.....	33
3.4.1.1 Funktionsweise der mehrstufigen Warteschlange.....	34
3.4.2 Die Ergebnismethoden.....	35
3.4.2.1 Die Klasse „WarteschlangenErgebnis“.....	35
3.4.2.2 Der Inhalt der Ergebnisklasse.....	35
3.4.2.3 Beispiel für den Ergebnisexport.....	37
3.4.3 Realisierung der Animation.....	38
3.4.3.1 Die Animation.....	39
3.4.4 Die Anbindung der Straßen an eine Kreuzung .....	39
3.4.4.1 Was für Straßenanbindungen sind erlaubt?.....	40
3.4.4.2 Tabelle der Abbiegewahrscheinlichkeiten.....	40
3.4.4.3 Falsches Abbiegen verhindern.....	42
3.4.5 Das Konzept der Segmente.....	42
3.4.5.1 Das Mehrsegment-Fahrzeug-Konzept.....	43
<b>3.5 Der Kreisverkehr.....</b>	<b>45</b>
3.5.1 Der Kreisverkehr in der graphischen Oberfläche.....	45
3.5.2 Das Modell.....	46
3.5.3 Die dynamische Berechnung der Segmentanzahl eines Kreisverkehrs.....	48
3.5.4 Die Abbiegelogik.....	48
3.5.5 Die Einfahrtregeln.....	49
3.5.6 Einschränkungen.....	49
<b>3.6 Die Rechts-vor-Links Kreuzung.....</b>	<b>49</b>
3.6.1 Die Rechts-vor-Links Kreuzung in der graphischen Oberfläche.....	49
3.6.2 Das Modell.....	50
3.6.3 Die Abbiegelogik.....	51
3.6.4 Die Einfahrtregeln.....	51
3.6.5 Einschränkungen.....	52
<b>3.7 Die Ampelkreuzungen.....</b>	<b>52</b>
3.7.1 Gemeinsamkeiten der vier Ampeltypen.....	52
3.7.1.1 Gemeinsamkeiten bei den Einstellungsdialogen.....	52
3.7.1.2 Die Einfahrtregeln.....	53
3.7.1.3 Die Ampelphasen.....	53
3.7.2 Die kleine Ampel.....	53
3.7.2.1 Die kleine Ampel in der graphischen Oberfläche.....	53
3.7.2.2 Das Modell.....	54
3.7.2.3 Die Abbiegelogik.....	55
3.7.3 Die mittelgroße Ampel.....	55
3.7.3.1 Die mittelgroße Ampel in der graphischen Oberfläche.....	55
3.7.3.2 Das Modell.....	56
3.7.3.3 Die Abbiegelogik.....	57
3.7.3.3.1 Das Problem der entgegenkommenden Linksabbieger.....	57
3.7.3.4 Einschränkungen.....	59
3.7.4 Die mittelgroße, optimierte Ampel.....	60

3.7.4.1 Die optimierte Ampel in der graphischen Oberfläche.....	60
3.7.4.2 Die Ampelphasen der optimierten Ampel.....	61
3.7.4.3 Einschränkung.....	62
3.7.5 Die große Ampel.....	62
3.7.5.1 Die große Ampel in der graphischen Oberfläche.....	62
3.7.5.2 Das Modell.....	63
3.7.5.3 Die Abbiegelogik.....	64
3.7.5.4 Einschränkungen.....	65
<b>3.8 Die Vorfahrtstraßenkreuzung.....</b>	<b>65</b>
3.8.1 Die Vorfahrtstraßenkreuzung in der graphischen Oberfläche.....	65
3.8.2 Das Modell.....	66
3.8.2.1 Das Problem der entgegenkommenden Linksabbieger.....	67
3.8.2.2 Das Problem der gleichzeitigen Rechts- und Linksabbieger.....	68
3.8.2.3 Die Lösung anhand des komplexeren Kreuzungsmodells.....	68
3.8.2.3.1 Lösung für das Problem der entgegenkommenden Linksabbieger.....	69
3.8.2.3.2 Die Lösung des Problems der gleichzeitigen Rechts- und Linksabbieger.....	69
3.8.3 Die Abbiegelogik.....	70
3.8.4 Die Einfahrtregeln.....	71
3.8.4.1 Einfahrt in eine Vorfahrtstraße.....	71
3.8.4.2 Abbiegeregeln für Abbiegen aus der Vorfahrtstraße.....	73
3.8.5 Besonderheiten.....	75
3.8.6 Einschränkungen.....	75
<b>4 DIE FALLSTUDIEN.....</b>	<b>76</b>
<b>4.1 Fallstudie 1.....</b>	<b>76</b>
4.1.1 Ergebnis.....	77
<b>4.2 Fallstudie 2.....</b>	<b>78</b>
4.2.1 Das Simulationsmodell.....	78
4.2.2 Ergebnis.....	79
<b>5 QUELLENVERZEICHNIS.....</b>	<b>81</b>
<b>6 ABBILDUNGSVERZEICHNIS.....</b>	<b>82</b>
<b>7 TABELLENVERZEICHNIS.....</b>	<b>84</b>
<b>ANHANG A – FALLSTUDIE 2.....</b>	<b>85</b>
<b>ANHANG B – PSEUDOCODE STRASSE.....</b>	<b>87</b>
<b>ANHANG C – PSEUDOCODE KREISVERKEHR.....</b>	<b>88</b>
<b>ANHANG D – PSEUDOCODE RECHTS-VOR-LINKS KREUZUNG.....</b>	<b>90</b>
<b>ANHANG E – PSEUDOCODE AMPEL.....</b>	<b>92</b>
<b>ANHANG F – PSEUDOCODE VORFAHRTKREUZUNG.....</b>	<b>95</b>

**ANHANG G – „PSEUDOCODE ZUM BERECHNEN DES ZEITBEDARFS ZUM  
BEFAHREN EINER STRECKE“.....97**

# 1 Einleitung

Seit den Anfängen des Automobils hat sich viel getan. Durch die gestiegenen Absatzzahlen der Automobilindustrie hat sich die Zahl der Fahrzeuge auf den Straßen explosionsartig vermehrt. Dadurch kommt es seit geraumer Zeit immer mehr zum Phänomen der überfüllten Straßen. Dieses Problem wird besonders in den Städten immer drängender. Um diesem Problem Herr zu werden gibt es zwei mögliche Ansätze. Zum einen den Ausbau bestehender Straßen bzw. den Bau neuer Straßen. Dieser Möglichkeit sind in Zeiten knapper Kassen enge Grenzen gesetzt. Zum anderen kann durch neue Verkehrsführungskonzepte eine verbesserte Verkehrsführung erreicht werden. Damit können Staus bereits in ihren Anfängen vermieden werden. Dies kann unter anderem durch Umgehung der Innenstädte bzw. durch neue Verkehrsregelungskonzepte, wie z.B. der allseits bekannten „Grünen Welle“ geschehen.

Das Problem für die Verkehrsplaner ist dabei, dass es teilweise sehr schwer sein kann, unter den verschiedenen Planungsalternativen die beste zu erkennen. Fehlentscheidungen bei der Verkehrsführung lassen sich im Nachhinein oft nur durch einen beträchtlichen finanziellen Aufwand wieder beheben. Daher ist es für moderne Verkehrsplanung beinahe unverzichtbar geworden, auf technische Hilfsmittel zu setzen. So ist es auch nicht verwunderlich, dass es sehr viele wissenschaftliche Untersuchungen zum Thema der Verkehrsplanung gibt. Diese haben viele wissenschaftliche Methoden und Werkzeuge zur Verkehrsplanung hervorgebracht. In diesem Bereich hat sich in der letzten Zeit vor allem das Hilfsmittel der Simulation als besonders nützlich erwiesen um die verschiedenen verkehrsplanerischen Alternativen miteinander zu vergleichen.

Soweit bekannt, basieren die meisten dieser Simulationslösungen auf dem Konzept der periodenorientierten Simulation. Als alternatives Konzept gibt es die Ereignisorientierung, welche aber von den meisten Herstellern von Verkehrssimulationen nicht verwendet wird. Gründe dafür könnten u.a. die Schwierigkeiten bei der Visualisierung, begrenzte Rechenkapazitäten und die Probleme beim Parallelisieren sein.

In meiner Diplomarbeit habe ich ein einfaches Modell eines ereignisorientierten, mikroskopischen Verkehrsmodells erstellt, welches in Form einer Simulationssoftware von mir in der Programmiersprache Java umgesetzt wurde. Ziel der Diplomarbeit war eine einfache Simulationssoftware auf Basis der Ereignisorientierung zu erstellen, welche eine Möglichkeit der Visualisierung in Form einer Animation enthält. Dabei wurden von mir auch die physikalische Grundlagen, sowie einfachen psychologischen Konzepten erarbeitet. Diese Grundlagen, zusammen mit verschiedenen Automatismen, ermöglichen auch die vergleichsweise einfache Bedienung der Simulationssoftware. Um nur die Schwerpunkte bei der Verkehrsplanung zu beachten, wurde das Modell nur für die Abbildung eines städtischen Verkehrsnetzes entworfen.

## 1.1 Vergleich mikroskopische und makroskopische Simulationen

In diesem Abschnitt wollen wir kurz auf wichtige Zusammenhänge und Begrifflichkeiten im Zusammenhang mit dem Thema Verkehrssimulationen eingehen.

### 1.1.1 Die mikroskopische Simulation

Bei der mikroskopischen Simulation hat jedes Element der Realität eine Abbildung innerhalb der Simulation. So wird jedes Auto innerhalb der Simulation als eigenständiges Objekt definiert und dessen Handlungen sowie alle wichtigen Eigenschaften virtuell abgebildet. Bei aufwendigeren Verkehrssimulationen werden dabei nicht nur die Autofahrer, sondern auch andere Objekte wie Straßenbahnen oder Fußgänger simuliert, um eine möglichst genaue Abbildung der Realität zu erhalten und die Wechselwirkungen zwischen den einzelnen Objekten möglichst präzise darzustellen. Dabei werden sowohl physikalische wie auch psychologische Aspekte der einzelnen Objekte beachtet. Als Nachteil der mikroskopischen Simulation kann ihr immenser Rechenaufwand betrachtet werden, der jedoch durch die moderne Computertechnik und die verbesserte, parallelisierte Datenverarbeitung immer mehr in den Hintergrund tritt.

### 1.1.2 Die makroskopische Simulation

Bei der makroskopischen Simulation werden die Fahrzeuge nicht mehr als eigenständige Objekte betrachtet, sondern sie werden gedanklich verflüssigt. [@ROSE01]

Ein Beispiel, wie man sich dieses Verflüssigen vorstellen kann, ist in Abbildung 1.1 aufgeführt. Zu sehen ist, dass die einzelnen Fahrzeuge zu zwei Kurven, der Geschwindigkeit  $v(x, t)$  und der Verkehrsdichte  $\rho(x, t)$  verflüssigt werden. Mithilfe der Fluidodynamik (Teil der Strömungslehre) könne so auch sehr große Verkehrsmodelle mit vergleichsweise geringem Rechenaufwand simuliert werden. Allerdings können nicht alle Verkehrsabläufe mit Hilfe dieser Modellierung simuliert werden, wodurch die makroskopische Simulation stets ungenauer als die mikroskopische Simulation desselben Modells ist.

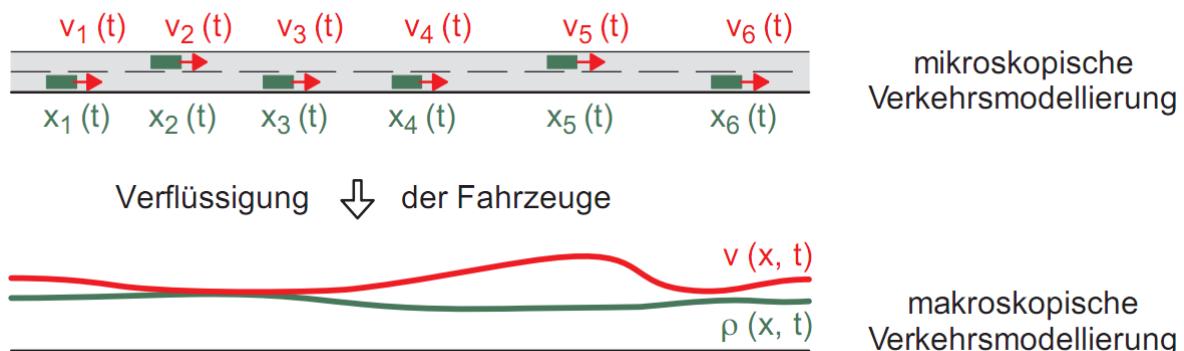


Abbildung 1.1: Übergang von der mikroskopischen zu makroskopischen Modellierung nach [@ROSE01]

### 1.1.3 Die mesoskopische Simulation

Unter der mesoskopischen Simulation versteht man jetzt die Vereinigung beider bisher angesprochenen Simulationsarten. Dies kann für einen Übergang von einer mikroskopischen auf eine makroskopische Simulation verwendet werden. Ein Beispiel dafür ist in Abbildung 1.2 zu sehen. Dabei wird die mikroskopische Simulation für ein kleinen Teilbereich verwendet und die makroskopische Simulation für das Umfeld. Dass dabei kein harter Bruch an den Grenzen der unterschiedlichen Simulationsbereichen auftritt, dafür sorgt die mesoskopische Simulation. Diese vermittelt quasi zwischen den beiden Simulationsarten. So können also die Vorteile beider Simulationsarten miteinander kombiniert werden. Es wird die präzise Darstellung der mikroskopischen Simulation für den Kernbereich der Simulation und die

ungenauere makroskopische Simulation für das eher unwichtigere Umfeld der Simulation verwendet. Somit kann trotz begrenzter Rechenleistung ein ziemlich genaues Modell für verschiedene Verkehrssituationen erstellt werden.

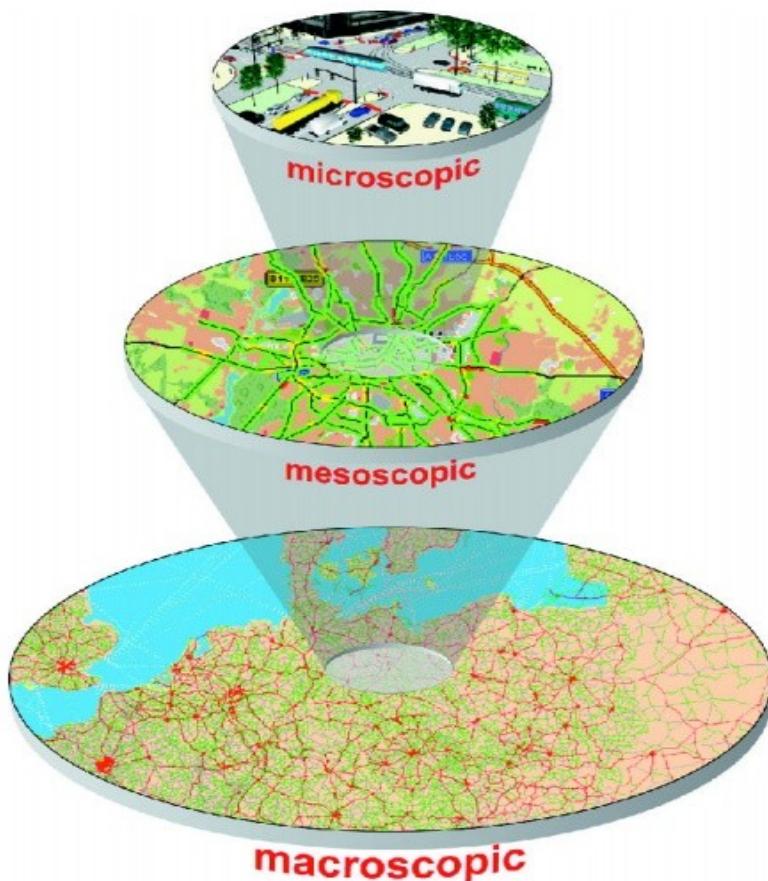


Abbildung 1.2: Wofür mikro-, meso- oder makroskopische Simulation. [VISF@]

### 1.1.4 Einordnung meiner Verkehrssimulation

Meine Verkehrssimulation orientiert sich an dem Modell der mikroskopischen Simulation. So werden alle Fahrzeuge als eigenständige Objekte dargestellt, welche über physikalische und psychologische Aspekte verfügen und innerhalb der Simulation miteinander interagieren. In meiner Ausarbeitung werde ich auch nochmals besonders auf die physikalischen Aspekte zu sprechen kommen. Auf die psychologischen Aspekte werde ich innerhalb meiner Ausarbeitung nicht gesondert eingehen, sondern diese werden eher nebenher bei den verschiedenen Modellen beschrieben.

Nach den Regeln der modernen Programmierung ist die Simulationssoftware nach dem Prinzip des Modell-View-Controller-Pattern in verschiedene Komponenten unterteilt. Die Simulationslogik ist komplett von der graphischen Oberfläche entkoppelt. Die Idee und Implementierung der Logik wurde komplett von mir allein gestaltet.

Teilideen für die Gestaltung der graphischen Oberfläche entstammt aus dem Programm Navigator 3 aus der Methodenbank der HTWG-Konstanz. Allerdings wurde diese Teilideen von mir überarbeitet, angepasst und teilweise auch komplett neu programmiert.

## 1.2 Stand der Technik

Kommen wir nun zu einer kurzen Übersicht über den aktuellen Stand der Technik im Bereich der Verkehrssimulationen.

### 1.2.1 Kommerzielle Verkehrssimulationen

Der Bereich der Verkehrssimulationen ist momentan einer der am besten erforschten Simulationsbereiche. Es hat sich in den letzten Jahrzehnten ein großes Angebot an kommerzieller Software für diesen Bereich etabliert. Als aktueller Marktführer ist dabei das Programm VISSIM für mikroskopische Simulationen und sein makroskopisches Pendant VISUM zu nennen. Wobei anzumerken ist, dass es den mikroskopischen Simulationen u.a. durch die gesteigerte Rechenleistung der aktuellen Systeme und der genaueren Abbildung der Verkehrslogik gelungen ist, die makroskopischen Simulationen aus den meisten Anwendungsgebieten der Verkehrssimulation zu verdrängen.

### 1.2.2 Die Theorie der Verkehrssimulation

Auch im Bereich der Theorie hat sich bei den Verkehrssimulationen viel getan. So haben sich die physikalische und vor allem die psychologischen Modelle in den letzten Jahrzehnten deutlich weiter entwickelt. In modernen Verkehrssimulationen werden viele unterschiedliche psychologische Aspekte beachtet. Dies geht so weit, dass manche Modelle sogar versuchen, die Verarbeitung der von den Augen erfassten Information im Gehirn nachzubilden. So gibt es Modelle für die Wahrnehmung von Geschwindigkeiten, Wahrnehmung von Abständen zu vorausfahrenden Fahrzeugen, Simulationsmodelle für Fußgänger und vieles mehr. Auch die Routenberechnung und die automatische Anpassung der Route zur Stauumfahrung helfen, dass moderne Modelle bereits eine sehr genaue Nachbildung der Realität erreichen. Dies alles, genauso wie die 3D-Visualisierung der Simulationen, wurde erst in jüngster Zeit durch die stark gesteigerte Rechenleistung und die starke Parallelverarbeitung modernen Rechnersysteme ermöglicht.

### 1.2.3 VISSIM

Das Simulationsprogramm VISSIM ist ein Produkt der PTV Planung Transport Verkehr AG aus Karlsruhe. VISSIM steht für „Verkehr In Städten SIMulation“ und ist eine der weltweit führenden, mikroskopischen Simulationsprogramme für Verkehrsnetze.

Dabei wird für „*die Bewegung der Fahrzeuge [...] ein wissenschaftlich fundiertes psychophysisches Fahrzeugfolge-Modell mit einer zeitlichen Auflösung von bis zu 1/10 Sekunde verwendet*“ [VISF@].

Es darf also davon ausgegangen werden das VISSIM, genauso wie die meisten anderen Verkehrssimulationen, nach dem Prinzip der periodenorientierte Simulation arbeitet.



Abbildung 1.3: 3D Rundflug durch eine von VISSIM animierte Stadt. [VISF@]

Dabei ist VISSIM inzwischen mehr als nur ein Simulationsprogramm für Verkehrsnetze. In der aktuellen Version werden unter anderem neben Fahrzeugen auch der öffentliche Nahverkehr, Fußgänger, Fahrradfahrer und vieles weiteres simuliert, wobei alle diese simulierte Objekte miteinander interagieren können.

[WIKI01@]

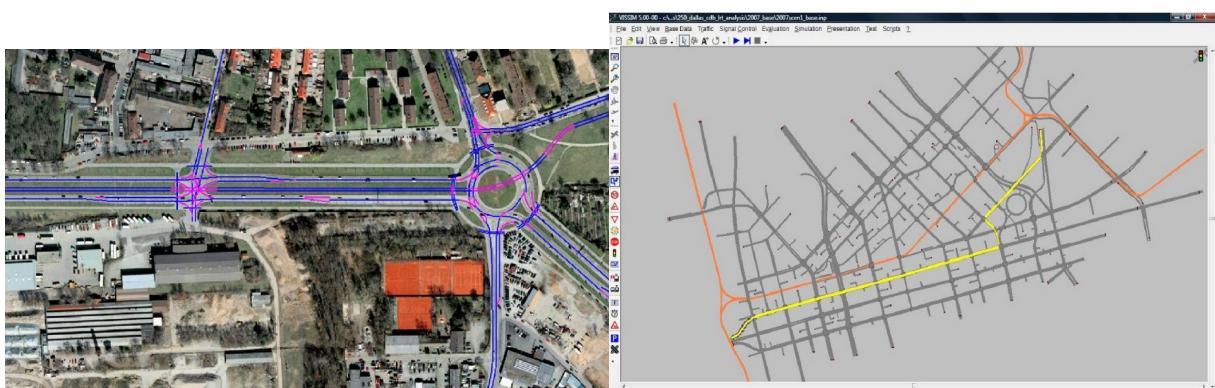


Abbildung 1.4: Planung des Verkehrsnetzes mit VISSIM. [VISF@]

Auch zur Planung und Visualisierung gibt es umfangreiche Möglichkeiten. Die grobe Planung des Verkehrsnetzes kann so z.B. direkt aus Luftbildern (siehe Abbildung 1.4) geschehen. Dabei können auch individuelle Kreuzungen mit komplexen Schaltlogiken (Abbildung 1.5) im Simulationspaket erstellt werden. Aus diesen Verkehrsnetzmodellen und der folgenden Simulation kann später dann, zur besseren Darstellung und für Werbezwecke, sogar auch

noch ein animierter 3D Rundflug durch die simulierte Stadt erstellt werden (siehe Abbildung 1.3). So können die verschiedenen Planungskonzepte besonders werbewirksam in Szene gesetzt werden.

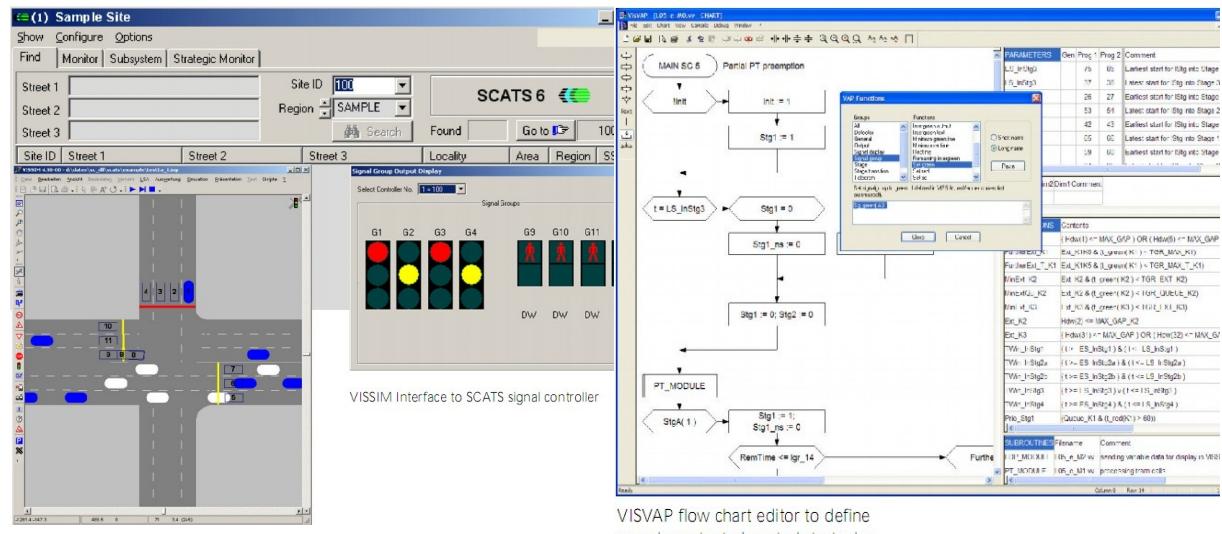


Abbildung 1.5: Planung von Kreuzungen und Ampelphasen mit VISSIM. [VISF@]

## 2 Die physikalischen Grundlagen

In diesem Abschnitt wollen wir uns mit den physikalischen Grundlagen, auf welchen diese Simulation basiert, befassen. Wir werden dabei von den grundlegenden Ideen über die physikalischen Formeln bis zur fertigen Implementierung (Pseudocode) die Entwicklung der Konzepte verfolgen.

Die Implementierung der physikalischen Berechnungen erfolgt in der Klasse Physik.

### 2.1 Die maximale Geschwindigkeit eines Fahrzeugs in einer Kurve

In diesem Abschnitt wird aufzeigt, wie die maximale Geschwindigkeit eines Fahrzeugs in der Kurve berechnet werden kann. Diese Methode wird in der Simulationssoftware dazu verwendet die maximale Geschwindigkeit von Rechts- und Linksabbiegern innerhalb einer Kreuzung zu ermitteln.

#### 2.1.1 Die grundlegende Idee

Ziel dieser Methode ist es, die maximale Geschwindigkeit mit welcher ein Fahrzeug durch eine Kurve fährt, zu ermitteln. Zur Berechnung dieser Geschwindigkeit gilt es erstmals die Faktoren, welche die Geschwindigkeit beeinflusse zu ermitteln.

Diese Faktoren sind:

- Der Kurvenradius:  
Natürlich beeinflusst der Kurvenradius die maximale, in einer Kurve erreichbare, Geschwindigkeit. So ist es durchaus möglich eine Kurve mit einem großen Kurvenradius wesentlich schneller zu durchfahren, ohne dass die Reifen den Bodenkontakt verlieren, als dies bei einer Kurve mit einem Minimalradius der Fall wäre.
- Der Fahrer:  
Des Weiteren hat jeder Fahrzeugführer seine individuelle Art zu fahren. Dies drückt sich u.a. dadurch aus, dass jeder Fahrer bei sonst gleichen Bedingungen eine Kurve mit einer leicht unterschiedlichen Geschwindigkeit durchfährt. So empfindet jeder Fahrer eine andere maximale Zentrifugalbeschleunigung (die Kraft, die einen in der Kurve nach außen drückt) als angenehm.
- Das Fahrzeug:  
Ein weiterer Faktor, welcher die Kurvengeschwindigkeit beeinflusst ist das Fahrzeug an sich. So kann ein PWK durch seinen niedrigen Schwerpunkt eine Kurve mit einer wesentlich höheren Geschwindigkeit durchfahren, ohne dass das Fahrzeug durch die wirkende horizontale Beschleunigungen (Zentrifugalbeschleunigung) umkippt, als dies bei einem LKW möglich wäre. Weitere Faktoren wie Reifen und Straßenbelag entscheiden, welche Zentrifugalbeschleunigung das Fahrzeug maximal aushält.
- Die Straße:  
Als letztes müssen noch die aktuellen Straßenverhältnisse beachtet werden. So wird bei Glatteis die Haftriebung der Reifen stark verringert, wodurch das Fahrzeug schon

bei geringer Kurvengeschwindigkeit aus der Spur ausbrechen kann. Um dies zu verhindern, muss jeder Fahrer eine zu hohe Zentrifugalbeschleunigung vermeiden und langsamer durch eine Kurve fahren.

Abschließend muss man noch erwähnen, dass die letzten drei genannten Faktoren sich alle auf dieselbe Problematik zusammenfassen lassen. Und zwar nämlich die Zentrifugalbeschleunigung, welche immer einen maximalen Wert nicht überschreiten darf.

Alle diese Punkte kann man also zusammengefasst eigentlich mit zwei grundlegenden Eigenschaften beschreiben.

1. Dem Kurvenradius
2. Die Zentrifugalbeschleunigung

So spielen auch in diesem Modell auch nur noch diese beiden Eigenschaften, bei der Berechnung der maximalen Kurvengeschwindigkeit, eine Rolle.

### 2.1.2 Die physikalische Formel

Die Berechnung der Zentrifugalbeschleunigung erfolgt durch folgende physikalische Formel:

$$a_z = \frac{v^2}{r}$$

Dabei gilt:

- $a_z$  ist die Zentrifugalbeschleunigung in  $\frac{\text{Meter}}{\text{Sekunde}^2}$
- $v$  ist die Geschwindigkeit in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $r$  ist der Radius der Kurve in  $\text{Meter}$

Da der Simulation die maximale Zentrifugalbeschleunigung  $a_z$  genauso wie der Kurvenradius  $r$  bekannt ist und wir die Geschwindigkeit ermitteln wollen, erhalten wir durch Umformen:

$$v^2 = r \cdot a_z$$

bzw.

$$v_1 = +\sqrt{r \cdot a_z}$$

$$v_2 = -\sqrt{r \cdot a_z}$$

Da wir annehmen können, dass unser Fahrzeug nur eine positive Geschwindigkeit hat und die Zentrifugalbeschleunigung auch positiv ist, gilt:

$$v = \sqrt{r \cdot a_z}$$

## 2.2 Der Geschwindigkeitsverlust bei kurzfristiger Blockade eines Fahrzeugs

Durch eine Blockade innerhalb einer Kreuzung kann es zu einem Stocken des Verkehrsflusses kommen. So muss bei einer kurzfristigen Blockade eines Segments das nachfolgende Fahrzeug warten bis das Segment wieder frei ist. Beim Modell müsste man also solange stehen bleiben bis die Blockade behoben ist und man weiterfahren kann. Dabei macht es keinen Sinn die Geschwindigkeit des Fahrzeugs auf 0 Kilometer pro Stunde zu setzen, weil in der Realität ein Fahrer diese Blockade durch vorausschauendes Fahren erkennen und frühzeitig seine Geschwindigkeit der Situation anpassen würde. Dadurch würde das Fahrzeug nicht komplett zum Stillstand kommen, sondern sich nur verlangsamen.

Da dieses vorausschauende Fahren sehr schwierig zu simulieren ist und diese Blockadesituation eher selten auf einer Kreuzung auftreten dürfte, behelfen wir uns hier mit einer näherungsweisen Lösung des Problems. Dazu nehmen wir an, dass der Fahrer die Blockade kommen sieht und anstelle stehen zu bleiben einfach mit einer gleichmäßigen Beschleunigung über die Segmentlänge beschleunigt, bzw. abremst und es so zu keinerlei Blockade kommt. Um dies zu erreichen brauchen wir neben der Eintrittsgeschwindigkeit des Fahrzeugs zum Beginn des Segments die bekannte Streckenlänge und die Zeitdauer, welche zwischen dem Eintrittszeitpunkt des Fahrzeugs und dem Zeitpunkt des Blockadeendes liegt.

### 2.2.1 Die physikalischen Formeln

Zum Berechnen des Geschwindigkeitsverlusts werden folgende Formeln benötigt:

$$s = v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2$$

$$v_1 = v_0 + a \cdot t$$

Wobei gilt:

- $v_0$  ist die Anfangsgeschwindigkeit des Fahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $v_1$  ist die neue Geschwindigkeit des Fahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $s$  ist die Segmentlänge in  $\text{Meter}$
- $t$  ist die Zeit, welche seit dem Verlassen des letzten Segments verstrichen sind in  $\text{Sekunden}$
- $a$  ist die Beschleunigung des Fahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}^2}$

Nun formen wir die erste Formel nach  $a$  um:

$$s - v_0 \cdot t = \frac{1}{2} \cdot a \cdot t^2$$

$$2s - 2v_0 \cdot t = a \cdot t^2$$

$$a = \frac{2s - 2v_0 \cdot t}{t^2}$$

Durch diese Berechnung erhalten wir die Beschleunigung, mit welcher das Fahrzeug konstant beschleunigen bzw. abbremsen (bei negativer Beschleunigung) müsste, um ohne eine Blockadesituation das Segment zu durchfahren. Mit dieser Beschleunigung können wir auch ziemlich einfach die Geschwindigkeit berechnen, welche das Fahrzeug durch das vorausschauende Fahren des Fahrers jetzt haben müsste. Dies machen wir mit Hilfe der zweiten Formel:

$$v_1 = v_0 + a \cdot t$$

Natürlich gilt auch hier die simulationstypische Einschränkung, dass das Fahrzeug nicht schneller als seine maximale Höchstgeschwindigkeit fahren darf. Dies muss leider beachtet werden, da wir in diesem Modell von einer konstanten Beschleunigung ausgehen, wohingegen die normale Streckenberechnung (siehe Kapitel 2.4 „Berechnung des Zeitbedarfs zum Zurücklegen einer Strecke“) wesentlich komplexere Modelle erlaubt.

## 2.3 Die Geschwindigkeit des nachfolgenden Fahrzeugs beim Anfahren in der Gruppe

Ein weiteres Phänomen welches beachtet werden muss, ist das Einfahren mehrerer stehender Fahrzeuge hintereinander aus einer Warteschlange in die Kreuzung. Bei diesem Anfahren in der Gruppe ist es so, dass das erste Fahrzeug noch aus dem Stehen anfährt, also mit 0 km/h Startgeschwindigkeit beginnt, während das zweite Fahrzeug schon eine längere Strecke zum Beschleunigen hatte.

Um uns erstmal genau zu verbildlichen was genau beim Anfahren in der Gruppe passiert, nehmen wir an, dass beide Fahrzeuge gleich stark beschleunigen. Dieser Fall wird auch in der Tabelle 2.1 nochmals aufgezeigt.

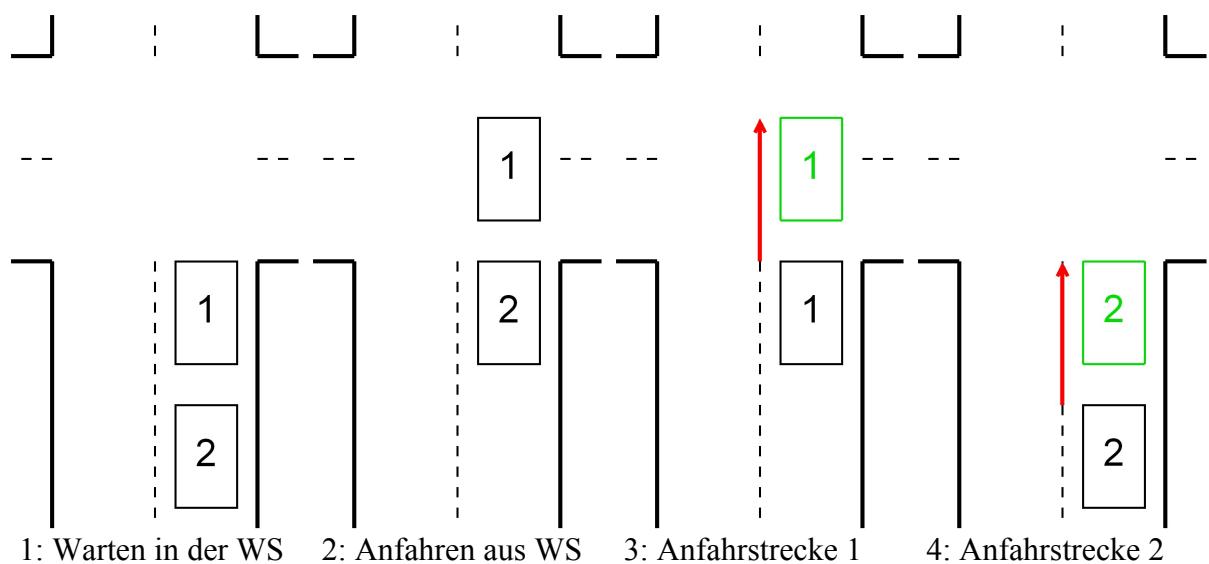


Tabelle 2.1: Anfahren aus einer Warteschlange

Wenn mehrere Fahrzeuge hintereinander aus einer Warteschlange anfahren (siehe Bild 1 und 2 der Tabelle 2.1), hat das zweite Fahrzeug dieselbe Zeitspanne wie auch dieselbe Streckenlänge zum Beschleunigen wie das erste Fahrzeug. Das heißt: es hat, bei identischer Beschleunigung, dieselbe Geschwindigkeit beim Einfahren in die Kreuzung wie das erste Fahrzeug zu diesem Zeitpunkt. Dies gilt unabhängig von der Größe eines Fahrzeugs.

Dieses Prinzip des Anfahrens ist nicht nur auf das erste und zweite Fahrzeug einer Warteschlange begrenzt, sondern gilt auch für alle weiter hinten anfahrende Fahrzeuge.

Um jetzt herauszufinden wie die Geschwindigkeit des nachfolgenden Fahrzeugs ist, wenn die Beschleunigungen des ersten und des zweiten Fahrzeugs nicht identisch sind brauchen wir eine Formel und es gelten folgende Einschränkung:

- Ein nachfolgendes Fahrzeug kann nicht schneller beschleunigen als das vorausfahrende Fahrzeug, da es sonst zu einem Auffahrungsfall kommen würde.
- Ein Fahrzeug darf seine maximale Höchstgeschwindigkeit für diese Kreuzung nicht überschreiten.
- Durch die unterschiedlich starken Beschleunigungen der Fahrzeuge kann nicht mehr angenommen werden, dass die Beschleunigungsstrecke der beiden Fahrzeuge gleich lang ist. Allerdings ist die Zeitspanne in welcher beide Fahrzeuge beschleunigen immer noch dieselbe.

### 2.3.1 Die physikalischen Formeln

Allgemein gilt zum Berechnen der Geschwindigkeit beim Anfahren aus dem Stehen folgende Formel:

$$v = a \cdot t$$

Für die nun folgenden Formeln gelten folgende Definitionen:

- $v_v$  = Geschwindigkeit des Vorgängerfahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $v_n$  = Geschwindigkeit des Nachfolgefahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $a_v$  = Beschleunigung des Vorgängerfahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}^2}$
- $a_n$  = Beschleunigung des Nachfolgefahrzeugs in  $\frac{\text{Meter}}{\text{Sekunde}^2}$
- $t$  = Zeitdauer in *Sekunden* zwischen der Einfahrt des Vorgängerfahrzeugs und der Einfahrt des Nachfolgefahrzeugs in die Kreuzung.

Für das Vorgängerfahrzeug gilt also:  $v_v = a_v \cdot t$

Durch das Umformen nach  $t$  kann man aus dieser Formel berechnen, wie lange das Fahrzeug zum Einfahren gebraucht hat:

$$t = \frac{v_v}{a_v}$$

Weiterhin gilt für das nachfolgende Fahrzeug:  $v_n = a_n \cdot t$

Durch das Einsetzen der umgeformten, ersten Formel in die zweite Formel erhalten wir:

$$v_n = \frac{a_n}{a_v} \cdot v_v$$

Dies gilt allerdings nur falls  $a_n \leq a_v$ , weil sonst das hintere Fahrzeug auf seine Vorgänger auffahren würde.

Ist dies nicht der Fall, gilt dass ein nachfolgendes Fahrzeug nicht schneller als das vorausfahrende Fahrzeug fahren kann. Es gilt also:

$$v_n = v_v$$

Als simulationstypische Einschränkung gilt, dass die berechnete Geschwindigkeit  $v_n$  nicht höher als die maximale Höchstgeschwindigkeit für das Fahrzeug innerhalb der Kreuzung sein darf. Falls das Fahrzeug seine maximale Höchstgeschwindigkeit für diese Kreuzung überschreiten würde, wird an Stelle der berechneten Geschwindigkeit  $v_n$  einfach die maximale Höchstgeschwindigkeit eingesetzt.

### 2.3.2 Einschränkung

Durch das Simulationsmodell ist es leider nicht möglich die Zeitspanne, die ein nachfolgendes Fahrzeug benötigt um an die erste Stelle der Warteschlange zu gelangen, zu berücksichtigen. Das bedeutet also, dass das zweite Fahrzeug direkt nach dem ersten Fahrzeug in die Kreuzung einfahren kann. Dies geht auch dann, wenn das nachfolgende Fahrzeug aufgrund seiner geringeren Beschleunigung eigentlich die Wegstrecke bis zur Kreuzungseinfahrt nicht hätte zurücklegen können. Allerdings dürfte sich dieser Fehler innerhalb der Simulation nur minimal auswirken. Deshalb kann dieses Problem im Modell vernachlässigt werden.

## 2.4 Berechnung des Zeitbedarfs zum Zurücklegen einer Strecke

Kommen wir nun zur Berechnung der Zeitdauer, welche von einem Fahrzeug benötigt wird um eine bestimmte Strecke zurückzulegen. Diese Methode wird in der fertigen Simulation sowohl für lange Straßen, genauso wie für das Abfahren einzelner Segmente verwendet.

### 2.4.1 Die grundlegende Idee

In der Realität erscheint das Befahren eines Streckenabschnitts auf den ersten Blick eigentlich nicht besonders komplex. Wenn man es allerdings mal etwas genauer betrachtet, so zeigt sich doch, dass die Sache nicht ganz so trivial ist, wie man zuerst denkt.

So ist es durchaus üblich auf einem längeren Streckenabschnitt innerhalb der Stadt auf die maximale Höchstgeschwindigkeit von 50 km/h zu beschleunigen, damit einen großen Teil der Strecke zu befahren, bevor man wieder an der nächsten Kreuzung abremst. Reicht der Straßenabschnitt allerdings nur aus, um kurzfristig auf die Höchstgeschwindigkeit zu beschleunigen und danach sofort wieder eine Vollbremsung hinzulegen, so wird dies kaum ein Autofahrer machen. Der Autofahrer wird eher schwach beschleunigen, mit kleinerer Geschwindigkeit ein Stück fahren und dann wieder vor der nächsten Kreuzung bremsen. Weiterhin ist es möglich, dass der Fahrer mit 50 km/h dem Verlauf der Vorfahrtstraße gefolgt ist und nun nur ein kurzes Straßenstück hat, um vor einer roten Ampel anzuhalten. Dadurch würde er gezwungen die komplette Straßenlänge zum Bremsen zu benutzen.

Alles in allem ist das Befahren eines Streckenabschnitts also nicht so einfach zu simulieren, wie man auf den ersten Blick denken könnte.

Um die spätere Implementierung zu verstehen, wollen wir alle möglichen Fälle genauer betrachten:

1. Der Autofahrer kann die Strecke optimal ausnutzen.  
Er beschleunigt am Anfang der Strecke auf die zulässige Höchstgeschwindigkeit, fährt mit dieser solange es geht und bremst, falls dies erforderlich ist, vor der nächsten Kreuzung wieder ab.
2. Um das Auto auf die zulässige Höchstgeschwindigkeit zu beschleunigen und später wieder abzubremsen ist die Straßenlänge zu kurz.  
Er beschleunigt also sein Fahrzeug auf eine angemessene Geschwindigkeit, fährt eine Zeit lang konstant mit dieser Geschwindigkeit und bremst dann wieder vor der nächsten Kreuzung ab.
3. Die Strecke ist so kurz, dass der Autofahrer beinahe die gesamte Strecke zum Bremsen vor der Kreuzung benötigt.  
In diesem Fall wird der Autofahrer wohl kaum beschleunigen, nur um direkt wieder Bremsen zu müssen. Er wird also versuchen, solange es noch geht, mit seiner aktuellen Geschwindigkeit zu fahren, um dann mit dem Bremsen erst vor der nächsten Kreuzung zu beginnen.
4. Bei einer kurzen Strecke kann auch der Fall eintreten, dass die Ampel an der folgenden Kreuzung gerade Grün zeigt. Dann wird der Fahrer bis zur nächsten Kreuzung sein Fahrzeug beschleunigen, um danach mit der zulässigen Höchstgeschwindigkeit in das nachfolgende Segment bzw. die nachfolgende Kreuzung einzufahren.
5. Die Strecke ist zu kurz für einen vollen Geschwindigkeitsausgleich zum nächsten Segment. Sollte dem Fahrer nur ein sehr kurzer Streckenabschnitt zur Verfügung stehen, so wird er versuchen möglichst die ganze Streckenlänge zum bremsen, beziehungsweise zum beschleunigen, zu nutzen.

Natürlich müssen auch noch Sonderfälle beachtet werden:

- So kann es z.B. sein, dass ein Fahrzeug zu schnell in einen Straßenabschnitt einfährt. Dann muss der Fahrer zuerst auf die maximal zulässige Höchstgeschwindigkeit abbremsen, bevor er mit dieser Geschwindigkeit die restliche Strecke befährt.

Des Weiteren nehmen wir als Vereinfachung an, dass alle Autofahrer mindestens 50 Prozent der Straßenlänge mit einer konstanten Geschwindigkeit durchfahren möchten, falls dies möglich ist.

## 2.4.2 Die physikalischen Formeln

In den nun folgenden Herleitungen setzen wir folgende Formeln als bekannt voraus.

Zum Berechnen der konstanten Geschwindigkeit  $v$  über eine Strecke  $s$  und einen Zeitraum  $t$ :

$$v = \frac{s}{t}$$

Zum Berechnen der Endgeschwindigkeit  $v_1$  bei konstanter Beschleunigung  $a$  über eine Strecke  $s$  mit einer Anfangsgeschwindigkeit  $v_0$ :

1. Formel:  $2as = v_1^2 - v_0^2$

2. Formel:  $v_1 = v_0 + a \cdot t$

### 2.4.2.1 Herleiten der Formel für den 2. Fall

Da es für den 2. Fall keine allgemein bekannte Formel zum Berechnen der Höchstgeschwindigkeit gibt, müssen wir zur Berechnung der Höchstgeschwindigkeit des Fahrzeugs eine eigene Formel herleiten.

Allgemein gilt die Formel:

$$2as = v_2^2 - v_0^2$$

Durch umformen der Formel nach  $v_2$  erhält man:

$$v_2^2 = 2as + v_0^2$$

Dabei gelten folgende Variablendefinitionen:

- $a$  = Beschleunigung in  $\frac{\text{Meter}}{\text{Sekunde}^2}$
- $b$  = Bremsbeschleunigung in  $\frac{\text{Meter}}{\text{Sekunde}^2}$  (wobei gilt:  $b$  ist einfach nur eine negative Beschleunigung)
- $s$  = Zum Beschleunigen und Bremsen zur Verfügung stehende Streckenlänge in Meter
- $v_0$  = Anfangsgeschwindigkeit in Meter pro Sekunde
- $v_1$  = maximal erreichbare Geschwindigkeit in Meter pro Sekunde
- $v_2$  = Endgeschwindigkeit in Meter pro Sekunde

Zuerst müssen wir die Gesamtlänge der Strecke in zwei Teile einteilen.

Es gibt einen Streckenabschnitt, welcher vom Fahrer mit einer konstanten Geschwindigkeit durchfahren wird (in unserem Fall sind dies 50 Prozent der Gesamtstrecke) und den Streckenabschnitt welcher dem Fahrer zum Beschleunigen und Bremsen zur Verfügung steht. Diesen letzten Streckenabschnitt bezeichnen wir in dieser Herleitung als  $s$ .

Diesen zur Verfügung stehenden Streckenabschnitt  $s$  teilen wir weiterhin in die Beschleunigungsstrecke  $s_1$  und die Bremsstrecke  $s_2$  ein. Darum gilt:

$$s = s_1 + s_2$$

Durch Umformen nach  $s_1$  erhalten wir:

$$s_1 = s - s_2$$

Weiterhin gilt für die beiden Streckenabschnitte die oben aufgeführte Formel.

Für die Beschleunigungsstrecke  $s_1$  gilt:  $v_1^2 = 2as_1 + v_0^2$

Da  $a = -b$  ist, gilt für die Bremsstrecke  $s_2$ :  $v_2^2 = -2bs_2 + v_1^2$

Nun setzen wir die Formel für die Beschleunigungsstrecke  $s_1$  in die Formel für die Bremsstrecke  $s_2$  einsetzt und erhalten:

$$v_2^2 = -2bs_2 + 2as_1 + v_0^2$$

Weil wir jetzt wissen, dass  $s_1 = s - s_2$  ist können wir die Variable  $s_1$  ersetzen:

$$v_2^2 = -2bs_2 + 2a(s - s_2) + v_0^2$$

Durch weiteres Umformen erhalten wir:

$$\Rightarrow 0 = -2bs_2 + 2a(s - s_2) + v_0^2 - v_2^2$$

$$\Rightarrow 0 = -2bs_2 + 2as - 2as_2 + v_0^2 - v_2^2$$

$$\Rightarrow 0 = (-2b - 2a)s_2 + 2as + v_0^2 - v_2^2$$

Schließlich erhalten wir für die Bremsstrecke  $s_2$  folgende Formel:

$$s_2 = \frac{v_0^2 + 2as - v_2^2}{2b + 2a}$$

Wenn wir jetzt noch den Geschwindigkeitsunterschied und den Streckenverbrauch zum Ausgleichen der Anfangs- und Endgeschwindigkeit aus der Strecke  $s$  heraus rechnen entspricht  $v_0 = v_1$  und damit gilt:

$$s_2 = \frac{2as}{2b + 2a}$$

Beziehungsweise vereinfacht:

$$s_2 = \frac{as}{b + a}$$

Daraus können wir dann die Streckenlänge der Strecke  $s_2$  berechnen.

Da wir jetzt die Bremsstrecke  $s_2$  haben, können wir daraus auch die Beschleunigungsstrecke  $s_1$  berechnen, da gilt:

$$s_1 = s - s_2$$

Dadurch können wir schließlich mithilfe der Hilfsfunktionen „geschwNachBeschleunigenUeberStrecke“ (Kapitel 2.4.3.3) und „zeitZumBeschleunigen“ (Kapitel 2.4.3.2) die Gesamtdauer für das Durchqueren der Strecke berechnen.

### 2.4.3 Verwendete Hilfsfunktionen

Soweit nicht anders definiert gelten folgende Variablendefinitionen:

- $v$  = konstante Geschwindigkeit in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $s$  = Streckenlänge in  $\text{Meter}$
- $t$  = Zeit in  $\text{Sekunden}$
- $a$  = Beschleunigung in  $\frac{\text{Meter}}{\text{Sekunde}^2}$
- $b$  = Bremsen entspricht einer negativen Beschleunigung ( $b = -a$ )
- $v_1$  = Endgeschwindigkeit in  $\frac{\text{Meter}}{\text{Sekunde}}$
- $v_0$  = Anfangsgeschwindigkeit in  $\frac{\text{Meter}}{\text{Sekunde}}$

#### 2.4.3.1 Die Methode „zeitZumKonstantFahren(double v, double s)“

Diese Methode gibt die Zeit zurück, welche benötigt wird um eine Strecke  $s$  mit einer konstanten Geschwindigkeit  $v$  zu durchfahren.

$$v = \frac{s}{t}$$

Umgeformt nach  $t$  gilt:

$$t = \frac{s}{v}$$

#### 2.4.3.2 Die Methode „zeitZumBeschleunigen(double v1, double v0, double a)“

Diese Methode gibt den Zeitbedarf, welcher zum Beschleunigen von einer Anfangsgeschwindigkeit  $v_0$  auf eine Endgeschwindigkeit  $v_1$  mit einer Beschleunigung  $a$  benötigt wird.

$$v_1 = v_0 + a \cdot t$$

Durch Umformen nach t erhält man:

$$t = \frac{v_1 - v_0}{a}$$

#### **2.4.3.3 Die Methode „geschwNachBeschleunigenUeberStrecke(double v<sub>0</sub>, double s, double a)“**

Diese Methode gibt die Endgeschwindigkeit v<sub>1</sub> nach einem Beschleunigen mit der Beschleunigung a über eine Strecke s von einer Anfangsgeschwindigkeit v<sub>0</sub> zurück.

$$2as = v_1^2 - v_0^2$$

Durch Umformen nach v<sub>1</sub> erhält man:

$$v_1^2 = 2as + v_0^2$$

Beziehungsweise, da man davon ausgehen kann dass die Endgeschwindigkeit v<sub>1</sub> positiv sein muss gilt:

$$v_1 = \sqrt{2as + v_0^2}$$

#### **2.4.3.4 Die Methode „streckeZumBeschleunigen(double v<sub>1</sub>, double v<sub>0</sub>, double a)“**

Diese Methode gibt die Streckenlänge s zurück, welche benötigt wird um von der Anfangsgeschwindigkeit v<sub>0</sub> auf die Endgeschwindigkeit v<sub>1</sub> mit der Beschleunigung a zu beschleunigen.

$$2as = v_1^2 - v_0^2$$

Durch Umformen nach s erhält man:

$$s = \frac{v_1^2 - v_0^2}{2a}$$

#### **2.4.3.5 Die Methode „beschleunigenUeberStrecke(double v<sub>1</sub>, double v<sub>0</sub>, double s)“**

Diese Methode gibt die benötigte Beschleunigung a, welche zum Beschleunigen von der Anfangsgeschwindigkeit v<sub>0</sub> auf die Endgeschwindigkeit v<sub>1</sub> über die Strecke s benötigt wird, zurück.

$$2as = v_1^2 - v_0^2$$

Durch Umformen nach a erhält man:

$$a = \frac{v_1^2 - v_0^2}{2s}$$

## 2.4.4 Die Implementierung

Für die verschiedenen Fälle gibt es jetzt verschiedene Vorgehensweisen zum Berechnen des Zeitbedarfs zum Durchqueren der Strecke:

1. Der Autofahrer kann die Strecke optimal ausnutzen.  
In diesem Fall berechnen mithilfe der Methode „*streckeZumBeschleunigen*“ die Strecken welche der Fahrer zum Beschleunigen auf die Höchstgeschwindigkeit und zum Abbremsen auf die Endgeschwindigkeit braucht. Dadurch können wir die Strecke berechnen, welche der Fahrer mit konstanter Geschwindigkeit zurücklegt. Nun können wir mit den Methoden „*zeitZumBeschleunigen*“ und „*zeitZumKonstantFahren*“ die Gesamtzeit berechnen, welche der Fahrer benötigt.
2. Um das Auto auf die maximal zulässige Höchstgeschwindigkeit zu beschleunigen und später wieder abzubremsen ist die Straßenlänge zu kurz.  
In dem Fall benutzen wir die in Kapitel 2.4.2.1 „Herleiten der Formel für den 2. Fall,“ beschriebene Formeln zum Berechnen der Gesamtzeit.
3. Die Strecke ist so kurz, dass der Autofahrer beinahe die gesamte Strecke zum Bremsen vor der Kreuzung benötigt.  
Hier wird mit Hilfe der Methode „*streckeZumBeschleunigen*“ die Strecke berechnet, welche der Fahrer zum Abbremsen benötigt. Damit errechnen wir die Strecke welche mit einer konstanten Geschwindigkeit durchfahren wird. So können wir auch wieder mit den Methoden „*zeitZumBeschleunigen*“ und „*zeitZumKonstantFahren*“ die Gesamtzeit berechnen, welche der Fahrer benötigt.
4. Der Fahrer benutzt beinahe die gesamte Strecke zum Beschleunigen auf die maximale Höchstgeschwindigkeit des nächsten Segments.  
Dies wird genauso wie der 3. Fall berechnet. Die einzigen Unterschiede sind, dass der Fahrer beschleunigt anstatt zu bremsen und dass er zuerst beschleunigt, bevor er mit der konstanten Geschwindigkeit weiterfährt.
5. Die Strecke ist zu kurz für einen vollen Geschwindigkeitsausgleich zum nächsten Segment.  
In diesem Fall wird mit der Methode „*geschwNachBeschleunigenUeberStrecke*“ die Endgeschwindigkeit des Fahrzeugs ermittelt. Mit ihrer Hilfe kann über die Methode „*zeitZumBeschleunigen*“ die Zeit zum Durchqueren der Strecke berechnet werden.

Die Implementierung des ganzen kann als Pseudocode im Anhang G – „Pseudocode zum Berechnen des Zeitbedarfs zum Befahren einer Strecke“ nochmals genau betrachtet werden.

## 3 Die Simulation im Detail

In diesem Kapitel werde ich genauer auf die Modellierung der Simulationskomponenten eingehen.

### 3.1 Allgemeiner Simulationsaufbau

Dieser Abschnitt beschäftigt sich mit den allgemeinen Konzepten, auf welche diese Simulation basiert. Dabei wird kurz darauf eingegangen, welche grundlegende Arten von Komponenten es gibt, wie diese initialisiert werden und wie die einzelnen Komponenten ineinander greifen.

#### 3.1.1 Grundprinzipien der ereignisorientierten Simulation

In meiner Simulation orientiere ich mich an den Grundsätzen der ereignisorientierten Simulation. Dies bedeutet: es gibt Ereignisse, welche chronologisch nach einer globalen Ereignisliste (auch Ereigniskalender genannt) abgearbeitet werden. Weiterhin gibt es Ereignisroutinen (im weiteren Ereignis-Handler genannt) welche die verschiedenen Ereignisse abarbeiten, wobei eventuell auch neue Ereignisse entstehen können. Diese Ereignisse und die Zustände der einzelnen Simulationsobjekte ergeben das klassische Modell einer ereignisorientierten Simulation.

##### 3.1.1.1 Das Ereignis

Die Ereignisse werden in dieser Simulation als eigenständige Objekte behandelt. Dazu gibt es die Klasse „*Ereignis*“. Dieses Ereignisobjekt enthält alle für das Abarbeiten eines Ereignisses innerhalb der Simulation wichtige Informationen. Diese sind unter anderem:

- Der Zeitpunkt an welchem das Ereignis eintritt.  
Zum Beispiel: Eintritt zum Simulationszeitpunkt 27,4523.
- Die Art des Ereignisses.  
Dies kann z.B. ein Ereignis „*Ankunft*“ oder ein „*Bedien Ende*“ sein.
- Den Simulationsabschnitt an welchem das Ereignis auftritt.
- Das Fahrzeug bei welchem das Ereignis eintritt.

##### 3.1.1.2 Die Ereignisliste

Alle Ereignisse werden in einer globalen Ereignisliste abgespeichert. Diese Ereignisliste ist damit eine der zentralen Komponenten der Simulation. Alle neu auftretende Ereignisse werden in dieser Liste, nach ihrem Eintrittzeitpunkt aufsteigend sortiert, gespeichert.

Bei jedem Schritt der Simulation wird jetzt das erste Ereignis aus dieser Liste geholt und an den im Ereignis hinterlegten Simulationsabschnitt weitergegeben. Der Simulationsabschnitt übernimmt dann mit seiner integrierten Logik die Behandlung dieses Ereignisses. Der Vorteil dieser Vorgehensweise ist, dass die grundlegende Logik auch bei neuen Ereignistypen nicht verändert werden muss.

### **3.1.2 Die grundlegende Komponentendefinition als Simulationsabschnitt.**

Da ein Straßennetz aus vielerlei Komponenten besteht, die fast beliebig zusammen kombinierbar sind, muss dieses Prinzip auch innerhalb der Simulation gelten. Daher sind alle Komponenten der Verkehrssimulation, wie z.B. Straßen, Kreuzungen, Quellen und Senken, durch die Schnittstelle „*Simulationsabschnitt*“ definiert. Diese Schnittstelle umfasst alle grundlegenden Methoden über welche ein Simulationsabschnitt verfügen muss (siehe Abbildung 3.1).

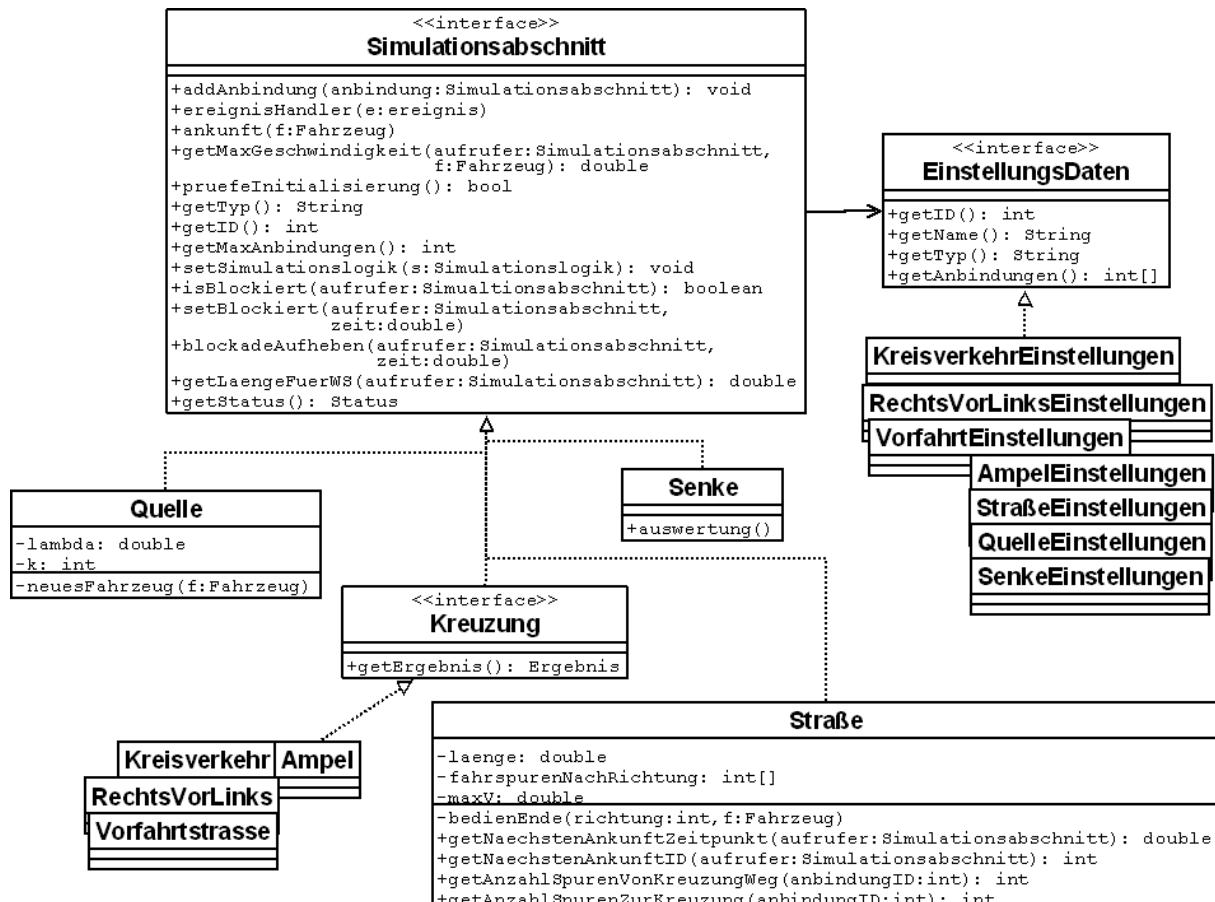


Abbildung 3.1: Vereinfachtes Klassendiagramm zum Simulationsaufbau

Diese Aufgaben sind unter anderem:

- Das Ermöglichen der Anbindung anderer Komponenten an vorgegebene Verbindungsstellen. Dies wird benötigt um die einzelnen Komponenten miteinander verbinden zu können.
  - Eine Ereignisroutine - auch Ereignis-Handler genannt - welche die Ereignisse für diese Komponente abarbeitet. Wie oben bereits erwähnt kommen diese Ereignisse von der globalen Simulationslogik, welche die Ereignisse nicht selbst abarbeiten kann und daher diese an die einzelnen Komponenten weiter gibt.
  - Eine Voranmeldung für Fahrzeuge, welche später von anderen Simulationsabschnitten an diesen Simulationsabschnitt übergeben werden. Bei dieser Voranmeldung wird auch gleichzeitig die maximale Geschwindigkeit mit welcher das Fahrzeug in den nächsten Abschnitt einfahren darf zurückgegeben.
  - Die Übergabe der individuellen Einstellungen für diese Komponente in Form einer Einstellungsklasse.

- Die Abfrage des aktuellen Status der Komponente innerhalb einer Animationsphase.
- Die Übergabe einer Referenz der Simulationslogik. Diese wird benötigt, damit die einzelnen Komponenten neue Ereignisse in die Ereignisliste eintragen können.
- Die Implementierung eines „Ankunft“-Ereignisses für neu ankommende Fahrzeuge. Dies wird benötigt um eine einheitliche Schnittstelle zur Übergabe von Fahrzeugen von einer Komponente zur nächsten Komponente der Simulation zu ermöglichen.
- Eine Möglichkeit zur Kontrolle der fehlerfreien Initialisierung der Komponente. Bei dieser Kontrolle wird auch eine umfangreiche Logik-Prüfung der Einstellungen vorgenommen.
- Zuletzt müssen auch noch Methoden für die Realisierung eines globalen Blockadesystems eingebaut werden. Dies wird später zur realistischen Simulation von Stau und den daraus folgenden Rückstaus benötigt.

### 3.1.3 Das globale Blockadesystem

Unter dem Begriff globales Blockadesystem verbirgt sich die Logik für das Simulieren von Staus und ihren Folgen. So kommt es oft vor, dass ein Stau nicht lokal auf einen Straßenabschnitt begrenzt ist, sondern sich auch auf die Kreuzungen und die dort einmündenden Straßen auswirkt. Dieses Phänomen wird im weiteren Verlauf von mir als Rückstau bezeichnet. Dieser Rückstau muss natürlich auch in der Simulation beachtet werden. Da die Simulation allerdings über eine Vielzahl von Komponenten realisiert ist, bleibt die Frage, wie man so ein Blockadesystem in der Simulation einführen kann.

Um dieses Rückstauen zu simulieren wurden von mir drei Methoden entworfen, welche dieses Blockadesystem ermöglichen. So gibt es die Methoden „*setBlockiert*“ und „*blockadeAufheben*“ zum Setzen und Aufheben einer Blockadesituation. Weiterhin kann über die Methode „*isBlockiert*“ abgefragt werden, ob in einem anderen Simulationsabschnitt eine Blockade vorliegt.

#### 3.1.3.1 Beispiel für einen Rückstau

In Abbildung 3.2 kann man einen Stau (angezeigt durch das rote Einfärben der Straße) entlang einer Straße erkennen. In diesem Fall ist es ein Stau auf der Bahnhofstraße. Diese Blockade wirkt sich in Abbildung 3.3 durch einen Rückstau auch auf die vorgelagerte Weimarstraße aus, wodurch sich auch dort ein Stau bildet. Dieser entsteht, weil die Fahrbahn der Bahnhofstraße durch wartende Fahrzeuge bereits komplett blockiert wird und deshalb die Fahrzeuge an der Vorfahrt-Kreuzung nicht von der Weimarstraße in die Bahnhofstraße einfahren können.

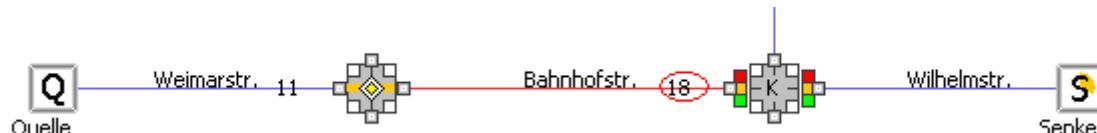


Abbildung 3.2: Beispiel für eine Blockade

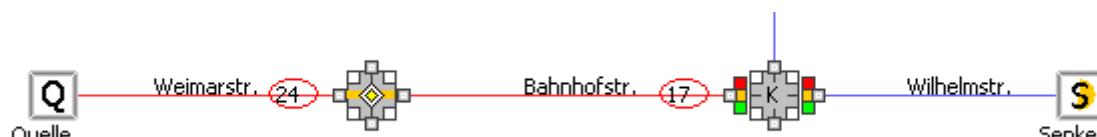


Abbildung 3.3: Beispiel für einen Rückstau

### **3.1.3.2 Funktionsweise des Blockadesystems**

Kommen wir nun zur Funktionsweise des Blockadesystems. Wie in Abbildung 3.2 zu sehen ist, beginnt der Stau bei der Ampel zwischen der Bahnhof- und der Wilhelmstraße. Diese Ampel schafft es nicht genügend Fahrzeuge aus der Bahnhofstraße abzufertigen, wodurch die Fahrzeuge sich in der Bahnhofstraße stauen. Dies geschieht so lange, bis diese durch die wartenden Fahrzeuge vollständig blockiert ist.

Dieses Blockieren wird dabei durch die Warteschlange der Ampelkreuzung initialisiert. Wenn die Warteschlange ihre maximale Ausdehnung erreicht (welcher der Straßenlänge entspricht), wird von der Warteschlangenlogik die angebundene Straße blockiert (siehe dazu Kapitel 3.4.1).

Nachdem diese Straße blockiert ist, ist es für Fahrzeuge nicht mehr möglich aus der Weimarstraße in die Bahnhofstraße einzufahren. Dies wird nämlich von der Kreuzungslogik verhindert. Dadurch bildet sich auch dort ein Stau, welcher auf Dauer (wie in Abbildung 3.3 zu sehen ist) auch diese Straße komplett blockiert. Dies geschieht sobald die Warteschlange, welche für die Weimarstraße verantwortlich ist, ihre maximale Ausdehnung erreicht hat.

### **3.1.3.3 Einschränkung des Blockadesystems**

Als Einschränkung für das globale Blockadesystem gilt, dass Fahrzeuge, welche noch nicht bei der nächsten Kreuzung in der Warteschlange stehen (also momentan noch auf der Straße unterwegs sind), nicht zur Länge der Warteschlange zählen. Daher kommt es zu dem Phänomen, dass obwohl eine Straße bereits blockiert ist, immer noch Fahrzeuge an der Warteschlange einer Kreuzung ankommen. Dies liegt daran, dass die Warteschlangen keinerlei Kenntnisse über die Fahrzeuge auf der angebundenen Straße haben. Die Warteschlangen kennen nur die Fahrzeuge, welche bereits an der Kreuzung angekommen sind und sich dort innerhalb der Warteschlange befinden. Eine Lösung dieses Problems wäre möglich, wenn die Warteschlange eine Möglichkeit hätte die Gesamtlänge und Anzahl der Fahrzeuge, welche auf sich auf der Straße befinden, abzufragen. Leider konnte ich diese Lösung aufgrund von Zeitmangel nicht in meiner Simulationssoftware realisieren.

## **3.2 Die grundlegenden Simulationskomponenten**

In diesem Abschnitt werden die grundlegenden Komponenten der Simulation betrachtet.

### **3.2.1 Die Senke**

Die Senke (siehe Abbildung 3.4) ist wohl das einfachste Element der Simulation. Fahrzeuge die in der Senke ankommen, fahren aus der Simulation heraus und spielen weiter keine Rolle mehr. Ankommende Fahrzeuge werden durch die Geschwindigkeit, mit welcher sich der orange farbige Kreis füllt, symbolisiert (vergleiche Abbildung 3.5). Dies geschieht umso schneller, je mehr Fahrzeuge pro Zeiteinheit die Senke erreichen.

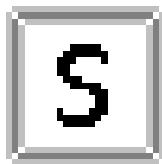


Abbildung 3.4:  
Die Senke in der  
GUI

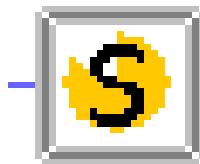


Abbildung 3.5: Fahrzeuge  
erreichen die Senke

Als einzige Einstellungsmöglichkeiten (siehe Abbildung 3.6) hat die Senke ihren Namen und die maximale Geschwindigkeit, mit welcher die Fahrzeuge in die Senke einfahren dürfen.



Abbildung 3.6: Der Einstellungsdialog der Senke

### 3.2.2 Die Quelle

Die Quelle (siehe Abbildung 3.7) ist wohl das wichtigste Element der gesamten Simulation. In der Quelle werden die Fahrzeuge erstellt, welche später die Simulation bevölkern. Sie entscheidet, zu welchem Simulationszeitpunkt, was für ein Fahrzeug in die Simulation eintritt. Aufgrund dieser Gegebenheit, sind die Einstellungen für die Quellen (siehe Abbildung 3.8) eine der wichtigsten Konfigurationsmöglichkeiten der gesamten Simulation. Als wichtige Einstellungen gibt es neben dem Erwartungswert  $E$  und der Stufenzahl  $k$ , welche zum Berechnen des Abstands der Ankunftsereignisse benötigt werden und deren Sinn weiter unten noch genauer beschrieben wird, noch die Ankunftsgeschwindigkeit der Fahrzeuge in km/h und der prozentuale Anteil der LKWs am gesamten Verkehrsaufkommen. Mit Hilfe der Ankunftsgeschwindigkeit wird definiert, mit welcher Geschwindigkeit die Fahrzeuge an der Quelle in das System einfahren. Weiterhin wird mit der „Anzahl der LKWs am Verkehr in Prozent“ angegeben, wie viele Prozent der ankommenden Fahrzeuge LKWs sind.

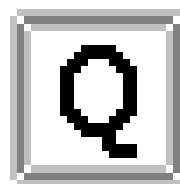


Abbildung 3.7: Die  
Quelle in der GUI

Zu beachten gilt, dass innerhalb meiner Simulation jede Quelle auch gleichzeitig die Funktionalität einer Senke übernimmt. Dies hat den Vorteil, dass bei normalen Straßen nicht eine Quelle für die in die Simulation einfahrenden Fahrzeuge und gleichzeitig eine Senke für die aus der Simulation ausfahrenden Fahrzeuge eingerichtet werden muss.

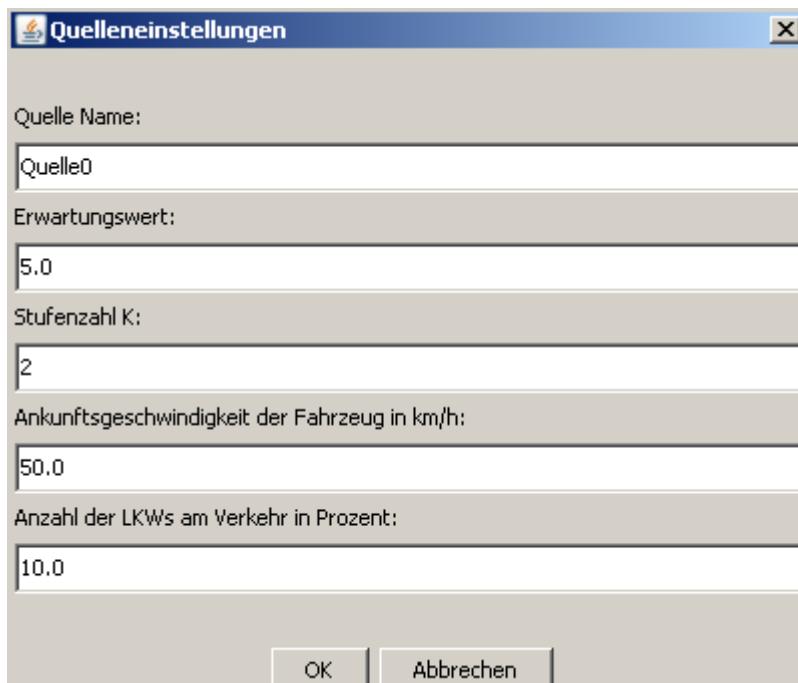


Abbildung 3.8: Der Einstellungsdialog der Quelle

### 3.2.2.1 Die Ankunftsverteilung

Zum Erzeugen der Ankunftsabstände zwischen den einzelnen Fahrzeugen wird die Erlang-Verteilung verwendet. Der Vorteil der Erlang-Verteilung ist, dass diese je nach Eingabe der Stufenzahl eine andere Verteilung nachbilden kann. So entspricht die Erlang-Verteilung bei einer Stufenzahl von 1 der Exponentialverteilung während bei hohen Stufenzahlen, sie sich der Normalverteilung annähert (siehe dazu Abbildung 3.9, wobei zu beachten ist, dass die Stufenzahl k in diesem Bild als n bezeichnet wird).

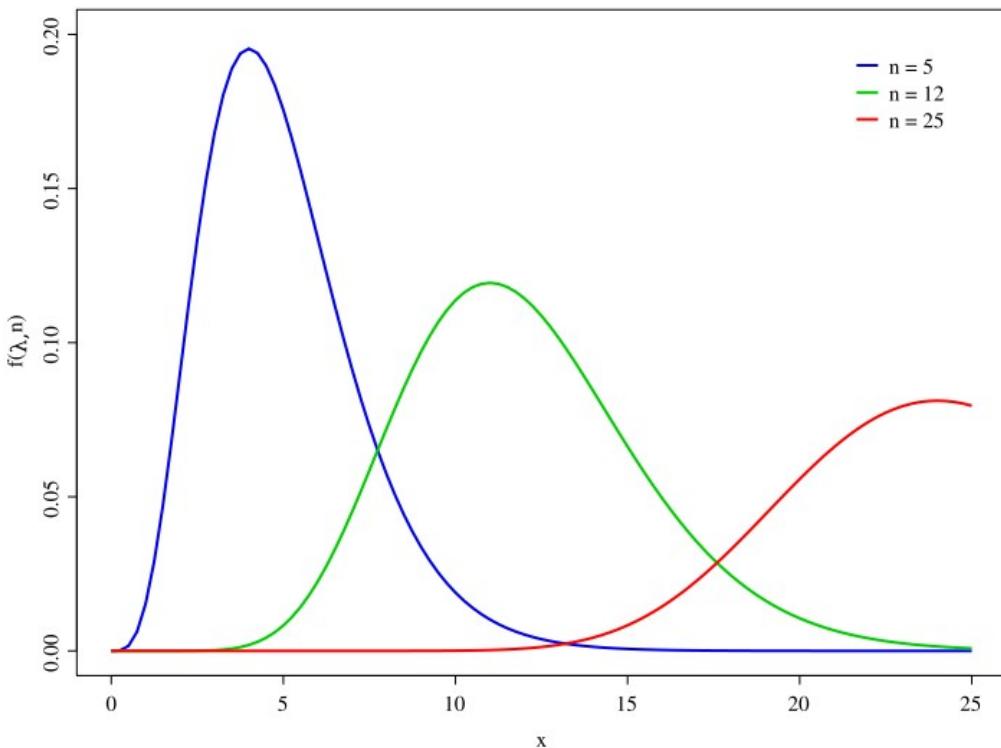


Abbildung 3.9: Die Dichtefunktion der Erlang-Verteilung. [WIKI02@]

### 3.2.2.2 Die Erlang-Verteilung

Die Erlang-Verteilung lautet:

$$F(x) = \frac{\lambda^k}{(k-1)!} \int t^{k-1} e^{-\lambda t} dt \quad \text{für } x \geq 0$$

$$F(x) = 0 \quad \text{für } x < 0$$

Es gilt folgende Dichtefunktion für die Erlang-Verteilung:

$$f(x) = \frac{(\lambda \cdot x)^{k-1}}{(k-1)!} \lambda e^{-\lambda x} \quad \text{für } x \geq 0$$

$$f(x) = 0 \quad \text{für } x < 0$$

Dabei gilt, dass der Erwartungswert  $E(x)$  nach folgender Formel aus  $\lambda$  und der Stufenzahl  $k$  berechnet wird:

$$E(x) = \frac{k}{\lambda}$$

Um die Bedienung der Simulation möglichst einfach zu gestalten, wird bei der Eingabe im Einstellungsdialog (Abbildung 3.8) der Erwartungswert  $E(x)$  verlangt, woraus später das  $\lambda$  berechnet werden kann. Dies geschieht durch Umformen obiger Formel:

$$\lambda = \frac{k}{E(x)}$$

Durch die Eingabe eines Erwartungswerts von 10 Sekunden und der Stufenzahl  $k = 2$ , erhalten wir folgende, in Abbildung 3.10 zu sehende Dichtverteilung. Mithilfe einer geringen

Stufenzahl können wir also dem Phänomen der normalerweise immer in Gruppen ankommenden Fahrzeuge gerecht werden. So nimmt die Wahrscheinlichkeit, dass ein weiteres Fahrzeug demnächst ankommt ab, wenn mehr Zeit seit dem letzten Ankunftsereignis verstrichen ist.

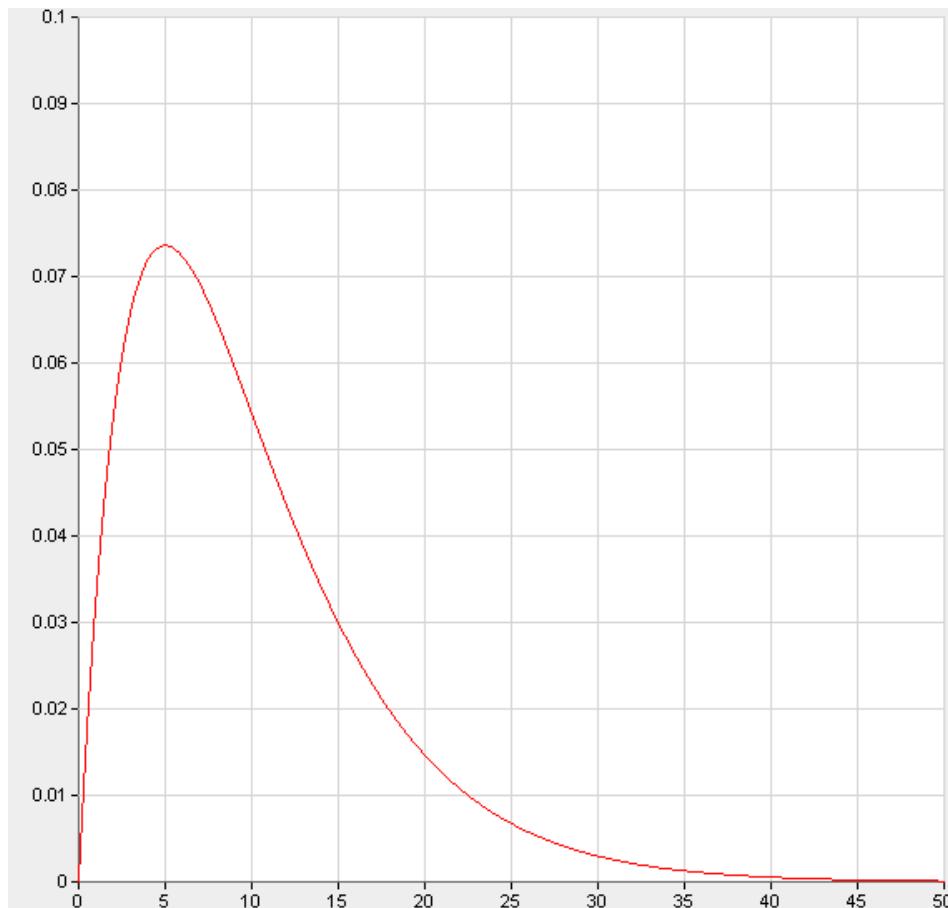


Abbildung 3.10: Beispiel der Dichtefunktion einer Erlang-Verteilung mit einem Erwartungswert von 10 Sekunden und  $k=2$ , erstellt mit dem Online-Plotter von [WAZO@].

### 3.2.3 Die Straße

Die Straße ist eine weitere zentrale Komponente der Simulation. Straßen verbinden, wie dies auch in der Realität der Fall ist, alle anderen Komponenten miteinander. Beim Bau einer Straße können drei verschiedene Straßentypen (siehe Abbildung 3.11) ausgewählt werden. Es gibt die Bundesstraße (Abbildung 3.12), die Hauptstraße (Abbildung 3.13) und die Nebenstraße (Abbildung 3.14). Diese verschiedenen Straßentypen haben zwei Hauptaufgaben.

1. Sie vereinfachen die Orientierung innerhalb der graphischen Abbildung der Simulation, indem sie für den Benutzer verschiedene Straßentypen visualisieren.
2. Sie vereinfachen das Erstellen einer einfachen Simulation, indem sie standardmäßig einen unterschiedlichen Fahrzeudurchsatz (Anzahl der Fahrzeuge, welche normalerweise pro Minute die Straße passieren) hinterlegt haben. Diesen Fahrzeudurchsatz wird später für die Berechnung der automatischen Abbiegewahrscheinlichkeiten gebraucht.

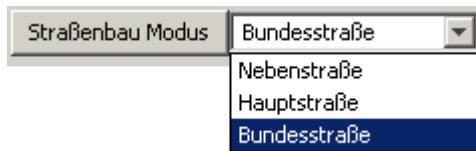


Abbildung 3.11: Auswahl des Straßentyps



Abbildung 3.12: Die Bundesstraße



Abbildung 3.13: Die Hauptstraße



Abbildung 3.14: Die Nebenstraße

In Abbildung 3.15 ist der Einstellungsdialog einer Straße zu sehen. Es kann sowohl die Länge der Straße, wie auch die erlaubte Höchstgeschwindigkeit angegeben werden. Als Besonderheiten, neben dem Fahrzeugdurchsatz welcher oben schon beschrieben wurde, sind die Anzahl der Fahrspuren je nach Fahrtrichtung zu erkennen. Mit Hilfe dieser Spuranzahl können sowohl mehrspurige Straßen, wie auch Einbahnstraßen definiert werden. Für eine Einbahnstraße wird einfach Anzahl der Fahrspuren der Gegenrichtung auf 0 gesetzt. Für mehrspurige Straßen wird einfach die Spuranzahl entsprechend erhöht.

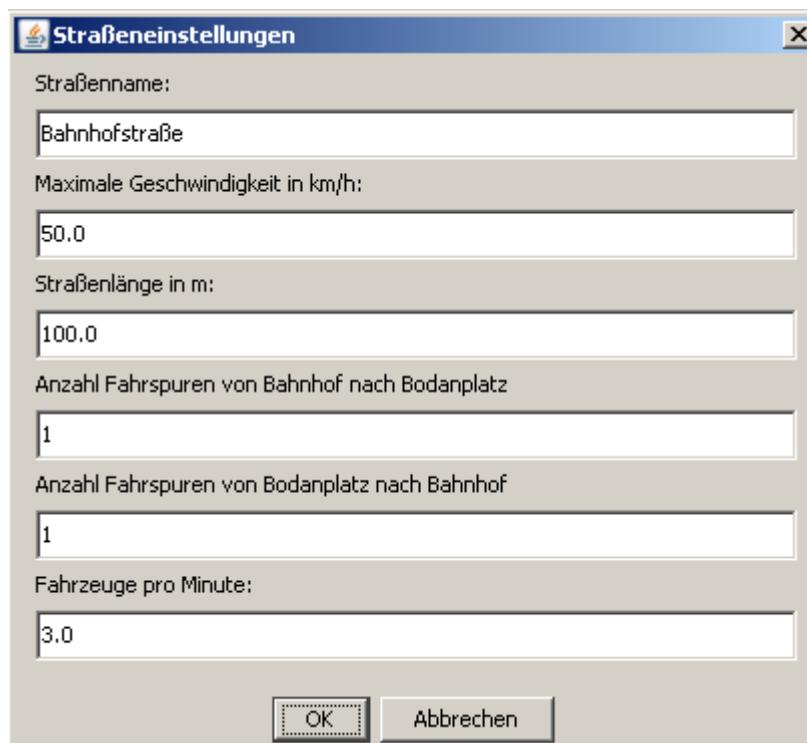


Abbildung 3.15: Der Einstellungsdialog der Straße

### 3.2.3.1 Das Modell

Das Modell der Straße entspricht dem der klassischen, ereignisorientierten Simulation. Die Straße wird also nur als ein Segment abgebildet, bei welchem es ein Ankunft-Ereignis und ein

Bedienende-Ereignis gibt. Zwischen diesen beiden Ereignissen finden keinerlei weitere Ereignisse beim Befahren der Straße statt (siehe dazu Anhang B – Pseudocode Straße).

Die Zeit zum Befahren der Straße hängt von der Länge der Straße, der maximalen Höchstgeschwindigkeit des aktuellen und des nachfolgenden Segments, sowie den Eigenschaften des Fahrzeuges ab. Wie diese Eigenschaften genutzt werden um den Zeitbedarf zu ermitteln, welche ein Fahrzeug zum Durchqueren der Straße benötigt, haben wir in Kapitel 2.4 bereits ausführlich behandelt.

Beim Berechnen des Ankunftszeitpunkts gilt es weiterhin noch zu beachten, dass bei einer einspurigen Straße das Überholen der vorausfahrenden Fahrzeuge nicht erlaubt ist. Weiterhin gilt es einen Sicherheitsabstand einzuhalten. Für diesen Sicherheitsabstand gilt Innerorts die Faustregel: eine Sekunde Abstand zum Vordermann. Auch dies wird von der Simulation beachtet.

Zu beachten bei mehrspurigen Straßen ist allerdings noch, dass aufgrund der Beschränkungen des Modells diese mehrspurigen Straßen nur den Vorteil haben, dass das Überholen von anderen Fahrzeugen erlaubt ist und die maximale Warteschlangenlänge durch sie erhöht wird (siehe dazu Kapitel 3.4.1).

Im Weiteren hat die angebundene Straße verschiedene Einflüsse auf die Kreuzungen, auf welche wir erst beim genaueren Betrachten der Kreuzungen eingehen wollen.

## 3.3 Die Fahrzeuge

In diesem Kapitel wollen wir die Fahrzeuge genauer betrachten. Zuerst betrachten wir, welche realen Eigenschaften ein Auto besitzt und welche dieser Eigenschaften für diese Simulation interessant sind. Danach betrachten wir, welche Einschränkungen ich für mein Modell annehme und wie eine einfache Differenzierung zwischen PKWs und LKWs erstellt werden kann. Schließlich betrachten wir noch, wie spezifische Informationen für die einzelnen Kreuzungen bei den Fahrzeugen gespeichert werden.

### 3.3.1 Was macht ein reales Fahrzeug aus?

Die Eigenschaften eines realen Fahrzeugs sind:

- Seine physikalische Eigenschaften:  
Die Länge, Breite, Höhe des Fahrzeugs in Meter und das Gewicht in Tonnen.
- Der Motor:  
Die Art des Motors, seine Zylinderzahl, Hubraum, Getriebe, die maximale Leistung in PS, Drehmoment, Fahrzeughöchstgeschwindigkeit, Kraftstoffverbrauch, ...
- Die Äußerlichkeiten:  
wie z.B. die Farbe, der Hersteller, das Fahrzeugmodell, ...
- Der Fahrer:  
Jeder Fahrer fährt ein Fahrzeug anders. Viele wichtige Fahreigenschaften, wie z.B. die Beschleunigen, die Kurvengeschwindigkeit und das Bremsverhalten, sind vom Fahrer abhängig.
- Innere Werte:

Natürlich hat jedes Fahrzeug auch innere Werte wie z.B. das Ladevermögen, die Anzahl der Sitzplätze und verschiedene Ausstattungsmerkmale wie z.B. ein Navigationssystem.

- Die Sicherheitsmerkmale:  
Jedes moderne Fahrzeug verfügt über Sicherheitsmerkmale wie z.B. ABS, ESP, ... welche dem Fahrer in Krisensituationen hilfreich zur Seite stehen.
- Eine aktuelle Geschwindigkeit:  
Jedes Fahrzeug hat zu jedem Zeitpunkt eine bestimmte Geschwindigkeit, auch wenn diese momentan gleich Null sein sollte.
- Einen Standort:  
Jedes Fahrzeug hat immer einen Standort, an welchem es sich befindet.

### **3.3.2 Welche dieser Punkte sind für die Simulation wichtig?**

Wichtige Eigenschaften eines Fahrzeugs im Modell:

- Jedes Fahrzeug hat eine individuelle Länge.
- Jedes Fahrzeug hat durch den individuellen Fahrstil des Fahrers eine individuelle Beschleunigung (dargestellt im Modell als Beschleunigung in Meter pro Quadratsekunde).  
Diese Beschleunigung hängt ab vom Fahrstil des Fahrers, der Leistung des Motors, dem Gewicht des Fahrzeugs und den Straßenverhältnissen.
- Jedes Fahrzeug hat durch den individuellen Fahrstil des Fahrers eine individuelle Bremsbeschleunigung (dargestellt im Modell als Bremsbeschleunigung in Meter pro Quadratsekunde).  
Die Bremsbeschleunigung hängt u.a. ab von den Bremsen, dem Fahrzeuggewicht, dem Fahrstil des Fahrers und den Straßenverhältnissen.
- Weiterhin bevorzugt jeder Fahrer in einer Kurve eine individuelle horizontale Beschleunigung (Zentrifugalkraft). Das ist die Kraft, die einen bei der Fahrt durch eine Kurve nach außen drückt.
- Jedes Fahrzeug hat zu jedem Ereigniszeitpunkt eine bestimmte Geschwindigkeit. Mit Hilfe dieser lassen sich verschiedene Situationen innerhalb der Simulation realistischer abbilden. So gibt es z.B. beim Einfahren in eine Kreuzung einen Unterschied zwischen dem Einfahren mit voller Geschwindigkeit und dem Anfahren aus der Warteschlange.
- Jedes Fahrzeug hat einen Simulationsabschnitt als Standort, an welchem sich das Fahrzeug aktuell befindet.

### **3.3.3 Welchen Einschränkungen unterliegt ein Fahrzeug innerhalb des Modells?**

Ein Fahrzeug im Modell unterliegt folgenden Einschränkungen:

- Es wird angenommen, dass ein Fahrzeug immer mit der gleichen Beschleunigungskraft beschleunigt – es gibt z.B. also keine Verzögerungen durch die Gangschaltung.
- Der quadratisch zur Geschwindigkeit wachsende Luftwiderstand wird ignoriert, da dieser bei Geschwindigkeiten bis 50 km/h innerhalb der Ortschaft eine untergeordnete Rolle spielt.
- Es wird angenommen, dass ein Autofahrer sich immer gleich verhält. Das heißt unter anderem, dass es keine Unterschiede beim Beschleunigen, Bremsen und der bevorzugten Kurvengeschwindigkeit gibt.

- Wir beachten keine Einschränkungen des Straßenverlaufs, wie z.B. für LKWs zu geringe Durchfahrtshöhen oder zu enge Kurven.
- Es wird angenommen, dass alle Fahrzeuge sich immer an die Geschwindigkeitsbeschränkungen halten und, falls dies möglich ist, auch die Maximalgeschwindigkeit ausnutzen.
- Es gibt keine Reaktionszeit des Fahrers: Durch vorausschauendes Fahren ist ein flüssiger Verkehrsverlauf möglich.

### **3.3.4 Wie wird der Unterschied zwischen einem PKW und einem LKW im Modell realisiert?**

Ein LKW unterscheidet sich im Modell gegenüber einem PKW in folgenden Punkten:

- LKWs haben eine größere Gesamtlänge als PKWs.
- LKWs beschleunigen langsamer und bremsen schwächer als PKWs.
- LKWs fahren langsamer durch eine Kurve. Sie benutzen nur eine geringe horizontale Beschleunigung, weil durch den höheren Schwerpunkt eines LKWs dieser sonst in einer Kurve zu kippen droht.

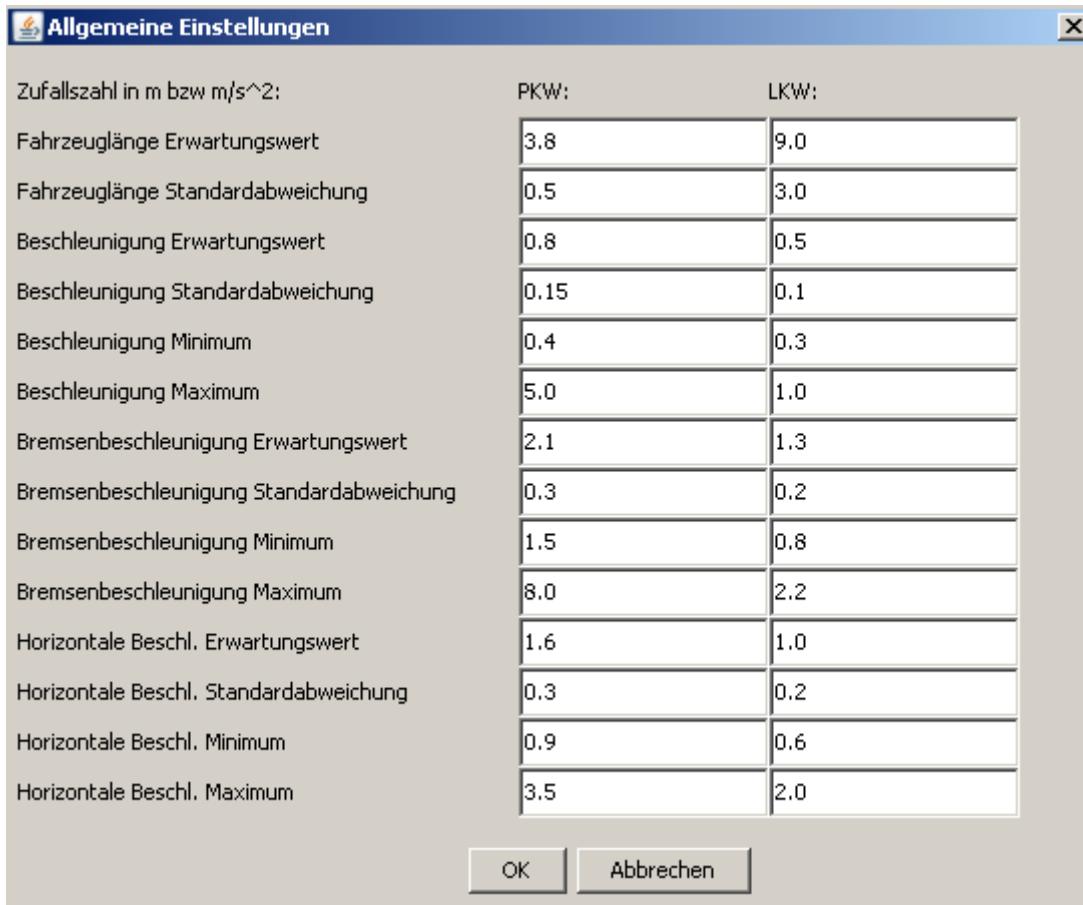
Alle diese Unterschiede werden durch extra Eingabefelder für LKWs beachtet.

### **3.3.5 Die Standardeinstellungen der Simulation**

Da wir jetzt alle Kennziffern die ein Fahrzeug ausmachen definiert haben, geht es nun daran alle diese Zahlen der Simulation bekannt zu machen. Dies geschieht über den Dialog der allgemeinen Einstellungen. In Abbildung 3.16 ist dieser Dialog mit den von mir geschätzten Standardeinstellungen zu sehen. Diese Einstellungen können natürlich auch von Hand angepasst und gespeichert werden.

Zu beachten ist, dass wir von jeder Fahrzeugkennziffer annehmen, dass diese normalverteilt, mit einer Ober- und Untergrenze ist. Somit brauchen wir für jede Kennziffer vier Variablen, welche natürlich jeweils für PKWs und LKWs angegeben werden müssen. Die Ober- und Untergrenzen sind notwendig, da jede Fahrzeugklasse z.B. immer über eine minimale und eine maximale Beschleunigung verfügt, weil diese sonst nicht für den Straßenverkehr zugelassen werden würde, bzw. es technisch überhaupt nicht realisierbar ist. Für jede Kennziffer müssen folgende Variablen eingegeben werden:

- Der Erwartungswert der Kennziffer
- Die Standardabweichung
- Das Minimum
- Das Maximum



The screenshot shows a dialog box titled "Allgemeine Einstellungen". It contains two columns: "PKW:" and "LKW:". The left column lists various parameters with their expected values, while the right column lists the corresponding values for trucks. At the bottom are "OK" and "Abbrechen" buttons.

Zufallszahl in m bzw m/s <sup>2</sup> :	PKW:	LKW:
Fahrzeuglänge Erwartungswert	3,8	9,0
Fahrzeuglänge Standardabweichung	0,5	3,0
Beschleunigung Erwartungswert	0,8	0,5
Beschleunigung Standardabweichung	0,15	0,1
Beschleunigung Minimum	0,4	0,3
Beschleunigung Maximum	5,0	1,0
Bremsenbeschleunigung Erwartungswert	2,1	1,3
Bremsenbeschleunigung Standardabweichung	0,3	0,2
Bremsenbeschleunigung Minimum	1,5	0,8
Bremsenbeschleunigung Maximum	8,0	2,2
Horizontale Beschl. Erwartungswert	1,6	1,0
Horizontale Beschl. Standardabweichung	0,3	0,2
Horizontale Beschl. Minimum	0,9	0,6
Horizontale Beschl. Maximum	3,5	2,0

Abbildung 3.16: Die allgemeinen Einstellungen

Als Minimum und Maximum der Fahrzeulgängen bei PKWs und LKWs gelten folgende Werte:

- Ein PKW hat eine minimale Länge von zwei Metern und eine maximale Länge von 10 Metern. Gesetzlich ist zwar ein PKW mit Anhänger bis zu 18 Metern erlaubt, allerdings entsprechen die sonstigen Fahreigenschaften eines solchen Fahrzeuggespanns eher einem LKW.
- Ein LKW hat eine minimale Länge von 10 Metern und eine gesetzliche maximale Länge von 18 Metern.

### 3.3.6 Wie wurde dies Implementiert?

Kommen wir nun zur Implementierung der Fahrzeuge. Wie in Abbildung 3.17 als Klassendiagramm zu erkennen ist, wurde die Implementierung komplizierter gelöst als dies auf dem ersten Blick sinnvoll wäre. Wir werden in den nun folgenden Abschnitten darauf eingehen, warum so eine Implementierung für unser Modell bedeutende Vorteile hat.

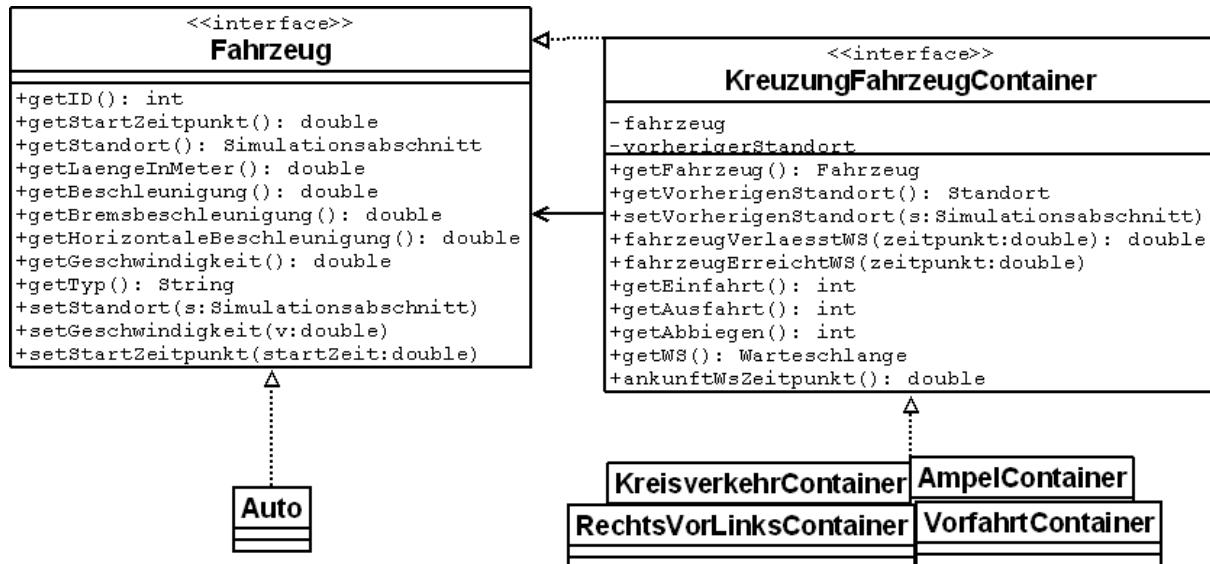


Abbildung 3.17: Das Fahrzeug-Klassendiagramm

### 3.3.6.1 Die Schnittstelle „Fahrzeug“

Die wichtigste Komponente ist wohl eindeutig die Schnittstelle „*Fahrzeug*“. Diese Schnittstelle definiert alle wichtigen Fahrzeugeigenschaften als Methoden. Man kann also sagen, dass diese Schnittstelle einem Fahrzeug innerhalb der Simulation entspricht. Die Implementierung erfolgt in der Klasse „*Auto*“.

### 3.3.6.2 Die Schnittstelle „KreuzungFahrzeugContainer“

Die Schnittstelle „*KreuzungFahrzeugContainer*“ erbt von der Schnittstelle „*Fahrzeug*“. Das heißt: alle Klassen, welche diese Schnittstelle implementieren, entsprechen einem vollwertigen Fahrzeug.

Wenn jetzt ein Fahrzeug eine Kreuzung erreicht, erstellt diese ein Objekt vom Typ des „*KreuzungFahrzeugContainer*“ und packt dort das ankommende Fahrzeug hinein. Dies wird gemacht um dem Fahrzeug wichtige zusätzliche Informationen für die Kreuzung mitzugeben. Diese Informationen werden also wie eine Hülle um ein bestehendes Fahrzeug herum gepackt (vergleiche Abbildung 3.18). Trotz dieser Hülle um das eigentliche Fahrzeug herum kann auf das Fahrzeug wie bisher zugegriffen werden. Nur die Kreuzung, zu welcher die Hülle gehört, weiß von der Existenz dieser und kann auf die dort abgelegten Informationen zugreifen. Schließlich wird beim Verlassen der Kreuzung das Fahrzeug wieder aus seiner Hülle befreit und dem nächsten Simulationsabschnitt übergeben.

### KreuzungsContainer

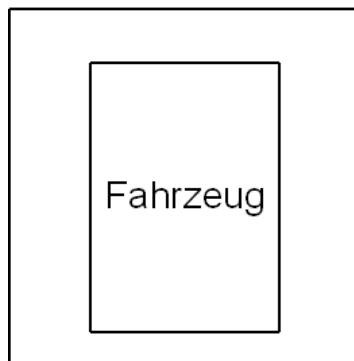


Abbildung 3.18: Kapselung eines Fahrzeugs in einem Container

## 3.4 Allgemeine Kreuzungskonzepte

In diesem Kapitel wollen wir auf die allgemeine Konzepte eingehen, welche alle Kreuzungsarten gemein haben.

### 3.4.1 Das Warteschlangenkonzept

Bei einer Kreuzung ohne spezielle Abbiegespuren entspricht das Warteschlangenkonzept dem der klassischen Warteschlange (vergleiche Abbildung 3.19).

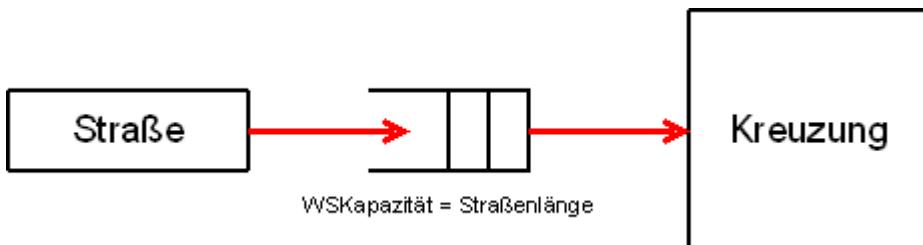


Abbildung 3.19: Einstufige Warteschlange

Wir habe also eine Warteschlange nach dem First-In-First-Out Prinzip. Beschränkt ist diese durch die Länge der angeschlossenen Straße. Ist die ankommende Straße mehrspurig, wird zum Berechnen der maximalen Warteschlangenkapazität die Länge der Straße mit der Anzahl ihrer Fahrspuren multipliziert.

Um zu ermitteln wann die Warteschlange voll ist, wird die Länge jedes ankommenden Fahrzeugs, inklusiv eines Sicherheitsabstands zwischen den Fahrzeugen, zur aktuellen Warteschlangenlänge addiert. Jedes in die Kreuzung einfahrende Fahrzeug reduziert die aktuelle Warteschlangenlänge wieder um seine Fahrzeulgänge und den Sicherheitsabstand.

Wenn jetzt eine Kreuzung allerdings über mehrere Abbiegespuren verfügt reicht das klassische Warteschlangenmodell nicht mehr aus.

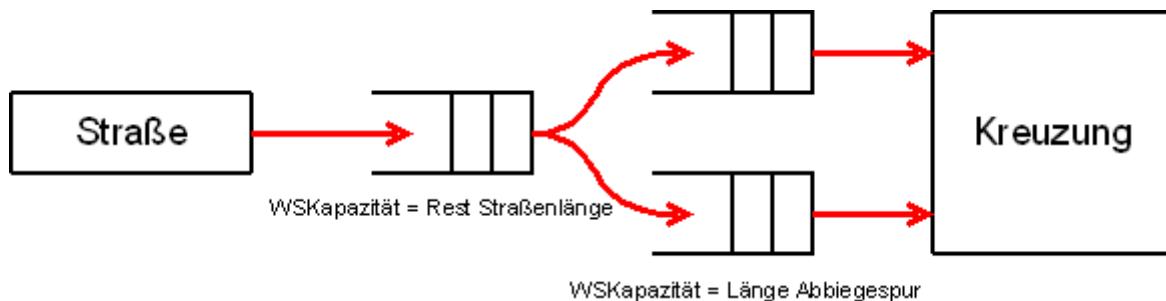


Abbildung 3.20: Mehrstufige Warteschlange

In diesem Fall wird ein mehrstufiges Warteschlangenmodell verwendet. Wie in Abbildung 3.20 ersichtlich ist, wird eine vorgelagerte Warteschlange verwendet. Diese verteilt die ankommenden Fahrzeuge auf die eigentlichen Warteschlangen für jede Abbiegespur. Die normalen Warteschlangen entlang der Abbiegespuren verwenden die in den Einstellungen hinterlegte Länge der Abbiegespur als maximale Warteschlangenkapazität. Wohingegen die vorgelagerte Warteschlange die Straßenlänge abzüglich der Länge der Abbiegespuren als maximale Warteschlangenkapazität benutzt.

### 3.4.1.1 Funktionsweise der mehrstufigen Warteschlange

Die Funktionsweise einer mehrstufigen Warteschlange ist jetzt natürlich etwas komplexer als dies bei einer einfachen Warteschlange der Fall ist. So gelangen alle neu ankommenen Fahrzeuge direkt in die normalen Warteschlangen solang diese ihre maximale Warteschlangenlänge noch nicht erreicht haben und die vorgelagerte Warteschlange leer ist. Wenn eine der beiden Bedingungen nicht erfüllt ist, muss das Fahrzeug in der vorgelagerten Warteschlange warten. Hat die normale Warteschlange wieder ihre maximale Länge unterschritten und das erste Fahrzeug in der vorgelagerten Warteschlange möchte in diese Abbiegespur einbiegen, so wird das Fahrzeug aus der vorgelagerten Warteschlange herausgeholt und in die normale Warteschlange eingereiht. Weiterhin muss dann überprüft werden, ob folgende Fahrzeuge jetzt auch in die Warteschlange ihrer gewünschten Richtung einfahren können.

Durch dieses Arbeitsprinzip kann es also - genauso wie in der Realität - zu einem Rückstau führen, wenn auch nur eine Abbiegespur total überlastet ist.

Abbildung 3.21 zeigt ein Beispiel für die Anzeige der Warteschlangenlänge (in diesem Fall wird die Anzahl Fahrzeuge in der Warteschlange angezeigt). Durch das rote Einfärben der Straße und dem Einkreisen der aktuellen Warteschlangenlänge wird eine volle Warteschlange gekennzeichnet.

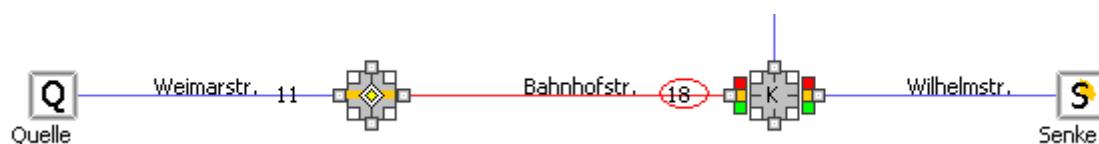


Abbildung 3.21: Beispiel für die Anzeige einer Warteschlangenlänge in der Animation

### 3.4.2 Die Ergebnismethoden

Die Ergebnisse einer Kreuzung werden in Form einer Ergebnisklasse von der Simulation zurückgegeben. Diese Ergebnisklasse enthält für jede Warteschlange der Kreuzung ein Ergebnis in Form einer „*WarteschlangenErgebnis*“-Klasse (vergleiche Abbildung 3.22).

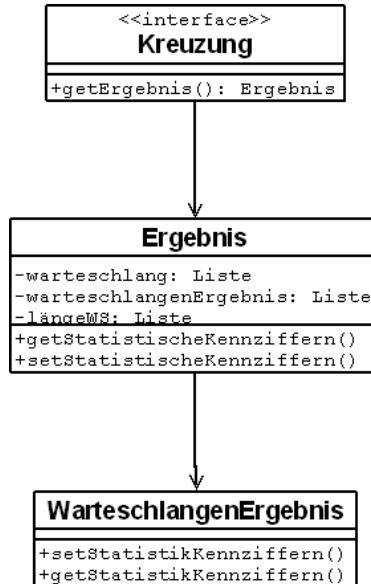


Abbildung 3.22: Die Ergebnisklassen im Klassendiagramm

#### 3.4.2.1 Die Klasse „WarteschlangenErgebnis“

Die Klasse „*WarteschlangenErgebnis*“ enthält alle Ergebniskennziffern, welche von einer Warteschlange für das Gesamtergebnis zur Verfügung gestellt werden. Jede Warteschlange erstellt ihr eigenes „*WarteschlangenErgebnis*“, welches u.a. folgende Kennziffern enthält:

- Die aktuelle Anzahl Fahrzeuge in der Warteschlange.
- Die maximale Anzahl der Fahrzeuge in der Warteschlange.
- Die Gesamtzahl der von dieser Warteschlange abgefertigten Fahrzeuge.
- Die durchschnittliche Anzahl wartender Fahrzeuge.
- Die aktuelle Warteschlangenlänge in Meter.
- Die durchschnittliche Verweilzeit (in Sekunden) eines Fahrzeugs in dieser Warteschlange.
- Die maximale Warteschlangenlänge in Meter.
- Die durchschnittliche Warteschlangenlänge in Meter.
- Den prozentualen Zeitanteil, welche diese Warteschlange blockiert war (d.h. Ihre maximale Kapazität erreicht hatte).

#### 3.4.2.2 Der Inhalt der Ergebnisklasse

Die Methode „*getErgebnis*“ sammelt die Ergebnisse aller Warteschlangen der Kreuzung. Neben den einzelnen Warteschlangen-Ergebnissen, welche u.a. in dem ausführlichen CSV-Bericht (CSV-Export) angegeben werden, befinden sich in der Ergebnis Klasse auch die berechneten Kurzergebnisse, welche über die GUI ausgegeben werden. Dies sind u.a.:

- Die Gesamtlänge der Warteschlangen einer Zufahrtsstraße zu Simulationsende. Also die Länge einer eventuellen vorgelagerten Warteschlange + alle Warteschlangen der Abbiegespuren.

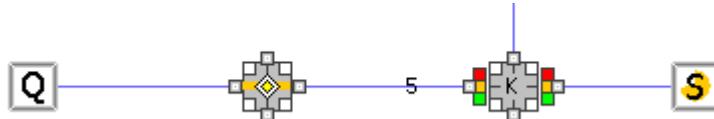


Abbildung 3.23: Die Gesamtlänge der Warteschlangen

- Die durchschnittliche Gesamtlänge der Warteschlangen einer Zufahrtsstraße.



Abbildung 3.24: Die durchschnittliche Gesamtlänge der Warteschlangen

- Die maximal erreichte Warteschlangenlänge (in Fahrzeuge) während der Simulationszeit.



Abbildung 3.25: Die maximal erreichte Warteschlangenlänge

- Die durchschnittliche Gesamtwartezzeit (in Sekunden) der Fahrzeuge aller Warteschlangen einer Zufahrtsstraße.

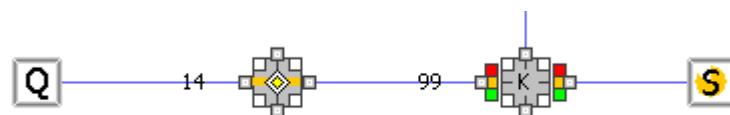


Abbildung 3.26: Die durchschnittliche Gesamtwartezzeit

- Die prozentuale Straßenauslastung der Straße in diese Richtung. Diese ist definiert durch die prozentuale Warteschlangenlänge an der maximal möglichen Warteschlangenlänge einer Richtung.



Abbildung 3.27: Die prozentuale Straßenauslastung

- Der prozentuale Stauanteil der Straße. Dies gibt an zu wie viel Prozent der gesamten Simulationszeit die Straße durch eine Stau (Blockade) blockiert war.

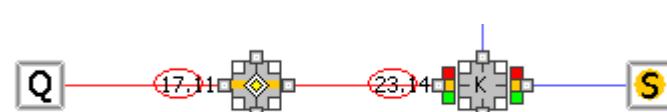


Abbildung 3.28: Der prozentuale Stauanteil

Da die GUI immer nur eine Kennziffer pro Anbindung darstellen kann, bleibt die Frage wie die vielen Kennziffern einer mehrstufigen Warteschlange zu einer einzigen Kennziffer verrechnet werden. Dies erfolgt am Beispiel der Abbildung 3.21 folgendermaßen:

- Die Gesamtlänge der Warteschlange in Fahrzeugen wird errechnet, indem die Anzahl der Fahrzeuge aus der vorgelagerten Warteschlange mit denen der beiden anderen Warteschlangen addiert wird.
- Die Berechnung der durchschnittlichen Warteschlangenlänge erfolgt durch die Addition der einzelnen durchschnittlichen Warteschlangenlängen.
- Genauso funktioniert auch die Berechnung der maximalen Warteschlangenlängen. Wir addieren einfach die maximalen Längen der einzelnen Warteschlangen. Damit erhalten wir zwar eventuell eine etwas zu große maximale Warteschlangenlänge, was dadurch bedingt ist, dass die Warteschlangen einer Richtung nicht unbedingt alle zum selben Zeitpunkt ihre maximale Länge haben. Allerdings lässt sich dieser Fehler nicht so einfach vermeiden und er dürfte auch nur eine untergeordnete Rolle spielen – also zu vernachlässigen sein.
- Die durchschnittliche Wartezeit der Fahrzeuge in der Warteschlange zu berechnen ist etwas komplizierter. Dazu berechnen wir das Produkt der durchschnittlichen Verweilzeit der beiden normalen Warteschlangen mit der Anzahl der von ihnen abgefertigten Fahrzeuge. Diese beiden Produkte addieren wir zusammen und dividieren dies dann durch die insgesamt von beiden Warteschlangen abgefertigten Fahrzeuge. Somit ist die durchschnittliche Wartezeit der Fahrzeuge beider Warteschlangen prozentual richtig zueinander verrechnet. Zu diesem Ergebnis addieren wir schließlich noch die durchschnittliche Verweilzeit der Fahrzeuge in der vorgelagerten Warteschlange hinzu, um die gesamte, durchschnittliche Verweilzeit zu erhalten.
- Zum Berechnen der prozentualen Straßenauslastung errechnen wir für alle drei Warteschlangen erstmals das Produkt aus der prozentualen Straßenauslastung und der für die Warteschlange zu Verfügung stehende Länge in Metern. Diese werden schließlich zusammenaddiert und durch die Summe der insgesamt zur Verfügung stehenden Länge dividiert.
- Zum Erhalten des prozentualen Anteils des Staus an einer Straße während der simulierten Zeit, erfragen wir diese einfach von der vorgelagerten Warteschlange, da diese für das Blockieren der Straßen zuständig ist.

### 3.4.2.3 Beispiel für den Ergebnisexport

Eine ausführliche Ergebnisansicht erreicht man durch die Detailansicht einer Kreuzung in der Ergebnisansicht oder über den Ergebnisexport. In diesem Abschnitt wollen wir kurz aufzeigen wie ein Ergebnisexport als CSV-Datei aussieht und welche Kennziffern dort gespeichert werden. Als Beispiel nehmen wir die in Abbildung 3.29 dargestellte Situation.

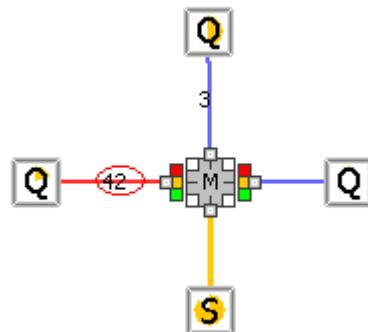


Abbildung 3.29: Beispiel für den Ergebnisexport

Dieses Beispiel liefert die in Tabelle 3.1 dargestellten Werte. Zu sehen ist, dass die Zahlen für vorgelagerte Warteschlangen – oberer Bereich in der Tabelle – von denen der normalen Warteschlangen – unterer Bereich der Tabelle – getrennt ist. Als Spalten treffen wir auf die aus dem vorherigen Abschnitt bereits bekannten verschiedenen Kennziffern, welche allerdings in diesem Fall elementar für jede einzelne Warteschlange ausgegeben werden.

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Vorgelagert-Nord	0	2,45	13	45	13,35	0
1	Vorgelagert-Ost	0	0,03	2	9	0,14	0
2	Vorgelagert-Süd	0	0	0	0	0	0
3	Vorgelagert-West	38	12,46	38	98	85,81	32,51

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung
0	Nord: Rechts/Gerade	2	2,99	7	59	76,95
1	Nord: Links	1	1,08	5	50	24,75
2	Ost: Rechts/Gerade	0	1,08	5	22	26,72
3	Ost: Links	0	0,74	4	20	18,97
4	Süd: Rechts/Gerade	0	0	0	0	0
5	Süd: Links	0	0	0	0	0
6	West: Rechts/Gerade	4	4,03	7	125	106,84
7	West: Links	0	1,84	5	119	50,73

Tabelle 3.1: Beispieldatenebene für den Ergebnisexport einer mittelgroßen Ampel

Noch zu erwähnen ist, dass die prozentuale Straßenauslastung aufgrund des Simulationsmodells (die Warteschlange blockiert erst dann, wenn sie überfüllt ist) in Extremfällen auch mal Werte über 100 Prozent liefern kann.

### 3.4.3 Realisierung der Animation

Über die Methode „*getStatus*“ wird während der Animationsphase die aktuellen Gesamtlänge der Warteschlangen einer Richtung und die Blockadesituation angezeigt. Dieser Status wird nach jedem Simulationsschritt (Ereignis) von dem Simulationsabschnitt, welcher dieses Ereignis abgearbeitet hat, an die GUI übertragen. Diese Ergebnisse werden dann an die entsprechende Visualisierungskomponente in der GUI übergeben. Es werden also nur für den Teil der Animation Daten übergeben, in welchem sich was verändert hat. Das Neuzeichnen der GUI-Animation geschieht davon unabhängig alle 100 Millisekunden, in welchen von der Simulation versucht wird, entsprechend der Animationsgeschwindigkeit, alle Ereignisse des simulierten Zeitraums abzuarbeiten. Reicht die Rechenleistung des Prozessors dafür nicht aus, so kann die gewünschte Animationsgeschwindigkeit leider nicht erreicht werden.

### 3.4.3.1 Die Animation

In Abbildung 3.30 ist eine Animation (links im Bild) zusammen mit einer Detailansicht einer Kreuzung (rechts im Bild) zu sehen. Während der Animation sieht man für jede Warteschlange die aktuelle Warteschlangenlänge (Anzahl der Fahrzeuge) als Zahl neben der Kreuzung angezeigt. Weiterhin wird durch rotes Einfärben der Straße eine aktuelle Blockade auf der Straße gekennzeichnet. Durch das Einkreisen der überfüllten Warteschlange kann man erkennen in welcher Fahrtrichtung sich der Stau gebildet hat. Will man mehr Details über eine Kreuzung, als nur ihre Warteschlangendaten, so kann man sich über die Detailansicht einen genaueren Überblick über die aktuelle Lage innerhalb der Kreuzung verschaffen. Diese Detailansicht aktualisiert sich, genauso wie der Rest der Animation, alle 100ms – man kann also detailliert die Vorgänge innerhalb einer Kreuzung betrachten. Im Beispiel kann man die Ampelphase erkennen in welcher sich momentan die Ampel befindet, genauso wie den momentanen Status (in diesem Fall ist die Ampel für alle Fahrzeuge aus dem Osten momentan grün) und die Segmentübersicht. Die Segmentübersicht stellt das Segmentmodell der Kreuzung dar und zeigt an in welchen Segmenten sich gerade was für ein Fahrzeug befindet. So fährt gerade in unserem Beispiel das Fahrzeug mit der ID 459 gerade nach Westen aus der Kreuzung aus. Im Weiteren biegen die Fahrzeuge 462 und 466 gerade hintereinander nach links ab.

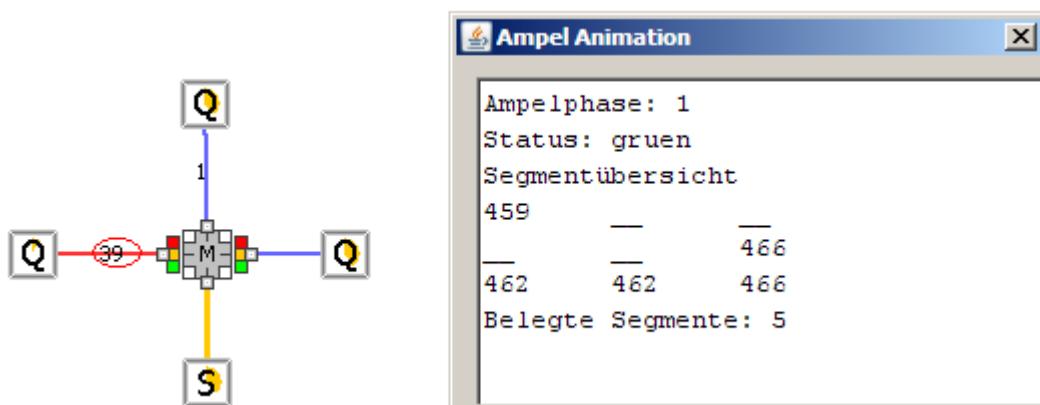


Abbildung 3.30: Beispiel einer Animation inklusive Detailansicht

### 3.4.4 Die Anbindung der Straßen an eine Kreuzung

Kommen wir nun zu der Anbindung von Straßen an Kreuzungen.

Jede Kreuzung hat vier Anbindungen, an welche jeweils eine Straße angebunden werden kann (vergleiche Abbildung 3.31).

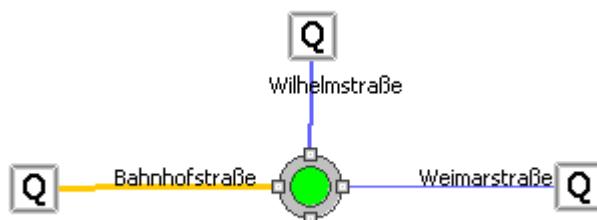


Abbildung 3.31: Eine Beispielkreuzung mit drei angebundenen Straßen

Diese Anbindungen sind nach Himmelsrichtungen sortiert. So kann jeweils eine Straße im Norden, Osten, Süden und Westen angebunden werden. Obwohl diese Angabe der Himmelsrichtung nicht verbindlich ist, so wird doch die Reihenfolge der Anbindungen und die 90 Grad Winkel zwischen den einzelnen Anbindungen innerhalb der Simulation als gegeben vorausgesetzt. Es ist daher nicht möglich individuelle Winkel zwischen den einzelnen angebundenen Straßen anzugeben oder mehr als vier Straßen an eine Kreuzung anzubinden. Allerdings ist es problemlos möglich weniger als vier Straßen an eine Kreuzung anzubinden. Zu beachten ist dann nur, dass die Beschaltung der Kreuzung überhaupt noch einen Sinn ergibt. Auf dieses Problem wird im nächsten Abschnitt nochmal genauer eingegangen.

Noch zu beachten ist, dass bei Quellen (werden als Q-Objekte dargestellt) und Senken (werden als S-Objekte dargestellt) jeweils nur eine Straßenanbindung erlaubt ist. Dies ist notwendig, weil z.B. Quellen keine Weichen-Logik zum Verteilen der ankommenden Fahrzeuge auf verschiedene Straßen enthalten.

#### **3.4.4.1 Was für Straßenanbindungen sind erlaubt?**

Beim Initialisieren des Simulationsmodells wird bei jeder Kreuzung überprüft, ob genug Straßen angebunden sind, damit das Kreuzungsmodell funktionieren kann. So sind nur sinnvoll beschaltete Kreuzungen erlaubt. Ist eine Kreuzung falsch mit Straßen beschaltet, beendet sich die Initialisierung und die Simulation wird nicht gestartet. So sind u.a. folgende unsinnige Beschaltungen verboten:

- Eine Kreuzung ohne Straßen.
- Eine Kreuzung mit nur einer angebundenen Straße.
- Eine Kreuzung mit angebundenen Einbahnstraßen, welche alle nur in die Kreuzung hineinführen.

Um solche fehlerhaften Konfigurationen bei Kreuzungsanbindungen zu erkennen, bedarf es einer Überprüfungslogik. So wird bei jedem Initialisieren überprüft, ob für jede Fahrspur welche in die Kreuzung mündet, mindestens eine Fahrspur der anderen Anbindungen aus der Kreuzung heraus führt. Damit ist es möglich alle unsinnigen Kreuzungen bereits vor dem Simulationsstart zu erkennen.

#### **3.4.4.2 Tabelle der Abbiegewahrscheinlichkeiten**

In diesem Abschnitt wollen wir uns mit der Tabelle der Abbiegewahrscheinlichkeiten der Kreuzungen befassen.

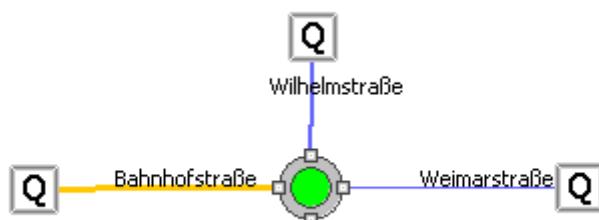


Abbildung 3.32: Unsere Beispielkreuzung

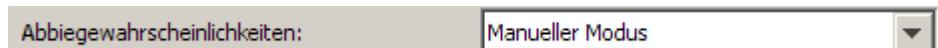


Abbildung 3.33: Aktivieren des Manuell-Modus

	N Wilhelmstraße	O Weimarstraße	S	W Bahnhofstraße
Rechtsabbieger	0.33	0.33	0.33	0.33
Geradeausfahrer	0.33	0.33	0.33	0.33
Linksabbieger	0.33	0.33	0.33	0.33

Abbildung 3.34: Beispiel für eine Abbiegewahrscheinlichkeitstabelle

In der Abbildung 3.34 ist die Standardvorgabe der Abbiegewahrscheinlichkeitstabelle der Kreuzung aus Abbildung 3.32 zu sehen, welcher über die Aktivierung des manuellen Modus, wie in Abbildung 3.33 gezeigt, zu erreichen ist. Wie zu sehen ist müssen alle Abbiegewahrscheinlichkeiten selbst eingetragen werden. Es muss also für die Richtungen Nord (aus welcher die Wilhelmstraße in unserem Beispiel angebunden ist), Ost (der Weimarstraße), Süd (welche in diesem Beispiel keine angebundene Straße hat) und West (der Bahnhofstraße) für alle Fahrzeuge die Wahrscheinlichkeiten zum Rechtsabbiegen, Geradeausfahren und Linksabbiegen angegeben werden. Die Simulation benutzt dann diese Angaben um die Fahrzeuge an den Kreuzungen zu verteilen. Noch zu beachten ist, dass das Abbiegen in eine nicht angebundene Richtung (hier Süden) beim automatischen Prüfen der Initialisierung durch einen Sicherungsmechanismus verhindert wird – man kann also ruhig die voreingestellten Werte so belassen und die Simulation startet trotzdem. Des Weiteren müssen die eingegebenen Wahrscheinlichkeiten für Geradeausfahrer, Rechts- und Linksabbieger auch zusammen nicht 100 Prozent ergeben. Die automatische Prüfung bei der Initialisierung rechnet die Abbiegewahrscheinlichkeiten automatisch auf prozentuelle Werte um.

Wenn man diese Tabelle der Abbiegewahrscheinlichkeiten nicht selbst ausfüllen möchte, kann man wie in Abbildung 3.35 zu sehen ist, auch den Automatik-Modus aktivieren (dies entspricht der Standardeinstellung).

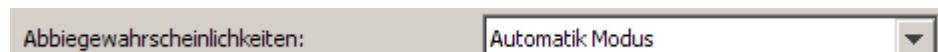


Abbildung 3.35: Aktivieren des Automatik-Modus

	N Wilhelmstraße	O Weimarstraße	S	W Bahnhofstraße
Rechtsabbieger	0.8696	0.3333	0.0909	0.0
Geradeausfahrer	0.0	0.6667	0.303	0.2308
Linksabbieger	0.1304	0.0	0.6061	0.7692

Abbildung 3.36: Beispiel für eine automatisch erstellte  
Abbiegewahrscheinlichkeitstabelle

Dieser füllt die Tabelle der Abbiegewahrscheinlichkeiten selbstständig auf (für unser Beispiel siehe Abbildung 3.36). Dabei verwendet er die bei den Straßeneinstellungen hinterlegten, durchschnittlichen Fahrzeugdurchsatz (Fahrzeuge pro Minute welche die Straße passieren). Die Logik dahinter ist einfach, dass ein Fahrzeug eher in eine viel befahrene Straße einbiegen wird, als in eine wenig befahrene Straße. So sind im obigen Beispiel folgende durchschnittliche Fahrzeuge pro Minute hinterlegt: in der Bahnhofstraße passieren 20

Fahrzeuge pro Minute die Straße, in der Wilhelmstraße 10 und in der Weimarstraße 3. Rechnet man dies nun zusammen, bekommt man die in der oberen Tabelle stehenden Abbiegewahrscheinlichkeiten zusammen. So ist die Wahrscheinlichkeit aus der Weimarstraße nach rechts in die Bahnhofstraße einzubiegen mit rund 66 % (weil 20 Fahrzeuge pro Minute diese passieren) doppelt so groß wie die Wahrscheinlichkeit für das Einbiegen in die Wilhelmstraße mit rund 33% (mit 10 Fahrzeugen pro Minute).

### 3.4.4.3 Falsches Abbiegen verhindern

Weiterhin werden bei der Initialisierung der Kreuzung eventuelle Fehler in den Abbiegewahrscheinlichkeiten erkannt und korrigiert. So kann es schnell durch Unachtsamkeit bei der Eingabe der Abbiegewahrscheinlichkeiten zu einem Fehler kommen. Es kann zum Beispiel sein, dass man ein Einbiegen in eine nicht vorhandene Straße, bzw. entgegen einer Einbahnstraße erlaubt. Um solche groben Fehler zu erkennen und zu korrigieren, wird bei der Initialisierung der Kreuzung für alle verbotenen Abbiegemöglichkeiten die Abbiegewahrscheinlichkeit auf 0 gesetzt, so dass ein falsches Abbiegen nicht auftreten kann.

### 3.4.5 Das Konzept der Segmente

Kommen wir nun zu einem weiteren grundlegenden Kreuzungskonzept - den Segmenten. Ein Segment ist die kleinste, unteilbare Einheit einer Kreuzung und hat eine individuelle Identifikationsnummer. Jedes Segment kann dabei nur drei Zustände einnehmen:

- frei – Wenn kein Fahrzeug momentan das Segment belegt.
- belegt – Wenn ein Fahrzeug sich momentan durch das Segment bewegt.
- blockiert – Wenn das Fahrzeug welches sich momentan in der Kreuzung befindet, nicht in das nächste Segment einfahren kann, weil dort sich momentan ein anderes Fahrzeug befindet.

Jede Kreuzung ist aus Segmenten aufgebaut, welche ein Fahrzeug beim Durchqueren der Kreuzung durchfährt (siehe dazu Abbildung 3.37: Beispiel Kreisverkehr).

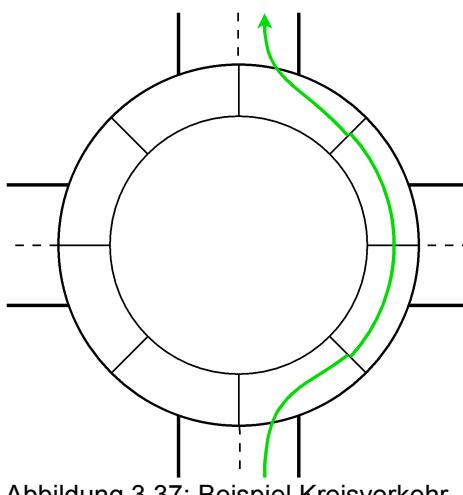


Abbildung 3.37: Beispiel Kreisverkehr

Wie in der Abbildung zu erkennen ist, muss dabei, je nach Einfahrt und Abbiegerichtung, jedes Fahrzeug andere Segmente durchqueren. Daher ist in jedem Fahrzeug (siehe dazu Kapitel 3.3.6.2 Die Schnittstelle „KreuzungFahrzeugContainer“) die individuelle Liste der Segmente und die Reihenfolge, wie diese durchfahren werden müssen, hinterlegt (Routenkonzep). Diese Route wird beim Voranmelden der Fahrzeuge bei der Kreuzung anhand der Tabelle der Abbiegewahrscheinlichkeiten für jedes Fahrzeug individuell erstellt.

Dies hat den Vorteil, dass jedes Fahrzeug seine eigene Route kennt und die Kreuzungslogik bequem auf diese Daten zugreifen kann.

Kommt es jetzt zu der Situation, dass das nächste Segment der Route belegt ist, setzt das Fahrzeug den Status des aktuellen Segments auf „blockiert“ und meldet sich bei dem Segment, von welchem es blockiert wird, als blockiertes Fahrzeug an. Später wenn das Segment wieder frei ist, wird dann von diesem Segment die Blockade aufgehoben und das Fahrzeug kann weiter fahren (Hollywood-Prinzip).

Das Problem bei dieser Vorgehensweise ist, dass in der Realität diese Blockade durch ein vorausfahrendes Fahrzeug vom Fahrer erkannt werden würde und dieser einfach seine Geschwindigkeit reduzieren würde. Damit tritt so etwas wie die Blockade im Normalfall nicht auf. Durch dieses Verhalten entsteht auch ein Geschwindigkeitsverlust des Fahrzeugs, welcher von der Simulation nicht im Voraus berechnet werden kann. So muss dieser Geschwindigkeitsverlust durch die Blockade von der Simulation (beim Aufheben der Blockade) berechnet werden (siehe Kapitel 2.2 Der Geschwindigkeitsverlust bei kurzfristiger Blockade eines Fahrzeugs) und die im Fahrzeug gespeicherte Geschwindigkeit entsprechend angepasst werden.

### 3.4.5.1 Das Mehrsegment-Fahrzeug-Konzept

In der klassischen Verkehrssimulation gilt das Prinzip, dass sich ein Fahrzeug immer nur in einem Segment befinden kann. Wenn man sich jetzt die beträchtlichen Größenunterschiede zwischen einem normalen PKW und einem LKW mit Anhänger vor Augen führt, so muss man sich doch fragen, ob dieses Modell der Realität überhaupt gerecht werden kann. Daher wird bei meinem Modell der Kreuzungen nicht mehr von diesem Konzept Gebrauch gemacht. Es kann daher Fahrzeuge geben, die eine unbekannte Anzahl von Segmenten lang sind. Weiterhin kann die Länge der Segmente deutlich verkürzt werden, was zu einer feineren Auflösung bei den Kreuzungsmodellen führt.

Allerdings darf man die Nachteile dieses Konzepts nicht vergessen. Ab sofort ist es nicht mehr bekannt, in welchem Segment sich ein Fahrzeug genau befindet - es erstreckt sich ja auf mehrere Segmente. Aber wie werden diese vielen Segmente, aus welchen das Fahrzeug besteht, verwaltet und wie fährt ein solches Fahrzeug durch eine Kreuzung (siehe Beispiel Abbildung 3.38)?

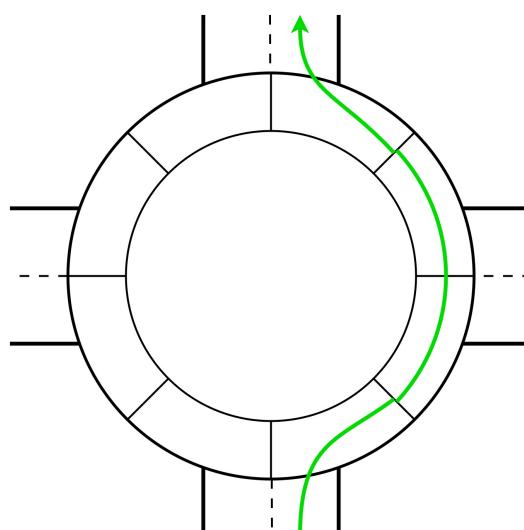


Abbildung 3.38: Segmentübersicht eines Kreisverkehrs mit 8 Segmenten

Eine Antwort für dieses Problem befindet sich in dem Videospielklassiker Snake (vergleiche Abbildung 3.39). In diesem Spiel steuert man eine Schlange, welche sich Segment für Segment durch die Gegend schlängelt. Bei jedem „Schritt“ den die Schlange dabei macht, wird am Schlangenende (Tail) ein Segment entfernt und beim Schlangenkopf (Head) ein neues Segment hinzugefügt.

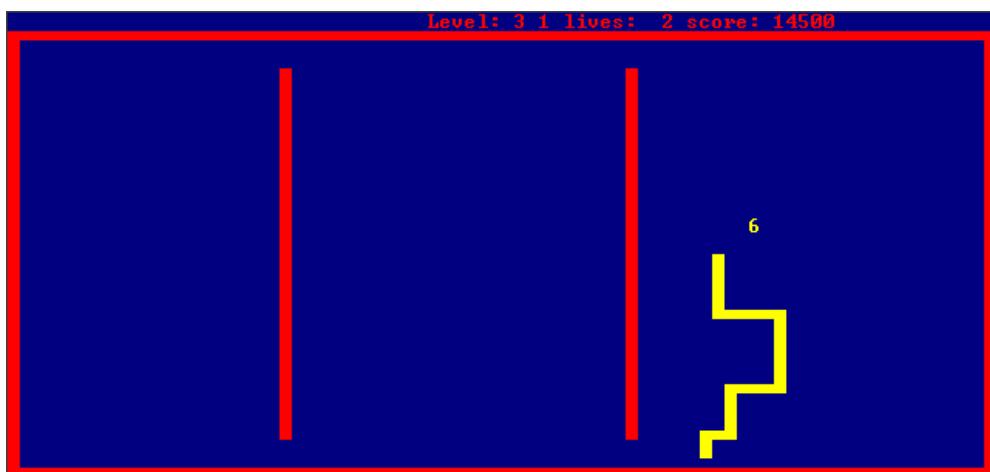
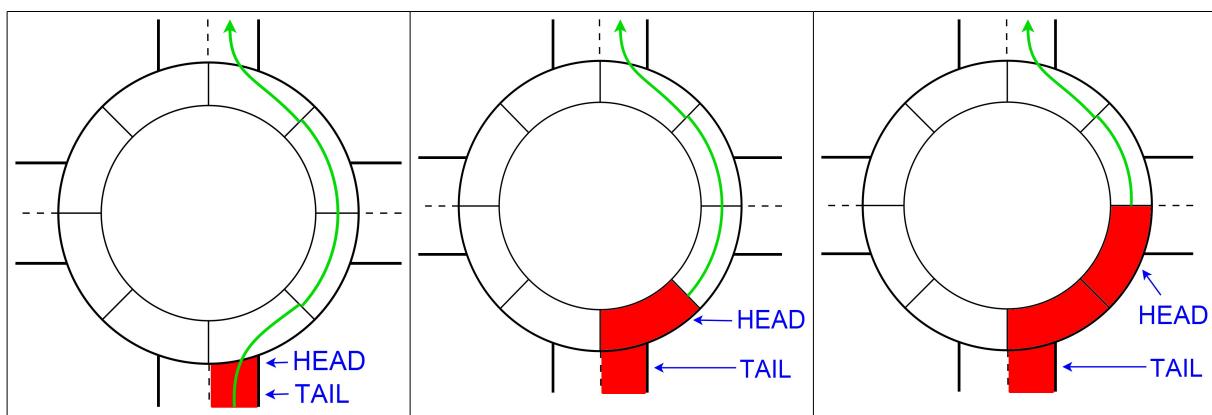


Abbildung 3.39: Nibbles: Remake des Spielesklassiker Snake [WIKI03@]

Dieses Prinzip wird auch bei meiner Simulation verwendet. Jedes Fahrzeug hat einen Head und einen Tail, wobei bei kurzen Fahrzeugen diese auch auf dasselbe Segment verweisen können, und bei jedem Bedienende-Ereignis werden sowohl der Head wie auch der Tail um ein Segment weiter bewegt.

In den nun folgenden Abbildungen in der Tabelle 3.2 wird Beispielhaft das oben erklärte Schema an einem Fahrzeug dargestellt, welches eine Länge von drei Segmenten hat. Dieses Fahrzeug bewegt sich entlang der in Abbildung 3.38 dargestellten Route.

Zu beachten sind dabei noch die Sonderfälle bei der Einfahrt und der Ausfahrt des Fahrzeugs. Bei diesen Sonderfällen bleibt der Head bzw. der Tail solange bei jedem Schritt an seinem Platz, bis das Fahrzeug mit seiner kompletten Länge eingefahren bzw. aus dem Kreisverkehr herausgefahren ist.



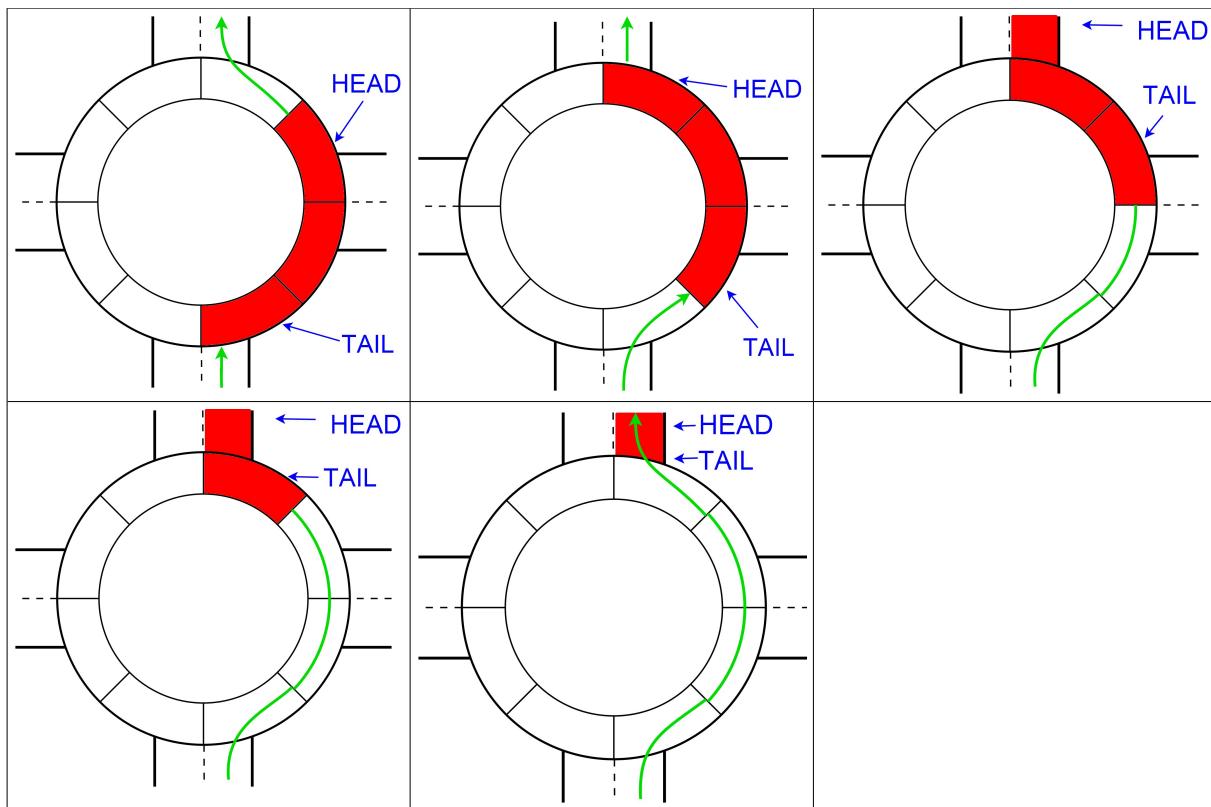


Tabelle 3.2: Das Head - Tail Konzept mit einem Drei-Segment-Fahrzeuge

## 3.5 Der Kreisverkehr

Kommen wir nun zum Modell des Kreisverkehrs. Der nun folgende Abschnitt erklärt grob den Modellaufbau. Die detaillierten Ereignisse und Hilfsfunktionen können als Pseudocode im Anhang C – Pseudocode Kreisverkehr betrachtet werden.

### 3.5.1 Der Kreisverkehr in der graphischen Oberfläche

In Abbildung 3.40 wird gezeigt, wie ein Kreisverkehr in der graphischen Oberfläche der Simulation aussieht. Zu beachten ist, dass in der Simulation generell nur Kreuzungen mit maximal vier angebundenen Straßen erlaubt sind. Daher hat auch der Kreisverkehr nur vier mögliche Anbindungen für Straßen.

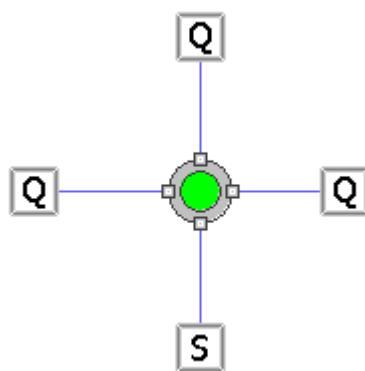


Abbildung 3.40: Der Kreisverkehr in der Simulation.

In der folgenden Abbildung 3.41 ist der Einstellungsdialog für einen Kreisverkehr zu sehen. Die einzige besondere Einstellung ist dabei die Eingabemöglichkeit für den Kurvenradius innerhalb des Kreisverkehrs. Diese Eingabe wird benötigt um die Anzahl und Länge der Segmente des späteren Modells genauso wie die maximale Geschwindigkeit des Fahrzeugs innerhalb der Kreisverkehrs (vergleiche Kapitel 2.1 „Die maximale Geschwindigkeit eines Fahrzeugs in einer Kurve“), zu berechnen. Als Eingabe wird dabei der mittlere Kurvenradius gefordert, welcher von einem Fahrzeug das den Kreisverkehr durchquert benutzt werden würde.

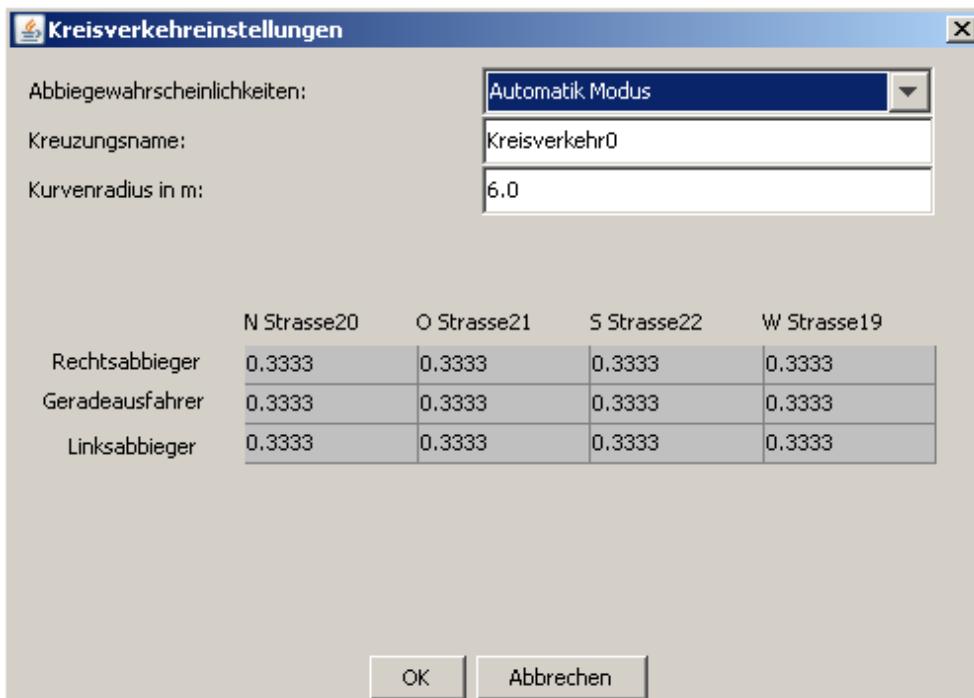


Abbildung 3.41: Die Kreisverkehreinstellungen

### 3.5.2 Das Modell

Nun wollen wir das Modell eines Kreisverkehrs genauer betrachten. Dazu betrachten wir erstmal ein Kreisverkehr mit vier Segmenten, bevor wir auf einen Kreisverkehr mit mehr Segmenten und den Algorithmus zum Erzeugen von Kreisverkehren mit beliebig vielen Segmenten eingehen.

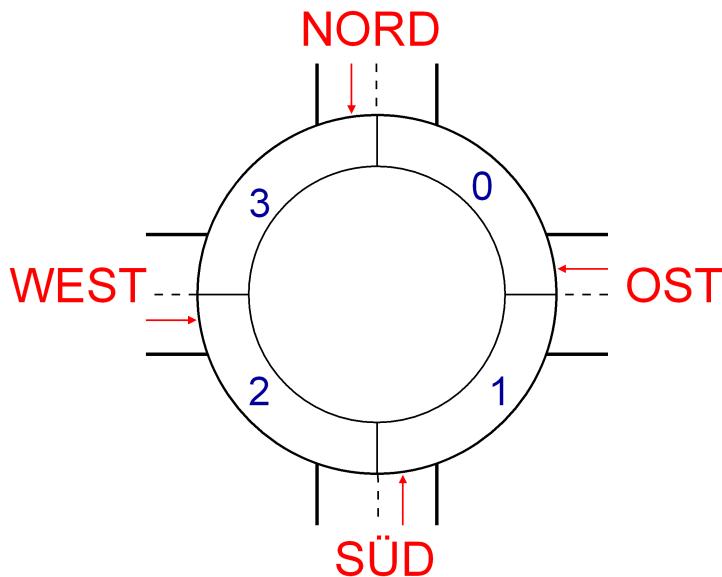


Abbildung 3.42: Segmentübersicht Kreisverkehr mit 4 Segmenten

Wie in Abbildung 3.42 zu sehen, ist bei diesem kleinsten aller möglichen Kreisverkehre jedes Segment gleichzeitig ein Einfahrt- und Ausfahrtsegment. Die Nummerierung verläuft im Uhrzeigersinn, beginnend bei der nördlichen Einfahrt. Die Fahrzeuge fahren, entgegen der Nummerierung und entgegen dem Uhrzeigersinn durch den Kreisverkehr. Zu beachten ist, dass vier Segmente die kleinste mögliche Segmentanzahl für einen Kreisverkehr mit vier angebundenen Straßen ist. Allerdings funktionieren auch alle Vielfache davon zum Aufbau eines Kreisverkehrs.

In der nun folgenden Abbildung 3.43 ist ein Kreisverkehr mit acht Segmenten zu sehen. Dieser entspricht außer der Tatsache, dass nicht mehr jedes Segment gleichzeitig ein Einfahrt- und Ausfahrtsegment ist, dem Modell mit vier Segmenten.

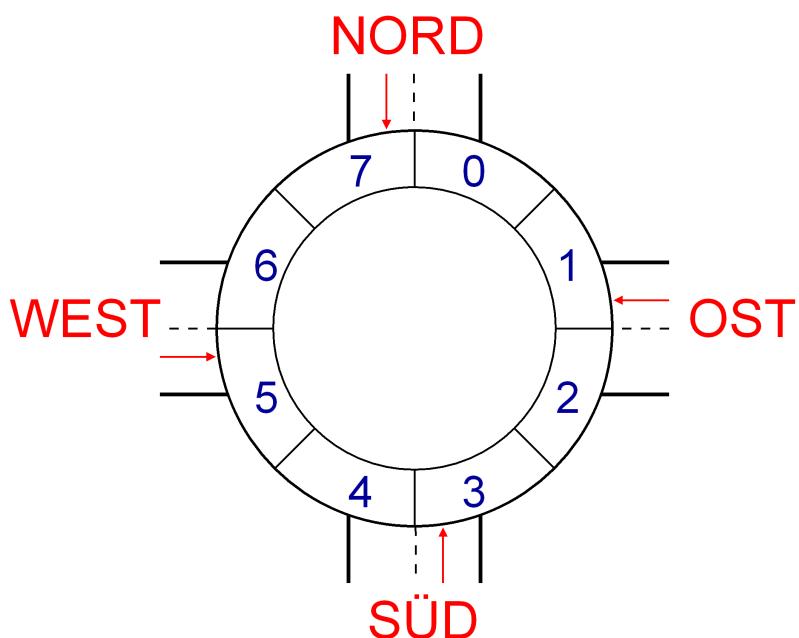


Abbildung 3.43: Segmentübersicht Kreisverkehr mit 8 Segmenten

### 3.5.3 Die dynamische Berechnung der Segmentanzahl eines Kreisverkehrs

In diesem Abschnitt wollen wir kurz darauf eingehen, welchen Bedingungen ein Kreisverkehr innerhalb der Simulation unterworfen ist und wie wir die Anzahl und Länge von Segmenten berechnen.

Zur Berechnung der Anzahl und Länge der Segmente verwenden wir folgende Vorgaben:

- Der Kreisverkehr hat einen minimalen Umfang von 9 Meter.
  - Der Kreisverkehr hat einen maximalen Umfang von 250 Meter.
  - Jedes Segment ist mindestens zwei, maximal vier Meter lang.
- Zu beachten ist, dass die maximale Segmentlänge nicht kleiner als das Doppelte der minimalen Segmentlänge sein darf, weil sonst der nachfolgende Algorithmus nicht funktioniert.

Weiterhin gilt:

- Der Kreisumfang  $u$  wurde mit folgender Formel berechnet:  

$$u = 2 \cdot \pi \cdot r$$
- Für das Vervielfachen der 4 Segmente führen wir die Variable  $i$  als Segmentmultiplikator ein. Daher gilt: Segmentanzahl =  $i * 4$ .
- Für die Segmentlänge verwenden wir die Variable  $s$ .

Unter Beachtung der Nebenbedingungen oben können wir nun folgenden Algorithmus anwenden:

```
for ( i = 1; i < 20; i++ ){
    s = u / ( i * 4 );
    if ( s <= 4.0 && s >= 2.0 ){
        „Es wurde der richtige Segmentmultiplikator gefunden.“
        „Anzahl der Segmente = i * 4“
        „Schleife abbrechen.“
    }
}
```

### 3.5.4 Die Abbiegelogik

In der Tabelle 3.3 sind die verschiedenen Routen für die verschiedenen Abbiegerichtungen abgebildet.

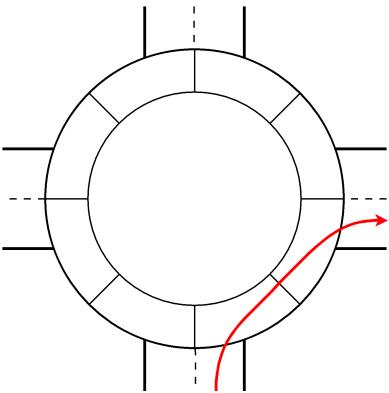
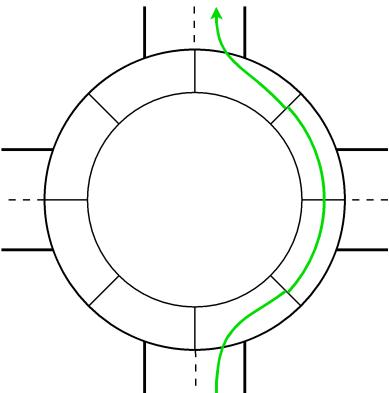
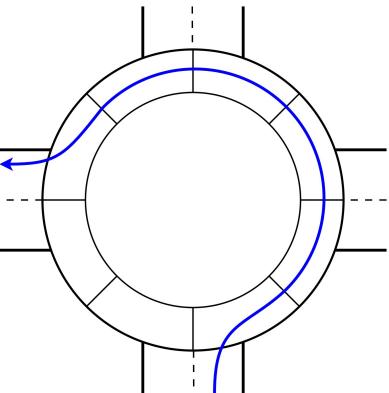
Rechtsabbiegen:	Geradeaus fahren:	Linksabbiegen:
		

Tabelle 3.3: Übersicht der Segmentnutzung nach Abbiegerichtung

### 3.5.5 Die Einfahrtregeln

Beim Einfahren eines Fahrzeugs in einen Kreisverkehr gilt es folgende Regeln zu beachten:

- Es darf nicht eingefahren werden, wenn ein weiteres Fahrzeug sich vor einem in der Warteschlange befindet.
- Ein Fahrzeug darf nicht in den Kreisverkehr einfahren, wenn das Einfahrsegment momentan belegt ist.
- Es darf nur eingefahren werden, wenn von links kein Fahrzeug kommt. Es muss genügend Platz zum nächsten - bereits im Kreisverkehr fahrenden - Fahrzeug geben, damit das komplette Fahrzeug einfahren kann. Es müssen also genügend Segmente entgegen der Fahrtrichtung frei sein, damit das ganze Fahrzeug hinein passt.

### 3.5.6 Einschränkungen

- Es gibt nur Kreisverkehre mit maximal vier Ein- und vier Ausfahrten.
- Es ist nur möglich einen einspurigen Kreisverkehr abzubilden.
- Die im Kreisverkehr fahrenden Fahrzeuge haben Vorfahrt. Ein Kreisverkehr nach der Rechts-vor-Links Regelung ist nicht möglich.
- Es ist nicht möglich auf seine Vorfahrt zu verzichten, um ein anderes Fahrzeug in den Kreisverkehr einfahren zu lassen.

## 3.6 Die Rechts-vor-Links Kreuzung

In diesem Abschnitt wollen wir uns mit einer einfachen Rechts-vor-Links Kreuzung befassen. Der nun folgende Abschnitt erklärt grob den Modellaufbau. Die detaillierten Ereignisse und Hilfsfunktionen können als Pseudocode im Anhang D – Pseudocode Rechts-Vor-Links Kreuzung betrachtet werden.

### 3.6.1 Die Rechts-vor-Links Kreuzung in der graphischen Oberfläche

In Abbildung 3.44 wird gezeigt, wie ein Rechts-vor-Links Kreuzung in der graphischen Oberfläche der Simulation aussieht.

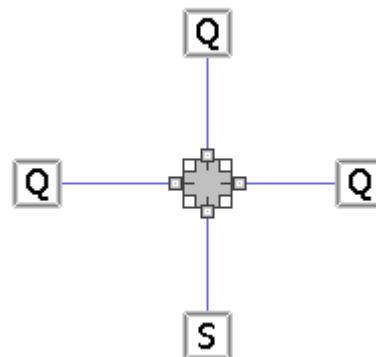


Abbildung 3.44: Die Rechts-vor-Links Kreuzung in der Simulation

In der Abbildung 3.45 ist der Einstellungsdialog einer Rechts-vor-Links Kreuzung zu sehen. Als einzige besondere Einstellung ist hier die Eingabe der Kreuzungsbreite zu sehen. Diese

Kreuzungsbreite wird bei der Initialisierung der Simulation zur Berechnung der Segmentlänge des Modells und der maximalen Höchstgeschwindigkeit des Fahrzeugs (siehe Kapitel 2.1 „Die maximale Geschwindigkeit eines Fahrzeugs in einer Kurve“) benötigt.

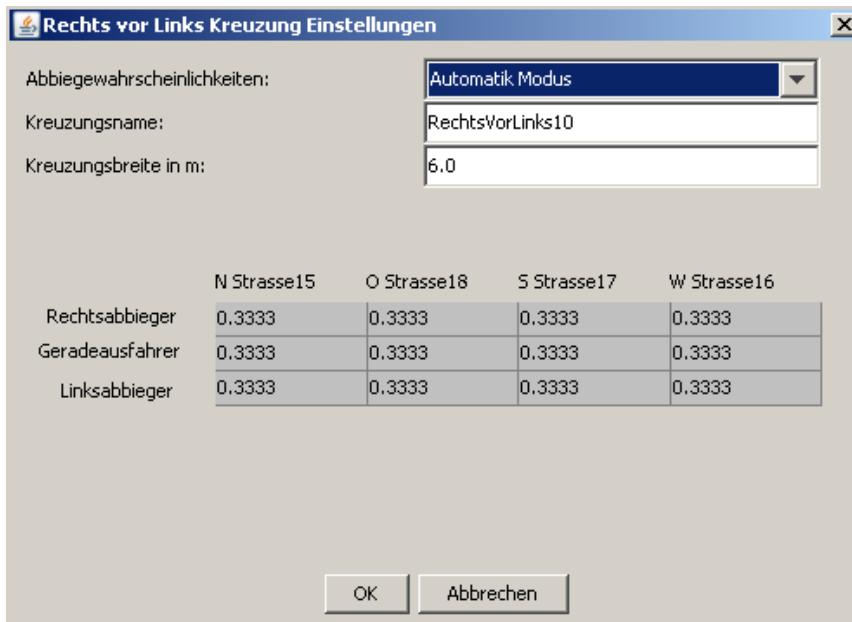


Abbildung 3.45: Der Einstellungsdialog für die Rechts-vor-Links Kreuzung

### 3.6.2 Das Modell

Wie in Abbildung 3.46 zu sehen ist, besteht die Rechts-vor-Links Kreuzung aus vier Segmenten, welche klassisch nach Zeile und Spalte durchnummeriert sind. Die Länge eines Segments entspricht dabei der halben Kreuzungsbreite.

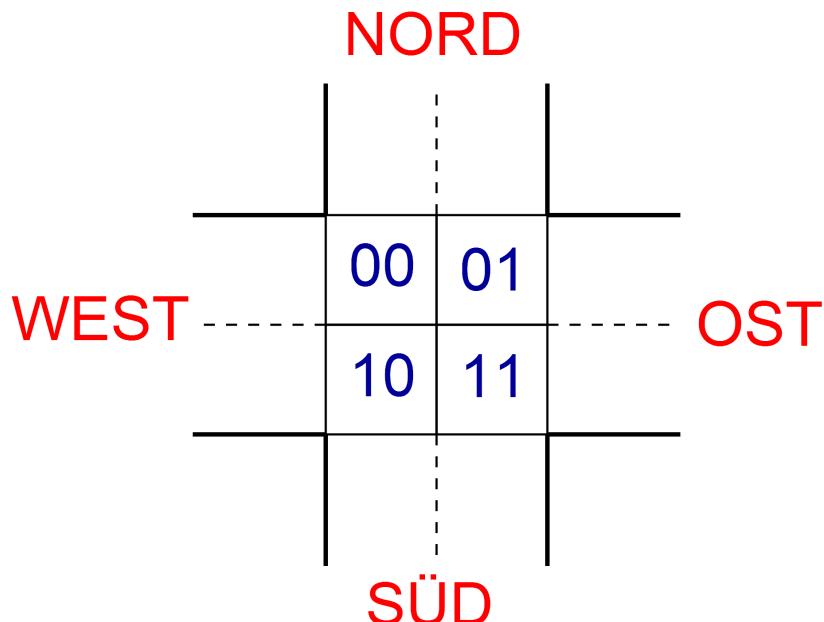


Abbildung 3.46: Segmentübersicht einer Rechts-vor-Links Kreuzung

### 3.6.3 Die Abbiegelogik

Nachdem das grundlegende Modellkonzept steht, bleibt die Frage offen welche Route die Fahrzeuge innerhalb einer Kreuzung zurücklegen müssen um an ihr Ziel zu kommen. Dazu werden erstmals alle verschiedenen Abbiegemöglichkeiten eines Fahrzeugs im grundlegenden Modell betrachtet (siehe Abbildung 3.47).

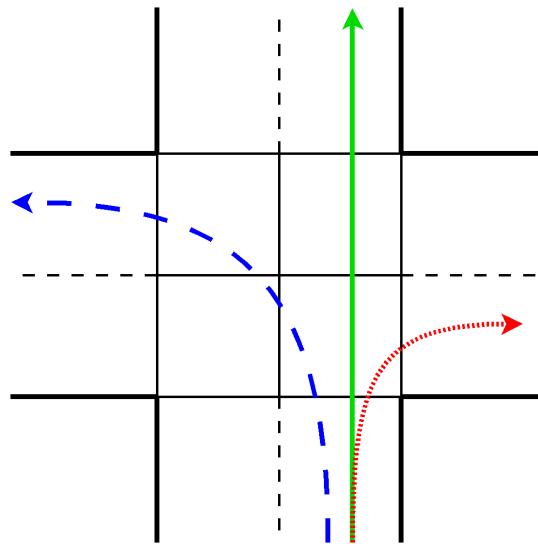


Abbildung 3.47: Kurzübersicht über die Abbiegemöglichkeiten

Durch diese Überlegung kommen wir nun zu folgenden Routen, welche in Abbildung 3.48 zu sehen sind. Zu beachten ist nur, dass wir, um die längere Strecke für die Linkssabbieger entsprechend abzubilden, eine ausgefahrene Linkskurve über drei Segmente benutzen. Als Folge davon ist es im Modell leider nicht möglich, wie dies in der Straßenverkehrsordnung vorgesehen ist, dass zwei entgegenkommende Fahrzeuge voreinander nach links abbiegen können.

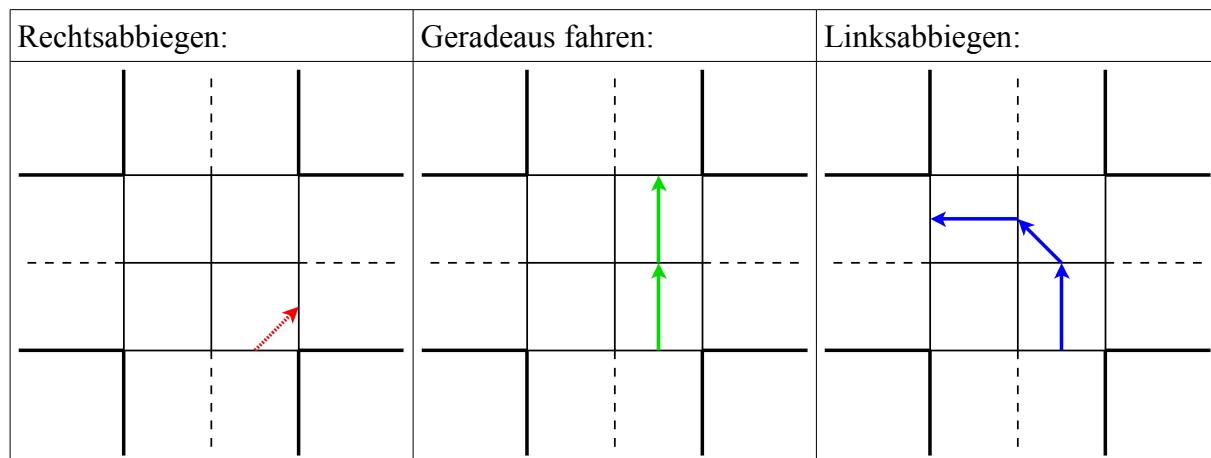


Abbildung 3.48: Übersicht der Nutzung von Segmenten beim Abbiegen

### 3.6.4 Die Einfahrtregeln

Beim Einfahren eines Fahrzeugs in eine Rechts-vor-Links Kreuzung gilt es folgende Regeln zu beachten:

- Es darf nicht in die Kreuzung eingefahren werden, solange ein anderes Fahrzeug sich in der Kreuzung befindet.
- Es darf nicht in die Kreuzung eingefahren werden, wenn sich von rechts ein Fahrzeug nähert bzw. es sich in der Warteschlange befindet.  
Um zu kontrollieren, ob sich ein Fahrzeug aus einer Richtung nähert, sich aber noch nicht an der Kreuzung befindet wird überprüft ob ein neues Ankunftsereignis eines Fahrzeugs aus dieser Richtung in den nächsten zwei Sekunden geplant ist.
- Sollten sich aus allen vier Richtungen Fahrzeuge nähern, wird zufällig ausgewählt welches Fahrzeug als erstes einfahren darf.

### 3.6.5 Einschränkungen

- Um das Kreuzungsprinzip möglichst einfach zu halten, darf sich immer nur ein Fahrzeug gleichzeitig in der Kreuzung befinden. Da allerdings Rechts-vor-Links-Kreuzungen nur bei Kreuzungen mit einem geringen Verkehrsaufkommen benutzt werden dürfte der Fehler, der dadurch innerhalb der Simulation entsteht, vernachlässigbar sein.

## 3.7 Die Ampelkreuzungen

In diesem Kapitel wollen wir uns mit dem wohl wichtigsten Kreuzungstyp für moderne Städteplanung befassen – der Ampel. Der nun folgende Abschnitt erklärt grob den Modellaufbau. Die detaillierten Ereignisse und Hilfsfunktionen können als Pseudocode im Anhang E – Pseudocode Ampel betrachtet werden.

### 3.7.1 Gemeinsamkeiten der vier Ampeltypen

In diesem Abschnitt wollen wir auf die Gemeinsamkeiten der verschiedenen Ampeltypen eingehen, um nicht bei jeder Ampel wieder dieselben Feststellungen treffen zu müssen.

#### 3.7.1.1 Gemeinsamkeiten bei den Einstellungsdialogen

Die Gemeinsamkeiten der Einstellungsdialoage aller Ampeltypen sind:

- Die Kreuzungsbreite:  
Die Breite der Kreuzung wird zum Berechnen der zurückzulegenden Wegstrecken (Segmentgröße) und den maximal erreichbaren Geschwindigkeiten der Fahrzeuge in der Kreuzung verwendet.
- Die Schaltzeiten für die Ampelphasen:  
Hier wird die Länge der Grünphasen für die verschiedenen Richtungen angegeben.
- Die Länge der Fußgängerphase:  
Hier kann die Länge der Grünphase für die Fußgänger angegeben werden. Wird der Wert bei 0.0 gelassen, wird von dem Programm die komplette Phase weggelassen.
- Der Phasenoffset:  
Mit Hilfe dieser Funktion ist es möglich über verschiedene Ampeln hinweg eine grüne Welle zu simulieren. Um dies zu ermöglichen muss es möglich sein manche Ampel früher bzw. später auf Grün schalten zu können, um den Fahrzeugen ein zügiges Durchkommen zu ermöglichen. Dazu wird der Phasenoffset in Sekunden verwendet. Mithilfe dieses Offsets berechnet die Ampel in welcher Phase sie sich beim Start der

Simulation befindet. Wird kein Offset eingegeben, startet die Ampel mit der ersten Phase und dem Status „Grün“.

### 3.7.1.2 Die Einfahrtregeln

Die Einfahrtregeln sind bei der Ampel ziemlich einfach definiert. Ein Fahrzeug darf in eine Kreuzung einfahren wenn es gerade Grün hat, das Einfahrtsegment frei ist und die Kreuzung momentan nicht blockiert ist.

Eine Blockade entsteht auf einer Kreuzung, wenn beim Beginn der Grünphase sich noch Fahrzeuge auf der Kreuzung befinden. Dies kann u.a. passieren, wenn ein großer, langsamer LKW kurz vor dem Ende der Grünphase in die Kreuzung einfährt und durch eine kleine Kurve fahren muss. In so einem Fall kann es sein, dass die verbleibende Grünphase und die folgende Rotphase nicht mehr zum Durchqueren der Kreuzung ausreichen. Verlässt das Fahrzeug, welches die Kreuzung blockiert, diese, so wird die Blockade aufgehoben und die anderen Fahrzeuge können in die Kreuzung einfahren.

### 3.7.1.3 Die Ampelphasen

Soweit nicht anders vermerkt gelten folgende Phasen für alle Ampeltypen:

1. Grün für die nördliche Einmündung.
2. Grün für die östliche Einmündung.
3. Grün für die südliche Einmündung.
4. Grün für die westliche Einmündung.
5. Grün für Fußgänger

Zwischen diesen Grünphasen wird jeweils noch eine Rotphase mit einer einheitlichen Länge von zwei Sekunden eingefügt.

Sollte z.B. durch eine fehlende Anbindung eine Phase sinnlos sein, so wird dies von der Simulation erkannt und diese Phase (inklusive ihrer Rotphase) automatisch übersprungen.

## 3.7.2 Die kleine Ampel

Das Modell der kleinen Ampel spiegelt eine einfache kleine Kreuzung wieder, welche entsprechend unkompliziert zu verstehen ist.

### 3.7.2.1 Die kleine Ampel in der graphischen Oberfläche

In Abbildung 3.49 wird gezeigt wie die kleine Ampel in der graphischen Oberfläche der Simulation aussieht.

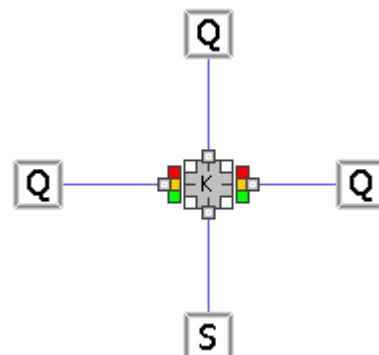


Abbildung 3.49: Die kleine Ampel  
in der Simulation

In der Abbildung 3.50 ist der Einstellungsdialog der Kreuzung zu sehen. Dieser entspricht dem Mindestumfang aller Ampelkreuzungen und ist daher schon besprochen.

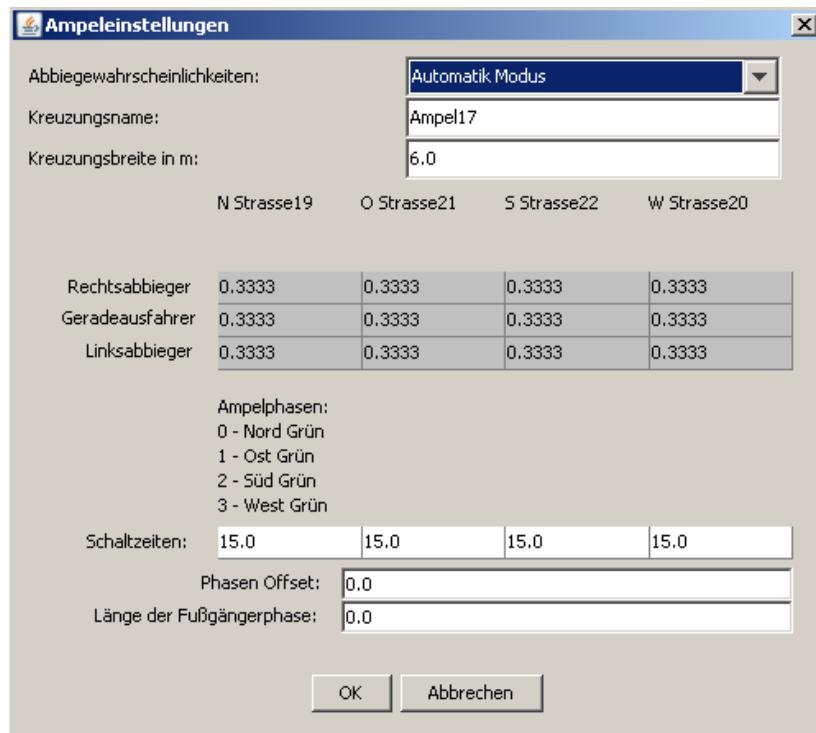


Abbildung 3.50: Der Einstellungsdialog der kleinen Ampel

### 3.7.2.2 Das Modell

Das Segmentkonzept der kleinen Ampel entspricht weitgehendsten dem normalen Vier-Segment-Konzept, welches auch schon bei der Rechts-vor-Links Kreuzung Verwendung findet. Wie in Abbildung 3.51 ersichtlich gibt es für jede Einfahrt genau eine Spur für alle Abbiegerichtungen.

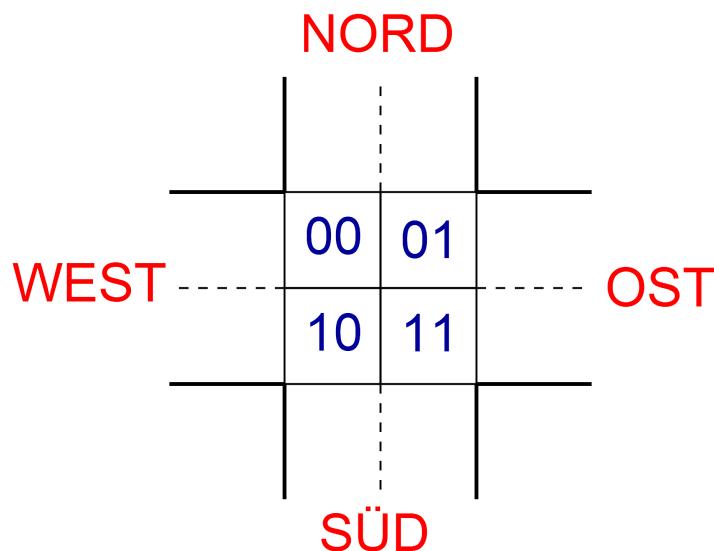


Abbildung 3.51: Segmentübersicht der kleinen Ampel

### 3.7.2.3 Die Abbiegelogik

Auch die genutzten Segmente beim Abbiegen sind genauso wie bei der Rechts-vor-Links Kreuzung (zu sehen in den Abbildungen 3.52 und 3.53).

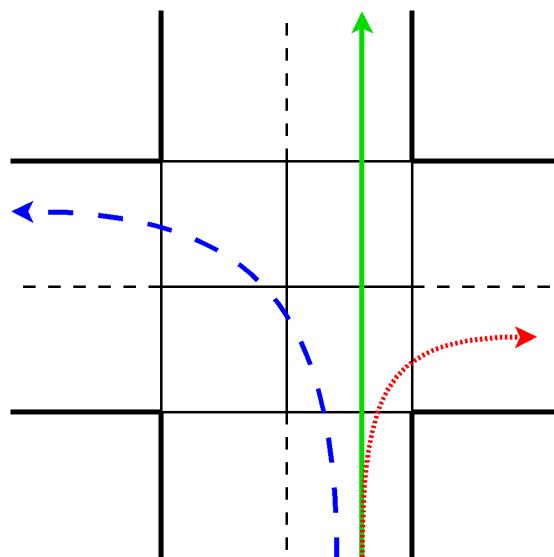


Abbildung 3.52: Kurzübersicht über die Abbiegemöglichkeiten in der Kreuzung

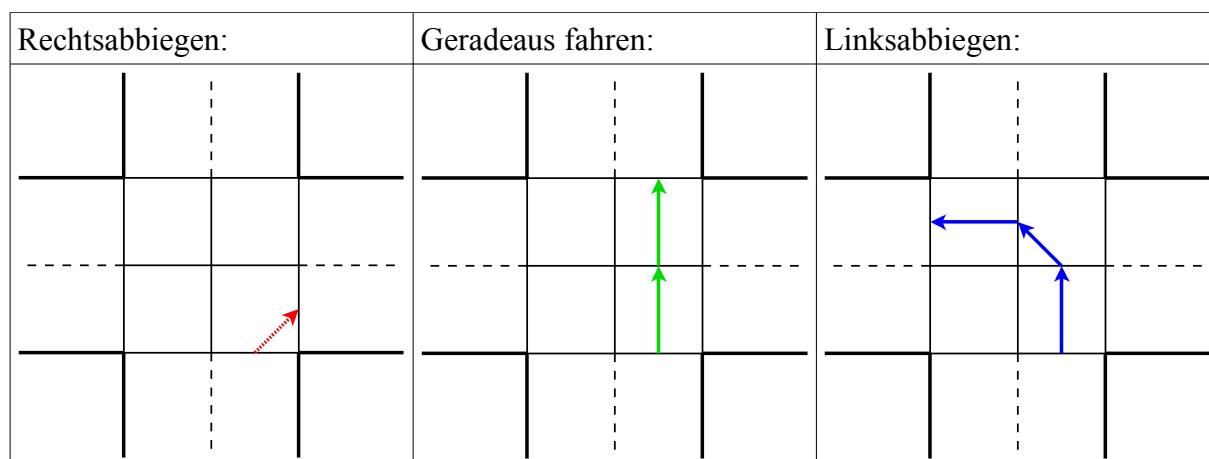


Abbildung 3.53: Übersicht der Segmentnutzung nach Abbiegerichtung

### 3.7.3 Die mittelgroße Ampel

Dieses Ampelmodell entspricht dem Modell einer einfachen Ampel mit einer extra Spur für Linksabbieger.

#### 3.7.3.1 Die mittelgroße Ampel in der graphischen Oberfläche

In Abbildung 3.54 ist die mittelgroße Ampel in der graphischen Oberfläche zu sehen. Gekennzeichnet wird diese durch das große M inmitten des Kreuzungssymbols.

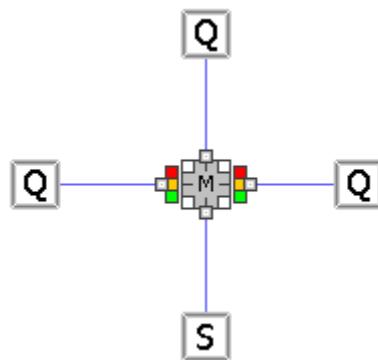


Abbildung 3.54: Die mittlere Ampel in der Simulation

Als einzige Neuerung, im Gegensatz zur kleinen Ampel, muss im Einstellungsdialog der Ampel (siehe Abbildung 3.55) für jede Richtung die Längen der linken Abbiegespuren angegeben werden. Diese wird dann für die Längen der einzelnen Spuren vor der Kreuzung verwendet (siehe dazu Kapitel 3.4.1.1 Funktionsweise der mehrstufigen Warteschlange).

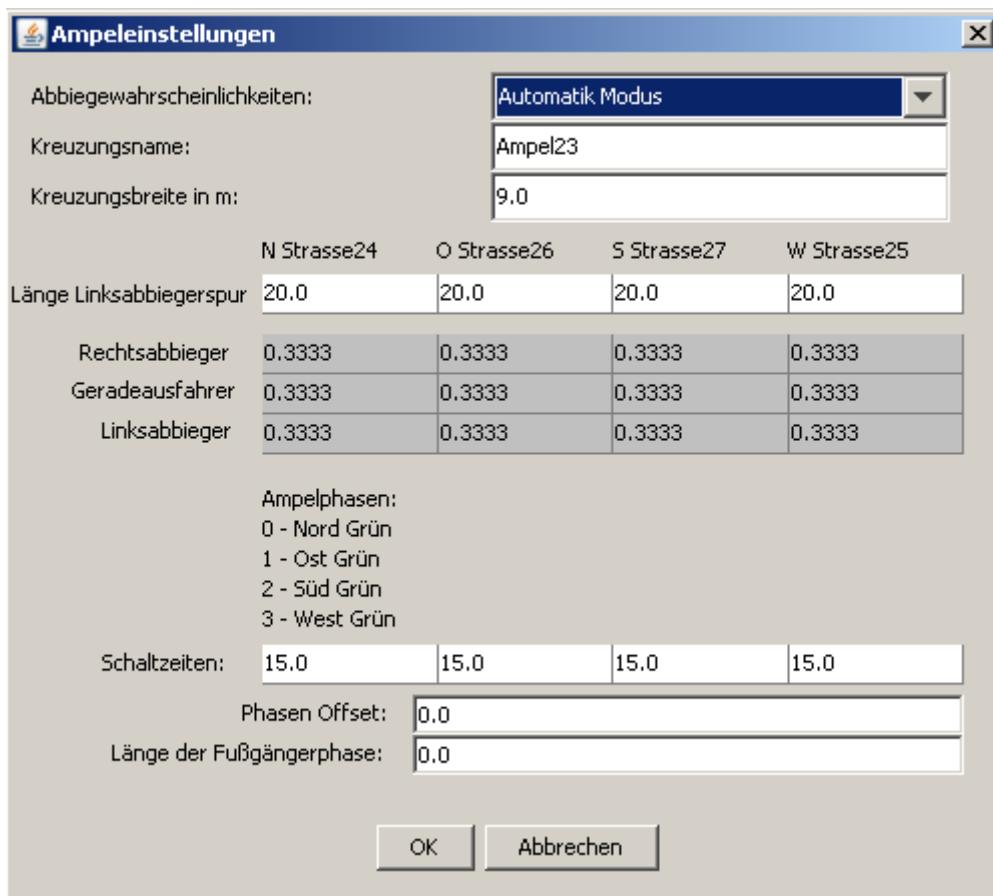


Abbildung 3.55: Der Einstellungsdialog der mittleren Ampel

### 3.7.3.2 Das Modell

Nun folgend, ist in Abbildung 3.56 das grundlegende Segmentmodell der Kreuzung zu sehen. Durch die zusätzliche Spur für Linksabbieger hat sich die Segmentanzahl auf insgesamt neun Segmente erhöht.

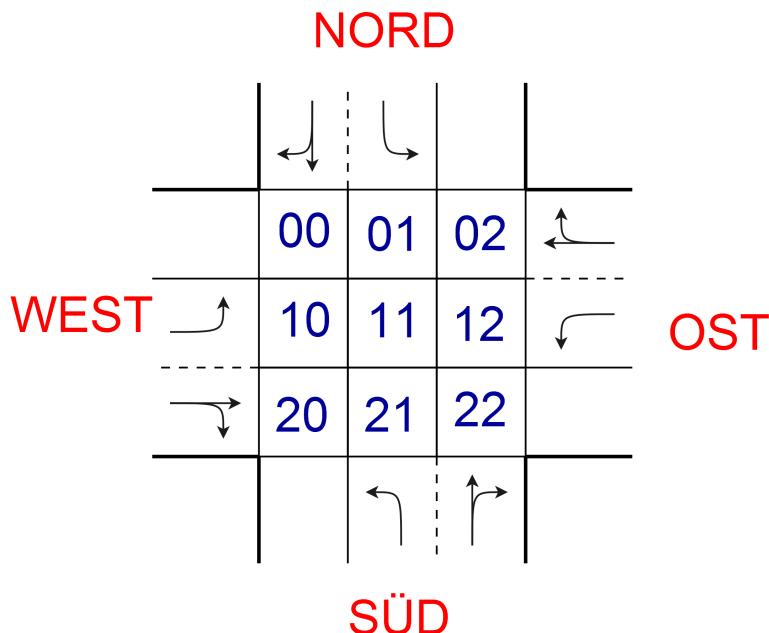


Abbildung 3.56: Segmentübersicht der mittleren Ampel

### 3.7.3.3 Die Abbiegelogik

In Abbildung 3.57 ist wieder das grundlegende Abbiegeverhalten der Fahrzeuge an der Kreuzung skizziert.

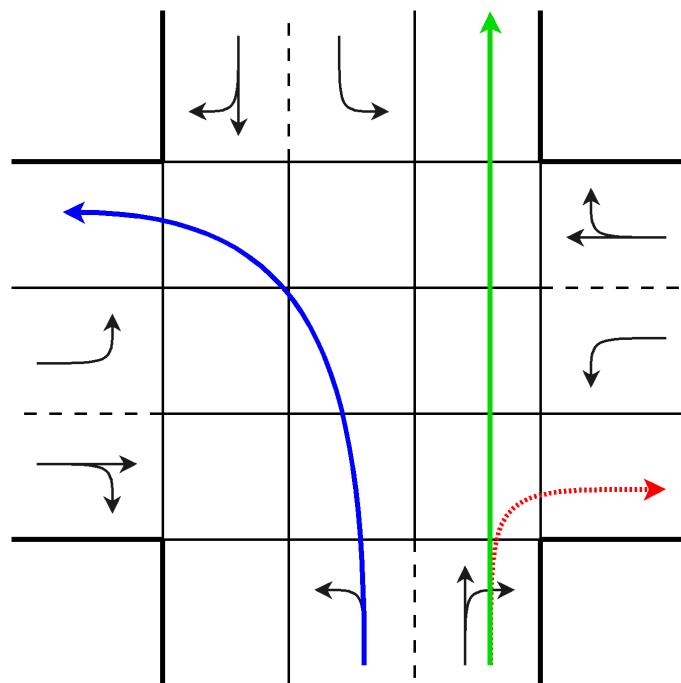


Abbildung 3.57: Kurzübersicht über die Abbiegemöglichkeiten

#### 3.7.3.3.1 Das Problem der entgegenkommenden Linksabbieger

Wenn wir jetzt die Route für die Linksabbieger aus dem Modell der kleinen Ampel auf unsere mittelgroße Ampel übertragen, erhalten wir für die Linksabbieger eine Route wie in Abbildung 3.58 zu sehen ist. Diese Route hat das Problem, dass wenn zwei Linksabbieger aus den gegenüberliegenden Richtungen entgegen kommen (wie dies in Abbildung 3.59 zu sehen ist) es zu einer Kollision der Routen der beiden Linksabbieger kommt (siehe Abbildung 3.60). Durch die Änderung der Route für die Linksabbieger (wie in Abbildung 3.61 zu sehen) wird dieses Problem gelöst, ohne dass andere Fahrzeuge derselben Phase in Mitleidenschaft gezogen werden.

Normales Linksabbiegermodell:	Problem: Zwei Linksabbieger
Abbildung 3.58: Die normale Route für Linksabbieger	Abbildung 3.59: Zwei entgegenkommende Linksabbieger
Kollision der beiden Linksabbieger	Lösung des Problems
Abbildung 3.60: Kollision im Modell	Abbildung 3.61: Lösung des Problems

Durch diese Veränderung der Route für die Linksabbieger erhalten wir die in Abbildung 3.62 gezeigten Routen.

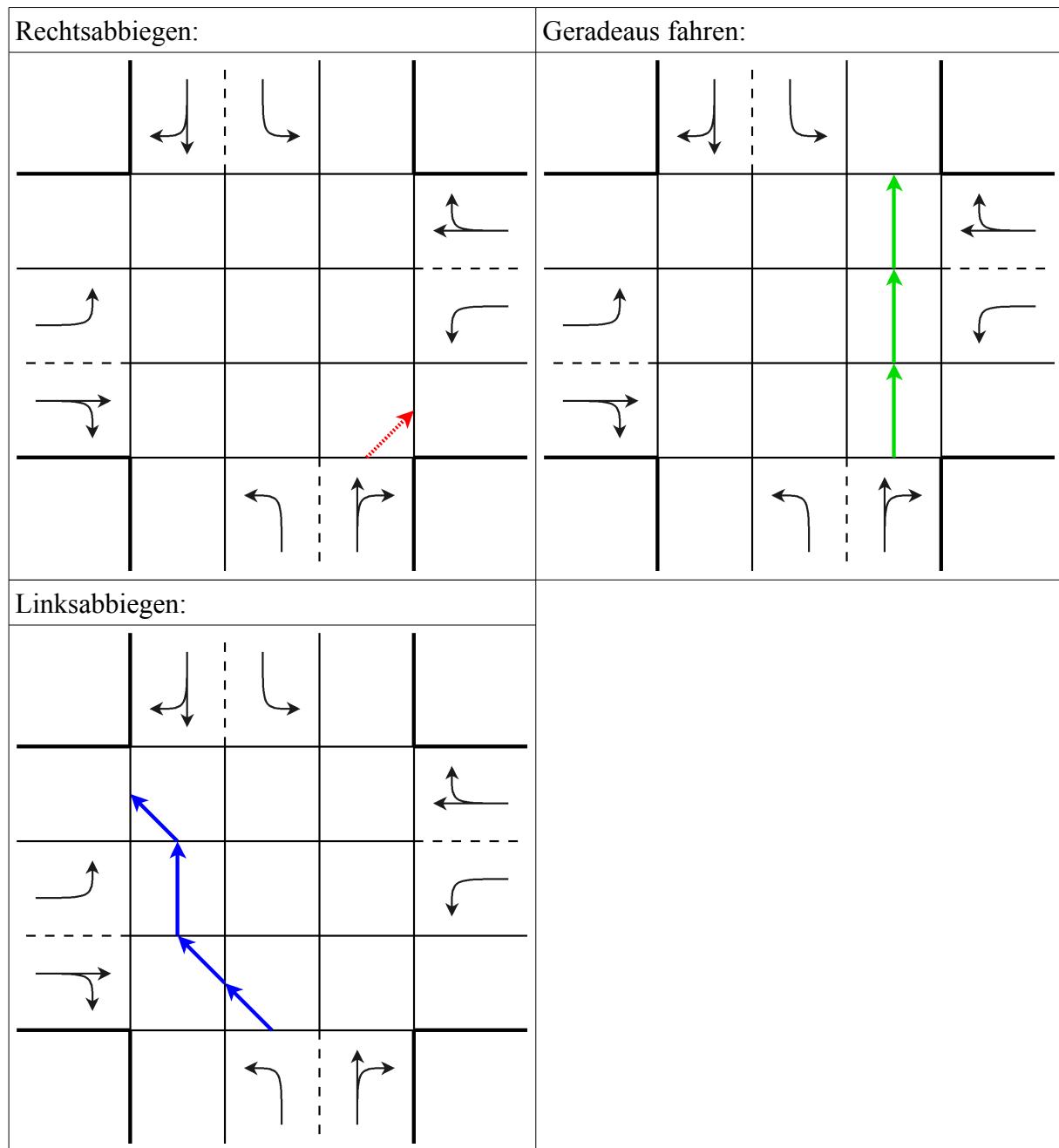


Abbildung 3.62: Segmentnutzungsübersicht beim Abbiegen

### 3.7.3.4 Einschränkungen

- Durch die Ampelphasen und die Kopplung der Spur für Rechtsabbieger und für Geradeausfahrer droht die Ampel ineffizient zu werden (siehe dazu auch Kapitel 3.7.4 „Die mittelgroße, optimierte Ampel,,“).

### 3.7.4 Die mittelgroße, optimierte Ampel

Die optimierte Ampel ist ein Versuch gewisse Nachteile der normalen, mittelgroßen Ampel zu beheben. Diese hat u.a. das Problem, dass ihre Ampelphasen in vielen Fällen ineffizient arbeiten. So dürfen alle Spuren einer Ampel während einer Grünphase gleich lang in die Kreuzung einfahren. Dabei muss die linke Spur nur die Linksabbieger bewältigen, wohingegen die rechte Spur aber die Rechtsabbieger und die Geradeausfahrer abfertigen muss. Wenn man diesen Aspekt betrachtet wird einem klar, dass die rechte Spur innerhalb einer Grünphase wesentlich mehr Fahrzeuge abfertigt als die linke Spur. Somit ist die linke Spur nicht optimal ausgelastet.

Dieses Problem wollen wir mithilfe veränderter Ampelphasen in dieser Kreuzung angehen, um so den Fahrzeugdurchsatz der Kreuzung zu verbessern.

Da die optimierte Ampel sich nur durch andere Ampelphasen von der normalen, mittelgroßen Ampel unterscheidet betrachten wir im Weiteren nur noch diesen Unterschied.

#### 3.7.4.1 Die optimierte Ampel in der graphischen Oberfläche

Die optimierte, mittelgroße Ampel wird in der graphischen Oberfläche, wie in Abbildung 3.63 durch ein O innerhalb des Kreuzungssymbols angedeutet.

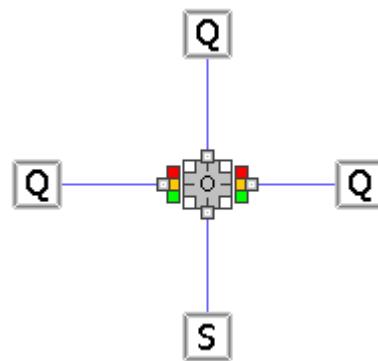


Abbildung 3.63: Die mittelgroße, optimierte Ampel in der Simulation

Im Einstellungsdialog der Ampel (siehe Abbildung 3.64) fällt nur die Anzeige der veränderten Ampelphasen auf.

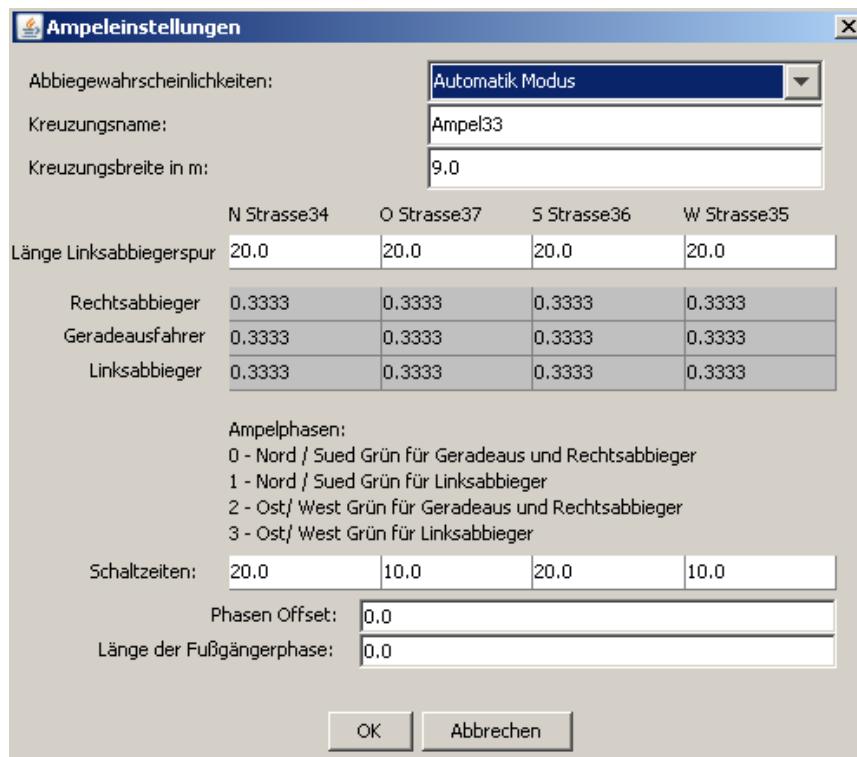


Abbildung 3.64: Der Einstellungsdialog der mittelgroßen, optimierten Ampel

### 3.7.4.2 Die Ampelphasen der optimierten Ampel

In Abbildung 3.65 sind die beiden grundlegenden Phasentypen der optimierten Ampel abgebildet. Diese bestehen aus:

- Links im Bild: Grünphase für die Geradeausfahrer und Rechtsabbieger der gegenüberliegenden Richtungen.
- Rechts im Bild: Grünphase für die Linksabbieger der gegenüberliegenden Richtungen.

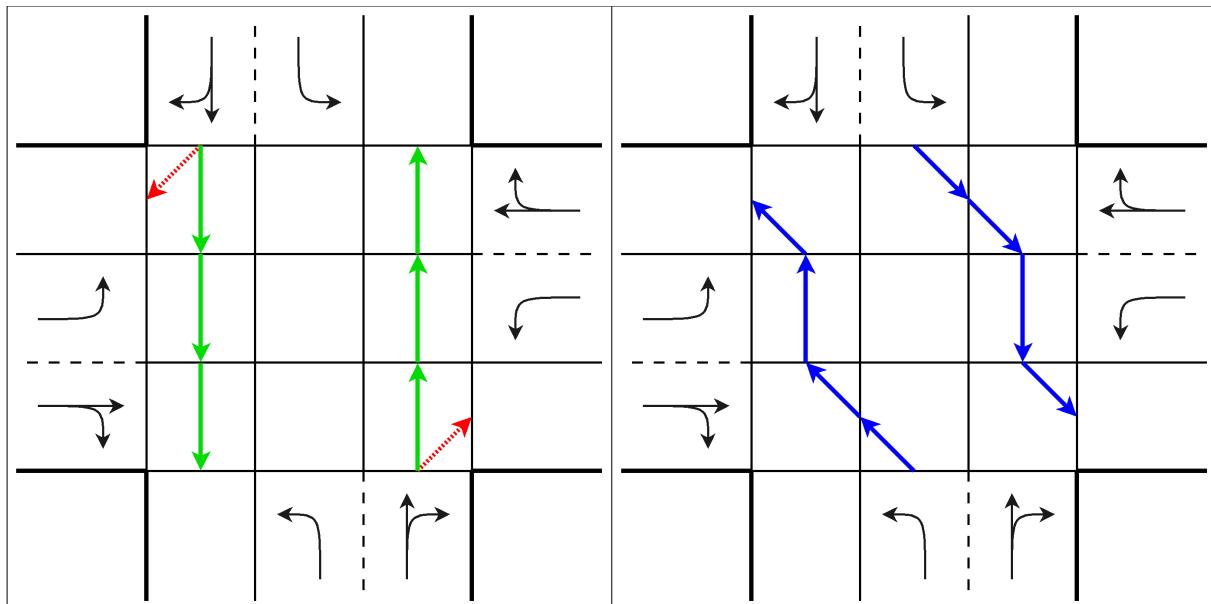


Abbildung 3.65: Die Phasen der mittelgroßen, optimierten Ampel

So kommen wir dann auch auf unsere neuen Ampelphasen:

1. Grün für Geradeaus und Rechtsabbieger aus dem Norden und Süden.
2. Grün für Linksabbieger aus dem Norden und Süden.
3. Grün für Geradeaus und Rechtsabbieger aus dem Osten und Westen.
4. Grün für Linksabbieger aus dem Osten und Westen.
5. Grün für Fußgänger.

### 3.7.4.3 Einschränkung

- Die optimierte Ampel ist nur besser als die normale Ampel, wenn das Verkehrsaufkommen der gegenüberliegenden Richtungen in etwa gleich groß ist.

## 3.7.5 Die große Ampel

Zuletzt betrachten wir schließlich noch das Modell der großen Ampel.

### 3.7.5.1 Die große Ampel in der graphischen Oberfläche

Wie in Abbildung 3.66 zu sehen, kennzeichnet ein großes G die große Ampel innerhalb der graphischen Oberfläche.

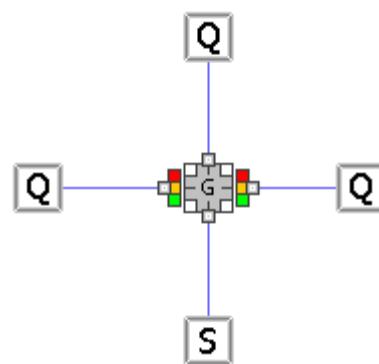


Abbildung 3.66: Die große Ampel in der Simulation

Der Einstellungsdialog aus Abbildung 3.67 ist eigentlich auch schon von der mittelgroßen Ampel bekannt und bedarf keinerlei Erklärung.

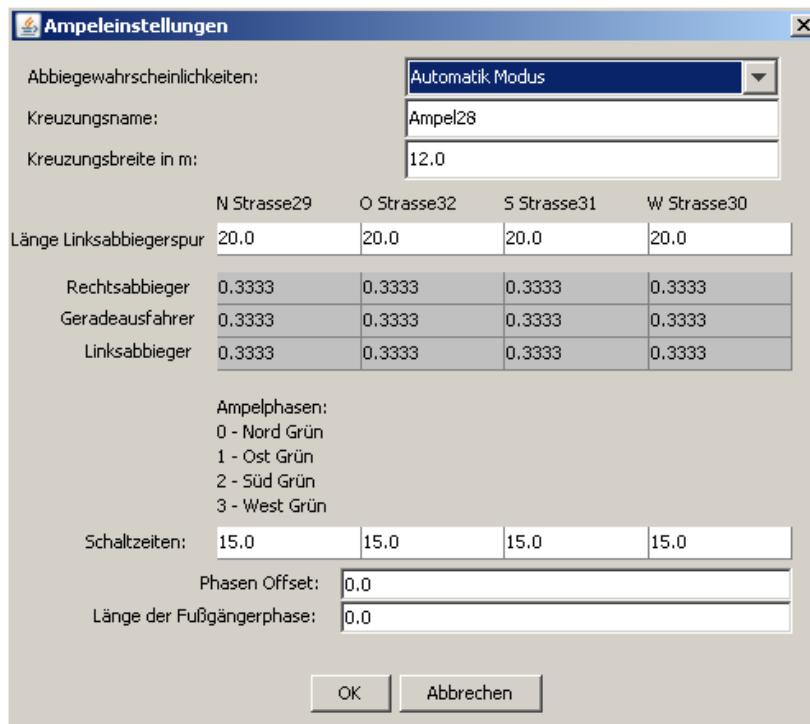


Abbildung 3.67: Der Einstellungsdialog der großen Ampel

### 3.7.5.2 Das Modell

Wie in Abbildung 3.68 zu sehen, ist das Segmentmodell der großen Ampel eigentlich nur die logische Erweiterung des Modells der mittelgroßen Ampel um eine zusätzliche Abbiegespur. Somit hat das Modell jetzt insgesamt 16 Segmente und jede Abbiegerichtung hat ihre eigene Spur.

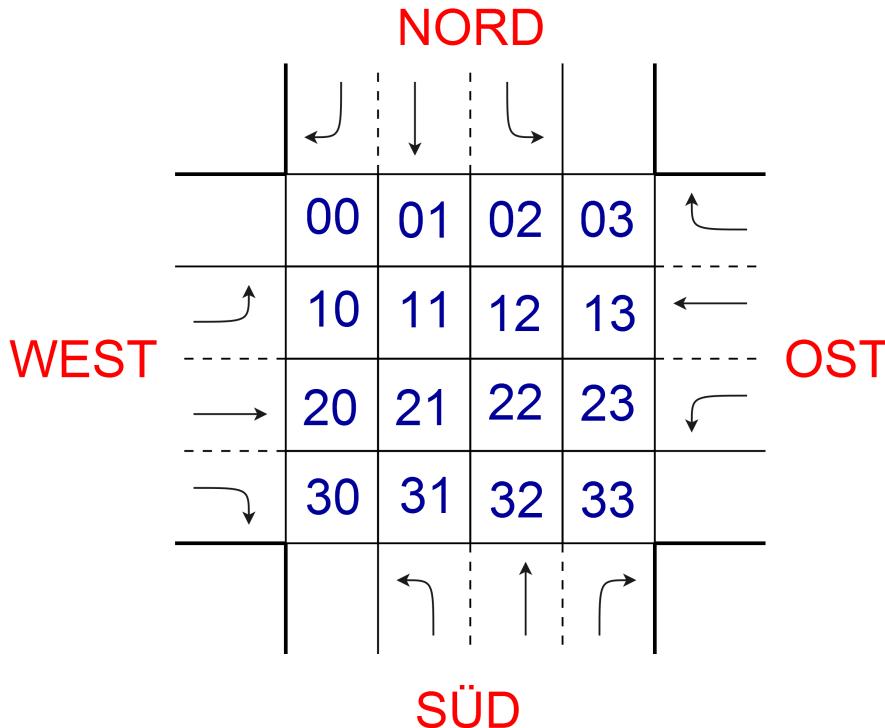


Abbildung 3.68: Segmentübersicht der großen Ampel

### 3.7.5.3 Die Abbiegelogik

Wie aus Abbildung 3.69 und Abbildung 3.70 ersichtlich, hat sich auch bei der Route der einzelnen Abbiegerichtungen nicht sonderlich viel getan. Vor allem wurden die Routen an die größere Kreuzung angepasst und durch die veränderte Lage der gegenüberliegenden linken Abbiegespuren konnte das Linksabbiegen wieder vereinfacht werden.

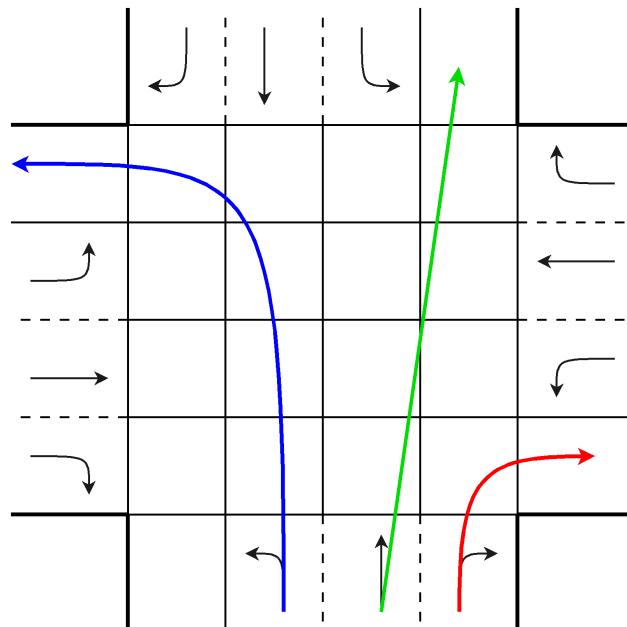


Abbildung 3.69: Kurzübersicht über die  
Abbiegemöglichkeiten

Rechtsabbiegen:	Geradeaus fahren:

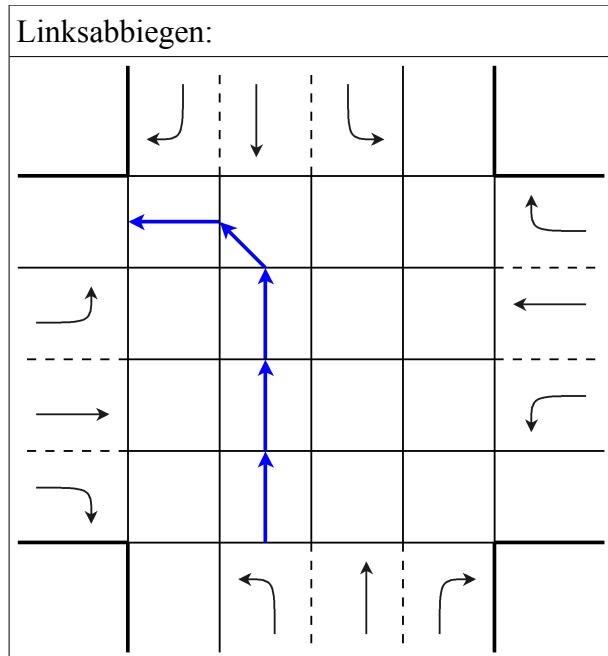


Abbildung 3.70: Übersicht der Nutzung von Segmenten beim Abbiegen

### 3.7.5.4 Einschränkungen

- Die Längen der Abbiegespuren sind für alle Spuren genau gleich lang. Es ist also nicht möglich, dass z.B. die linke Spur kürzer ist als die rechte Spur.
- Es sind für diesen Ampeltyp noch viele weitere Ampelphasenmodelle denkbar, welche nicht implementiert wurden.

## 3.8 Die Vorfahrtstraßenkreuzung

In diesem Abschnitt wollen wir auf das Modell einer Vorfahrtstraßenkreuzung eingehen. Der nun folgende Abschnitt erklärt grob den Modellaufbau. Die detaillierten Ereignisse und Hilfsfunktionen können als Pseudocode im Anhang F – Pseudocode Vorfahrtkreuzung betrachtet werden.

### 3.8.1 Die Vorfahrtstraßenkreuzung in der graphischen Oberfläche

In der Abbildung 3.71 ist eine abbiegende Vorfahrtstraße von Westen nach Süden zu sehen. Die orangen Linien innerhalb des Kreuzungssymbols signalisieren den Verlauf der Vorfahrtstraße.

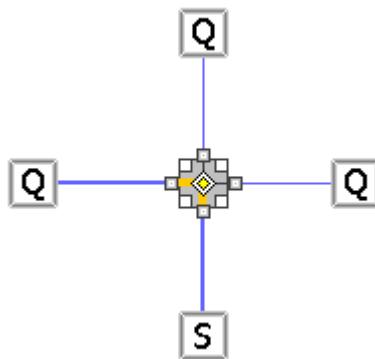


Abbildung 3.71: Eine abbiegende Vorfahrtstraße in der Simulation

Die einzige Besonderheit im Einstellungsdialog der Vorfahrt-Kreuzung ist die Einstellung zum Vorfahrtstraßenverlauf (siehe Abbildung 3.72). Standardmäßig ist der Automatikmodus für diese Einstellung aktiviert. Dieser Automatikmodus legt den Verlauf der Vorfahrtstraße entlang der zwei am meisten befahrenen Straßen fest. Dazu wird der bei den Straßen hinterlegte voraussichtliche Fahrzeugdurchsatz verwendet (siehe dazu Kapitel 3.2.3 „Die Straße“).

Alternativ kann der Verlauf der Vorfahrtstraße auch über die Auswahlbox gewählt werden.

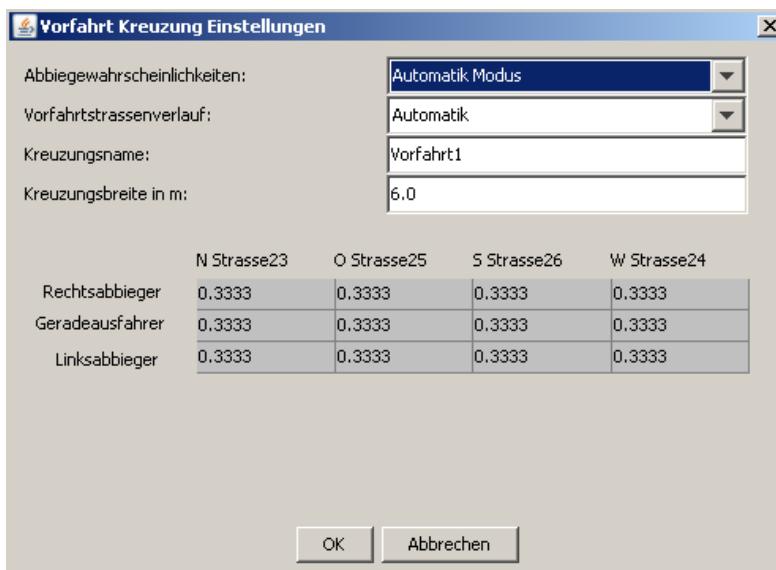


Abbildung 3.72: Der Einstellungsdialog der abbiegenden Vorfahrtstraße

### 3.8.2 Das Modell

Kommen wir nun zu dem eigentlichen Modell der Vorfahrtkreuzung. Dieses Modell ist wie in Abbildung 3.73 zu erkennen wesentlich komplexer aufgebaut als die anderen Kreuzungsmodelle. So erinnern die zwei ineinander geschachtelten Segmentkreise eher an einen Kreisverkehr als an eine normale Kreuzung. Allerdings ist so ein komplexes Modell notwendig, um die verschiedenen Kreuzungssituationen nachbilden zu können.

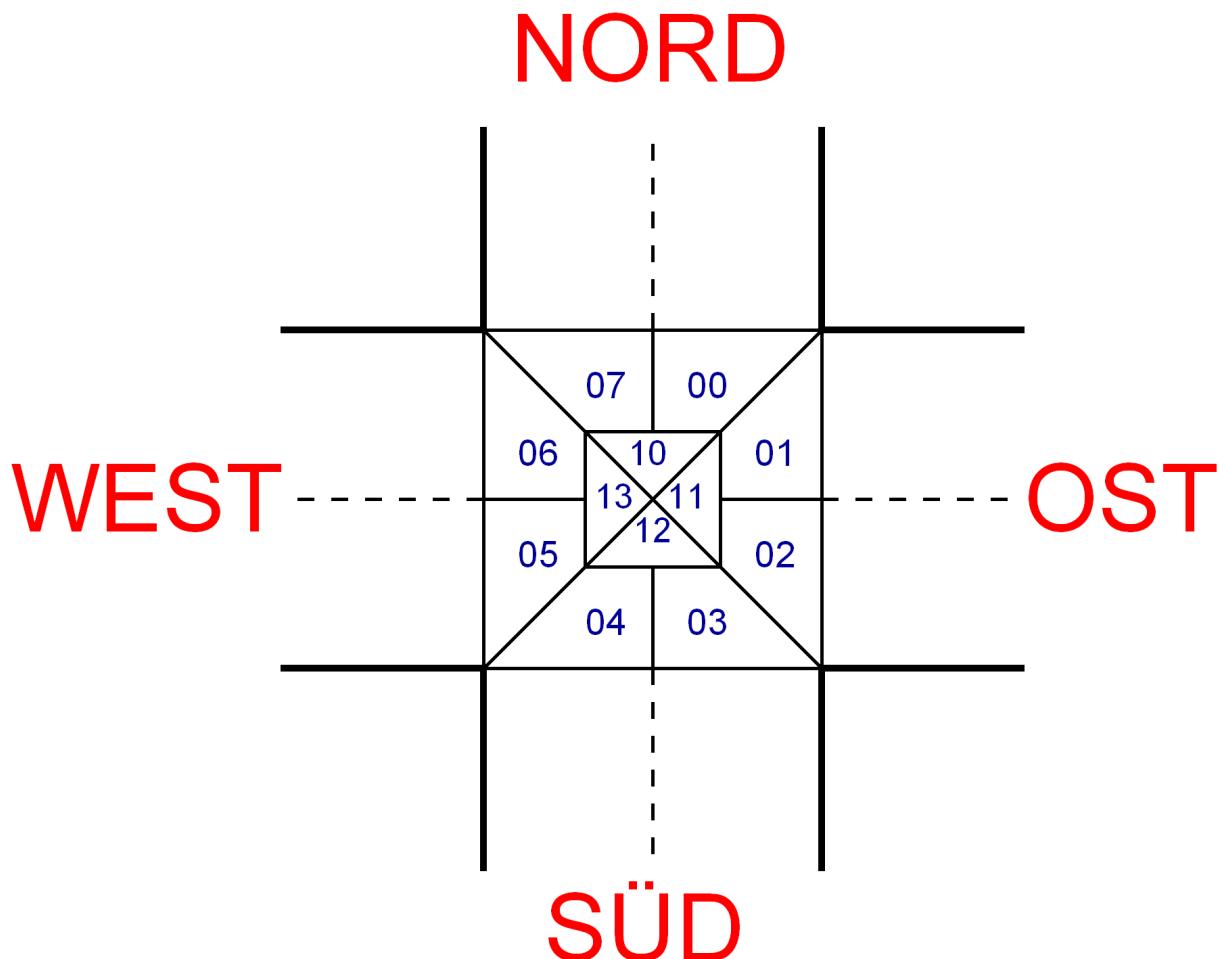


Abbildung 3.73: Segmentübersicht einer Vorfahrt-Kreuzung

Nun folgend wollen wir uns die Modellierungsprobleme, welche das Verwenden eines einfachen Modells verhindern, genauer betrachten.

### 3.8.2.1 Das Problem der entgegenkommenden Linksabbieger

Die deutsche Straßenverkehrsordnung sieht vor, dass zwei entgegenkommende Fahrzeuge, welche beide an derselben Kreuzung jeweils nach links abbiegen wollen, gleichzeitig voreinander abbiegen müssen. Dieses gleichzeitige Linksabbiegen ist mit einem einfachen Modell (siehe Abbildung 3.74) nicht möglich. Es kommt dabei zu einer Überschneidung der beiden Routen der Linksabbieger, was zu einer gegenseitigen Blockade führt.

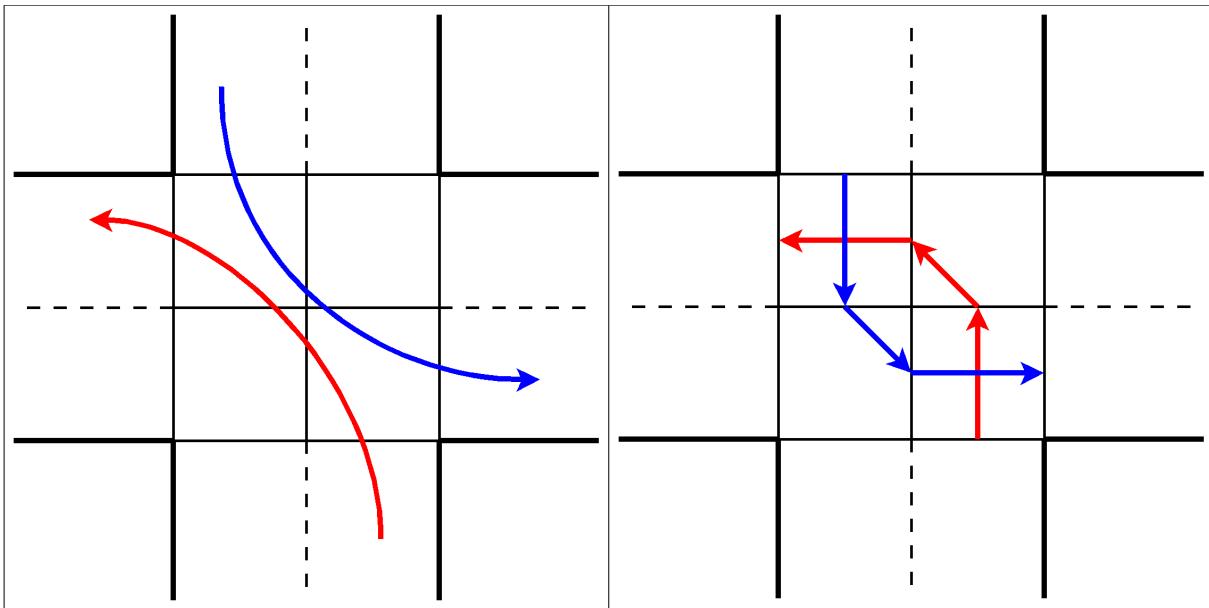


Abbildung 3.74: Linksabbiegerproblem beim Vier-Segment-Modell

### 3.8.2.2 Das Problem der gleichzeitigen Rechts- und Linksabbieger

Weiterhin gibt es denn Fall, dass wie in Abbildung 3.75 es gleichzeitig einen Linksabbieger und einen Rechtsabbieger (die beiden roten Pfeile links) gibt, welche in die Kreuzung einfahren wollen. Bei einer Kreuzung mittlerer Größe ist dies kein Problem. Im einfachen Vier-Segment-Modell funktioniert dies allerdings nicht, weil sowohl der Linksabbieger, wie auch der Rechtsabbieger dasselbe Segment in Anspruch nehmen (Abbildung 3.75 rechts).

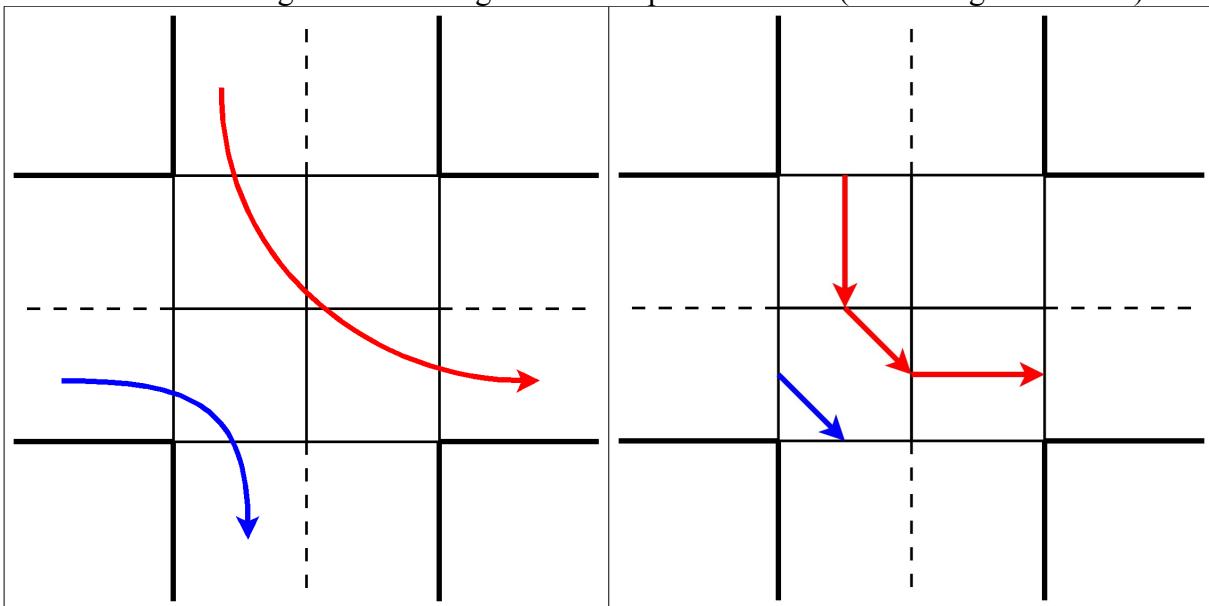


Abbildung 3.75: Problem Links- und Rechtsabbieger

### 3.8.2.3 Die Lösung anhand des komplexeren Kreuzungsmodells

Durch das komplexere Zwölf-Segment-Modell können wir jetzt die angeführten Probleme lösen.

### 3.8.2.3.1 Lösung für das Problem der entgegenkommenden Linksabbieger

Wie in Abbildung 3.76 zu erkennen, löst das neue Modell das Problem der entgegenkommenden Linksabbieger. Dies gelingt über das Benutzen der inneren Segmente des Modells.

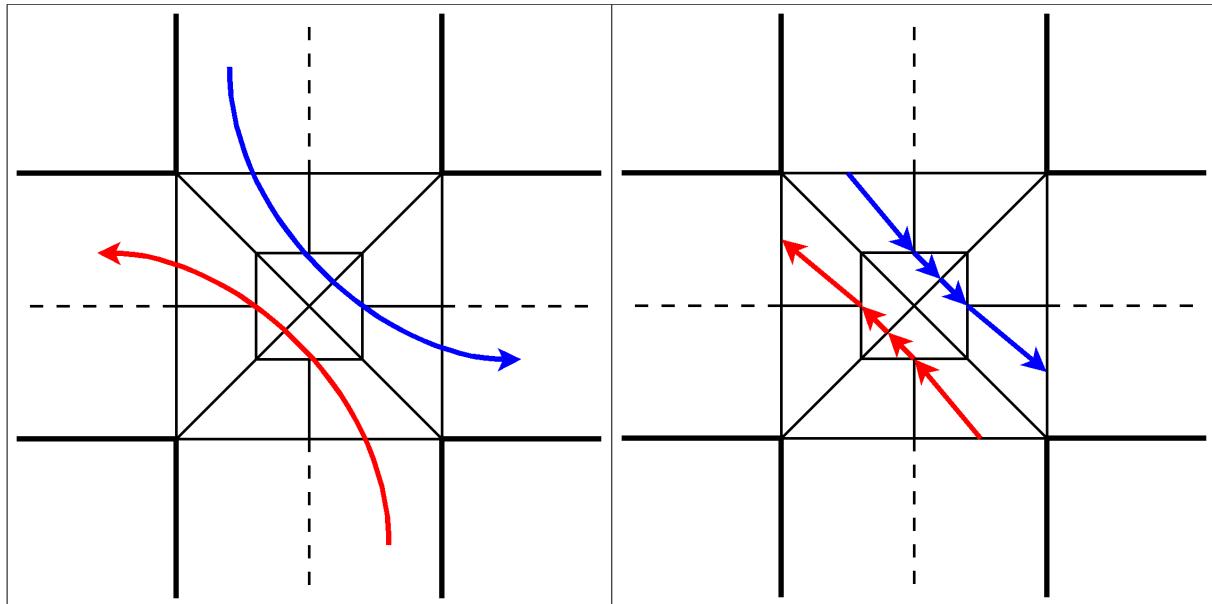


Abbildung 3.76: Lösung des Linksabbiegerproblems

### 3.8.2.3.2 Die Lösung des Problems der gleichzeitigen Rechts- und Linksabbieger

Auch dieses Problem ist durch die geänderte Route des Linksabbiegers gelöst. Es kommt also nicht mehr zur gegenseitigen Blockade der Fahrzeuge (vergleiche Abbildung 3.77).

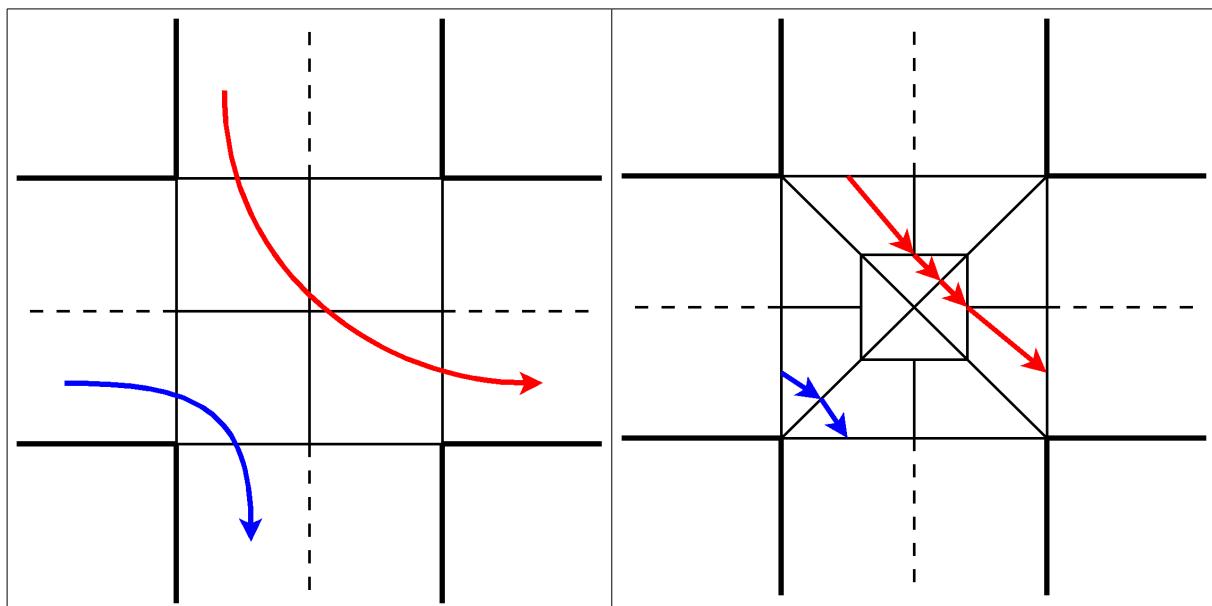


Abbildung 3.77: Lösung des Problems des Links- und Rechtsabbieger

### 3.8.3 Die Abbiegelogik

Nachdem das grundlegende Modellkonzept steht, bleibt die Frage offen, welche Route die Fahrzeuge innerhalb einer Kreuzung zurücklegen müssen, um an ihr Ziel zu kommen. Dazu werden erstmals alle verschiedenen Abbiegemöglichkeiten im zugrunde liegenden Modell betrachtet (siehe Abbildung 3.78).

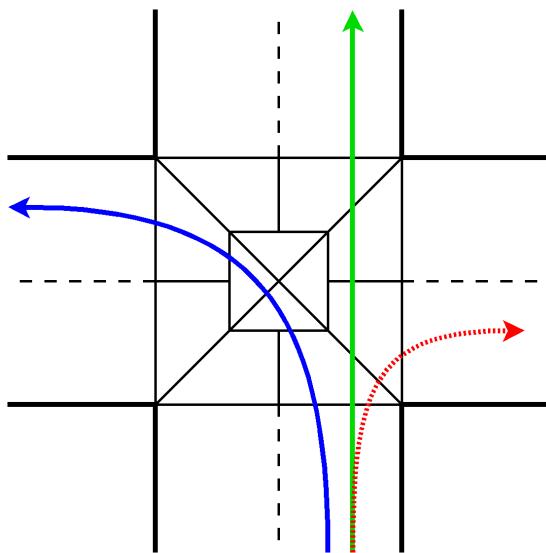


Abbildung 3.78: Abbiegen in der Kreuzung

Durch diese Überlegung kommen wir nun zu den folgenden Routen, welche in der Tabelle 3.4 abgebildet sind. Zu beachten ist, dass die vier inneren Segmente nur beim Linksabbiegen durchfahren werden. Als Problem tritt jetzt auf, dass die Segmente des Kreuzungsmodells nicht mehr alle gleich lang sein können. Dies ist dadurch geschuldet, dass die Linksabbieger eine größere Strecke zurücklegen müssen, als dies bei gleicher Länge aller Segmente der Fall wäre. Somit würden die Linksabbieger eine viel zu kurze Strecke innerhalb der Kreuzung zurücklegen. Um dieses Problem zu umgehen und die längere Strecke einer Linkskurve zu simulieren, wurde einfach die Segmentlänge der inneren Segmente so vergrößert, so dass im Endeffekt wieder dieselbe Kurvenlänge wie in der Realität erreicht wird.

Rechtsabbiegen	Geradeaus fahren	Linksabbiegen

Tabelle 3.4: Übersicht der Segmentnutzung nach Abbiegerichtung

### 3.8.4 Die Einfahrtregeln

Beim Einfahren eines Fahrzeugs in eine Vorfahrtkreuzung muss auf viele andere Fahrzeuge Rücksicht genommen werden. Es kommt dabei nicht nur auf die Richtung an, aus welcher die anderen Fahrzeuge in die Kreuzung einfahren, sondern auch darauf, wie diese abbiegen und wie der Verlauf der Vorfahrtstraße ist.

Um alle diese Anforderungen zu erfüllen, mussten alle möglichen Szenarien betrachtet werden, um anschließend eine Entscheidungstabelle mit den daraus resultierenden Regeln zu implementieren. Wir betrachten nun die aus diesen Überlegungen folgenden Vorfahrtregeln, die sowohl in Form von Entscheidungstabellen, wie auch als Abbildungen dargestellt werden. Alle diese Betrachtungsweisen haben gemeinsam, dass das Fahrzeug von unten in die Kreuzung einfährt.

In den Entscheidungstabellen gelten folgende vereinfachte Schreibweisen:

- R steht für rechts Abbiegen
- G steht für geradeaus fahrende Fahrzeuge
- L steht für links Abbiegen
- Die Zeilen definieren die verschiedenen Abbiegemöglichkeiten des Fahrzeugs, welches gerade in die Kreuzung einfahren möchte.
- Die Spalten Links, Oben, Rechts stehen für die aus dieser Richtung kommenden Fahrzeuge.
- Ein X markiert, dass das einfahrende Fahrzeug diesem anderen Fahrzeug die Vorfahrt gewähren muss.

In den Abbildungen gelten folgende Definitionen:

- Der Abbiegewunsch des betrachteten Fahrzeugs wird durch den durchlaufenden roten Pfeil signalisiert.
- Die gestrichelten blauen Pfeile sind die Fahrzeuge, welche ein Vorfahrtsrecht vor dem einfahrenden Fahrzeug haben.
- Der Verlauf der Vorfahrtstraße wird durch die Vorfahrtstraßenschilder symbolisiert.
- Eine Zeile in der Entscheidungstabelle entspricht einer Teil-Abbildung (Zeile R => Abbildung Links / Zeile G => Abbildung Mitte / Zeile L => Abbildung Rechts).

#### 3.8.4.1 Einfahrt in eine Vorfahrtstraße

In den nun folgenden drei Tabellen betrachten wir die Einfahrt eines Fahrzeugs auf einer nicht vorfahrtberechtigten Straße.

##### Vorfahrtstraße Oben/Links

Zuerst betrachten wir die Einfahrt eines Fahrzeugs in eine von Oben kommende Vorfahrtstraße, welche in die linke Straße mündet. Das Regelwerk für Rechtsabbiegen, Geradeaus fahren und Linksabbiegen ist in der Tabelle 3.5 und der Abbildung 3.79 abgebildet.

	Links			Oben			Rechts		
	R	G	L	R	G	L	R	G	L
R		X				X			
G	X	X				X	X	X	X
L	X	X	X	X			X	X	

Tabelle 3.5: Einfahrtregeln: Einfahrt in Oben/Links Vorfahrtstraße

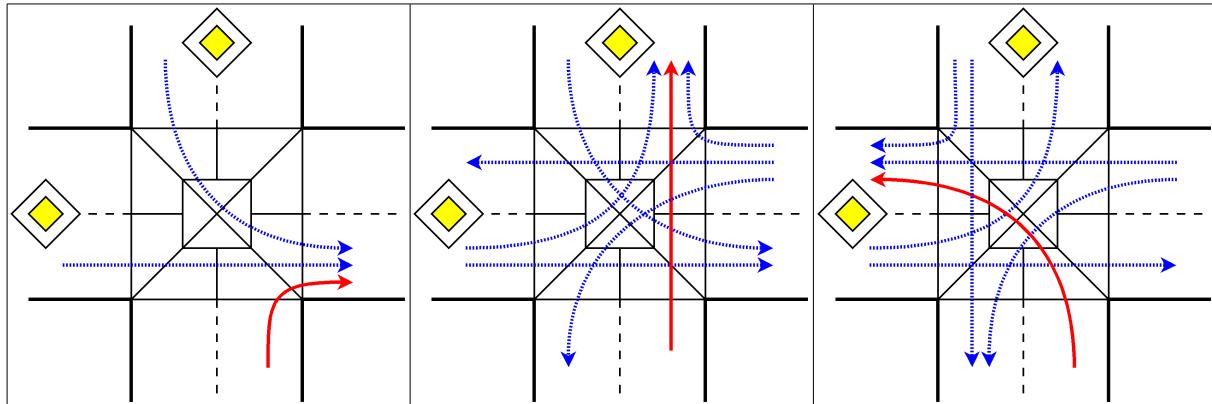


Abbildung 3.79: Einfahrtregeln: Einfahrt in Oben/Links Vorfahrtstraße

### Vorfahrtstraße Links/Rechts

Nun betrachten wir in der Tabelle 3.6 und der Abbildung 3.80 dieselbe Situation, nur dass diesmal die Vorfahrtstraße gerade von Links nach Rechts verläuft.

	Links			Oben			Rechts		
	R	G	L	R	G	L	R	G	L
<b>R</b>		X							
<b>G</b>		X	X				X	X	X
<b>L</b>	X	X	X	X			X	X	X

Tabelle 3.6: Einfahrtregeln: Einfahrt in Links/Rechts Vorfahrtstraße

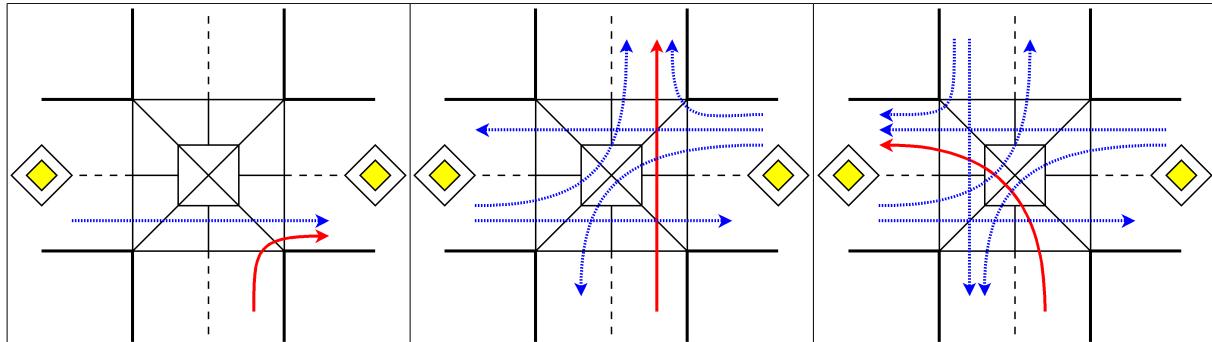


Abbildung 3.80: Einfahrtregeln: Einfahrt in Links/Rechts Vorfahrtstraße

### Vorfahrtstraße Oben/Rechts

Schließlich bleibt noch die Möglichkeit, dass die Vorfahrtstraße von Oben nach Rechts verläuft, genauso wie dies in der Abbildung 3.81 und der Tabelle 3.7 abgebildet ist.

	Links			Oben			Rechts		
	R	G	L	R	G	L	R	G	L
<b>R</b>							X		
<b>G</b>							X	X	X
<b>L</b>				X	X		X	X	X

Tabelle 3.7: Einfahrtregeln: Einfahrt in Oben/Rechts Vorfahrtstraße

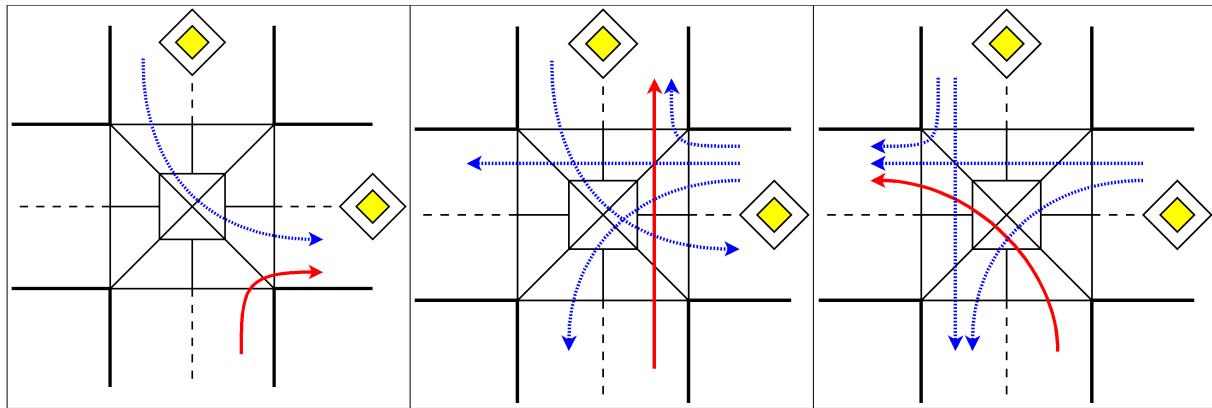


Abbildung 3.81: Einfahrtregeln: Einfahrt in Oben/Rechts Vorfahrtstraße

### 3.8.4.2 Abbiegeregeln für Abbiegen aus der Vorfahrtstraße

Nun müssen wir nur noch die Abbiegeregeln für das Abbiegen aus einer Vorfahrtstraße betrachten.

#### Vorfahrtstraße biegt nach links ab

Zuerst betrachten wir, in Tabelle 3.8 und Abbildung 3.82 den Fall, dass die Vorfahrtstraße nach links abbiegt.

Links			Oben			Rechts		
R	G	L	R	G	L	R	G	L
R								
G								
L								

Tabelle 3.8: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Links

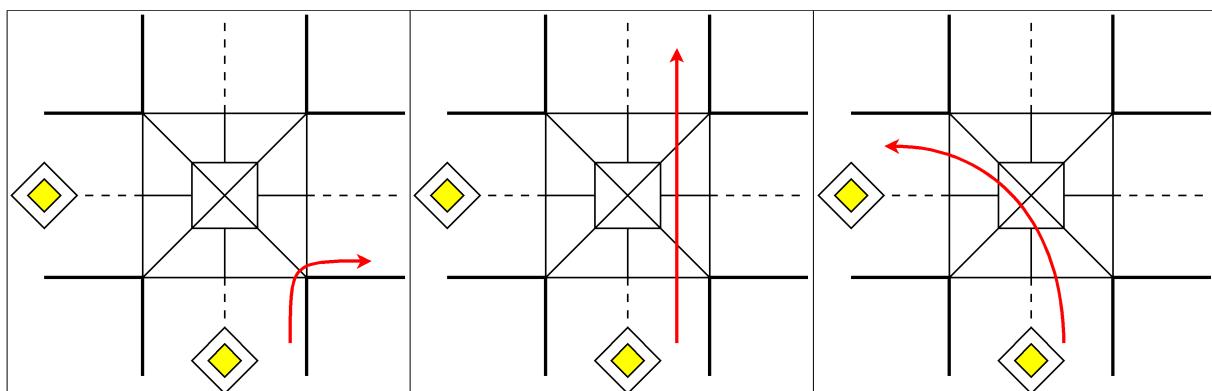


Abbildung 3.82: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Links

#### Vorfahrtstraße führt geradeaus

Im Weiteren betrachten wir in der Abbildung 3.83 und der Tabelle 3.9 den Fall, dass die Vorfahrtstraße weiter geradeaus führt.

	Links			Oben			Rechts		
	R	G	L	R	G	L	R	G	L
R									
G									
L				X	X				

Tabelle 3.9: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Oben

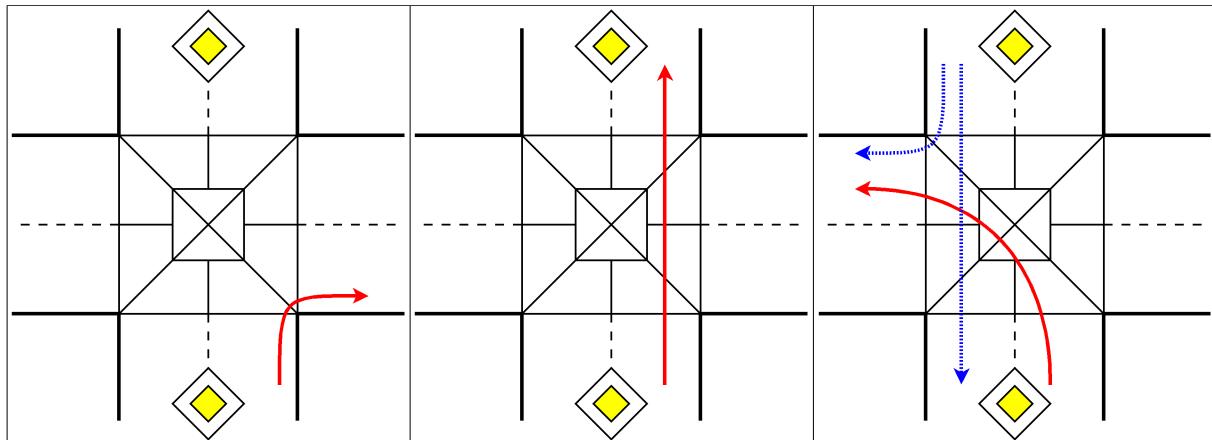


Abbildung 3.83: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Oben

### Vorfahrtstraße biegt nach rechts ab

Zuletzt betrachten wir noch den Fall einer nach rechts abbiegenden Vorfahrtstraße (siehe Tabelle 3.10 und Abbildung 3.84).

	Links			Oben			Rechts		
	R	G	L	R	G	L	R	G	L
R									
G							X	X	X
L							X	X	

Tabelle 3.10: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Rechts

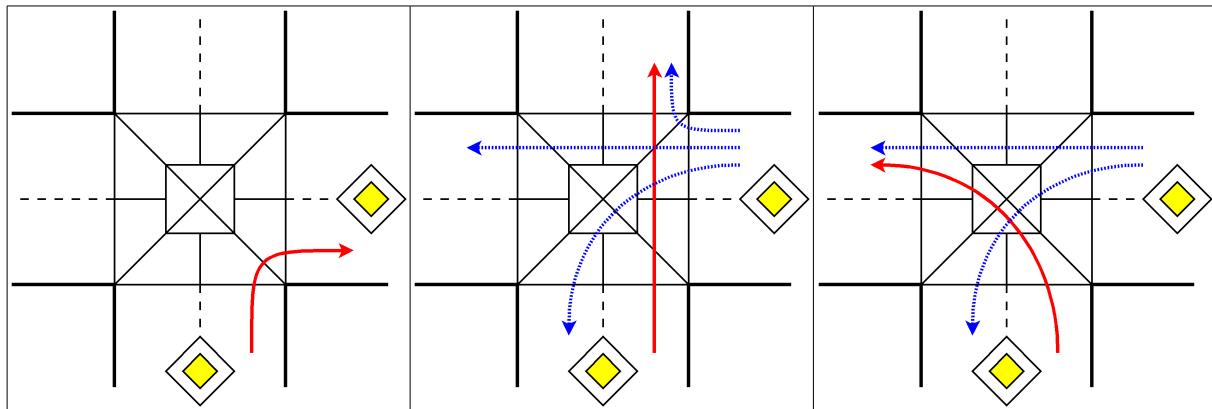


Abbildung 3.84: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Rechts

### 3.8.5 Besonderheiten

- Die Segment 00 bis 07 haben eine Segmentlänge welche  $\frac{1}{4}$  der Kreuzungsbreite entspricht, wohingegen die inneren Segmente 10 bis 13 sogar  $\frac{1}{3}$  der Kreuzungsbreite entspricht, um die längere Wegstrecke bei der Kurvenfahrt zu simulieren.
- Um zu verhindern, dass es durch die komplizierten Einfahrtregeln zu einer gegenseitigen Blockade der Fahrzeuge auf der Kreuzung kommt, muss jedes Fahrzeug vor der Einfahrt kontrollieren, ob alle Segmente seiner Route frei sind. Ist dies der Fall, reserviert das Fahrzeug diese Segmente. So kann kein anderes Fahrzeug dieselben Segmente durchfahren. Beim Verlassen eines Segments wird diese Reservierung wieder aufgehoben.

### 3.8.6 Einschränkungen

- Dieses Modell kann den Platzmangel bei kleinen Kreuzungen nicht nachahmen. Es sind trotz der begrenzten Größe bei kleinen Kreuzungen alle Abbiegekombinationen möglich.
- Es ist nicht möglich, dass ein vorfahrtberechtigtes Fahrzeug auf sein Vorfahrtrecht verzichtet und ein anderes Fahrzeug in die Kreuzung einfahren lässt.

## 4 Die Fallstudien

In diesem Kapitel wollen wir kurz auf zwei verschiedenen Fallstudien eingehen, mit welchen wir die Fähigkeiten der Simulation verdeutlichen. Da allerdings diese Fallstudien mit geschätzten Werten initialisiert wurden, kann man die Ergebnisse dieser Fallstudien nicht eins zu eins in die Realität übertragen. Um die Realität innerhalb der Simulation abbilden zu können bedarf es umfangreicher Daten, mit welcher die Simulation gefüttert werden müsste. Da diese Daten mir beim Erstellen der Fallstudien nicht zur Verfügung standen, dienen die Fallstudien nur zum Aufzeigen allgemeiner Verkehrssphänomene und zur Demonstration der Simulationssoftware.

### 4.1 Fallstudie 1

In dieser Fallstudie, wollen wir denn allmorgendlichen Berufsverkehr nach Konstanz hinein nachbilden. Dafür wurde das in Abbildung 4.1 zu sehende Modell des Straßennetzes entworfen.

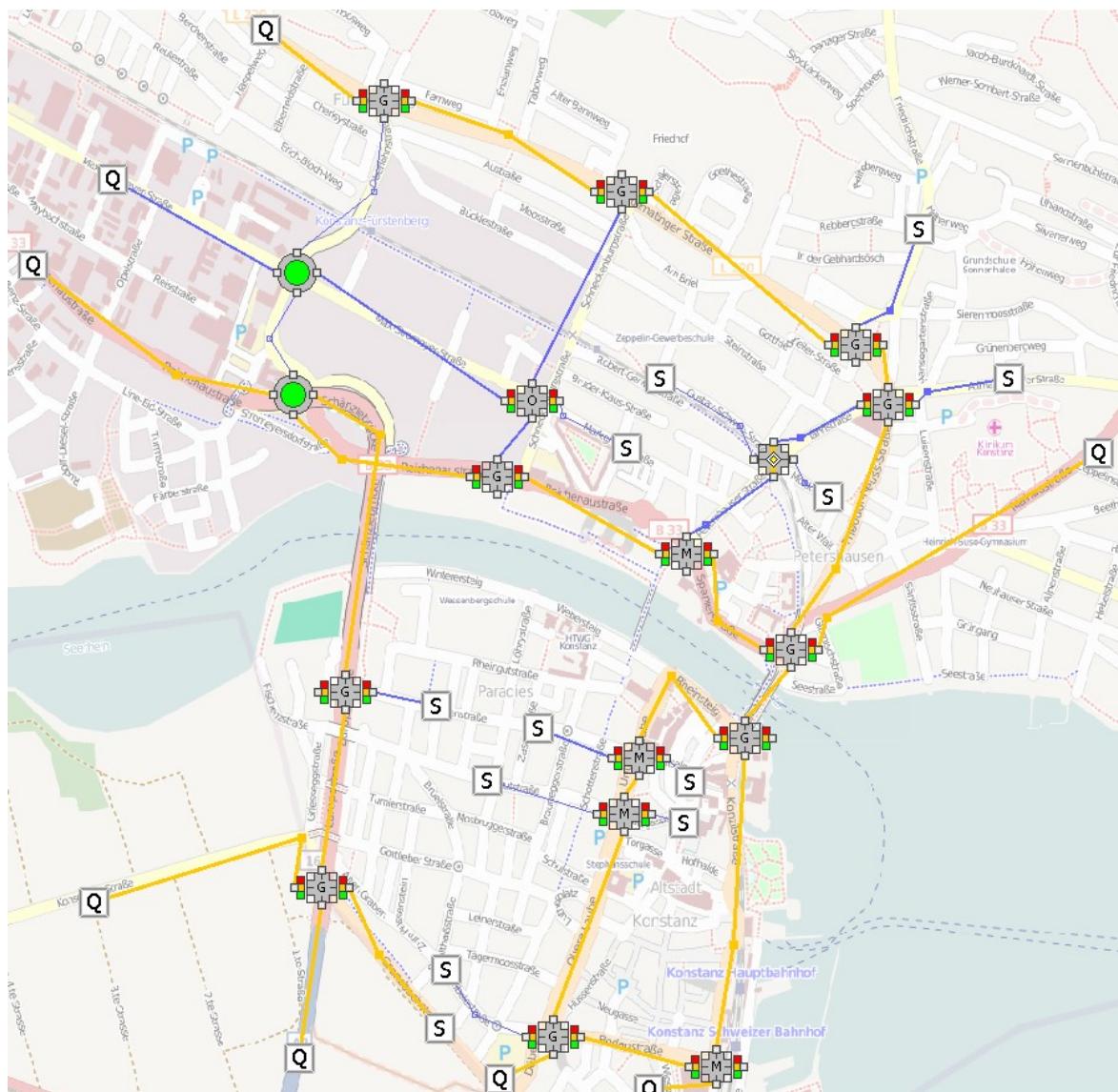


Abbildung 4.1: Modellaufbau. Hintergrundbild von [OSTM@]

### 4.1.1 Ergebnis

Nach mehreren Simulationsläufen können wir verschiedene Schlussfolgerungen ziehen, welche wir Beispielhaft an einem der Ergebnisse aufzeigen wollen. In Abbildung 4.2 ist angegeben wie viel Prozent der gesamten Simulationszeit die Straße aufgrund von Überlastung blockiert war. Straßen welche zum Simulationsende blockiert waren sind rot Hervorgehoben.

Durch dieses Ergebnis und durch das Betrachten der Simulation kommen wir zu folgenden Schlüssen:

- In der Reichenaustraße, der Mainaustraße und der Fürstenbergstraße stauen sich die Fahrzeuge nach Konstanz hinein bis zur Rheinbrücke. Allerdings verringert sich der Verkehr auf dem Weg zur Rheinbrücke sowohl auf der Fürstenbergstraße, sowie der Reichenaustraße immer mehr, je näher sie der Rheinbrücke kommen.
- Der stockende Verkehr löst sich innerhalb der Innenstadt erst in der Laube auf.
- Am schnellsten löst sich der stockende Verkehr entlang der Fürstenberg- / Wollmatinger Straße auf.

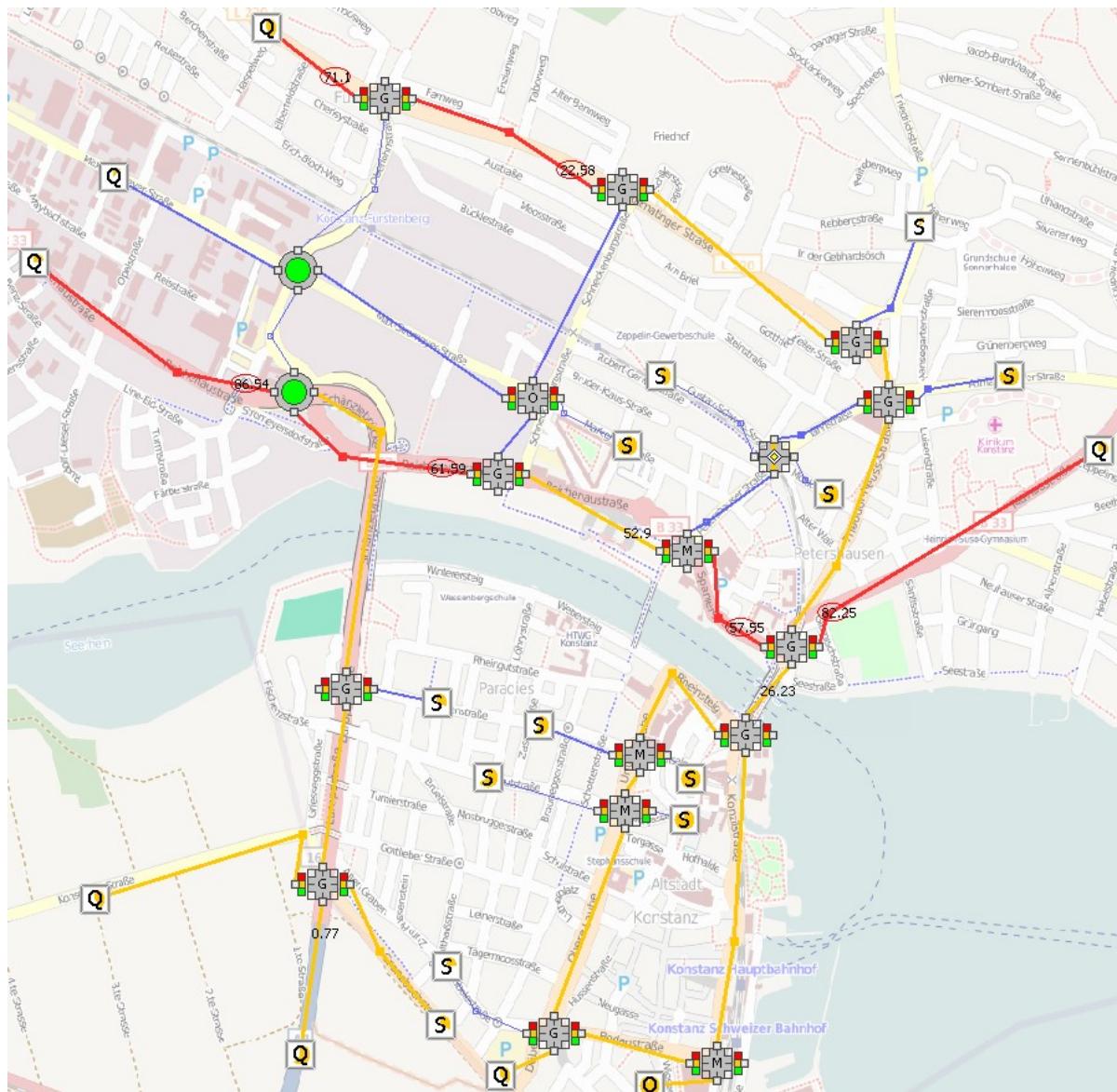


Abbildung 4.2: Stauanteil in Prozent während der Simulationszeit – Hintergrundbild von [OSTM@]

## 4.2 Fallstudie 2

In der zweiten Fallstudie wollen wir den abendlichen Berufsverkehr der stadtauswärts gesperrten Reichenastraße entlang des Flughafens simulieren.

Am 27.9.2008 berichtete der Südkurier folgendes:

**Konstanz**

Engpass auf B33 bis Dezember [9]

Die Auffahrt vom Schänzlekreisel auf die Neue Rheinbrücke wird komplett neu gebaut. Der Damm hat nicht gehalten. Bis Ende November oder Anfang Dezember soll die neue Zufahrt fertig sein. In den kommenden Wochen werden dann die Bauarbeiten an der B33-neu beim Landeplatz losgehen. Zunächst wird die Zufahrt zu den Flughafengebäuden neu gebaut.



Die Auffahrt vom Schänzlekreisel auf die Neue Rheinbrücke wird komplett neu gebaut. Arbeiter entfernen Hartschaumstoff, mit dem der Damm unter dem Asphalt aufgebaut ist. Die Konstruktion hielt nicht und muss erneuert werden.

Foto: Hanser

Konstanz - Die Auffahrt zur Brücke beim Schänzlekreisel hat nach der Freigabe im Mai 2007 nicht gehalten. "Wir tragen den gesamten Damm ab", berichtete Heinz Jenne von der Neubauleitung des Regierungspräsidiums in Singen. Der Damm unter dem Asphalt ist mit Hartschaumstoff aufgebaut. Als ein Teil absackte, sperrten die Straßenbauer eine Spur. Schließlich stellte sich heraus, dass es ein Gewährleistungsschaden ist. Die zuständige Firma müsse für die Reparatur aufkommen, sagte er. "Der Bund muss nichts bezahlen."

Nun wird der Damm neu aufgebaut (der SÜDKURIER berichtete). Heinz Jenne hofft darauf, dass die Witterung mitmacht und so der Zeitplan eingehalten werden kann. Autofahrer müssen während der Bauarbeiten über die Oberlohn-, Max-Stromeyer- und Opelstraße fahren. Der Transitverkehr wird großräumig über die Landesstraßen umgeleitet.

Mit dem Bau der B33-neu entlang des Landeplatzes gehe es Mitte Oktober weiter, kündigte Heinz Jenne an. Zunächst werde die Zufahrt zu den Flughafen-Gebäuden gebaut, denn eine Abfahrt von der Bundesstraße gebe es künftig nicht mehr. Die Arbeiten an der dritten und vierten Spur der neuen Straße werden dann im Frühjahr beginnen. Die Planer rechnen mit keinen großen Verkehrsbehinderungen. Ende 2009 soll der Abschnitt zwischen Ried- und Byk-Gulden-Straße komplett fertig sein. Die Planer wollen im Anschluss die jetzt schon vierstreifige Straße zwischen Landeplatz und Schänzlebrücke ausbauen.

Abbildung 4.3: Bauarbeiten auf der B33

Quelle: [SUEK@]

„Mit dem Bau der B33-neu entlang des Landeplatzes gehe es Mitte Oktober weiter, kündigte Heinz Jenne an. Zunächst werde die Zufahrt zu den Flughafen-Gebäuden gebaut, denn eine Abfahrt von der Bundesstraße gebe es künftig nicht mehr. Die Arbeiten an der dritten und vierten Spur der neuen Straße werden dann im Frühjahr beginnen. Die Planer rechnen mit keinen großen Verkehrsbehinderungen. Ende 2009 soll der Abschnitt zwischen Ried- und Byk-Gulden-Straße komplett fertig sein. Die Planer wollen im Anschluss die jetzt schon vierstreifige Straße zwischen Landeplatz und Schänzlebrücke ausbauen.“ [SUEK@]

Momentan ist die B33 am Flughafen durch diese Bauarbeiten nur einseitig stadteinwärts befahrbar. Beim abendlichen Berufsverkehr müssen sich also alle aus der Stadt hinausfahrenden Autofahrer über die Umleitung über die Riedstraße und die Byk-Gulden-Straße (vergleiche Abbildung 4.4) quälen. Durch diese Umleitung kommt es regelmäßig zu Staus im abendlichen Berufsverkehr. Diesem Phänomen wollen wir mit Hilfe dieser Simulation auf den Grund gehen.

### 4.2.1 Das Simulationsmodell

In Abbildung 4.4 ist die oben erwähnte Situation als Modell nochmals abgebildet. Wir haben vier Quellen, aus welchen die Fahrzeuge versuchen im abendlichen Berufsverkehr aus der Stadt heraus zu kommen:

- Die Quellen Fritz-Arnold-Straße und Reidstraße, welche nur wenige Fahrzeuge zur Simulation beisteuern und vor allem der Vollständigkeit halber abgebildet sind.
- Die Quelle der Max-Stromeyer-Straße, aus welcher schon etwas mehr Fahrzeuge kommen.
- Die Quelle Innenstadt, bei welcher sich die meisten Fahrzeuge sammeln, um aus der Stadt hinauszukommen.

Natürlich gibt es auch weiterhin Fahrzeuge, welche in die Stadt hinein wollen. Diese werden durch die Quelle Allensbach simuliert. Allerdings spielen diese in der Simulation nur eine untergeordnete Rolle.

Wie jetzt im Modell schön zu erkennen ist müssen alle Fahrzeuge, welche von der B33 aus der Stadt hinaus wollen, bei der Ampel Reichenau/Ried rechts Abbiegen, beim Kreisverkehr Ried/Byk-Gulden links Abbiegen und schließlich die ganze Byk-Gulden-Straße entlang fahren, bis sie an der Ampel Reichenau/Byk-Gulden wieder nach rechts in die B33 einbiegen können.

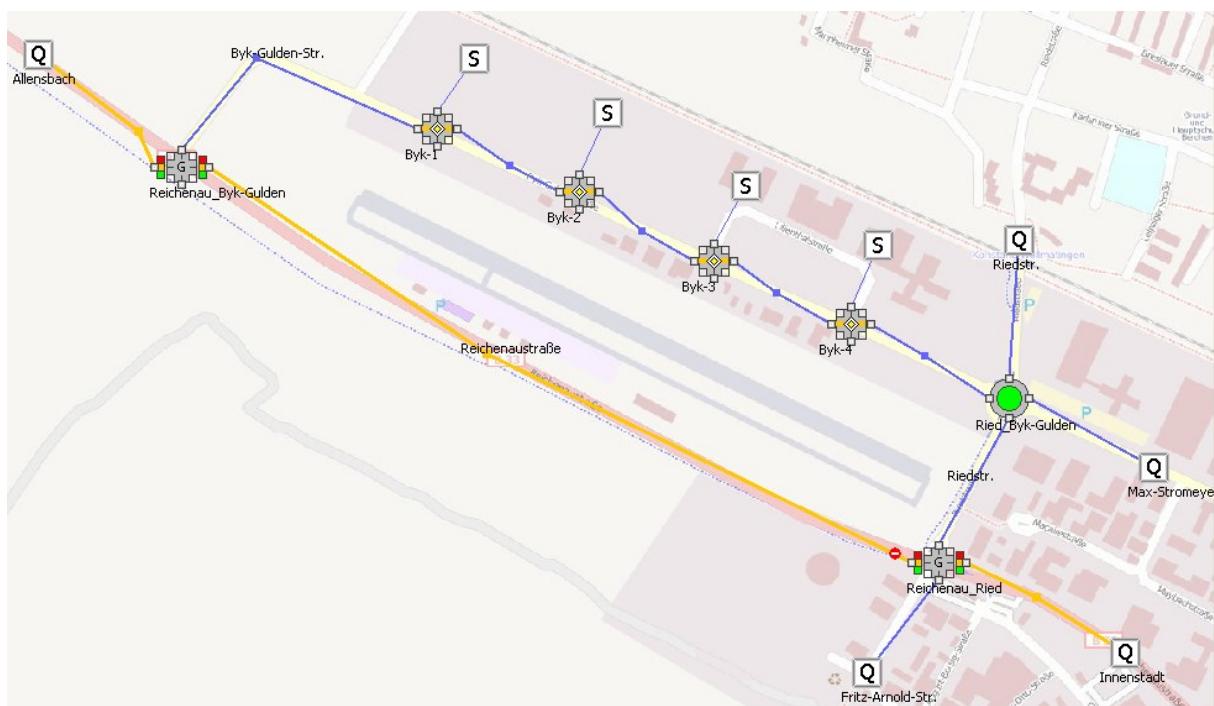


Abbildung 4.4: Modellaufbau. Hintergrundbild von [OSTM@]

## 4.2.2 Ergebnis

Nach mehreren Simulationsläufen, mit teils leicht unterschiedlichen Eingaben, gelangen wir zu verschiedenen Schlussfolgerungen. Diese wollen wir Beispielhaft an einem der Ergebnisse aufzeigen. In Abbildung 4.5 ist angegeben in wie viel Prozent der gesamten Simulationszeit die Straße aufgrund von Staus blockiert war. Der ausführliche Ergebnisexport ist im Anhang A – Fallstudie 2 zu finden.



Abbildung 4.5: Stauanteil in Prozent während der Simulationszeit – Hintergrundbild von [OSTM@]

Durch dieses Ergebnis und durch das Betrachten der Simulation kommen wir zu folgenden Schlussfolgerungen:

- Wie erwartet, bildet sich ein Stau entlang der Umleitung aus der Stadt hinaus. Dieser Stau sorgt auch für Staubildung bei den Zufahrtstraßen.
- Beim Betrachten der Animation fällt auf, dass sobald der erste Streckenabschnitt entlang der Umleitung durch Stau blockiert ist, es zu einer Kettenreaktion kommt. Es kommt zu einem Rückstau, der schnell auch die anderen Straßen blockiert.
- Nach einer Weile kommt es zu einem „Stop and Go“ entlang der Umleitung. Dies ist bedingt durch die Ampelphase an der Reichenau/Byk-Gulden Ampel. Bei jeder Grünphase kommt es zu einem kurzen Auflösen des totalen Staus. Diese Lücke im Stau bewegt sich dann entgegen der Fahrtrichtung fort. Dies kann in der Abbildung 4.5 durch die aktuelle Stauffreiheit der beiden ersten Straßenabschnitte der Byk-Gulden-Straße erkannt werden. Diese zwei Straßenabschnitte sind im Gegensatz zu den anderen Straßenabschnitten momentan nicht blockiert. Dies ist daran zu erkennen, dass sie nicht rot eingefärbt sind.
- Durch die vielen Fahrzeuge, welche durch die Umleitung an dem Kreisverkehr Ried/Byk-Gulden von Süden kommen und danach nach links abbiegen, stauen sich die Fahrzeuge aus der Riedstraße (Norden) und der Max-Stromeyer-Straße (Osten). Dies geschieht, weil es ihnen durch die vielen von Süden kommenden Fahrzeuge kaum noch möglich ist in den Kreisverkehr einzufahren.
- Der Verkehr welcher stadteinwärts entlang der normalen B33 fährt, kommt flüssig voran.
- Fahrzeuge aus der Richtung Allensbach, welche in das nördlich vom Flughafen gelegene Industriegebiet wollen, müssen mit längeren Wartezeiten rechnen, um in der Byk-Gulden-Straße links abbiegen zu können. Sie müssen die vielen entgegenkommenden Fahrzeuge erst vorbeilassen.

## 5 Quellenverzeichnis

- [ROSE01@] Makroskopische Simulation der Verkehrsabläufe auf Autobahnen, 2001 ,  
Dipl.-Ing. Martin Rose, Institut für Bauinformatik, Universität Hannover,  
<http://www.martinrose.net/>
- [SUEK@] Engpass auf B33 bis Dezember, Südkurier, 27.09.2008,  
<http://www.suedkurier.de/region/kreis-konstanz/konstanz/art372448,3434701,1>
- [VISF@] VISSIM-Flyer, PTV Planung Transport Verkehr AG, Stand 06.01.2009  
<http://www.vissim.de/>
- [WIKI01@] Wikipedia-Eintrag über VISSIM, Stand 06.01.2009  
<http://de.wikipedia.org/wiki/VISSIM>
- [WIKI02@] Wikipedia-Eintrag über die Erlangverteilung, Stand 06.01.2009  
<http://de.wikipedia.org/wiki/Erlangverteilung>
- [WAZO@] Der Online Plotter von Walter Zorn,  
<http://www.walterzorn.de/grapher/grapher.htm>
- [WIKI03@] Wikipedia-Eintrag über den Snake-Clon Nibbles, Stand 06.01.2009  
[http://commons.wikimedia.org/wiki/File:Nibbles\\_level3.png](http://commons.wikimedia.org/wiki/File:Nibbles_level3.png)
- [OSTM@] The Free Wiki World Map, Stand 28.12.2008  
OpenStreetMap.org

## 6 Abbildungsverzeichnis

Abbildung 1.1: Übergang von der mikroskopischen zu makroskopischen Modellierung nach [@ROSE01] .....	2
Abbildung 1.2: Wofür mikro-, meso- oder makroskopische Simulation. [VISF@].....	3
Abbildung 1.3: 3D Rundflug durch eine von VISSIM animierte Stadt. [VISF@].....	5
Abbildung 1.4: Planung des Verkehrsnetzes mit VISSIM. [VISF@].....	5
Abbildung 1.5: Planung von Kreuzungen und Ampelphasen mit VISSIM. [VISF@].....	6
Abbildung 3.1: Vereinfachtes Klassendiagramm zum Simulationsaufbau.....	20
Abbildung 3.2: Beispiel für eine Blockade.....	21
Abbildung 3.3: Beispiel für einen Rückstau.....	21
Abbildung 3.4: Die Senke in der GUI.....	23
Abbildung 3.5: Fahrzeuge erreichen die Senke.....	23
Abbildung 3.6: Der Einstellungsdialog der Senke.....	23
Abbildung 3.7: Die Quelle in der GUI.....	23
Abbildung 3.8: Der Einstellungsdialog der Quelle.....	24
Abbildung 3.9: Die Dichtefunktion der Erlang-Verteilung. [WIKI02@].....	25
Abbildung 3.10: Beispiel der Dichtefunktion einer Erlang-Verteilung mit einem Erwartungswert von 10 Sekunden und k=2, erstellt mit dem Online-Plotter von [WAZO@].	26
Abbildung 3.11: Auswahl des Straßentyps.....	27
Abbildung 3.12: Die Bundesstraße.....	27
Abbildung 3.13: Die Hauptstraße.....	27
Abbildung 3.14: Die Nebenstraße.....	27
Abbildung 3.15: Der Einstellungsdialog der Straße.....	27
Abbildung 3.16: Die allgemeinen Einstellungen.....	31
Abbildung 3.17: Das Fahrzeug-Klassendiagramm.....	32
Abbildung 3.18: Kapselung eines Fahrzeugs in einem Container.....	33
Abbildung 3.19: Einstufige Warteschlange.....	33
Abbildung 3.20: Mehrstufige Warteschlange.....	34
Abbildung 3.21: Beispiel für die Anzeige einer Warteschlangenlänge in der Animation.....	34
Abbildung 3.22: Die Ergebnisklassen im Klassendiagramm.....	35
Abbildung 3.23: Die Gesamtlänge der Warteschlangen.....	36
Abbildung 3.24: Die durchschnittliche Gesamtlänge der Warteschlangen.....	36
Abbildung 3.25: Die maximal erreichte Warteschlangenlänge.....	36
Abbildung 3.26: Die durchschnittliche Gesamtwartezzeit.....	36
Abbildung 3.27: Die prozentuale Straßenauslastung.....	36
Abbildung 3.28: Der prozentuale Stauanteil.....	36
Abbildung 3.29: Beispiel für den Ergebnisexport.....	38
Abbildung 3.30: Beispiel einer Animation inklusive Detailansicht.....	39
Abbildung 3.31: Eine Beispieldreieckung mit drei angebundenen Straßen.....	39
Abbildung 3.32: Unsere Beispieldreieckung.....	40
Abbildung 3.33: Aktivieren des Manuell-Modus.....	41
Abbildung 3.34: Beispiel für eine Abbiegewahrscheinlichkeitstabelle.....	41
Abbildung 3.35: Aktivieren des Automatik-Modus.....	41
Abbildung 3.36: Beispiel für eine automatisch erstellte Abbiegewahrscheinlichkeitstabelle.	41
Abbildung 3.37: Beispiel Kreisverkehr.....	42
Abbildung 3.38: Segmentübersicht eines Kreisverkehrs mit 8 Segmenten.....	43
Abbildung 3.39: Nibbles: Remake des Spieleklassiker Snake [WIKI03@].....	44
Abbildung 3.40: Der Kreisverkehr in der Simulation.....	45
Abbildung 3.41: Die Kreisverkehrsstellungen.....	46

Abbildung 3.42: Segmentübersicht Kreisverkehr mit 4 Segmenten.....	47
Abbildung 3.43: Segmentübersicht Kreisverkehr mit 8 Segmenten.....	47
Abbildung 3.44: Die Rechts-vor-Links Kreuzung in der Simulation.....	49
Abbildung 3.45: Der Einstellungsdialog für die Rechts-vor-Links Kreuzung.....	50
Abbildung 3.46: Segmentübersicht einer Rechts-vor-Links Kreuzung.....	50
Abbildung 3.47: Kurzübersicht über die Abbiegemöglichkeiten.....	51
Abbildung 3.48: Übersicht der Nutzung von Segmenten beim Abbiegen.....	51
Abbildung 3.49: Die kleine Ampel in der Simulation.....	53
Abbildung 3.50: Der Einstellungsdialog der kleinen Ampel.....	54
Abbildung 3.51: Segmentübersicht der kleinen Ampel.....	54
Abbildung 3.52: Kurzübersicht über die Abbiegemöglichkeiten in der Kreuzung.....	55
Abbildung 3.53: Übersicht der Segmentnutzung nach Abbiegerichtung.....	55
Abbildung 3.54: Die mittlere Ampel in der Simulation.....	56
Abbildung 3.55: Der Einstellungsdialog der mittleren Ampel.....	56
Abbildung 3.56: Segmentübersicht der mittleren Ampel.....	57
Abbildung 3.57: Kurzübersicht über die Abbiegemöglichkeiten.....	57
Abbildung 3.58: Die normale Route für Linksabbieger.....	58
Abbildung 3.59: Zwei entgegenkommende Linksabbieger.....	58
Abbildung 3.60: Kollision im Modell.....	58
Abbildung 3.61: Lösung des Problems.....	58
Abbildung 3.62: Segmentnutzungsübersicht beim Abbiegen.....	59
Abbildung 3.63: Die mittelgroße, optimierte Ampel in der Simulation.....	60
Abbildung 3.64: Der Einstellungsdialog der mittelgroßen, optimierten Ampel.....	61
Abbildung 3.65: Die Phasen der mittelgroßen, optimierten Ampel.....	61
Abbildung 3.66: Die große Ampel in der Simulation.....	62
Abbildung 3.67: Der Einstellungsdialog der großen Ampel.....	63
Abbildung 3.68: Segmentübersicht der großen Ampel.....	63
Abbildung 3.69: Kurzübersicht über die Abbiegemöglichkeiten.....	64
Abbildung 3.70: Übersicht der Nutzung von Segmenten beim Abbiegen.....	65
Abbildung 3.71: Eine abbiegende Vorfahrtstraße in der Simulation.....	66
Abbildung 3.72: Der Einstellungsdialog der abbiegenden Vorfahrtstraße.....	66
Abbildung 3.73: Segmentübersicht einer Vorfahrt-Kreuzung.....	67
Abbildung 3.74: Linksabbiegerproblem beim Vier-Segment-Modell.....	68
Abbildung 3.75: Problem Links- und Rechtsabbieger.....	68
Abbildung 3.76: Lösung des Linksabbiegerproblems.....	69
Abbildung 3.77: Lösung des Problems des Links- und Rechtsabbieger.....	69
Abbildung 3.78: Abbiegen in der Kreuzung.....	70
Abbildung 3.79: Einfahrtregeln: Einfahrt in Oben/Links Vorfahrtstraße.....	72
Abbildung 3.80: Einfahrtregeln: Einfahrt in Links/Rechts Vorfahrtstraße.....	72
Abbildung 3.81: Einfahrtregeln: Einfahrt in Oben/Rechts Vorfahrtstraße.....	73
Abbildung 3.82: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Links.....	73
Abbildung 3.83: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Oben.....	74
Abbildung 3.84: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Rechts.....	74
Abbildung 4.1: Modellaufbau. Hintergrundbild von [OSTM@].....	76
Abbildung 4.2: Stauanteil in Prozent während der Simulationszeit – Hintergrundbild von [OSTM@].....	77
Abbildung 4.3: Bauarbeiten auf der B33 Quelle: [SUEK@].....	78
Abbildung 4.4: Modellaufbau. Hintergrundbild von [OSTM@].....	79
Abbildung 4.5: Stauanteil in Prozent während der Simulationszeit – Hintergrundbild von [OSTM@].....	80

## 7 Tabellenverzeichnis

Tabelle 2.1: Anfahren aus einer Warteschlange.....	10
Tabelle 3.1: Beispieltabelle für den Ergebnisexport einer mittelgroßen Ampel.....	38
Tabelle 3.2: Das Head - Tail Konzept mit einem Drei-Segment-Fahrzeuge.....	45
Tabelle 3.3: Übersicht der Segmentnutzung nach Abbiegerichtung.....	48
Tabelle 3.4: Übersicht der Segmentnutzung nach Abbiegerichtung.....	70
Tabelle 3.5: Einfahrtregeln: Einfahrt in Oben/Links Vorfahrtstraße.....	71
Tabelle 3.6: Einfahrtregeln: Einfahrt in Links/Rechts Vorfahrtstraße.....	72
Tabelle 3.7: Einfahrtregeln: Einfahrt in Oben/Rechts Vorfahrtstraße.....	72
Tabelle 3.8: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Links.....	73
Tabelle 3.9: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Oben.....	74
Tabelle 3.10: Einfahrtregeln: Verlauf Vorfahrtstraße Unten/Rechts.....	74

## Anhang A – Fallstudie 2

Kreuzungs-ID: 0 Kreuzungs-Name: Reichenau\_Byk-Gulden

Daten der vorgelagerten WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Vorgelagert-Nord	34	32,31	50	299	89,4	44,53
1	Vorgelagert-Ost	0	0	0	0	0	0
2	Vorgelagert-Süd	0	0	0	0	0	0
3	Vorgelagert-West	1	0,03	2	5	0,18	0

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung
0	Nord: Rechts	4	4,56	9	332	112,73
1	Nord: Gerade	0	0	0	0	0
2	Nord: Links	0	1,04	2	292	25,55
3	Ost: Rechts	0	0	0	0	0
4	Ost: Gerade	0	0	0	0	0
5	Ost: Links	0	0	0	0	0
6	Süd: Rechts	0	0	0	0	0
7	Süd: Gerade	0	0	0	0	0
8	Süd: Links	0	0	0	0	0
9	West: Rechts	0	0	0	0	0
10	West: Gerade	5	1,3	6	17	31,76
11	West: Links	0	0,02	1	20	0,48

Kreuzungs-ID: 1 Kreuzungs-Name: Reichenau\_Ried

Daten der vorgelagerten WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Vorgelagert-Nord	0	0	0	0	0	0
1	Vorgelagert-Ost	345	129,51	345	762	132,81	53,57
2	Vorgelagert-Süd	50	9,92	50	237	67,93	28,13
3	Vorgelagert-West	36	2,08	36	42	2,03	0

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung
0	Nord: Rechts	0	0	0	0	0
1	Nord: Gerade	1	0,06	1	41	1,32
2	Nord: Links	0	0,17	2	45	4,66
3	Ost: Rechts	4	4,43	8	682	108,62
4	Ost: Gerade	0	0	0	0	0
5	Ost: Links	0	0,87	2	514	23,65
6	Süd: Rechts	0	0,98	3	170	24,52
7	Süd: Gerade	4	3,23	7	199	83,89
8	Süd: Links	0	0	0	0	0
9	West: Rechts	0	1,2	5	54	29,92
10	West: Gerade	0	2,17	6	49	57,38
11	West: Links	6	2,41	8	132	56,02

## Anhang A – Fallstudie 2

---

Kreuzungs-ID: 2 Kreuzungs-Name: Ried\_Byk-Gulden

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Nord	145	59,92	145	887	285,03	66,17
1	Ost	217	98,3	240	1389	471,22	71,88
2	Süd	24	14,38	31	170	71,93	40,65
3	West	0	0	1	3	0,01	0

Kreuzungs-ID: 3 Kreuzungs-Name: Vorfahrt3

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Nord	0	0	0	0	0	0
1	Ost	22	19,76	40	158	98,83	60,77
2	Süd	0	0	0	0	0	0
3	West	0	0	0	0	0	0

Kreuzungs-ID: 4 Kreuzungs-Name: Vorfahrt4

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Nord	0	0	0	0	0	0
1	Ost	16	14,98	30	145	75,46	36,04
2	Süd	0	0	0	0	0	0
3	West	0	0	0	0	0	0

Kreuzungs-ID: 5 Kreuzungs-Name: Vorfahrt5

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Nord	0	0	0	0	0	0
1	Ost	22	15,69	30	163	79,67	38,44
2	Süd	0	0	0	0	0	0
3	West	5	1,02	5	0	4,34	0

Kreuzungs-ID: 6 Kreuzungs-Name: Vorfahrt6

Daten der WS:

WS-ID	Beschreibung	Länge	durchs. Länge	max Länge	durchs. Wartezeit	proz. Straßenauslastung	proz. Stau
0	Nord	0	0	0	0	0	0
1	Ost	25	16,64	29	149	83,82	40,92
2	Süd	0	0	0	0	0	0
3	West	2	0,03	2	0	0,07	0

## Anhang B – Pseudocode Straße

### Die Ereignisse einer Straße:

- Ankunft
- BedienEnde

### ANKUNFT Ereignis:

„Anmelden des Fahrzeugs beim nächsten Abschnitt“;  
„Ermitteln der maximalen Geschwindigkeit des Fahrzeugs im nächsten Abschnitt“;  
„Berechnen des Zeitaufwands zum befahren der Strecke“;

```
if(anzahlFahrspuren < 2 && bedienEndeZeitpunkt < letzterBedienEndeZeitpunkt){  
    bedienEndeZeitpunkt = letzterBedienEndeZeitpunkt + sicherheitsabstand;  
}
```

„neues BedienEnde Ereignis mit dem Zeitpunkt bedienEndeZeitpunkt berechnen“;

### BEDIENENDE Ereignis:

„Neues Ankunft-Ereignis beim nächsten Abschnitt mit der aktuellen Zeit planen“;

## Anhang C – Pseudocode Kreisverkehr

### Ereignisse des Kreisverkehrs:

- BedienEnde
- Ankunft

### Hilfsfunktionen:

- Was passiert, wenn ein Segment frei wird.
- Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses.
- Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge im Kreisel
- Hilfsfunktion für das globale Blockade-System

### Ereignis BedienEnde

```
if(auto.Head == „Ausfahrtsegment des Autos“){  
    if(„Das Auto-Heck (Tail) verlässt den Kreisel“){  
        „neues Ankunft-Ereignis am nächsten Abschnitt zum aktuellen Zeitpunkt planen“;  
        „aktuelle Segment wird frei und eine eventl. Blockade wird aufgehoben“;  
        „andere Autos, die wegen diesem Fahrzeug mit dem Einfahren warten mussten, dürfen jetzt  
einfahren.“  
    }  
    else{ //Auto fährt weiter aus dem Kreisel aus  
        „neues Bedien Ende Ereignis planen“;  
    }  
}  
else{  
    //Auto hat sein Ausfahrtsegment noch nicht erreicht  
    if(„nächstes Segment ist frei“){  
        //Auto fährt normal weiter  
        „neues Beiden Ende Ereignis planen“;  
    }  
    else{  
        //Auto muss darauf warten, dass das nächste Segment frei wird  
        segmentstatus = „blockiert“;  
    }  
}
```

### Das Ankunft Ereignis:

„Fahrzeug aus der Liste der angekündigten Fahrzeuge holen“;  
„Fahrzeug beim nächsten Segment vor-Ankündigen“;  
„Geschwindigkeit für das nächste Segment ermitteln“; //Zum Rausbeschleunigen

```
if(„Einfahrtsegment frei“ && „Von links kommt kein Fahrzeug bzw. fährt gerade aus“  
&& ws == 0 && „Ausfahrtstraße ist nicht blockiert“){  
    „neues Bedien Ende Ereignis planen“;  
}  
else{ //Auto muss in die Warteschlange  
    „Geschwindigkeit des Fahrzeugs auf 0 setzen“;  
    ws++;  
}
```

### Hilfsfunktion: Was passiert, wenn ein Segment frei wird

segmentstatus = frei;

```
if(„ein anderes Fahrzeug wurde durch dieses belegte Segment blockiert“){  
    „Status des Vorgängersegments auf belegt setzen“;
```

```

,,Den Geschwindigkeitsverlust ermitteln, welcher durch die Blockade entstanden ist“
,, und die Geschwindigkeit des Fahrzeugs entsprechend anpassen“;

,,Bedien Ende für das Vorgängersegments aufrufen“;
}

else if(„falls dieses Segment eine WS hat“ && ws > 0 ){

    if(„Ausfahrtstraße des Autos ist nicht blockiert“ &&
       „von Links kommt kein Auto bzw. fährt gerade aus dem Kreisel aus“ ){

        if(„das einfahrende Auto konnte bereits etwas beschleunigen //Anfahren in der Gruppe“){
            „neue Geschwindigkeit für das Auto berechnen“;
        }

        „neues Bedien Ende Ereignis für das Auto planen“;
    }
}

```

#### **Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses**

maxGeschwindigkeit = „ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs im Kreisel“;  
 „Sowohl den Fahrzeug Head, wie auch das Tail um 1 Segment weiter bewegen“;

```

if(„Fahrzeug Head verlässt den Kreisel“){ //Fahrzeug verlässt den Kreisel
    segmentStatus = „ausfahrend“;
    maxGeschwindigkeit = „maximale Geschwindigkeit des nächsten Abschnitts“; //Rausbeschleunigen
}
else{
    Head.segmentStatus = „belegt“;
}

„vorheriges Tail Segment wird frei und eine eventl. Blockade wird aufgehoben“;

```

zeitpunkt = „Anhand der Geschwindigkeit und Segmentlänge: Berechnen des Zeitpunkts des nächsten Bedien Ende“;  
 „Bedien Ende Ereignis planen“;

#### **Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge im Kreisel**

„Anhand der Abbiege Wahrscheinlichkeiten entscheiden wie das Fahrzeug abbiegt“;  
 „ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs im Kreisel“;  
 „ermitteln der Route der Segmente, welche das Fahrzeug in der Kreuzung entlangfährt“;  
 „Erstellen eines Containers für das Fahrzeug und einstellen in die Liste der angekündigten Fahrzeuge“;

#### **Hilfsfunktion für das globale Blockade-System**

„Eventuelle durch eine ehemalige Blockade blockierte Fahrzeuge jetzt einfahren lassen“;

## Anhang D – Pseudocode Rechts-Vor-Links Kreuzung

### Ereignisse der Kreuzung:

- BedienEnde
- Ankunft

### Hilfsfunktionen:

- Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses.
- Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge in der Kreuzung
- Hilfsfunktion: AutosAusWsEinfahrenLassen
- Hilfsfunktion für das globale Blockade-System

### Ereignis BedienEnde

```
if(auto.Head == „Ausfahrtsegment des Autos“){  
    if(„Das Auto-Heck (Tail) verlässt die Kreuzung“){  
        „neues Ankunft-Ereignis am nächsten Abschnitt zum aktuellen Zeitpunkt planen“;  
        Head.segmentStatus = frei;  
        „Aufruf der Hilfsfunktion: AutosAusWsEinfahrenLassen“  
    }  
    else{ //Auto fährt weiter aus dem Kreisel aus  
        „neues Bedien Ende Ereignis planen“;  
    }  
}  
else{  
    //Auto hat sein Ausfahrtsegment noch nicht erreicht  
    if(„nächstes Segment ist frei“){  
        //Auto fährt normal weiter  
        „neues Beiden Ende Ereignis planen“;  
    }  
    else{  
        //Auto muss darauf warten, dass das nächste Segment frei wird  
        segmentstatus = „blockiert“;  
    }  
}
```

### Das Ankunft Ereignis:

„Fahrzeug aus der Liste der angekündigten Fahrzeuge holen“;  
„Fahrzeug beim nächsten Segment vor-Ankündigen“;  
„Geschwindigkeit für das nächste Segment ermitteln“; //Zum Rausbeschleunigen

```
if(„Kreuzung ist leer“ && „Von Rechts kommt nichts“ && ws == 0 && „Ausfahrtstraße ist nicht blockiert“){  
    „neues Bedien Ende Ereignis planen“;  
}  
else{ //Auto muss in die Warteschlange  
    „Geschwindigkeit des Fahrzeugs auf 0 setzen“;  
    ws++;  
    autosAusWsEinfahrenLassen();  
}
```

### Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses

maxGeschwindigkeit = „ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs in der Kreuzung“;  
„Sowohl den Fahrzeug Head, wie auch das Tail um 1 Segment weiter bewegen“;

if(„Fahrzeug Head verlässt die Kreuzung“) //Fahrzeug verlässt die Kreuzung

```

maxGeschwindigkeit = „maximale Geschwindigkeit des nächsten Abschnitts“; // Rausbeschleunigen
}
else{
    Head.segmentStatus = „belegt“;
}

„das alte Tail-Segment wird frei gegeben“;

zeitpunkt = „Anhand der Geschwindigkeit und Segmentlänge: Berechnen des Zeitpunkts des nächsten Bedien Ende“;
„Bedien Ende Ereignis planen“;

```

**Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge in der Kreuzung**

„Anhand der Abbiege Wahrscheinlichkeiten entscheiden wie das Fahrzeug abbiegt“;  
 „ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs in der Kreuzung“;  
 „ermitteln der Route der Segmente, welche das Fahrzeug in der Kreuzung entlangfährt“;  
 „Erstellen eines Containers für das Fahrzeug und einstellen in die Liste der angekündigten Fahrzeuge“;

**Hilfsfunktion: AutosAusWsEinfahrenLassen**

```
int zaehlerWS=0;
```

//Auswahl einer zufälligen Einfahrt.

```
int zufallszahl = „Ganzzahl zwischen 0 und 3 (inklusive) ziehen“;
```

```
for(int i=0;i<4;i++){
    if(ws[zufallszahl] > 0 && „Von Rechts kommt nichts“ && „Kreuzung ist leer“){
        „neues BedienEnde Ereignis für diese Richtung planen“;
    }
    else if(ws[zufallszahl] > 0){
        zaehlerWS++;
    }
}
```

//Die nächste Einfahrt entgegen des Uhrzeigersinns auswählen

```
zufallszahl--;
if(zufallszahl < 0)
    zufallszahl += 4;
```

```
}
```

//Falls von allen Seiten Fahrzeuge kommen -- Zufällig ein Fahrzeug einfahren lassen

```
if(zaehlerWS == 4 && „Kreuzung ist leer“){
    „neues BedienEnde für das erste Fahrzeug aus der WS[zufallszahl] planen“;
}
```

**Hilfsfunktion für das globale Blockade-System**

„Aufruf der Hilfsfunktion: *autosAusWsEinfahrenLassen*“;

## Anhang E – Pseudocode Ampel

### Ereignisse der Ampel:

- AmpelWirdRot
- AmpelWirdGrün
- BedienEnde
- Ankunft

### Hilfsfunktionen:

- Was passiert, wenn ein Segment frei wird.
- Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses.
- Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge im Kreisel.
- Hilfsfunktion für das globale Blockade-System.

### Ereignis: Ampel wird Rot

```
ampelStatus = „rot“;
zeitpunkt = aktuellerZeitpunkt;
zeitpunkt = zeitpunkt + dauerDerGelbphase;

if(ampelPhase == 3 && „es gibt eine Fußgängerphase“){
    zeitpunkt = zeitpunkt + „Länge der Fußgängerphase“;
}
„Neues Ampel wird Grün Ereignis zum Zeitpunkt zeitpunkt planen“;
```

### Ereignis: Ampel wird Grün

```
ampelStatus = „grün“;
„Ampelphase weiterschalten“;
„Neues Ampel Wird Rot Ereignis planen“;

if( „Es befinden sich noch Fahrzeuge auf der Kreuzung“){
    ampelStatus=“blockiert“;
}
else{
    „Fahrzeuge dürfen aus den zur Ampelphase passenden Warteschlange in die Kreuzung einfahren“;
}
```

### Ereignis: BedienEnde

```
if(auto.Head == „Ausfahrtsegment des Autos“){
    if(„Das Auto-Heck (Tail) verlässt die Kreuzung“){
        „neues Ankunft-Ereignis am nächsten Abschnitt zum aktuellen Zeitpunkt planen“;
        „aktuelle Segment wird frei und eine eventl. Blockade wird aufgehoben“;
    }
    else{
        //Auto fährt weiter aus der Kreuzung aus
        „neues Bedien Ende Ereignis planen“;
    }
}
else{
    //Auto hat sein Ausfahrtsegment noch nicht erreicht
    if(„nächstes Segment ist frei“){
        //Auto fährt normal weiter
        „neues Beiden Ende Ereignis planen“;
    }
    else{
        //Auto muss darauf warten, dass das nächste Segment frei wird
    }
}
```

```

        head.segmentstatus = „blockiert“;
    }
}

```

### **Das Ankunft Ereignis:**

„Fahrzeug aus der Liste der angekündigten Fahrzeuge holen“;  
 „Fahrzeug beim nächsten Segment vor-Ankündigen“;  
 „Geschwindigkeit für das nächste Segment ermitteln“; //Zum Rausbeschleunigen

```

if(„Einfahrsegent frei“ && ws==0 && „Die Richtung hat gerade GRÜN“ && „Ausfahrtstraße ist nicht
blockiert“){
    „neues Bedien Ende Ereignis planen“;
}
else{
    //Auto muss in die Warteschlange
    „Geschwindigkeit des Fahrzeugs auf 0 setzen“;
    ws++;
}

```

### **Was passiert, wenn ein Segment frei wird**

segmentstatus = frei;

```

if(ampelStatus == „blockiert“ && „Kreuzung ist jetzt leer“){
    ampelStatus = gruen;
    „Autos aus WS der aktuellen ampelPhase dürfen jetzt in die Kreuzung einfahren, außer es gibt eine
blockierte Ausfahrt“;
}

if(„ein anderes Fahrzeug wurde durch dieses belegte Segment blockiert“){
    „Status des Vorgängersegments auf belegt setzen“;

    „Den Geschwindigkeitsverlust ermitteln, welcher durch die Blockade entstanden ist“
    „ und die Geschwindigkeit des Fahrzeug entsprechend anpassen“;

    „Bedien Ende für das Vorgängersegments aufrufen“;
}
else if(„Segment hat eine WS“ && ws > 0 && „die Richtung hat gerade GRÜN“
&& „Ausfahrtstraße des Autos ist nicht blockiert“){

    //Falls das aktuelle Segment eine WS hat, kann das Auto jetzt einfahren

    if(„das einfahrende Auto konnte bereits etwas beschleunigen //Anfahren in der Gruppe“){
        „neue Geschwindigkeit für das Auto berechnen“;
    }

    „neues Bedien Ende Ereignis für das Auto planen“;
}

```

### **Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses**

maxGeschwindigkeit = „ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs im Kreisel“;  
 „Sowohl den Fahrzeug Head, wie auch das Tail um 1 Segment weiter bewegen“;

```

if(„Fahrzeug Head verlässt die Kreuzung“){           //Fahrzeug verlässt die Kreuzung
    maxGeschwindigkeit = „maximale Geschwindigkeit des nächsten Abschnitts“; //Rausbeschleunigen
}
else{
    Head.segmentStatus = „belegt“;
}

```

„vorheriges Tail Segment wird frei und eine eventl. Blockade wird aufgehoben“;

zeitpunkt = „Anhand der Geschwindigkeit und Segmentlänge: Berechnen des Zeitpunkts des nächsten Bedien Ende“;

„Bedien Ende Ereignis zum Zeitpunkt zeitpunkt planen“;

**Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge im Kreisel**

„Anhand der Abbiege Wahrscheinlichkeiten entscheiden wie das Fahrzeug abbiegt“;

„ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs in der Kreuzung“; //Links, Rechts, Geradeaus

„ermitteln der Route der Segmente, welche das Fahrzeug in der Kreuzung entlangfährt“;

„Erstellen eines Containers für das Fahrzeug und einstellen in die Liste der angekündigten Fahrzeuge“;

**Hilfsfunktion für das globale Blockade-System**

„Eventuelle durch eine ehemalige Blockade blockierte Fahrzeuge jetzt einfahren lassen, falls die Ampelphase stimmt“;

## Anhang F – Pseudocode Vorfahrtkreuzung

### Ereignisse der Kreuzung:

- BedienEnde
- Ankunft

### Hilfsfunktionen:

- Was passiert, wenn ein Segment frei wird
- Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses.
- Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge im Kreisel
- Hilfsfunktion einfahrtFrei.
- Hilfsfunktion autosAusWsEinfahrenLassen.
- Hilfsfunktion für das globale Blockade-System

### Ereignis BedienEnde

```

if(auto.Head == „Ausfahrtsegment des Autos“){
    if(„Das Auto-Heck (Tail) verlässt die Kreuzung“){
        „neues Ankunft-Ereignis am nächsten Abschnitt zum aktuellen Zeitpunkt planen“;
        „aktuelle Segment wird frei und eine eventl. Blockade wird aufgehoben“;

        autosAusWsEinfahrenLassen();
    }
    else{ //Auto fährt weiter aus der Kreuzung aus
        „neues Bedien Ende Ereignis planen“;
    }
}
else{ //Auto hat sein Ausfahrtsegment noch nicht erreicht
    if(„nächstes Segment ist frei“){
        //Auto fährt normal weiter
        „neues Beiden Ende Ereignis planen“;
    }
    else{ //Auto muss darauf warten, dass das nächste Segment frei wird
        head.segmentstatus = „blockiert“;
    }
}
}

```

### Das Ankunft Ereignis:

„Fahrzeug aus der Liste der angekündigten Fahrzeuge holen“;  
 „Fahrzeug beim nächsten Segment vor-Ankündigen“;  
 „Geschwindigkeit für das nächste Segment ermitteln“; //Zum Rausbeschleunigen

```

if(„Aufruf Hilfsfunktion: einfahrtFrei“ && ws==0){
    „neues Bedien Ende Ereignis planen“;
}
else{ //Auto muss in die Warteschlange
    „Geschwindigkeit des Fahrzeugs auf 0 setzen“;
    ws++;
}

```

### Was passiert, wenn ein Segment frei wird

segmentstatus = frei;

```

if(„ein anderes Fahrzeug wurde durch dieses belegte Segment blockiert“){
    „Status des Vorgängersegments auf belegt setzen“;
}

```

## Anhang F – Pseudocode Vorfahrtkreuzung

---

```
„Den Geschwindigkeitsverlust ermitteln, welcher durch die Blockade entstanden ist“  
„ und die Geschwindigkeit des Fahrzeug entsprechend anpassen“;  
  
„Bedien Ende für das Vorgängersegments aufrufen“;  
}  
else if(„Segment hat eine WS“ && ws > 0 && „Ausfahrtstrasse des Autos ist nicht blockiert“){  
    //Falls das aktuelle Segment eine WS hat, kann das Auto jetzt einfahren  
  
    if(„das einfahrende Auto konnte bereits etwas beschleunigen //Anfahren in der Gruppe“){  
        „neue Geschwindigkeit für das Auto berechnen“;  
    }  
    „neues Bedien Ende Ereignis für das Auto planen“;  
}
```

### Hilfsfunktion zum Erstellen eines neuen Bedien Ende Ereignisses

maxGeschwindigkeit = „ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs im Kreisel“;  
„Sowohl den Fahrzeug Head, wie auch das Tail um 1 Segment weiter bewegen“;

```
if(„Fahrzeug Head verlässt die Kreuzung“){  
    maxGeschwindigkeit = „maximale Geschwindigkeit des nächsten Abschnitts“; //Rausbeschleunigen  
}  
else{  
    Head.segmentStatus = „belegt“;  
}
```

„vorheriges Tail Segment wird frei und eine eventl. Blockade wird aufgehoben“;  
zeitpunkt = „Anhand der Geschwindigkeit und Segmentlänge: Berechnen des Zeitpunkts des nächsten Bedien Ende“;  
„Bedien Ende Ereignis zum Zeitpunkt zeitpunkt planen“;

### Hilfsfunktion für Ankündigungen und berechnen der maximalen Geschwindigkeit der Fahrzeuge im Kreisel

„Anhand der Abbiege Wahrscheinlichkeiten entscheiden wie das Fahrzeug abbiegt“;  
„ermitteln der maximalen Geschwindigkeit dieses Fahrzeugs in der Kreuzung“; //Links, Rechts, Geradeaus  
„ermitteln der Route der Segmente, welche das Fahrzeug in der Kreuzung entlangfährt“;  
„Erstellen eines Containers für das Fahrzeug und einstellen in die Liste der angekündigten Fahrzeuge“;

### Hilfsfunktion einfahrtFrei:

„Kontrolliert ob das Fahrzeug einfahren darf, die Route frei ist, und die Ausfahrtstraße nicht blockiert ist“;

### Hilfsfunktion autosAusWsEinfahrenLassen:

„Zuerst überprüfen ob Fahrzeuge aus WS von den Vorfahrtstraßen einfahren dürfen“;  
„Überprüfen ob sonstige Fahrzeuge einfahren dürfen“;

### Hilfsfunktion für das globale Blockade-System

„Aufrufen der Hilfsfunktion autosAusWsEinfahrenLassen“

## Anhang G – „Pseudocode zum Berechnen des Zeitbedarfs zum Befahren einer Strecke“

In diesem Abschnitt gehen wir auf die Implementierung der Methode zum „Berechnen des Zeitbedarfs zum Befahren einer Strecke“ in Form eines Pseudocods ein.

### Die Variablen

- $v_0$  = Anfangs-Geschwindigkeit in Meter pro Sekunde
- $v_1$  = Endgeschwindigkeit in Meter pro Sekunde
- $a$  = Anfahrbeschleunigung in Meter pro Quadratsekunde
- $b$  = Bremsbeschleunigung in Meter pro Quadratsekunde
- $\max V$  = Maximal zulässige Höchstgeschwindigkeit in Meter pro Sekunde
- $strecke$  = Länge der zu fahrenden Strecke in Meter

### Interne Variablen

- $effektiveBeschleunigungsStrecke$  = Prozentuale Streckenlänge, welche maximal zum Beschleunigen und Bremsen verwendet werden darf (Hier 0,5 also 50%).
- $bMax$  = Globale Bremsbeschleunigungsvariable, definiert die maximale negative Beschleunigung, welche bei einer Vollbremsung auftritt (maximal erreicht ein Auto bis zu 1g Bremsbeschleunigung – was einer Beschleunigung von -9,81 Meter pro Quadratsekunde entspricht)
- $endGeschw$  = Variable zum Speichern der Geschwindigkeit des Fahrzeugs am Ende des Streckenabschnitts
- $maxStreckeBedarf$  = Streckenbedarf, welcher zum Beschleunigen auf die maximale Geschwindigkeit und danach zum Abbremsen auf die maximale Geschwindigkeit des nächsten Segments, benötigt wird.

### Die Implementierung

```
double s = effektiveBeschleunigungsStrecke*strecke;
endGeschw = 0.0; // Endgeschwindigkeit initialisieren

// Falls das nachfolgende Segment höhere Geschwindigkeiten erlaubt, darf die Endgeschwindigkeit
// trotzdem nicht höher als die maximale Geschwindigkeit in diesem Segment sein
if(maxV < v1){
    v1 = maxV;
}

if(v0 > maxV){
    // Falls das Auto schneller einfährt als dies eigentlich erlaubt ist
    // (dies kann u.a. durch eine viel zu geringe Bremsstrecke im Vorgängersegment verursacht sein).
    // Sollte eigentlich niemals im fertigen Modell auftreten

    // Berechnen der Strecke, welche zum Bremsen von der Anfangsgeschwindigkeit
```

```
// bis zur maximalen Geschwindigkeit, benötigt wird  
maxStreckeBedarf = streckeZumBeschleunigen(maxV,v0,-b) ;  
  
// Berechnen der Strecke, welche zum Bremsen von der maximalenGeschwindigkeit  
// bis zur maximalen Geschwindigkeiten des nächsten Segments, benötigt wird.  
maxStreckeBedarf += streckeZumBeschleunigen(v1,maxV,-b);  
}  
else{  
    // Berechnen der Strecke, welche zum Beschleunigen von der Anfangsgeschwindigkeit  
    // bis zur maximalen Geschwindigkeit, benötigt wird  
    maxStreckeBedarf = streckeZumBeschleunigen(maxV,v0,a) ;  
  
    // Berechnen der Strecke, welche zum Bremsen von der maximalenGeschwindigkeit  
    // bis zur maximalen Geschwindigkeiten des nächsten Segments, benötigt wird.  
    maxStreckeBedarf += streckeZumBeschleunigen(v1,maxV,-b);  
}  
  
if(maxStreckeBedarf <= s){  
    // Die Streckenlänge reicht um damit man mindestens 50% der Strecke mit  
    // der maximalen Geschwindigkeit fahren kann.  
  
    // Berechnen der Strecke, in welcher das Fahrzeug mit der konstanten Geschwindigkeit maxV fahren  
    // kann.  
    double normaleStrecke = strecke-maxStreckeBedarf  
  
    // Variable t zum Zwischenspeichern des Zeitbedarfs zum fahren.  
    Double t;  
  
    if(v0>maxV){  
        // Falls die Anfangsgeschwindigkeit höher als die maximale Geschwindigkeit ist.  
  
        // Berechnen der Zeitspanne zum Bremsen von der Anfangsgeschwindigkeit  
        // auf die maximale Geschwindigkeit  
        t = zeitZumBeschleunigen(maxV,v0,-b);  
  
        // Berechnen der Zeitspanne zum Bremsen von der maximalen Geschwindigkeit  
        // auf die Endgeschwindigkeit.  
        t += zeitZumBeschleunigen(v1,maxV,-b);  
    }  
    else{  
        // Anfangsgeschwindigkeit ist niedriger als die maximaleGeschwindigkeit  
  
        // Berechnen der Zeitspanne zum Beschleunigen von der Anfangsgeschwindigkeit  
        // auf die maximale Geschwindigkeit  
        t = zeitZumBeschleunigen(maxV,v0,a);  
  
        // Berechnen der Zeitspanne zum Bremsen von der maximalen Geschwindigkeit  
        // auf die Endgeschwindigkeit.  
        t += zeitZumBeschleunigen(v1,maxV,-b);  
    }  
  
    // Berechnen der Zeitspanne, welche benötigt wird um die verbleibende Strecke  
    // mit der maximalen Geschwindigkeit zu befahren.  
    t += zeitZumKonstantFahren(maxV,normaleStrecke);  
  
    // Abspeichern der Endgeschwindigkeit des Fahrzeugs  
    // (die aufrufende Funktion speichert diese dann in das Fahrzeug).  
    endGeschw = v1;  
  
    return t;  
}
```

```

else{
    // Die Streckenlänge reicht nicht um damit man mindestens 50% der Strecke
    // mit der maximalen Geschwindigkeit gefahren werden kann.

    // Variable zum Zwischenspeichern der Strecke, welche zum Ausgleichen
    // zwischen der Anfangsgeschwindigkeit und der Endgeschwindigkeit benötigt wird.
    double vAusgleichStrecke;

    if(v0 > v1){
        // Berechnen der Strecke, welche zum Bremsen
        // von der Anfangsgeschwindigkeit v0 zur Endgeschwindigkeit v1 benötigt wird.
        vAusgleichStrecke = streckeZumBeschleunigen(v1,v0,-b);
    }
    else{
        // Berechnen der Strecke, welche zum Beschleunigen
        // von der Anfangsgeschwindigkeit v0 zur Endgeschwindigkeit v1 benötigt wird.
        vAusgleichStrecke = streckeZumBeschleunigen(v1,v0,a);
    }

    // Es gibt 3 Möglichkeiten was sein kann:
    // 1. Die Gesamtstrecke reicht nicht ganz zum Ausgleichen des Geschwindigkeitsunterschied
    // zwischen Anfangsgeschwindigkeit und Endgeschwindigkeit.
    // 2. Die Gesamtstrecke reicht zum Ausgleichen des Geschwindigkeitsunterschied
    // zwischen der Anfangsgeschwindigkeit und der Endgeschwindigkeit,
    // benötigt allerdings mehr als für die effektive Beschleunigungsstrecke vorgesehen ist.
    // 3. Die effektive Beschleunigungsstrecke reicht zum Beschleunigen auf mehr als die
    Endgeschwindigkeit,
    // allerdings nicht zum Beschleunigen auf die maximale Geschwindigkeit.

    if(vAusgleichStrecke > strecke){
        // 1. Möglichkeit.

        if(v0 > v1){
            // Die Streckenlänge reicht nicht um normal zu Bremsen
            // --> Vollbremsung mit der Bremsbeschleunigung bMax.

            // Berechnen der Strecke für die Vollbremsung.
            double vergl = streckeZumBeschleunigen(v1,v0,-bMax);

            if(vergl > strecke){
                // Die Strecke reicht nicht einmal für eine Vollbremsung.

                // Berechnen der Endgeschwindigkeit bei einer Vollbremsung über die
                Streckenlänge.
                endGeschw = geschwNachBeschleunigungUeberStrecke(v0,strecke,-bMax);

                // Berechnen und Rückgabe des Zeitbedarfs für diese Vollbremsung.
                return zeitZumBeschleunigen(endGeschw,v0,-bMax);
            }
            else{
                // Die Gesamtstrecke reicht für eine Vollbremsung.

                endGeschw = v1;

                // Berechnen welche Brems-Beschleunigung für das Bremsen benötigt wird.
                double aStrecke = beschleunigenUeberStrecke(v1,v0,strecke);

                // Berechnen und Rückgabe des Zeitbedarfs für die Vollbremsung
                return zeitZumBeschleunigen(v1,v0,aStrecke);
            }
        }
    }
}

```

```

else{
    // Die Streckenlänge reicht nicht zum vollen Beschleunigen
    // auf die maximale Geschwindigkeit des nächsten Segments.

    // Berechnen der Endgeschwindigkeit
    endGeschw = geschwNachBeschleunigungUeberStrecke(v0,strecke,a);

    // Berechnen und Rückgabe des Zeitbedarfs zum Beschleunigen.
    return zeitZumBeschleunigen(endGeschw,v0,a);
}

else if(vAusgleichStrecke > s && vAusgleichStrecke < strecke){
    // 2. Möglichkeit.

    // Berechnen der Reststrecke, in welche die Fahrzeuge mit konstanter Geschwindigkeit fahren
    können.
    double sRest = strecke - vAusgleichStrecke;

    // Variable für den Zeitbedarf
    double t;

    if(v0 > maxV){
        // Falls die Anfangsgeschwindigkeit > maxV
        //--> Bremsen notwendig, danach darf erst mit konstanter Geschwindigkeit gefahren
        werden

        // Berechnen Zeitbedarf zum Geschwindigkeitsausgleich
        // Eigentlich muss von v0 auf maxV abgebremst werden, danach dann von maxV auf
        v1.
        // Da allerdings die Geschwindigkeit und die Zeit sich linear zueinander verhalten,
        // kann dies auf einmal berechnet werden.
        t = zeitZumBeschleunigen(v1,v0,-b);

        // Berechnen des Zeitbedarfs um die restliche Strecke zu fahren
        t += zeitZumKonstantFahren(maxV,sRest);

        endGeschw = v1;
    }
    else if(v0 > v1){
        // Falls die Anfangsgeschwindigkeit > Endgeschwindigkeit --> Bremsen notwendig

        // Berechnen Zeitbedarf zum Geschwindigkeitsausgleich
        t = zeitZumBeschleunigen(v1,v0,-b);

        // Berechnen des Zeitbedarfs um die restliche Strecke zu fahren
        t += zeitZumKonstantFahren(v0,sRest);

        endGeschw = v1;
    }
    else{
        // Falls die Anfangsgeschwindigkeit < Endgeschwindigkeit --> Beschleunigen
        möglich

        // Berechnen Zeitbedarf zum Geschwindigkeitsausgleich
        t = zeitZumBeschleunigen(v1,v0,a);

        // Berechnen des Zeitbedarfs um die restliche Strecke zu fahren
        t += zeitZumKonstantFahren(v1,sRest);

        endGeschw = v1;
    }
}

```

```

// Rückgabe des Zeitbedarfs
return t;
}

else{
    // 3. Möglichkeit

    // Berechnen der restlichen Strecke, nach dem Ausgleichen des Geschwindigkeitsunterschied
    // (die Strecke zum konstant Fahren ist bereits abgezogen)
    double sRest = s - vAusgleichStrecke;

    // Variable für den Zeitbedarf.
    double t;

    // Berechnen der Streckenteilabschnitte s1 und s2 nach der Formel in Kapitel 2.4.2.1
    double s2 = (a*sRest)/(b+a);
    double s1 = sRest - s2;

    // Variable zum Zwischenspeichern der max. erreichbaren Geschwindigkeit.
    double vRestMax;

    endGeschw = v1;

    if(v0 > v1){
        //Falls v0 > v1

        // Zeitbedarf für den Geschwindigkeitsausgleich berechnen (Bremsen von v0 auf v1)
        t = zeitZumBeschleunigen(v1,v0,-b);

        // Berechnen der maximal erreichbaren Geschwindigkeit
        vRestMax = geschwNachBeschleunigungUeberStrecke(v0,s1,a);

        // Berechnen des Zeitbedarfs für das Beschleunigen von v0 auf vRestMax
        t += zeitZumBeschleunigen(vRestMax,v0,a);

        // Berechnen des Zeitbedarfs für das Bremsen von vRestMax auf v0
        t += zeitZumBeschleunigen(v0,vRestMax,-b);
    }
    else{
        // Falls v0 < v1
        // Zeitbedarf für den Geschwindigkeitsausgleich berechnen (Beschleunigen von v0 auf
        v1)
        t = zeitZumBeschleunigen(v1,v0,a);

        // Berechnen der maximal erreichbaren Geschwindigkeit
        vRestMax = geschwNachBeschleunigungUeberStrecke(v1,s1,a);

        // Berechnen des Zeitbedarfs für das Beschleunigen von v1 auf vRestMax
        t += zeitZumBeschleunigen(vRestMax,v1,a);

        // Berechnen des Zeitbedarfs für das Bremsen von vRestMax auf v1
        t += zeitZumBeschleunigen(v1,vRestMax,-b);
    }

    if(vRestMax > maxV){
        // Falls vRestMax > maxV ist darf die konstante Strecke nur mit maxV gefahren
        werden.
        vRestMax = maxV;
    }

    // Zeitbedarf zum konstanten Fahren über die restliche Strecke berechnen
    t += zeitZumKonstantFahren(vRestMax, (strecke-s));
}

```

```
// Rückgabe des Gesamtzeitbedarfs  
return t;  
}  
}
```