



TYP DER AUSARBEITUNG

# Verkehrssimulation der Kreuzung

angefertigt von

**Honghao Zhang**

Matr.-Nr.: 4825434

**Qilin Wei**

Matr.-Nr.: 4865300

am

**Institut für Dynamik und Schwingungen  
Technische Universität Braunschweig**

Betreuer:

**Prof. Dr.-Ing. habil. G.-P. Ostermeyer**

**February 2018**

# Eidesstattliche Erklärung

Hiermit erkläre ich eidesstattlich, dass ich diese Arbeit eigenständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Braunschweig den 12. Feb. 2018





# Inhaltsverzeichnis

<b>1 Einleitung.....</b>	<b>3</b>
1.1 Zielsetzung und Gliederung.....	3
1.2 Geschichte/Literatur.....	3
<b>2 Verkehrssimulation.....</b>	<b>4</b>
2.1 Bilder von Kreuzung.....	4
2.2 Bewegungsfälle.....	6
2.2.1 Grundkonzept.....	6
2.2.2 Regel 1 (Beschleunigung).....	6
2.2.3 Regel 2 (Abbremsen).....	7
2.2.4 Regel 3(Trödeln).....	7
2.2.5 Regel 4 (Überholmanöver).....	7
2.2.6 Regel 5 (Abbiegen).....	8
2.3 Zusätzliche Faktoren.....	9
2.3.1 Maximalgeschwindigkeit.....	9
2.3.2 Straßennetz.....	9
2.3.3 Stau.....	9
2.3.4 Trambahn.....	10
2.3.5 Ampel.....	11
2.4 Programmierung.....	12
2.4.1 Die Klasse TIntFeld.....	12
2.4.2 Die Klasse TUser.....	12
2.4.3 Der Ablauf der Simulation.....	23
2.4.4 Zusätzliche Funktionen.....	24
2.4.5 Funktionen des Buttons.....	25
<b>3 Zusammenfassung und Ausblick.....</b>	<b>26</b>
3.1 Ergebnisse.....	26
3.1.1 Maximale- und Durchschnittliche Geschwindigkeit.....	26
3.1.2 Dauer des Grünlichts und Durchschnittliche Geschwindigkeit.....	28
3.1.3 Dauer des Grünlichts und Leerzeit.....	30
3.1.4 Trambahn und Leerzeit.....	34
3.2 Ausblick.....	36

<b>Literaturverzeichnis.....</b>	<b>37</b>
<b>Anhang.....</b>	<b>38</b>
A.1 Vollständiges Programm.....	38

## Symbolverzeichnis

Formelzeichen	Bedeutung	Einheit
$V_{Max}$	Die Maximalgeschwindigkeit eines Fahrzeuges	
$n$	Das beobachtete Fahrzeug	
$n+1$	Das nächste Fahrzeug von $n$	
$v_n(t)$	Die aktuelle Geschwindigkeit von $n$	
$p_n(t)$	Die aktuelle Position von $n$	
$k$	Abstand zwischen dem Fahrzeug $n+1$ und $n$	
$p_{n+1}(t)$	Die aktuelle Position von $n+1$	
$v_n(t+1)$	Die nächste Geschwindigkeit von $n$	
$p_n(t+1)$	Die nächste Position von $n$	
$v_n^*(t)$	Geschwindigkeit in senkrechte Richtung zum Abbiegen	
$p_n^*(t+1)$	Position in senkrechte Richtung zum Abbiegen	

Indizes	Bedeutung
	Das bewegende Fahrzeug
	Das unbewegliche Fahrzeug
	Tram 1
	Tram 2

# 1 Einleitung

In dieser Arbeit wird die Verkehrssituation über einer Kreuzung durch den Zellulärer Automat simuliert.

## 1.1 Zielsetzung und Gliederung

Bei der Arbeit geht es um das Thema Verkehrssimulation. Der Zweck dieser Arbeit ist, ein offenes Straßennetz über einer Kreuzung von Hamburgstraße, die in der Nähe von TU Braunschweig ist, zu simulieren. Mit der in der Vorlesung implementierten Zellulärer Automat werden mehrere Ein- und Ausgänge und ein Straßenbahngleis an der Kreuzung aufgebaut. Durch das Modell könnten die Einflüsse von Ampeln unter unterschiedlichen Verkehrslage z.B. mit oder ohne Stau ermittelt werden. Dabei kann auch der Einfluss der Straßenbahn betrachtet werden. Mit der Ergebnisse der Simulation könnte man die Dauer der Ampeln oder die maxmale erlaubte Geschwindigkeit des Fahrzeuges regulieren, um die Straßennetz über Kreuzung zu verbessern.

## 1.2 Geschichte/Literatur

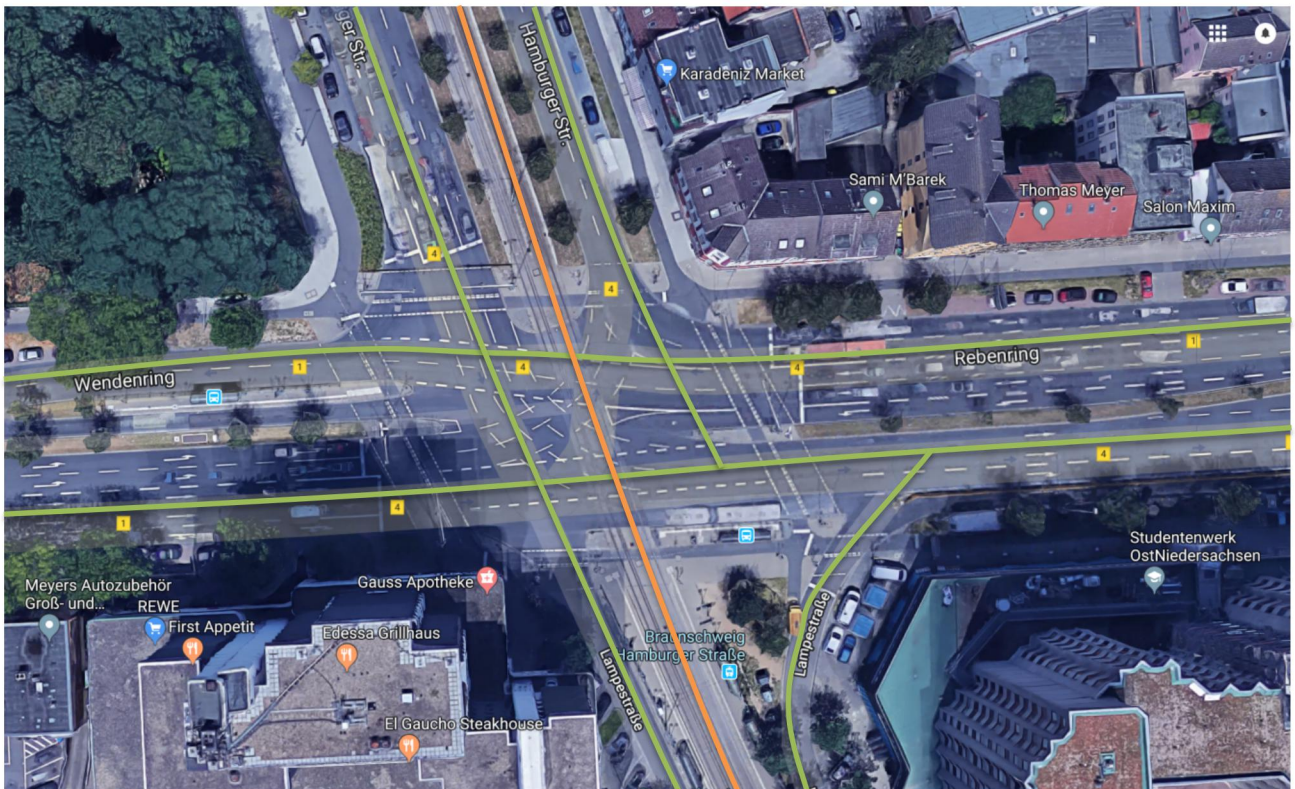
Das Nagel-Schreckenberg-Modell<sup>[1]</sup> (kurz NaSch-Modell) wurde 1992 von den Festkörperphysikern Kai Nagel und Michael Schreckenberg festgelegt. Mit dem Modell können die Verkehrslagen, insbesondere die Verkehrsdichte (Fahrzeuge je Streckenabschnitt) und der Verkehrsfluss (vorbeifahrende Fahrzeuge je Zeiteinheit), vorhergesagt werden. Es erklärt den Stau aus dem Nichts als Folge der Nicht-Einhaltung des Sicherheitsabstandes.

## 2 Verkehrssimulation

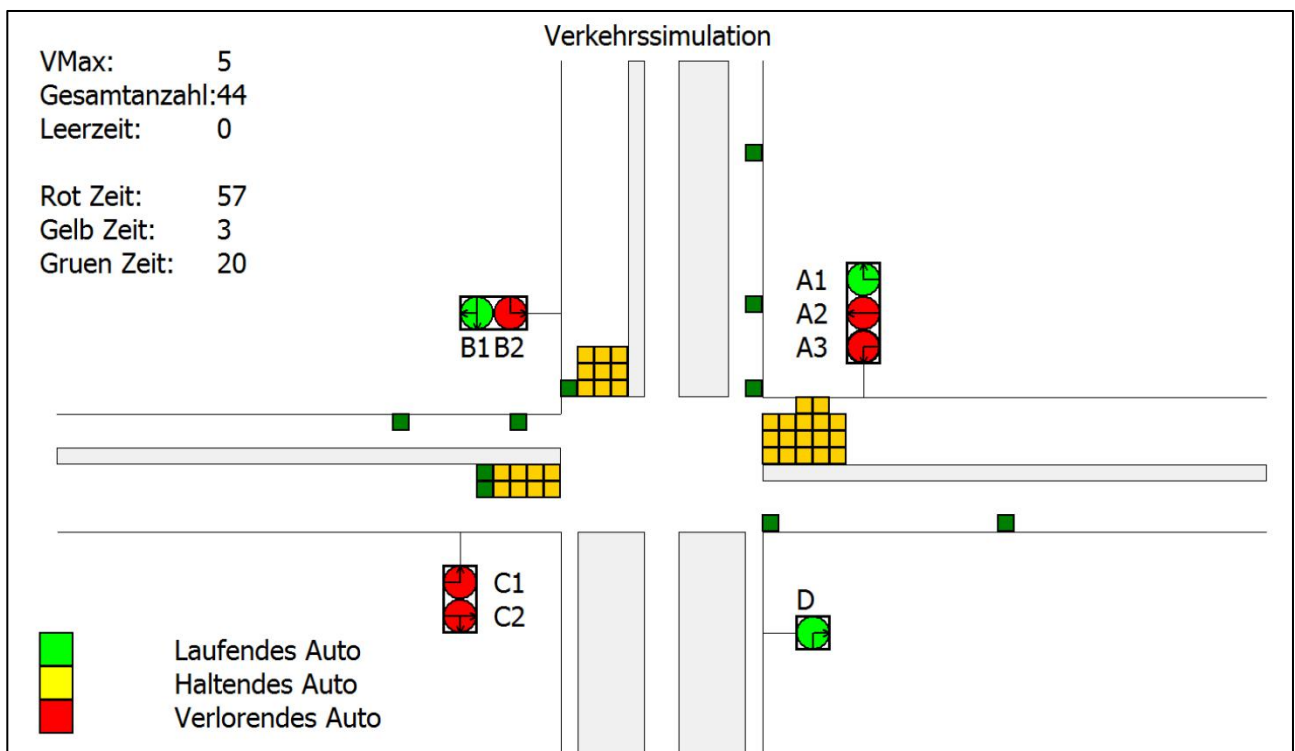
### 2.1 Bilder von Kreuzung



aktuelles Bild von der Kreuzung von Hamburgstraße



Satelliten-Draufsicht der Kreuzung von Hamburgstraße



Simulationsdiagramm der Kreuzung von Hamburgstraß

## 2.2 Bewegungsfälle

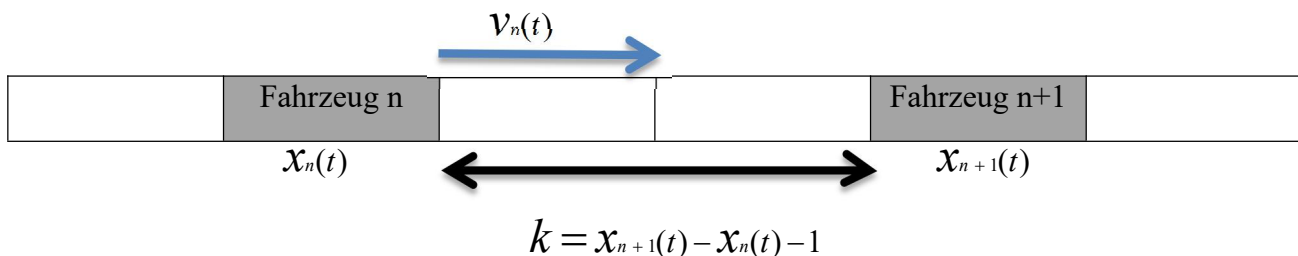
In den Zellularautomaten zur Verkehrssimulation kann jede Zelle den Zustand besetzt oder frei annehmen, wobei besetzten Zellen zusätzlich die aktuelle Geschwindigkeit des entsprechenden Fahrzeuges zugeordnet wird<sup>[2]</sup>.

Die Aktualisierung der Fahrzeuggeschwindigkeit und -orte wird durch folgenden Regeln festgelegt.

### 2.2.1 Grundkonzept

Der Wert  $V_{Max}$  entspricht der erlaubten Maximalgeschwindigkeit eines Fahrzeuges und wird als Defaultwert häufig zu  $V_{Max} = 5$  gesetzt.

Sei zu irgendeinem Zeitpunkt  $t$  die Geschwindigkeit des Fahrzeuges  $n$  gleich  $v_n(t)$  und die Position des Fahrzeuges  $n$  gleich  $x_n(t)$ . Und seien  $k$  Zellen zwischen das nächsten Fahrzeug  $n+1$  und das Fahrzeug  $n$ , dann gilt:



Zur Berechnung von  $v_n(t+1)$  und  $x_n(t+1)$  müssen folgende Regeln durchgeführt werden<sup>[2]</sup>.

### 2.2.2 Regel 1 (Beschleunigung)

$$v_n(t+1) = v_n(t) + 1, \text{ wenn } v_n(t) < V_{Max}.$$

$$x_n(t+1) = x_n(t) + v_n(t), \text{ wenn } v_n(t) \leq k.$$

Wenn das Fahrzeug  $n$  langsamer fährt als  $V_{Max}$ , erhöht es seine aktuelle Geschwindigkeit um 1 und erneuert es seine aktuelle Position.



### 2.2.3 Regel 2 (Abbremsen)

Sei  $x_n(t) + v_n(t) \leq x_{n+1}(t) + v_{n+1}(t)$  oder die Zelle von  $x_n(t+1) = x_n(t) + v_n(t)$  nicht frei, dann gilt:

$$v_n(t) = k, \text{ wenn } v_n(t) \geq k+1.$$

$$x_n(t+1) = x_n(t) + v_n(t).$$

Wenn es innerhalb von  $v_n(t)$  Zellen vor dem Fahrzeug  $n$  noch ein Fahrzeug  $n+1$  läge und die entsprechende Zelle  $x_n(t+1)$  nicht frei wäre, würde die Fahrzeugsgeschwindigkeit  $v_n(t)$  gleich wie die Anzahl der freien Zellgitter  $k$  eingestellt.

### 2.2.4 Regel 3(Trödeln)

Mit einer Wahrscheinlichkeit von  $p$  gilt:

$$v_n(t+1) = v_n(t) - 1, \text{ wenn } v_n(t) > 0.$$

d.h. das Fahrzeug könnte ohne Grund mit einer vorgegebenen Wahrscheinlichkeit  $p$  zufällig abbremsen.

### 2.2.5 Regel 4 (Überholmanöver)

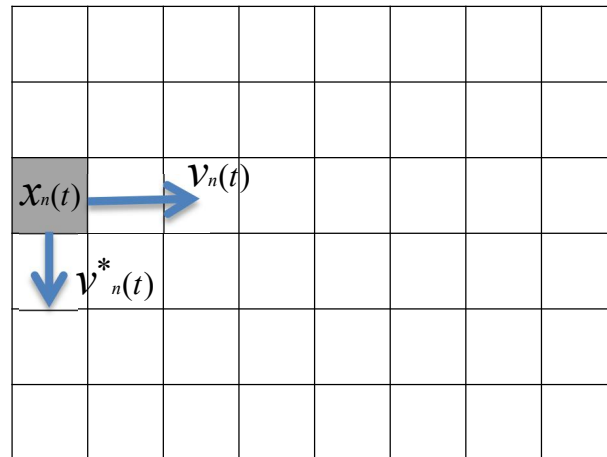
Sei  $x_n(t) + v_n(t) > x_{n+1}(t) + v_{n+1}(t)$  und die Zelle von  $x_n(t+1) = x_n(t) + v_n(t)$  frei, dann gilt:

$$v_n(t) = v_n(t), \text{ wenn } v_n(t) \geq k+1.$$

$$x_n(t+1) = x_n(t) + v_n(t).$$

Wenn es innerhalb von  $v_n(t)$  Zellen vor dem Fahrzeug  $n$  noch ein Fahrzeug  $n+1$  läge und die entsprechende Zelle  $x_n(t+1)$  frei wäre, würde das Fahrzeug in diesem Zellgitter ohne Geschwindigkeitsabnahme direkt bewegen.

### 2.2.6 Regel 5 (Abbiegen)



Sei das Fahrzeug  $n$  im zentralen Gebiet an der Kreuzung, dann begönne das Fahrzeug gleichzeitig eine Geschwindigkeit in senkrechte Richtung zum Abbiegen, und nach den konkreten Umständen gilt:

Kleine Abbiegung:

$v_n(t) = 1$ , Wenn  $x_n(t+1)$  frei wäre.

$v_n^*(t) = 1$ , Wenn  $x_n^*(t+1)$  frei wäre.

Große Abbiegung:

$v_n(t) = 2$ , Wenn  $x_n(t+1)$  frei wäre.

$v_n^*(t) = 1$ , Wenn  $x_n^*(t+1)$  frei wäre.

## 2.3 Zusätzliche Faktoren

Mit den eingesteckten Drucktasten könnten zusätzlicher Faktor realisiert werden.

### 2.3.1 Maximalgeschwindigkeit

#### Beschreibung:

Mit der folgenden Drucktaste könnte die Maximalgeschwindigkeit verändert werden.

**Drucktaste:** “VMax++” und “VMax--”



### 2.3.2 Straßennetz

#### Beschreibung:

Mit der folgenden Drucktaste “open” würde ein offenes Straßennetz aufgebaut. Dabei würde die Zahl des Fahrzeuges reduziert, wenn die Fahrzeuge aus dem Straßennetz gefahren hätten. Dagegen würde ein geschlossenes Straßennetz mit der Drucktaste “close” aufgebaut. Wenn ein Fahrzeug im diesen Fall aus dem Straßennetz gefahren hätte, würde gleichzeitig ein Fahrzeug von anderer Seite des Straßennetzes einfahren. Damit würde die Zahl des Fahrzeuges nicht verändert, sodass könnte der Lauf der Verkehrssimulation stets beobachtet werden.

**Drucktaste:** “open” und “close”



### 2.3.3 Stau

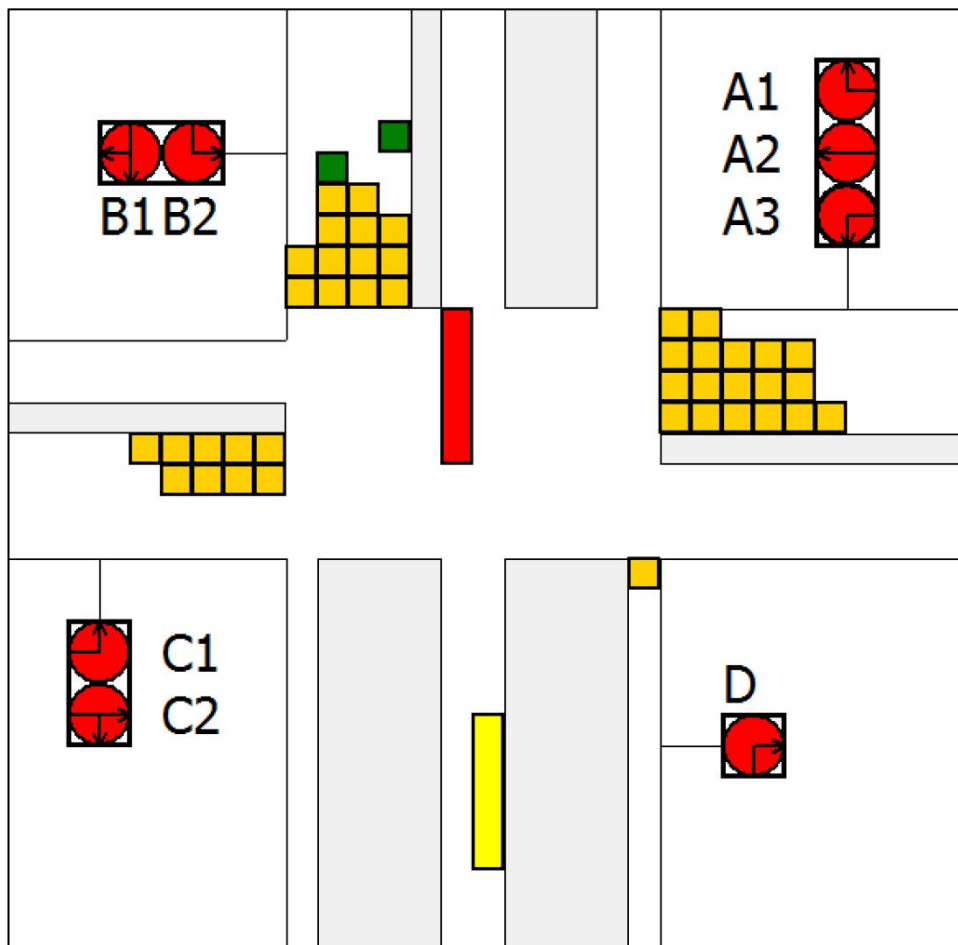
#### Beschreibung:

Die Zahl des Fahrzeuges bei einem Stau beträgt 2-3 mal als die bei einem normalen Fall.

**Drucktaste:** “Stau”



### 2.3.4 Trambahn



#### Beschreibung:

In der Kreuzung gibt es zwei Trambahnen von zwei unterschiedlichen Fahrrichtungen. Mit den folgenden Drucktasten könnte jeweils eine Trambahn gerufen werden. Jede Straßenbahn würde 5 Gitter im Modell besitzen. Straßenbahn hätte Vorfahrt im Vergleich zur anderen Fahrzeuge, d.h. wenn eine Straßenbahn einführe, würden alle Ampeln auf Rot wechseln. Wenn die Straßenbahn aus der Kreuzung gefahren hätte, arbeiten alle Ampeln wieder normal.

**Drucktaste:** “Tram1” und “Tram2”



### 2.3.5 Ampel

#### Beschreibung:

Als Defaultwerte würde die Dauer an einem Ampel nach folgendem Formular einstellen.

Die Dauer der Ampel unter verschiedenen Fälle

Fälle \ Ampeln Dauer (Sekunden)	Rot	Grün	Gelb	Endsumme
Normaler Fall	57	20	3	80
Stau	37	40	3	80

Mit der folgenden Drucktaste könnte die Dauer der grünen Lichte bzw. der roten Lichte eingestellt werden.

**Drucktaste:** “green++” und “green--”

green++	green--
---------	---------

Folgendes Formular zeigt die maximale und minimale Dauer des Rot- und Grünlichts.

	Min	Max
Rot	37	57
Grün	20	40
Gelb	3	3

## 2.4 Programmierung

Das Programm der Verkehrssimulation besteht aus 3 Klassen, nämlich TIntFeld, TUser und Main.

Die Klasse TIntFeld liefert eine 3-dimensionale Matrix, um die Geschwindigkeiten der Fahrzeugen in X- und Y-Richtung zu speichern.

Die Klasse TUser, die Klasse Tplan erweitert, bietet die Möglichkeiten, die Verkehrsregeln zu behandeln, die Elemente des GUI wie Fahrzeugen, Ampeln, Straßen und Plots zu zeichnen.

In der Klasse Main wird die Klasse TUser implementiert. In den folgenden Abschnitten sind die Klassen TIntFeld und TUser detailliert zu erläutern. Schließlich wird den Ablauf der Simulation anhand der Funktionen vorgestellt.

### 2.4.1 Die Klasse TIntFeld

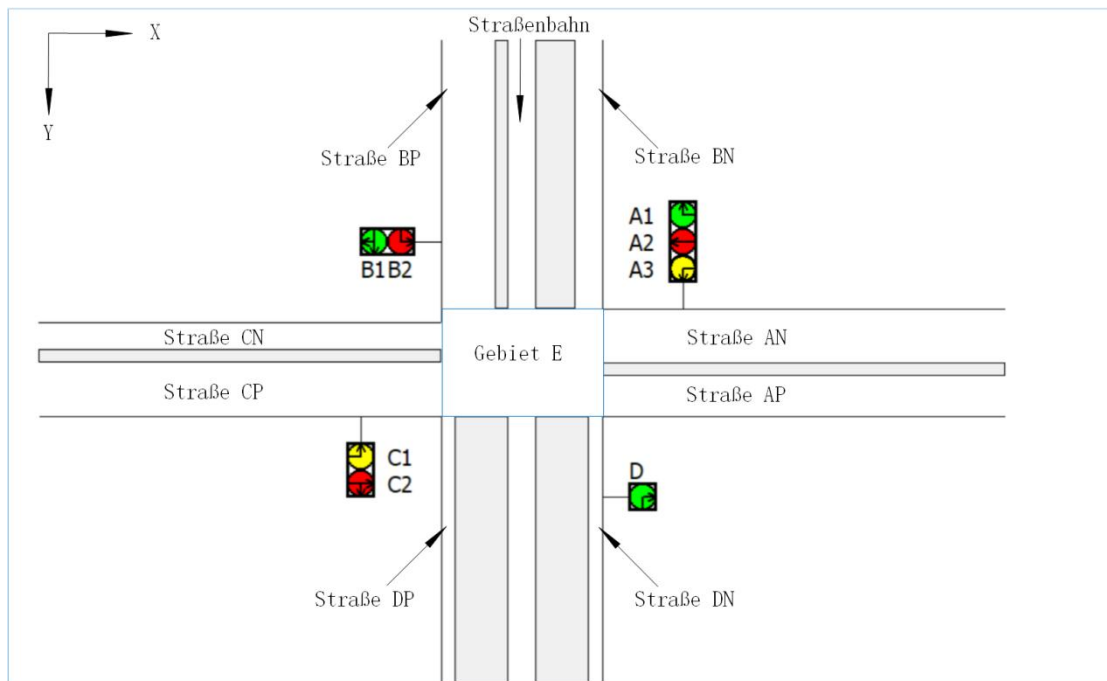
Die Klasse TIntFeld besitzt ein inneres Feld im Typ von int als Datenspeicher. Sie bietet die Möglichkeiten, Daten für Zelluläre Automaten im Form von Matrix zu speichern, kopieren sowie aufgrund Zeile- und Spalteziffer zuzugreifen. Die Veränderung der Klasse TIntFeld, die in diesem Programm genutzt wird, zu der originalen Version liegt daran, dass eine weitere Dimension erweitert wird. Da unsere Simulation sich auf eine Kreuzung bezieht, sollen die Fahrzeugen Geschwindigkeit in beiden Orientierungen haben.

Seien Feld ein TIntFeld-Objekt, i und j die Ziffern für Zeile und Spalte, dann liefert Feld(i, j, 0) die Geschwindigkeit in X-Richtung und Feld(i, j, 1) die Geschwindigkeit in Y-Richtung.

### 2.4.2 Die Klasse TUser

Der Klasse TUser hat zwei Hauptaufgabe. Eine ist, die Geschwindigkeiten und Position der Fahrzeugen gemäß Verkehrsregeln zu erneuern. Die andere ist, die Fahrzeugen, Ampeln und Plots auf dem Bildschirm zu zeichnen.

Die Kreuzung sieht wie im Abb.2.4.1 dargestellt aus.



**Abbildung 2.4.1 Kreuzung**

Es werden einige globalen Variablen inner der Klasse definiert. Die Datentype und Funktionen sind wie im Formular 2.4.1 beschreibt.

**Formular 2.4.1 Globale Variable der Klasse TUser**

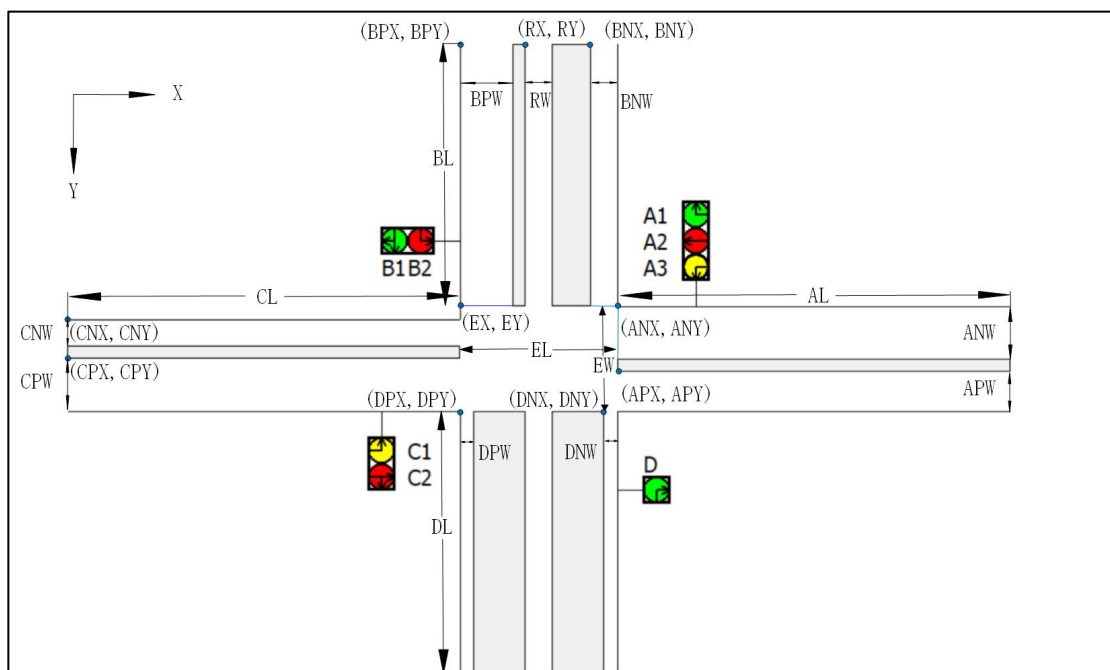
Variablenname	Datentyp	Bedeutung
Feld	TIntFeld	Feld für die Geschwindigkeiten der Fahrzeugen
HFeld	TIntFeld	Hilfsfeld für die Geschwindigkeiten der Fahrzeugen
N	int	Zeile der Matrix, die in Feld gespeichert wird.
M	int	Spalte der Matrix, die in Feld gespeichert wird.
W	int	Länge und Breite der Zelle
D	int	Abstand zwischen Zeichensbereich und Ränder der Bildschirm
t	int	Laufzeit der Ampel

T1, T2, T3, T4	int	Zeitbereich der Ampel
Runtime	int	Leerzeit
VMax	int	Maximale erlaubte Geschwindigkeit
VXS, VYS	int	Durchschnittliche Geschwindigkeit in X und Y Richtung
ANX, ANY, BNX, BNY, CNX, CNY, DNX, DNY, EX, EY, APX, APY, BPX, BPY, CPX, CPY, DPX, DPY, RX, RY	int	Koordinaten der jeweiligen Straßen(Abb.3.2)
AL, BL, CL, DL, EL	int	Länge der jeweiligen Straßen
APW, ANW, BPW, BNW, CPW, CNW, DPW, DNW, RW, EW	int	Bereite der jeweiligen Straßen
PTurn	int	Abbiegenswahrscheinlichkeit in der Straße BP
POverrun	int	Überholenswahrscheinlichkeit
PTrodel	int	Trödelswahrscheinlichkeit
Sum	int	Summe der Fahrzeugen
SumX	int	Summe der Fahrzeugen, die auf Straßen AN,AP,CN,CP fahren
SumY	int	Summe der Fahrzeugen, die auf Straßen BN,BP,DN,DP fahren
Tram1, Tram2	bool	Symbole,die bezeichnen, ob Tram1 oder Tram2 vorhanden ist.
OpenRoad	bool	Symbole,die bezeichnen, ob die Kreuzung offen ist.



Verkehrstau	bool	Symbole, die bezeichnen, ob Verkehrstau vorhanden ist.
RLamp	int	Radius der Ampeln
DXLamp, DYLamp	int	Abstand zwischen Ampeln und Straßen in X- und Y-Richtung
LampA1, LampA2, LampA3, LampB1, LampB2, LampC1, LampC2, LampD	int	Variablen, die die Zustände der jeweiligen Ampeln beschreiben. 0, 1, 2 repräsentieren jeweils grün, gelb, rot.

**Abbildung 2.4.2 Koordinatensystem der Kreuzung**



#### 2.4.2.1 Feststellen des Pturns

Die Fahrzeugen, die auf der linken Spur der Straße BP fahren, können gerade nach Straße DP fahren oder nach rechts abbiegen. Das hängt von einer Wahrscheinlichkeit PTurn ab. Zum Festlegen des Pturns werden die Fahrzeugen auf diesen Straßen

beobachtet. Fast 80% der Fahrzeugen biegen nach rechts ab, die resten fahren gerade nach vorne. Daher ist PTurn als 80 eingestellt.

Zum Bearbeiten der Geschwindigkeit wird die Funktion Road() definiert. Die Funktion Road() ruft die Funktionen RoadAN(), RoadAP(), RoadBN(), RoadBP(), RoadCN(), RoadCP(), RoadDN(), RoadDP(), RoadE() auf, um die Fahrzeugen in jeder Straße zu bearbeiten. Gemäß den unterschiedlichen Verkehrsregeln können die obengenannten Funktionen in 3 Gruppen geteilt werden. Die erste Gruppe beinhaltet die Funktionen RoadAN(), RoadBP(), RoadCP(), RoadDN(). Hierbei fahren die Fahrzeugen von jeweiligen Straßen nach dem Gebiete E (Abb.2.4.1). Im Gegensatz dazu, fahren auch einigen Fahrzeugen von dem Gebiet E nach jeweiligen Straßen. Diese entspricht den Funktionen wie RoadAP(), RoadBN(), RoadCN(), RoadDP(). Die Fahrzeugen, die von einer Seite zu einer anderen Seite des Gebietes E fahren, werden von RoadE() bearbeitet. Diese Funktionen werden im Formular 2.4.2 detailliert. Die entsprechenden Regeln der drei Gruppen sind in den unten Abschnitten zu erläutern.

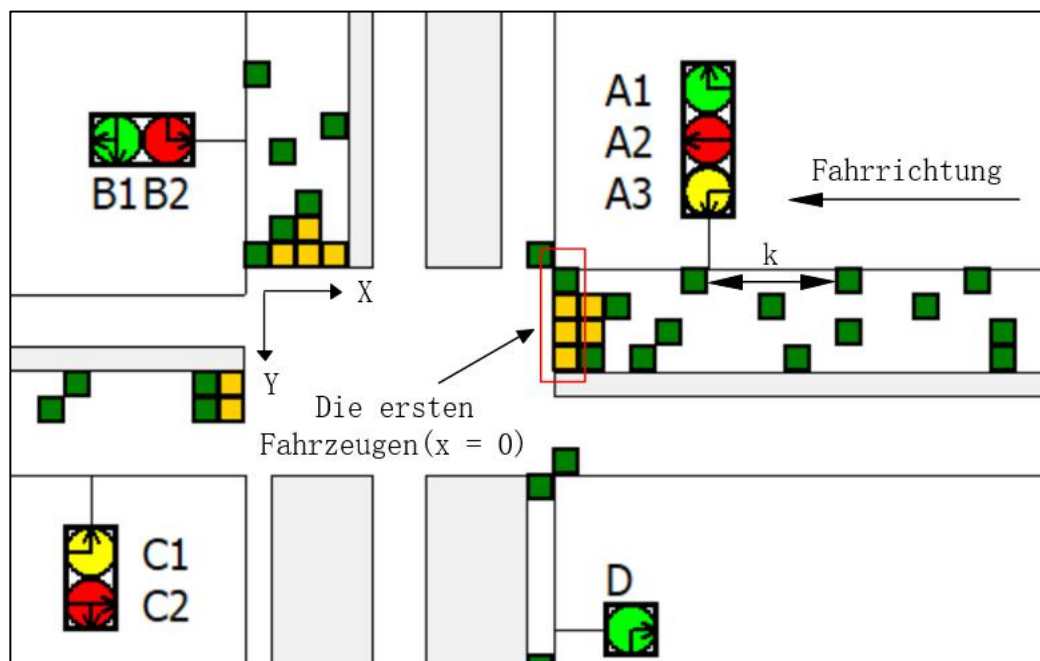
#### **Formular2.4.2 Funktionen für die Straßen**

<b>Typ</b>	<b>Name</b>	<b>Funktion</b>
Public	Road()	Erneuen die Zustände aller Zellen
	RoadAN()	Erneuen die Zustände der Zellen auf der Straße AN
	RoadAP()	Erneuen die Zustände der Zellen auf der Straße AP
	RoadBN()	Erneuen die Zustände der Zellen auf der Straße BN
	RoadBP()	Erneuen die Zustände der Zellen auf der Straße BP
	RoadCN()	Erneuen die Zustände der Zellen auf der Straße CN
	RoadCP()	Erneuen die Zustände der Zellen auf der Straße CP
	RoadDN()	Erneuen die Zustände der Zellen auf der Straße DN
	RoadDP()	Erneuen die Zustände der Zellen auf der Straße DP
	RoadE()	Erneuen die Zustände der Zellen in der Kreuzung
	Tram()	Erneuen die Zustände der Zellen des Trams

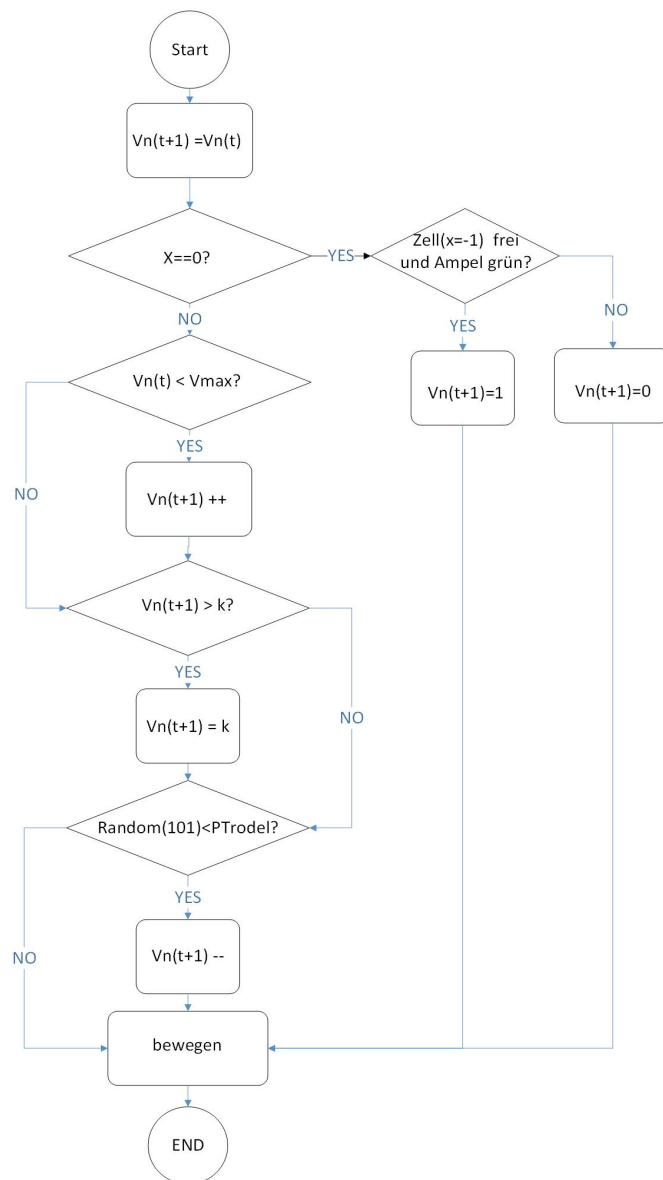
### 2.4.2.2 Die Verkehrsregel der einfahrenden Gruppe

Die Geschwindigkeiten der meisten Fahrzeuge in dieser Gruppe ändert sich aufgrund den Regeln „Beschleunigung, Abbremsen, Trödeln, Bewegung“. Die Geschwindigkeiten der Fahrzeugen, die an dem Gebiet E liegen, haben eine andere Regel. Die Geschwindigkeiten werden zuerst auf 0 gesetzt, wenn die erste Zelle in dem Gebiet E vor diesem Fahrzeug frei ist und gleichzeitig die Ampel der Spur, auf der der Fahrzeug gerade läuft, grün ist, soll die Geschwindigkeit um 1 erhöht werden. Danach bewegt der Fahrzeug mit der schon eingestellten Geschwindigkeit nach vorne.

**Abbildung 2.4.3 Die Fahrzeuge in der Straße AN**



Seien  $x$  die lokale Koordinate, die die lokale Position beschreibt,  $V_n(t)$  die Geschwindigkeit des  $n$ -ten Fahrzeugs in der X-Richtung in der Zeit  $t$ ,  $k$  die Summe der freien Zellen vor dem Fahrzeug,  $V_n(t+1)$  die Geschwindigkeit in der Zeit  $t+1$ . Das Programm läuft wie im Abb. 2.4.4 beschreibt.



**Abbildung 2.4.4 Ablauf der Funktion RoadAN()**

Schritt 1: Wenn der Fahrzeug an dem Gebiet E liegt, bzw.  $x=0$ ,  $n=1$ , wird diese Schritt durchgeführt. Wenn die Zelle( $x=-1$ ) vor dem Fahrzeug frei ist und gleichzeitig die Ampel grün ist, ändert  $V_1(t+1)$  zu 1. Sonst ändert sie sich zu 0. Danach soll direkt Schritt 5 statt Schritt 2 durchgeführt.

Schritt 2: Wenn  $V_n(t) < V_{Max}$ , dann gilt :  $V_n(t) = V_n(t)+1$ .

Schritt 3: Wenn  $V_n(t) > k$ , dann gilt :  $V_n(t) = k$ .

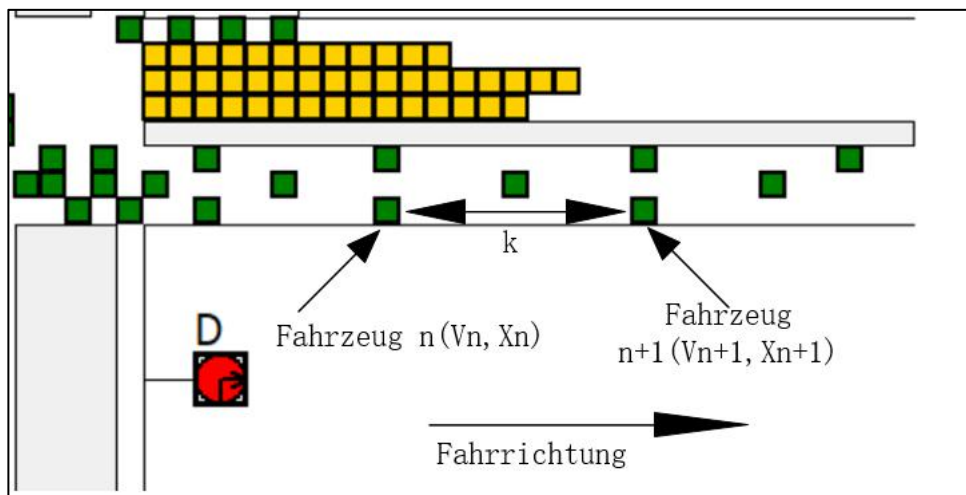
Schritt 4: Wenn  $\text{random}(101) < \text{PTrodel}$ , dann gilt :  $V_n(t+1) = V_n(t)-1$ .

Schritt 5: Der Fahrzeug wird um  $V_n(t+1)$  Zellen nach vorne bewegt.

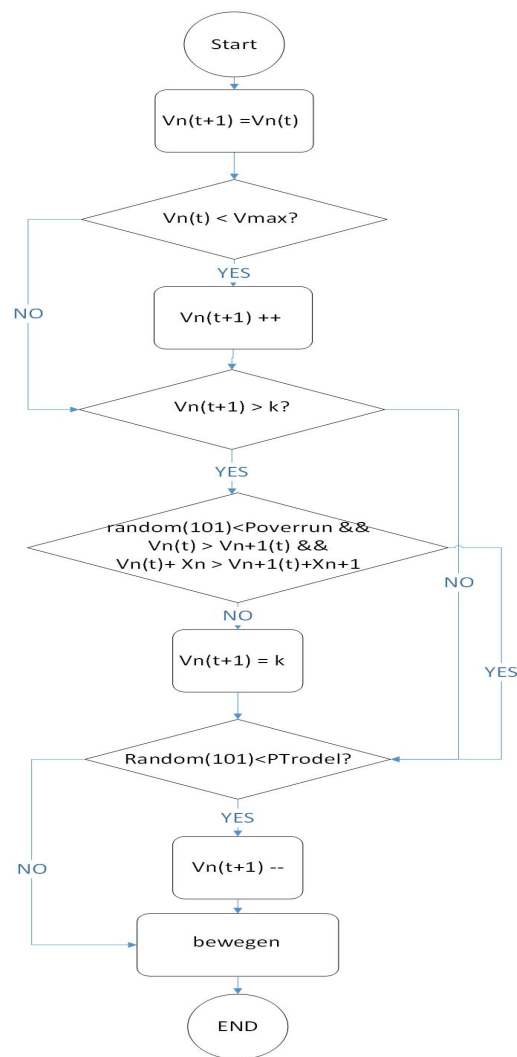
### 2.4.2.3 Die Verkehrsregel der ausfahrenden Gruppe Abb.

Die Besonderheit der Verkehrsregel für die aus dem Gebiet E fahrenden Gruppe liegt an der Möglichkeit des Überholmanövers. Wenn die Geschwindigkeit höher als die Summe der freien Zellen ist, die vor diesem Fahrzeug sich befindet, gibt es die Möglichkeit, das Fahrzeug statt abzubremseeln sondern den Fahrzeug, der vor diesem Fahrzeug fährt, zu überholen.

Abbildung 2.4.5 Straße AP



Seien  $X_n$ ,  $X_{n+1}$  die Position der  $n$ -ten und  $n+1$ -ten Fahrzeugen in der  $X$ -Richtung,  $V_n(t)$ ,  $V_{n+1}(t)$  die Geschwindigkeit in der Zeit  $t$ ,  $k$  die Summe der freien Zellen vor dem Fahrzeug  $n$ ,  $V_n(t+1)$ ,  $V_{n+1}(t+1)$  die Geschwindigkeiten in der Zeit  $t+1$ . Das Programm läuft wie im Abb. 2.4.6 beschreibt.



**Abbildung 2.4.6 Ablauf der Funktion RoadAP()**

Schritt 1: Wenn  $V_n(t) < V_{Max}$ , dann gilt :  $V_n(t) = V_n(t)+1$ .

Schritt 2: Wenn  $V_n(t) > k$ , dann gilt :

Wenn  $\text{random}(101) < \text{Poverrun}$  und  $V_n(t) > V_{n+1}(t)$  und  $V_n(t) + X_n > V_{n+1}(t) + X_{n+1}$ , dann  $V_{n(t+1)} = V_n(t)$ . Es wird direkt Schritt 4 durchgeführt. Sonst  $V_n(t) = k$ .

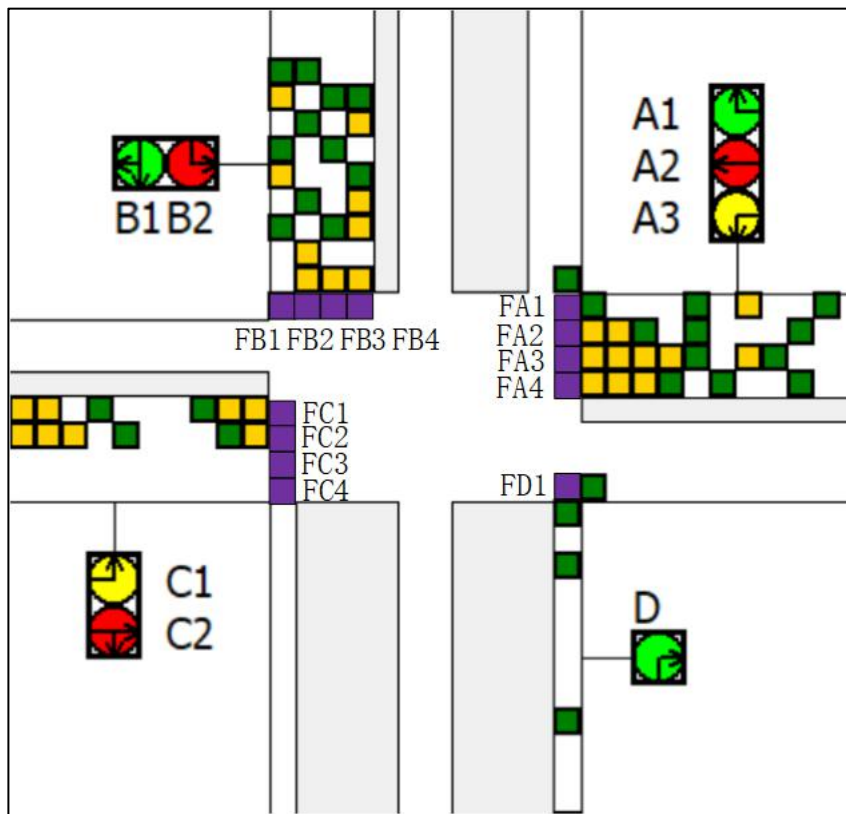
Schritt 3: Wenn  $\text{random}(101) < \text{PTrodel}$ , dann gilt :  $V_{n(t+1)} = V_n(t)-1$ .

Schritt 4: Der Fahrzeug wird um  $V_{n(t+1)}$  Zellen nach vorne bewegt.

#### 2.4.2.4 Die Verkehrsregel der Kreuzung

Alle Fahrzeugen in der Kreuzung bewegt mit einer bestimmten Geschwindigkeit wie im Formular 2.4.3 dargestellt.

**Abbildung 2.4.7 Zelle des Gebietes**



**Formular 2.4.3 Geschwindigkeiten der Zellen des Gebietes E**

Zelle	V <sub>x</sub>	V <sub>y</sub>
FA1	0	-1
FA2 /FA3	-2	0
FA4	-2	1
FB1	-1	1
FB2 /FB3 /FB4	1	1
FC1	2	-1
FC2 /FC3	2	0
FC4	0	1
FD	1	0

#### 2.4.2.5 Die Verkehrsregel des Trams

Der Tram bewegt mit einer festen Geschwindigkeit um 2 Zelle/ Zeitschritt und nach einer festen Richtung, d.h. Tram1 fährt von oben nach unten und Tram2 läuft gegenüber.

#### Formular2.4.4 Funktionen für das Tram

Typ	Name	Funktion
public	Tram()	Erneuen die Zustände der Zellen des Trams

#### 2.4.2.6 Die Regel der Ampeln

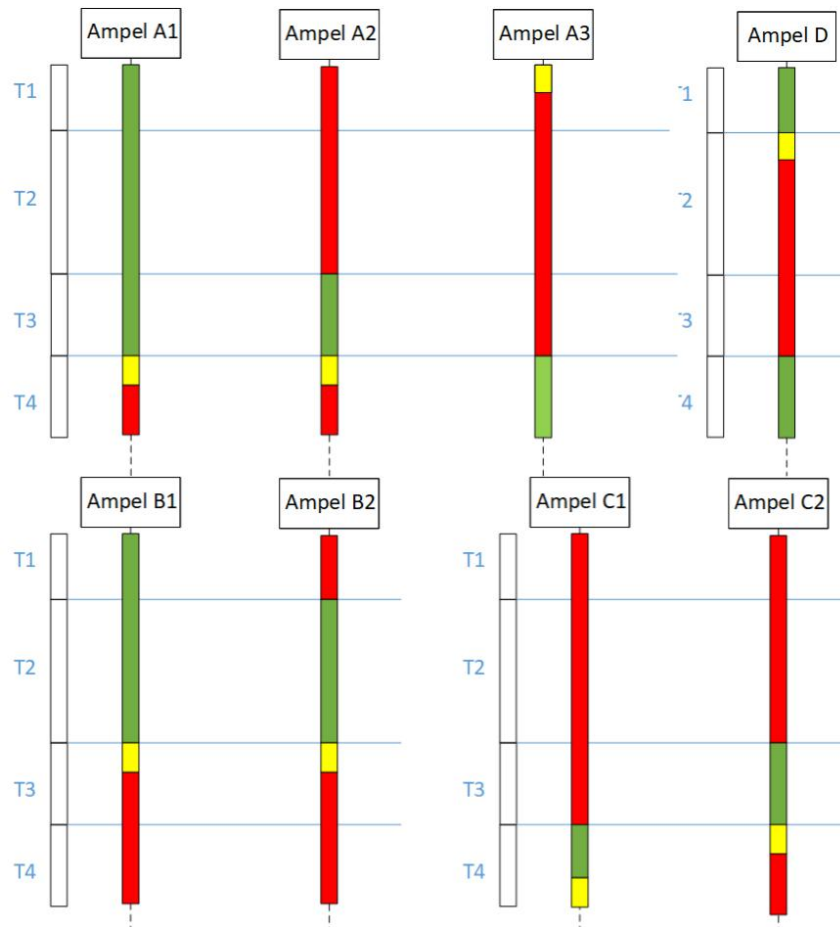
Die Ampel verwechselt die Farbe anhand der Zeit. Der Zyklus der Ampeln dauert 80s. Es wird in 4 Teilen geteilt. Die Farbe der Ampel in einem Zyklus wird in Abb.2.4.2 beschreibt. Die grüne Dauer der Ampel A2 ist veränderbar. Da die meisten Fahrzeugen in den zwei entsprechenden Spuren fahren, durch die Verlängerung der grüne Dauer ist die Erleichterung des Staus zu erwarten.

#### Formular2.4.5 Funktionen für die Ampeln

Typ	Name	Funktion
public	Lamp()	Bearbeiten die Zustände der Ampeln



**Abbildung 2.4.8 Zeitbereich der Ampeln**



### 2.4.3 Der Ablauf der Simulation

Der Ablauf der Simulation ist einfach zu formulieren. Es wird am allen ersten die Initialisierungsfunktion `Init()` durchgeführt. Dabei werden die Globalen Variablen initialisiert, die Ampeln vorbereitet, die Fahrzeugen erzeugt, die Straßen und die Ampeln gezeichnet. Nachdem die Run Taste gedruckt ist, läuft die Funktion `Run()` im Zyklus. In jeder Schleife wird zuerst der Zustand der Ampeln erneut. Danach wird der Zustand der Fahrzeugen und der Straßenbahnen bearbeitet. Zum Schluss werden nach einer Rereinigung wider die Straßen, die Ampeln, die Fahrzeugen, die Plots und die Informationen auf dem Bildschirm gezeichnet.

#### 2.4.4 Zusatzliche Funktionen

**Formular 2.4.6 Zusatzliche Funktionen**

Typ	Name	Funktion
privat	SetLampColor(int lamp)	Festlegen die Farbe gemäß Zustand der Ampel
	Arrow(int x1, int y1, int x2, int y2)	Zeichnen einen Pfeil von Punkt(x1,y1) zu Punkt(x2,y2)
public	Init()	Initialisieren das Objekt der Klasse TUser
	InitVerkehr()	Initialisieren die Variable Feld
	InitLamp()	Initialisieren die Variablen T1,T2,T3,T4
	InitPlot()	Initialisieren die Plots
	InitRoad()	Initialisieren die Koordinaten der Straßen
	DrawRoad()	Zeichnen alle Straße
	DrawLamp()	Zeichnen die Ampeln
	DrawCars()	Zeichnen die Fahrzeugen
	DrawTram()	Zeichnen die Trams
	ShowMsg()	Vorstellen wichtige Informationen
	Plots()	Zeichnen die Plots

### 2.4.5 Funktionen des Buttons

**Formular 2.4.7 Funktionen für die Buttons**

Button	Funktion
Stau	Erzeugen Verkehrstau
Plot	Erzeugen Plot für Gesamtzahl der Fahrzeugen und Plot für durchschnittliche Geschwindigkeiten in X- und Y-Richtung
Tram1	Erzeugen Tram1
Tram2	Erzeugen Tram1
Open	Öffnen die Kreuzung, damit die Fahrzeugen ausfahren können.
Close	Schließen die Kreuzung, damit die Fahrzeugen nicht ausfahren dürfen.
Vmax++	Erhöhen die erlaubte Geschwindigkeit um 1.
Vmax--	Senken die erlaubte Geschwindigkeit um 1.
Green++	Erhöhen die grüne Dauerzeiten der Ampeln A2 und C2 um 2, gleichzeitig senken die rote um 2 .
Green--	Erhöhen die rote Dauerzeiten der Ampeln A2 und C2 um 2, gleichzeitig senken die grüne um 2 .

## 3 Zusammenfassung und Ausblick

### 3.1 Ergebnisse

#### 3.1.1 Maximale- und Durchschnittliche Geschwindigkeit

Mit verschiedenen Maximalgeschwindigkeit  $V_{Max}$  wie im Abb.3.1.1 könnte folgendes Formular 3.1.1 in der Verkehrssimulation erhalten werden.

**Formular 3.1.1 Lauf mit verschiedenen  $V_{Max}$**

$V_{Max}$	2	5	8
$V$			
$V_{xmax}$	1.5	1.8	2.0
$V_{ymax}$	1.7	2.3	2.1
$V_{oft}$	0.2—1.3	0.2—1.5	0.3—1.6

Erklärung:

$V$ : Durchschnittliche Geschwindigkeit.

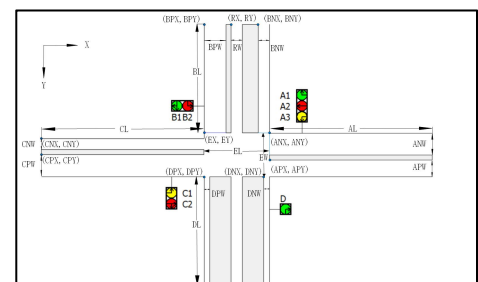
$V_{xmax}$ : Durchschnittliche Geschwindigkeit in der X Richtung.

$V_{ymax}$ : Durchschnittliche Geschwindigkeit in der Y Richtung.

$V_{oft}$ : Die Interval ,in der die durchschnittliche Geschwindigkeit oft liegt.

#### Zusammenfassung:

Mit der deutlichen Steigung der Maximalgeschwindigkeit, steigt die durchschnittliche Geschwindigkeit ein bisschen. In der ganzen Zeit sind die durchschnittliche Geschwindigkeit die Maximalgeschwindigkeit nicht erreicht.



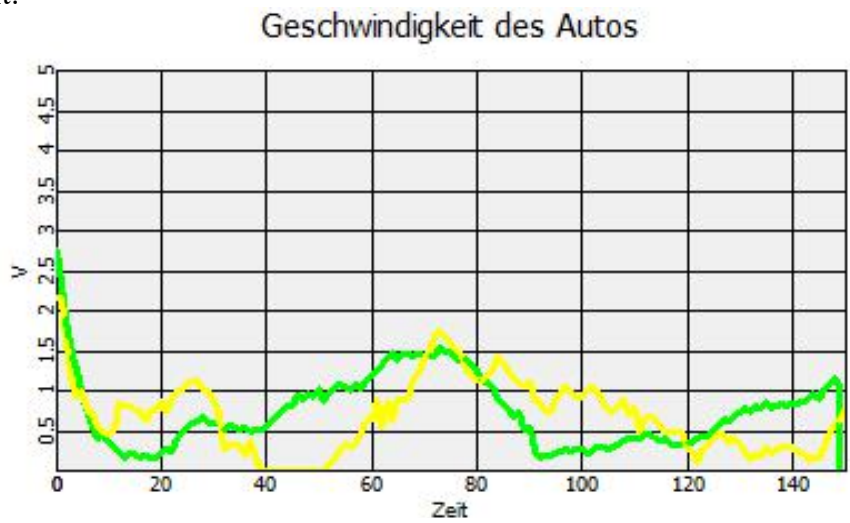
### Abbildung 3.1.1 Lauf mit verschiedenen VMax

Durchschnittliche Geschwindigkeit:

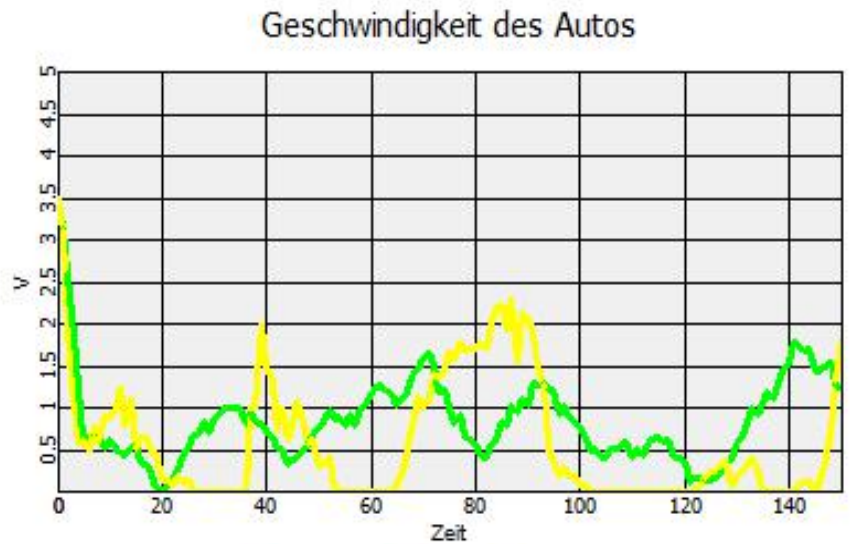
X Richtung: 

Y Richtung: 

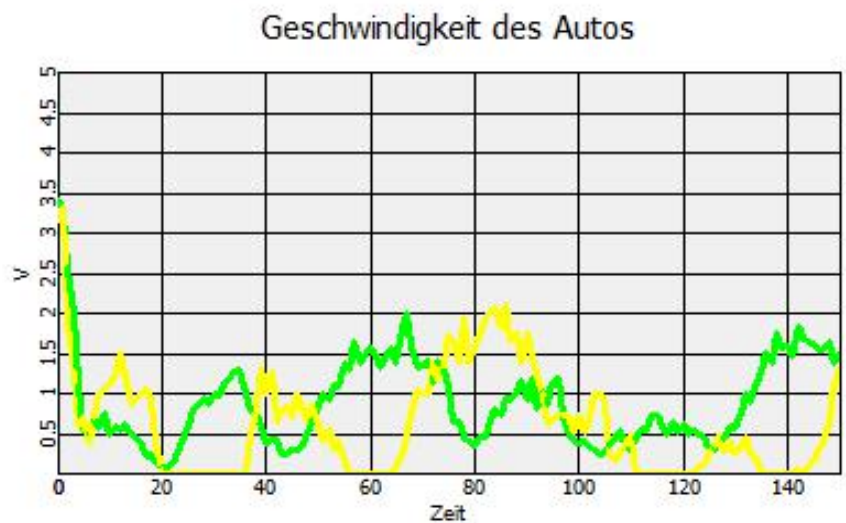
VMax: 2  
Gesamtanzahl: 45  
Leerzeit: 0  
  
Rot Zeit: 57  
Gelb Zeit: 3  
Gruen Zeit: 20



VMax: 5  
Gesamtanzahl: 45  
Leerzeit: 0  
  
Rot Zeit: 57  
Gelb Zeit: 3  
Gruen Zeit: 20



VMax: 8  
Gesamtanzahl: 45  
Leerzeit: 0  
  
Rot Zeit: 57  
Gelb Zeit: 3  
Gruen Zeit: 20



### 3.1.2 Dauer des Grünlichts und Durchschnittliche Geschwindigkeit

Mit verschiedener Dauer der Lichter am Ampel wie im Abb.3.1.2 könnte folgendes Formular 3.1.2 unter  $V_{\text{Max}} = 5$  erhalten werden.

**Formular 3.1.2 Lauf mit verschiedener Dauer der Lichter**

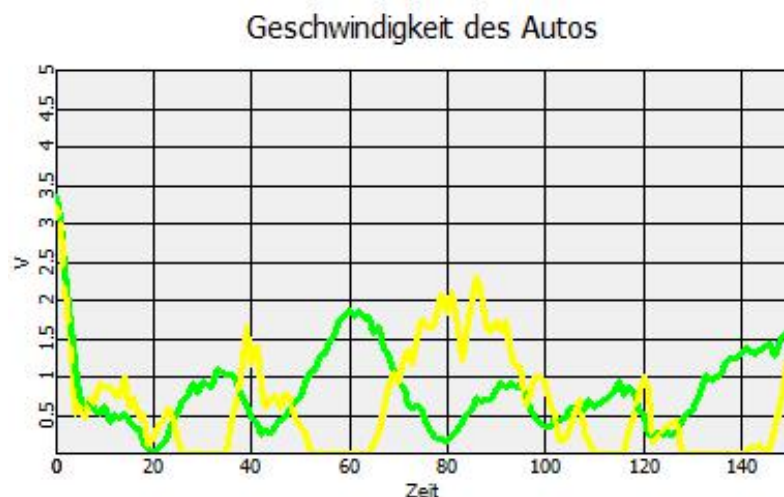
$V$ \ Dauer des Grünlichts	20	30	40
$V_{\text{xmax}}$	1.9	1.7	2.1
$V_{\text{ymax}}$	2.3	2.3	2.1
$V_{\text{oft}}$	0.2—1.6	0.2—1.7	0.2—1.7

#### Zusammenfassung:

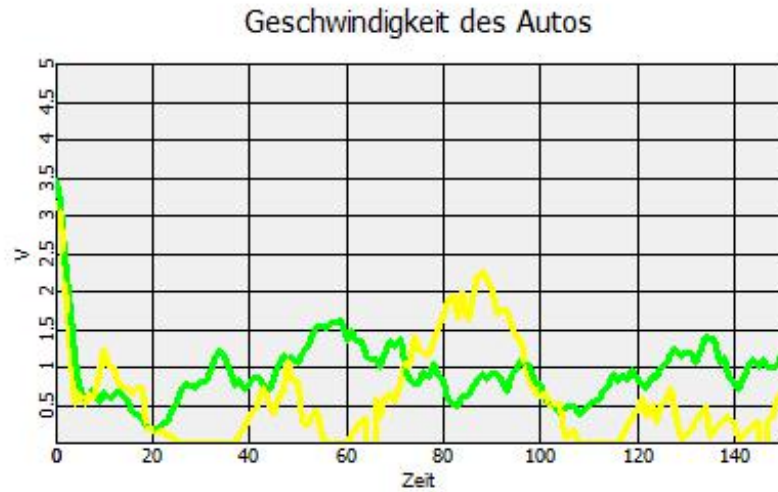
Mit der Steigung der Dauer der Grünlichter, steigt auch die durchschnittliche Geschwindigkeit, aber ein bisschen.

**Abbildung 3.1.2 Lauf mit verschiedener Dauer der Lichter**

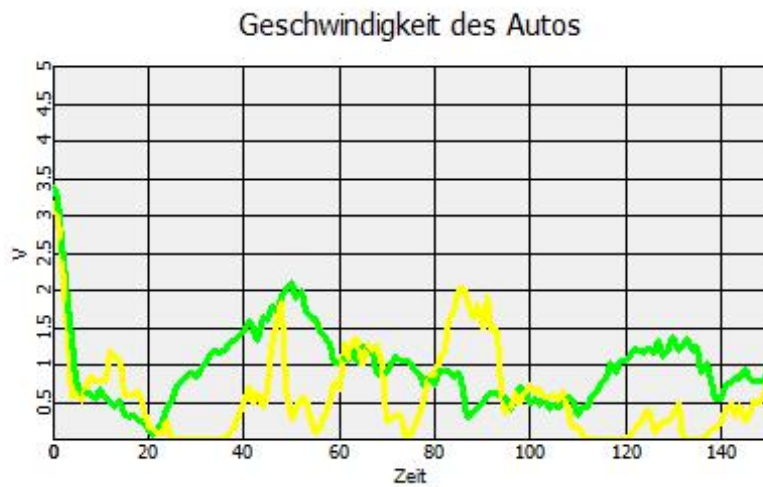
$V_{\text{Max}}$ : 5  
 Gesamtanzahl: 45  
 Leerzeit: 0  
  
 Rot Zeit: 57  
 Gelb Zeit: 3  
 Gruen Zeit: 20



VMax: 5  
 Gesamtanzahl: 45  
 Leerzeit: 0  
  
 Rot Zeit: 47  
 Gelb Zeit: 3  
 Gruen Zeit: 30



VMax: 5  
 Gesamtanzahl: 45  
 Leerzeit: 0  
  
 Rot Zeit: 37  
 Gelb Zeit: 3  
 Gruen Zeit: 40



### 3.1.3 Dauer des Grünlichts und Leerzeit

In einem offenen Straßennetz würde die Zahl des Fahrzeuges reduziert, wenn die Fahrzeuge aus dem Straßennetz gefahren hätten. Die Zeit, in der alle Fahrzeuge aus dem Straßennetz fahren könnten, ist die sogenannte Leerzeit. Mit einer kurzen Leerzeit könnte die Fahrzeuge schneller durch dem Straßennetz der Kreuzung fahren.

Mit verschiedener Dauer der Grünlichter am Ampel könnte folgendes Formular 3.1.3 erhalten werden.

**Formular 3.1.3 Leerzeit**

Leerzeit Dauer des Grünlichts	VMax			
		2	5	8
20		256	249	320
30		300	336	329
40		419	480	489

Folgende Abbildung 3.1.3 wäre der besten Fall Aus dem Formular 3.1.3 unter VMax=2, Dauer =20. Und unter VMax=8, Dauer =40 wäre dagegen der schlechten Fall wie im Abbildung 3.1.4.

Darunten ist die ausführliche Analyse zwischen den besten- und schlechten Fall.



Abbildung 3.1.3 Der besten Fall(Zeit:0—150)

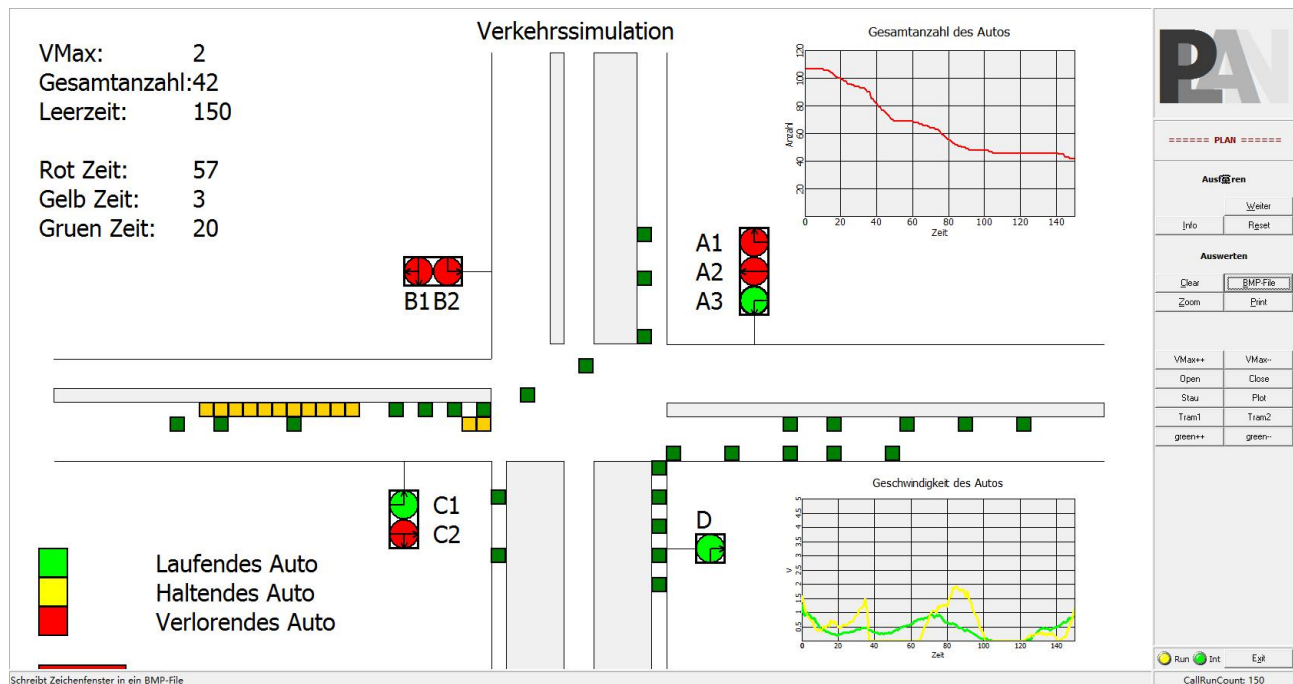
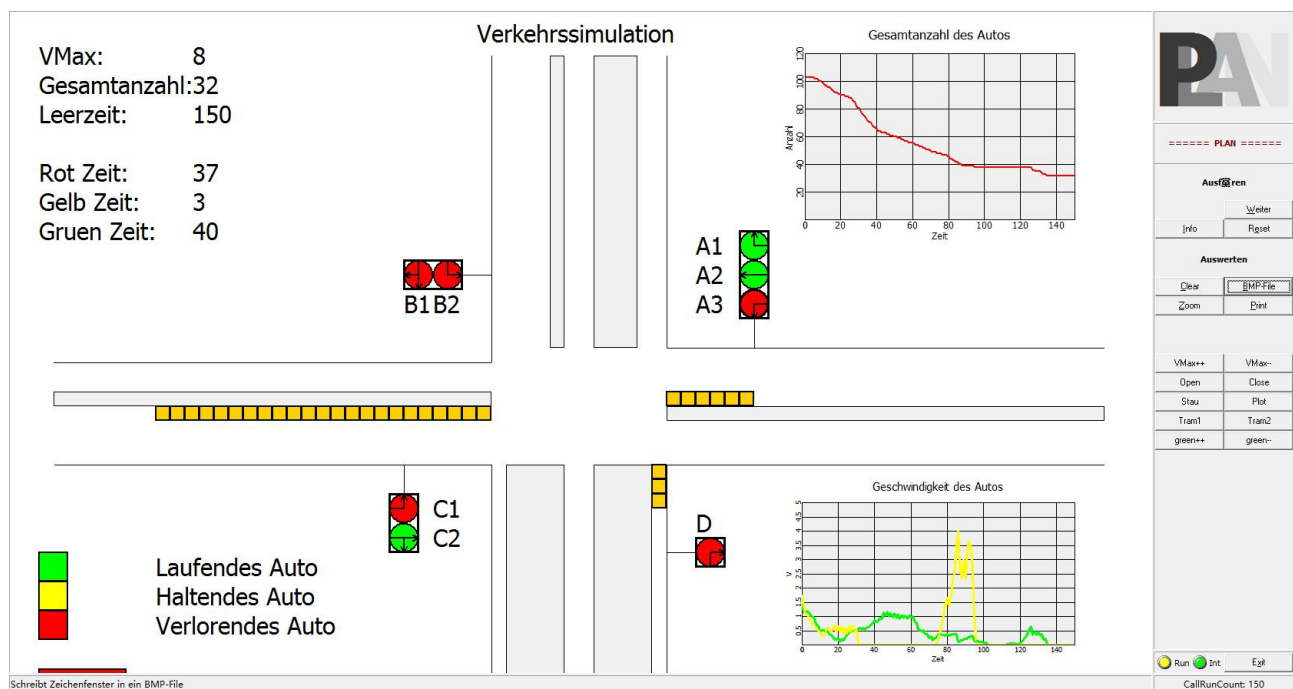


Abbildung 3.1.4 Der schlechten Fall(Zeit:0—150)



Im Zeitraum 0—150 sind die Kurven der Gesamtanzahl des Autos von dem besten Fall im Vergleich zu dem schlechten Fall ähnlich und Gesamtanzahl ist im besten

Fall größer( $42 > 32$ ), d.h. mehrere Fahrzeuge, die noch nicht aus der Kreuzung gefahren hätten.

Abbildung 3.1.5 Der besten Fall(Zeit:150—300)

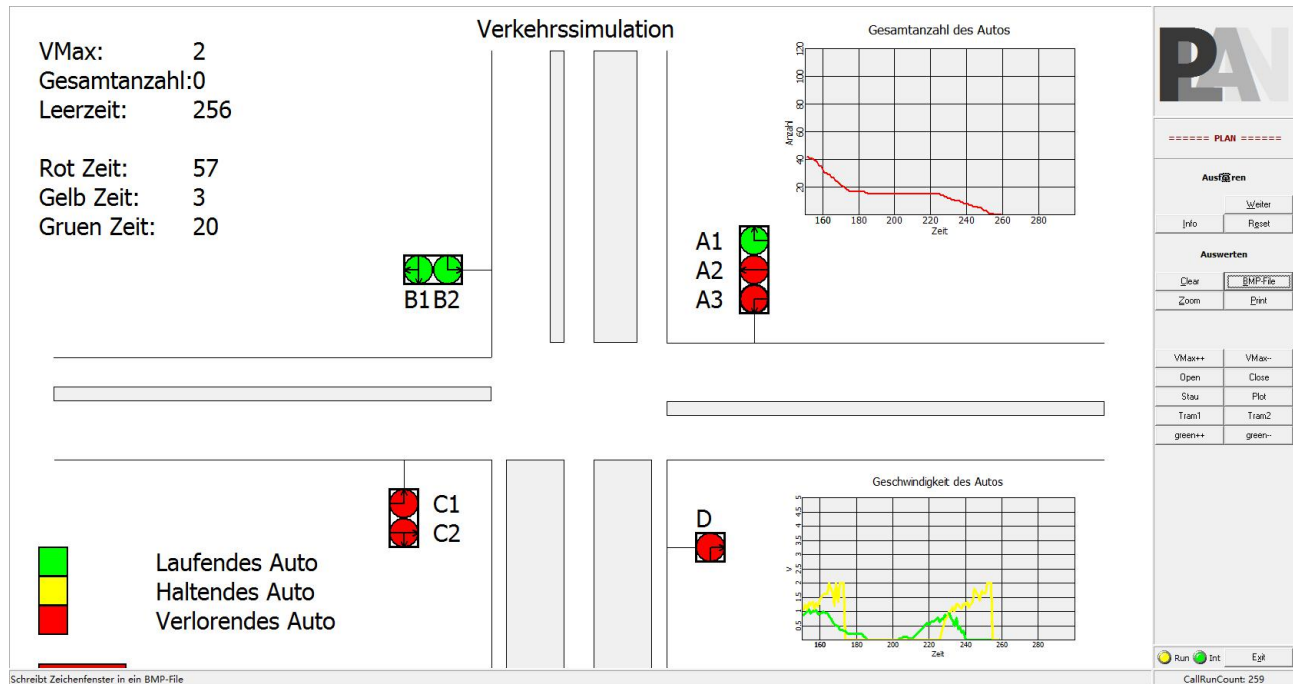
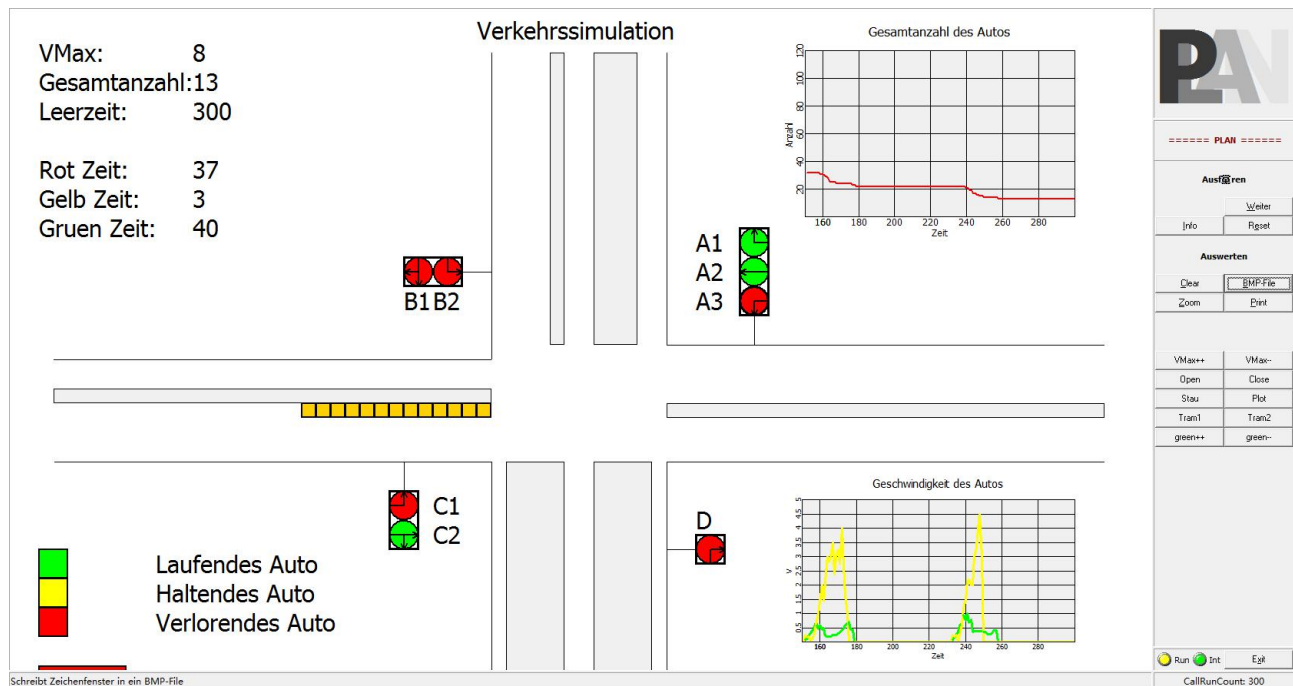
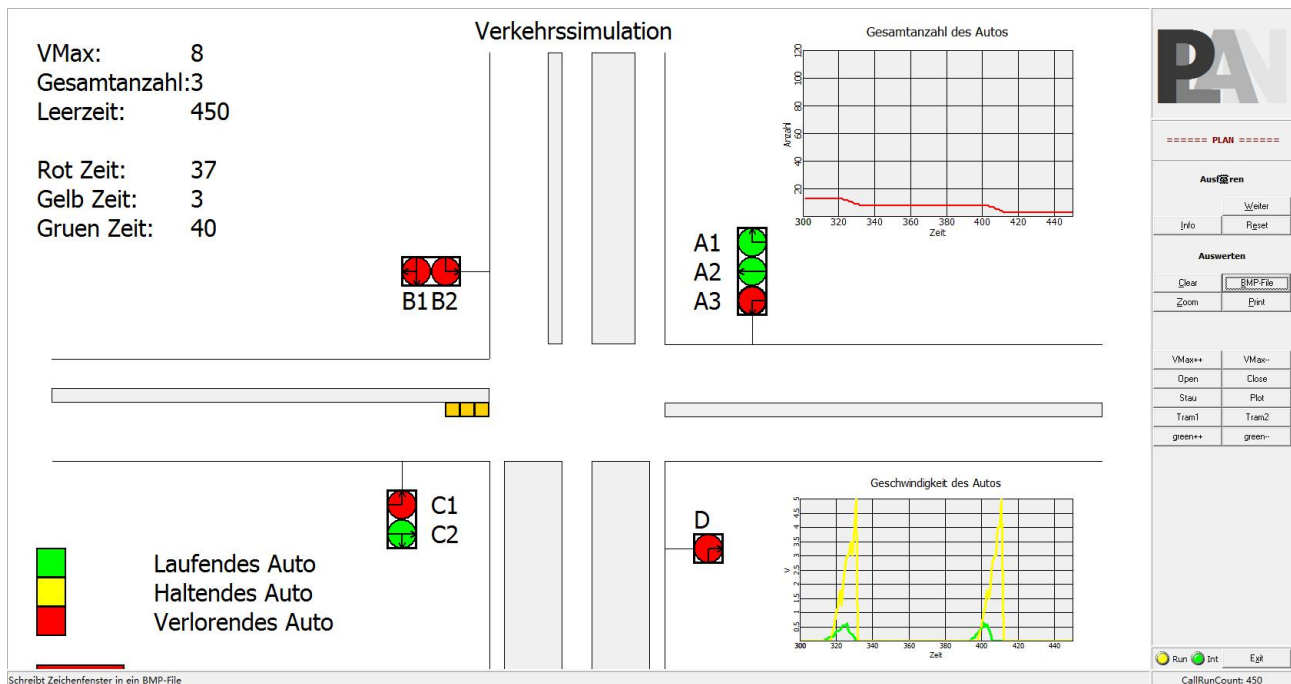


Abbildung 3.1.6 Der schlechten Fall(Zeit:150—300)



**Abbildung 3.1.7 Der schlechten Fall(Zeit:300—450)**



Im Zeitraum 150—450 aus der Abbildung 3.1.5, Abbildung 3.1.6, Abbildung 3.1.7, könnte man beobachten, dass die Kurve im schlechten Fall meistens gleich bleibt, d.h. im diesen Fall hätten die nach links biegenden Fahrzeuge lange auf die Grünlichte gewartet. Sodass wäre die Leerzeit von den schlechten Fall deutlich größer als die von den besten Fall bzw.  $489 > 256$ .

Andere schlechten Fälle sind analog.

### **Zusammenfassung:**

Im Zeitraum 0 — 150 könnte man ergeben, dass die Fahrzeuge mit höhere Maximalgeschwindigkeit bzw.  $V_{Max} = 8$  schneller aus der Kreuzung fahren könnten.

Im Zeitraum 150 — 450 könnte man ergeben, dass die Dauer des Grünlichts am Ampel von die Menge der Fahrzeuge abhängen soll, d.h. je mehr die Fahrzeuge auf einer Spur wären, desto länger soll das entsprechende Grünlicht bleiben.

### 3.1.4 Trambahn und Leerzeit

Wenn eine Trambahn in der Kreuzung einf hrt, w rden alle Ampeln auf Rot wechseln und alle Fahrzeuge auf der Trambahn warten.

Abbildung 3.1.8 Trambahn und Leerzeit

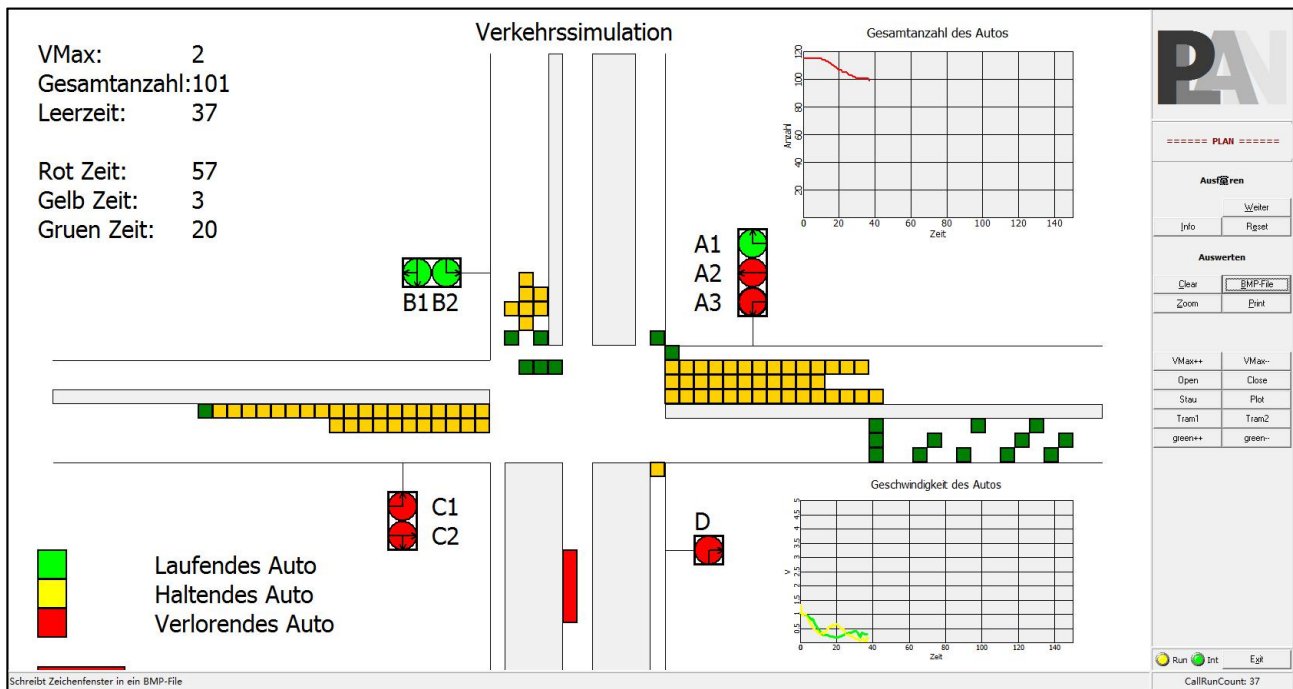
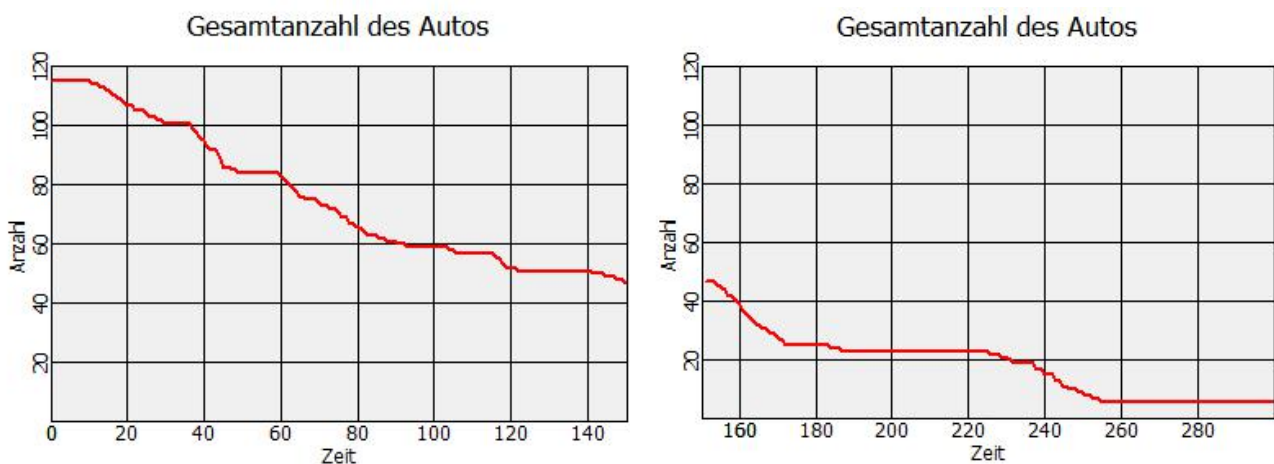
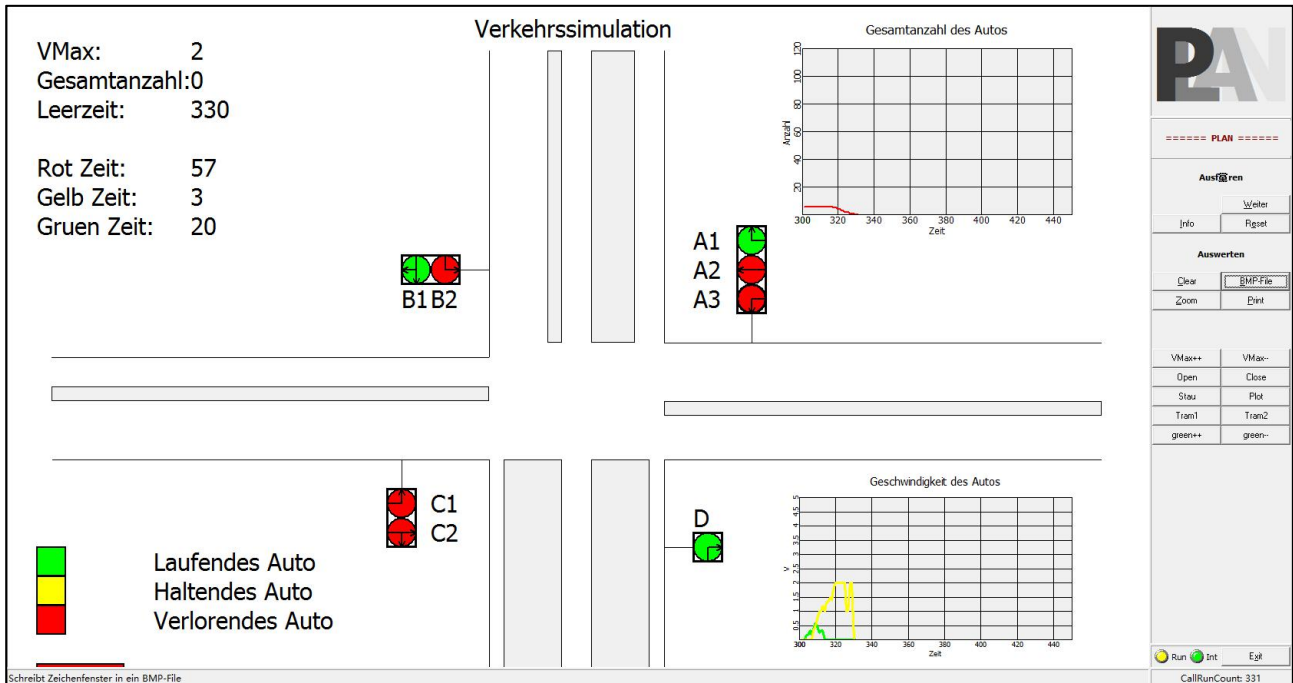


Abbildung 3.1.9 Trambahn und Leerzeit (0—300)



**Abbildung 3.1.10 Trambahn und Leerzeit (300—330)**



Unter der Anfangsbedingungen  $V_{Max} = 2$ , Dauer des Grünlichts = 20 bzw. der besten Fall aus dem Formular 3.1.3 wäre die Leerzeit davor 256 gleich. Mit einer Trambahn wäre die Leerzeit dagegen 330 gleich, also um 74 größer als die in dem besten Fall.

Wenn eine Trambahn in der Kreuzung einführe, dauert die Wartenzeit theoretisch gegen 15, also viel kleiner als die zugenommene Zeit 74.

### **Zusammenfassung:**

Unter der Anfangsbedingungen von dem besten Fall könnte man ergeben, obwohl die Trambahn eine kurze Zeit braucht, aus der Kreuzung zu fahren, hätte die Eintritt der Trambahn einen großen Einfluss auf die Leerzeit.

## 3.2 Ausblick

1. Als zusätzlicher Faktor könnte Baustelle hinzugefügt werden.
2. Als zusätzlicher Faktor könnten Passanten und Fahrräder hinzugefügt werden.
3. Als zusätzlicher Faktor könnte Feuerwehrfahrzeug und Krankenwagen hinzugefügt werden.
4. Momentan ist das Überholmanöver nicht auffällig, da das in dieser Verkehrssimulation aufgebaute Straßennetz zu kurz ist. Zum Verbessern könnte das Straßennetz verlängert werden.
5. Ein weiteres Modell könnte mit Verbindung von 2 Kreuzungen aufgebaut werden, um die Verkehrslage zwischen diesen 2 Kreuzungen zu beobachten.
6. Darüber hinaus könnten Mehreren Kreuzungen miteinander verbunden werden, um die Verkehrslage einer Stadt zu simulieren.

# Literaturverzeichnis

- [1] *Kai Nagel, Michael Schreckenberg: A cellular automaton model for freeway traffic. (PDF, 504 KB) In: J. Phys. I France, 2, 1992, S. 2221–2229*
  
- [2] Ostermeyer, G.-P.: *Simulation mechatronischer Systeme*, Vorlesungsskript Institut für Dynamik und Schwingungen, TU Braunschweig, Braunschweig, 2014

# Anhang

## A.1 Vollständiges Programm