EASY FLOW

VERKEHRSSIMULATION

BENUTZERHANDBUCH

Projektmitarbeiter:

Werner Böhm

Petra Jordan

Thomas Paumann

Studienfach:

Softwareentwicklung für Telematikanwendungen

Betreuer:

Philipp Urbauer, MSc

Mathias Ballner, MSc

Projektzeitraum:

WS 2014 / 2015

INHALTSANGABE

Technische Voraussetzungen

Eclipse Version Kepler
Java Version 1.8.0_20-b26
Die Letztversion des Programms liegt auch auf dem SVN Server

(auch die Komponente JDOM, die für das Lesen vom XML File benötigt wird)
Vielleicht auch, was für eine Version von Java und Eclipse wir verwendet haben
Anleitung wie die Daten zur Verfügung gestellt werden sollen (keine Ahnung was er hier meint)
Beschreibung der verwendeten Regeln in der Simulation (dito, k.A. was er hier haben will)

Eckdaten des Projekts, Besonderheiten

- Es wurde darauf Wert gelegt, dass der gesamte Code inklusive den Kommentaren einheitlich in Englisch verfasst wurde.

- Es wurde ein MVC Modell verwendet

- Die Planung wurde in UML durchgeführt (siehe beiliegendes Dokument im Projekt Workspace)

- Für die Dateneinspielung wurde ein XML-File verwendet

Umsetzungen der geforderten Punkte

1. *Optisch angenehme Darstellung einer Stadt und des Verkehrsaufkommens - umgestetzt*
2. *Datenmodell - umgesetzt*
3. *Unterschiedliche Städte und Verkehrsteilnehmer können modelliert und geladen werden.*
    Größe: 8-12 Kreuzungen - umgesetzt, aber nur mit
    Vereinfachung: eine einzige Stadt ist vordefiniert
4. *Verkehrsteilnehmer werden nach definierten Regeln in das Modell eingespielt bzw. entfernt. - umgesetzt*
5. *Fahrzeuge suchen sich eine optimale Fahrroute mittels Dijkstra und werden am Ziel wieder aus dem Verkehr entfernt*
- (Vereinfachung: Punkt wird nicht implementiert; Suche nach einem optimalen Weg

fehlt) - noch nicht umgesetzt

6. *Manuelle Manipulation von Signallichtanlagen - einzelne Manipulation umgesetzt*

7. *Einbau von Baustellen/Hindernissen - umgesetzt, Einbau und Entfernen*

8. *Fahrzeuge können andere Fahrzeuge und Hindernisse und Baustellen überholen/passieren.* - Baustellen können passiert werden

Vereinfachung: Andere Fahrzeuge können nicht überholt werden wegen Punkt 14

9. *Straßenarten- umgesetzt*

10. *KFZ-Arten* - umgesetzt, alle drei: - PKW, LKW, Busse

11. *Identifikation von Stau* - Umgesetzt, auf 2 Arten (über Alarm in ControlPanel UND farbige Markierung direkt in der Simulation

12. *Ein beliebiges Optimierungsverfahren für die Signallichtanlagen um Staus zu reduzieren.* (Benotung: Z.B wenn bei Stau die Fahrzeuge stehen und entsprechend als Reaktion die Ampel auf Grün schaltet => volle Punkte)

- voll umgesetzt

13. *Vorrangregeln entsprechend STVO* (Benotung: individuell) - umgesetzt

14. *Fahrzeuge können mit unterschiedlicher Geschwindigkeit fahren (unterschiedliche diskrete Geschwindigkeitsstufen z.B. 1-5)* (Benotung: *6P => alle Fahrzeuge, individuell => 3P alle Fahrzeuge gemeinsam (=Simulationsgeschwindigkeit))*
Vereinfachung in Anspruch genommen - alle Fahrzeuge gemeinsam, Simulationsgeschwindigkeit über Regler, daher auch kein Überholen möglich (siehe Punkt oben)

15. *Dokumentation in Form eines Benutzerhandbuch* - umgesetzt

Aufgabenaufteilung

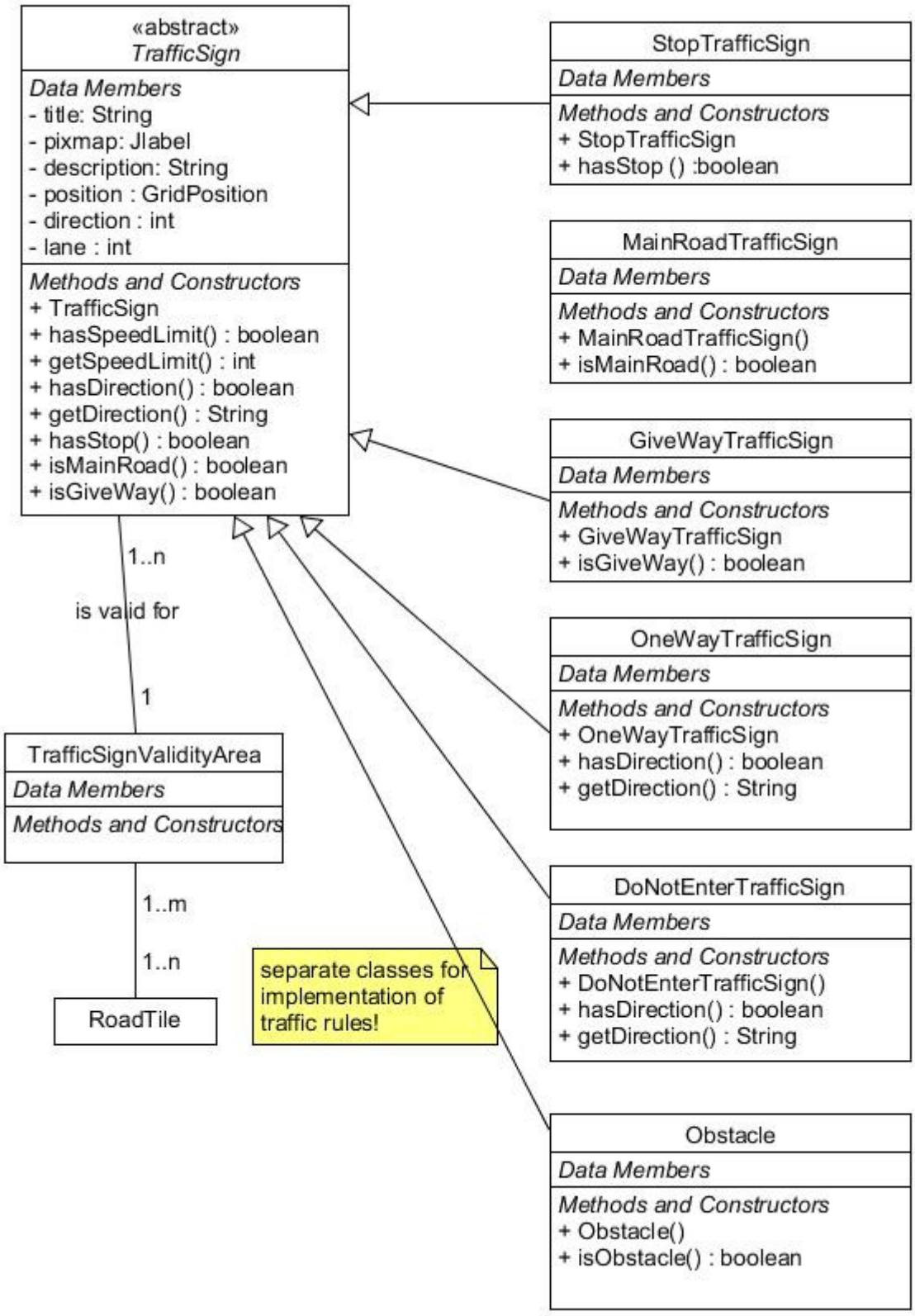
Werner Böhm: Design(MVC, Framework), Datenmodell

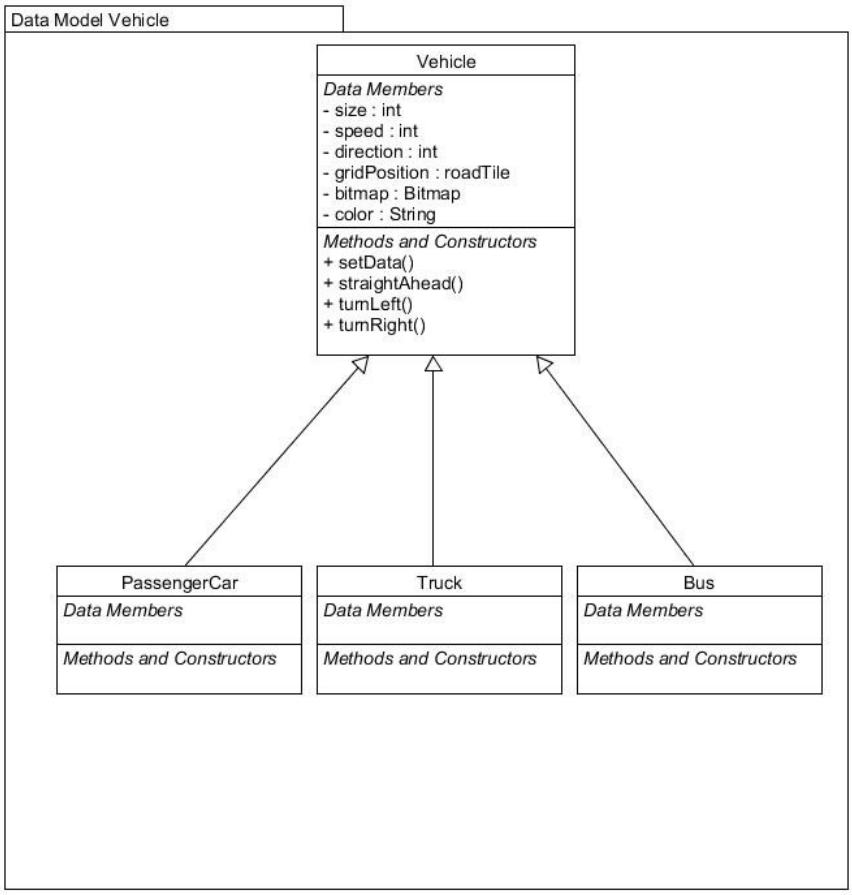
Petra Jordan: Bewegungslogik, Grafik

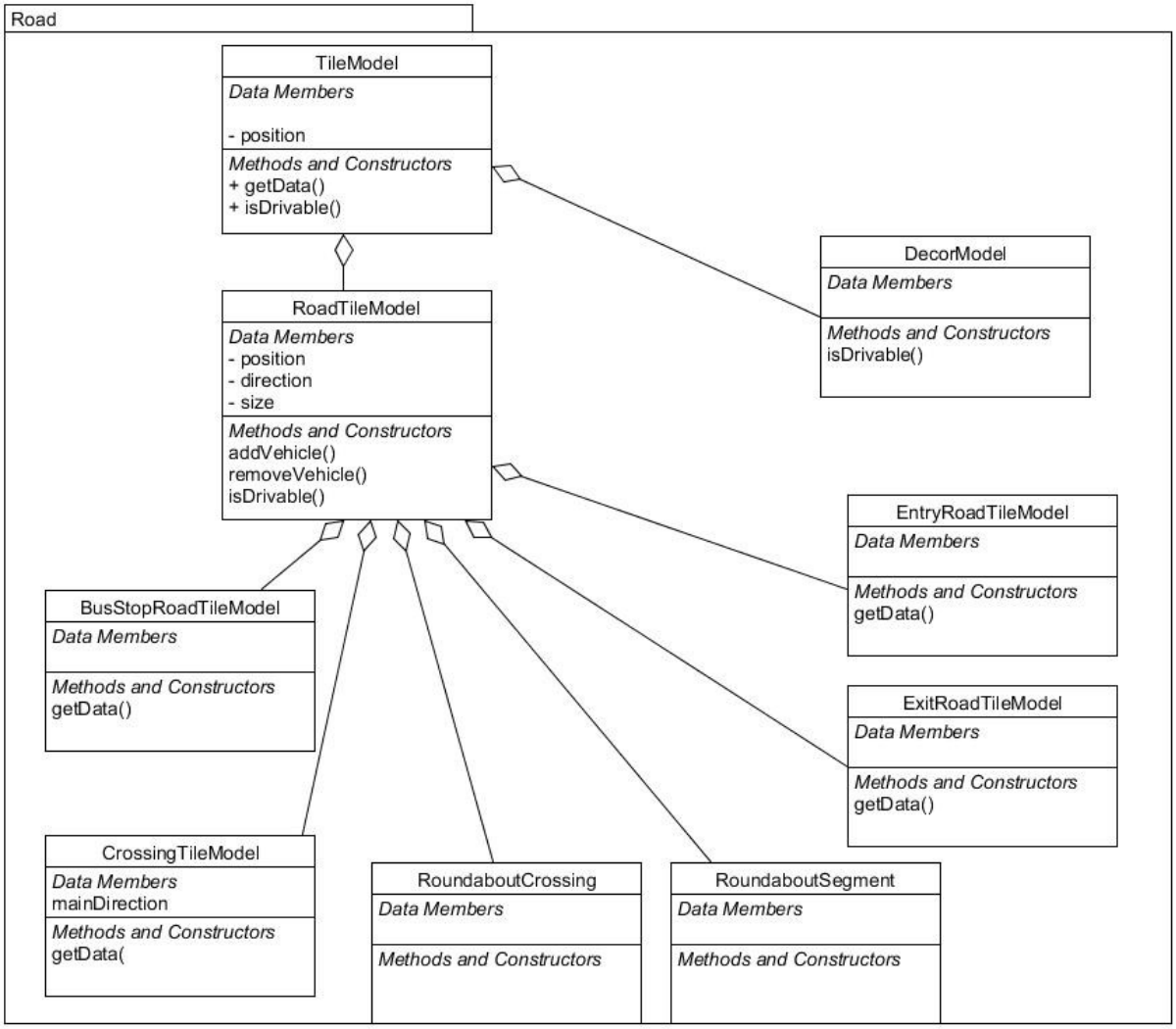
Thomas Paumann: XML, Dijkstra

DAS DATENMODELL

Basis der Applikation EaysFlow ist ein realitätsnahes Objektmodell.
Durch Bereitstellung von Basisklassen und der Möglichkeit der Vererbung kann das Modell mit geringem Aufwand an individuelle Kundenwünsche angepasst werden.
Das Datenmodell hält beides die Daten selbst als auch die zugrundeliegende Business-Logik.
Neue Klassen von Straßen, Verkehrszeichen oder Fahzeugen werden auf diese Weise bestmöglich unterstützt. Als Beispiel für das ausgereifte Obejektmodell sind im Folgenden UML Diagramme der Verkehrszeichen, Straßen-Bausteine und Fahrzeuge abgebildet.


«abstract»
TrafficSign
Data Members
- title: String
- pixmap: Jlabel
- description: String
- position : GridPosition
- direction : int
- lane : int
Methods and Constructors
+ TrafficSign
+ hasSpeedLimit() : boolean
+ getSpeedLimit() : int
+ hasDirection() : boolean
+ getDirection() : String
+ hasStop() : boolean
+ isMainRoad() : boolean
+ isGiveWay() : boolean
StopTrafficSign
Data Members
Methods and Constructors
+ StopTrafficSign
+ hasStop () :boolean
MainRoadTrafficSign
Data Members
Methods and Constructors
+ MainRoadTrafficSign()
+ isMainRoad() : boolean
GiveWayTrafficSign
Data Members
Methods and Constructors
+ GiveWayTrafficSign
+ isGiveWay() : boolean
OneWayTrafficSign
Data Members
Methods and Constructors
+ OneWayTrafficSign
+ hasDirection() : boolean
+ getDirection() : String
DoNotEnterTrafficSign
Data Members
Methods and Constructors
+ DoNotEnterTrafficSign()
+ hasDirection() : boolean
+ getDirection() : String
Obstacle
Data Members
Methods and Constructors
+ Obstacle()
+ isObstacle() : boolean
1..n
is valid for
1
TrafficSignValidityArea
Data Members
Methods and Constructors
1..m
1..n
RoadTile
separate classes for
implementation of
traffic rules!

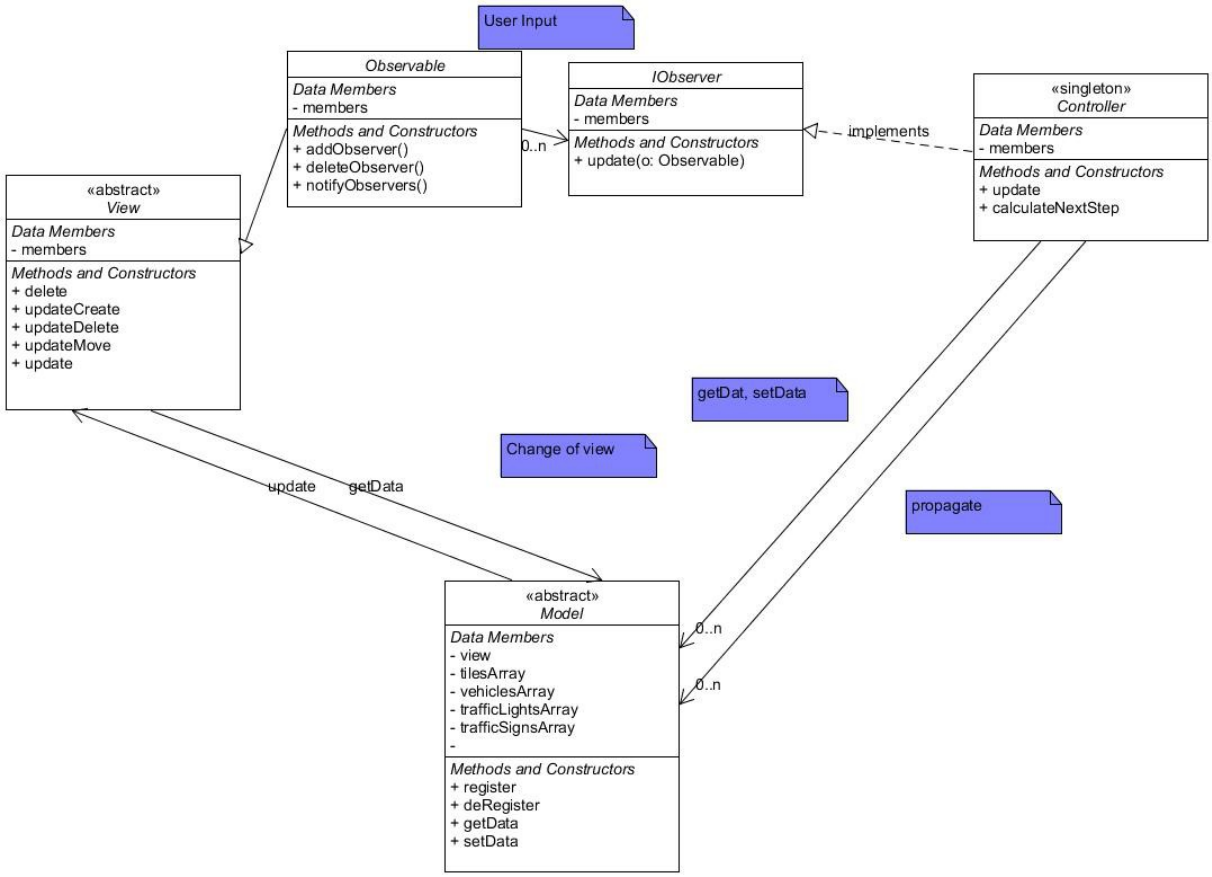

Data Model Vehicle
Vehicle
Data Members
- size : int
- speed : int
- direction : int
- gridPosition : roadTile
- bitmap : Bitmap
- color : String
Methods and Constructors
+ setData()
+ straightAhead()
+ turnLeft()
+ turnRight()
PassengerCar
Data Members
Methods and Constructors
Truck
Data Members
Methods and Constructors
Bus
Data Members
Methods and Constructors



Model View Controller (MVC)

MVC ist ein Design Pattern, dass in der Software Entwicklung häufig für GUI Applikationen verwendet wird. Durch die Auftrennung in Präsentation, Modell und Controller gewährleistet MVC die Flexibilität hinsichtlich späterer Erweiterungen und Portierungen.

Das DatenModell enthält die Business Logik und die Daten selbst und bleibt unabhängig von der plattform-spezifischen Implementierung der Präsentation.



DAS GRID

Um Verzerrungsprobleme mit unterschiedlichen Screenformaten und Auflösungen zu vermeiden, wurde der Frame auf 1024x768 fixiert.
Die Oberfläche ist eingeteilt in das Grid, die Gridbeschriftung und das ControlPanel. Das Grid wiederum wurde mittels GridBagLayout erstellt, per Aufteilung in 16x16 Pixel grosse Zellen ohne Gap dazwischen, was ein Raster von 48x45 Zellen erzeugt, in dem die Simulation abläuft.
Jede dieser Zellen bekommt per XML Import eine Typenzuordnung (Decor Tile, Road Tile, Crossing Tile usw) und zugehörige Eigenschaften (zB Richtung).
Es gibt 8 Richtungen, die der groben Windrose im Uhrzeigersinn folgen (1=Norden, 2=Nordosten, 3=Osten, ... , 7=Westen, 8=Nordwesten). Fahrzeuge können sich auf einem Tile mit Eigenschaft Richtung 3 (Osten) auch nur nach Osten bewegen - es sei

denn, das Fahrzeug führt ein Spezialmanöver durch, wie zB es weicht einem Hindernis aus.
Die Achsenbeschriftung dient dem Überblick und der besseren Orientierung, das Grid selbst gibt der View die Positionen vor, in die Labels geladen werden können. Die meisten Labels sind 1x1 Gridzelle gross (PassengerCars, Verkehrszeichen und Ampeln), es gibt allerdings auch Labels, die 2x2 Zellen füllen (Busse und Trucks), und auch welche, die 5x1 bzw 1x5 füllen (Optischer Alarm für Staudetektion). Mehr zur Darstellung in den jeweils entsprechenden Kapiteln.

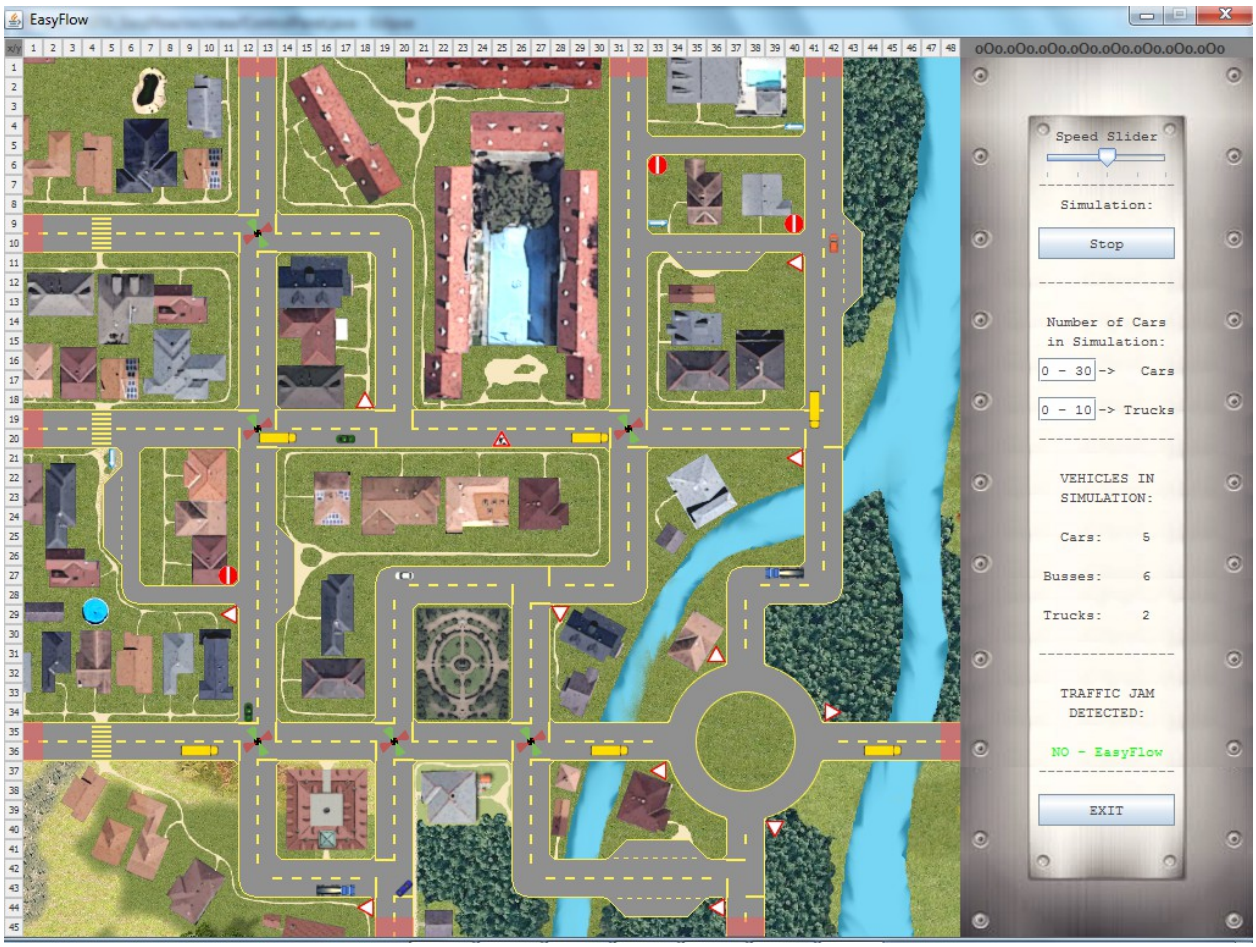


Abbildung: Ansicht der Programmoberfläche

Die Strassen, Kreuzungen, Exit oder Entrypunkte und alles andere an Dekor ist ein Hintergrundbild, das per Hand mit GIMP2 gezeichnet wurde und mit fixer Grösse in das Panel geladen wird und mit den Gridpositionen und damit auch den Labels abgestimmt ist.

XML FILE “city.xml”

In diesem File sind einige grundlegende Daten definiert.

1. Der Aufbau des GRIDs

Das Grid besteht aus 45 (0-44) Zeilen und 48 Spalten (0-47)
Alle Straßen-Tiles sind im XML-File definiert. Es wird für jede GRID-Position geprüft, ob es ein Straßenelement ist. Wenn im XML-File ein Eintrag für die GRID-Position gefunden wird, so wird der Eintrag gelesen, ausgewertet und das entsprechende Tile erzeugt und im GRID an der entsprechenden Stelle eingehängt.

2. Verkehrszeichen und Ampeln

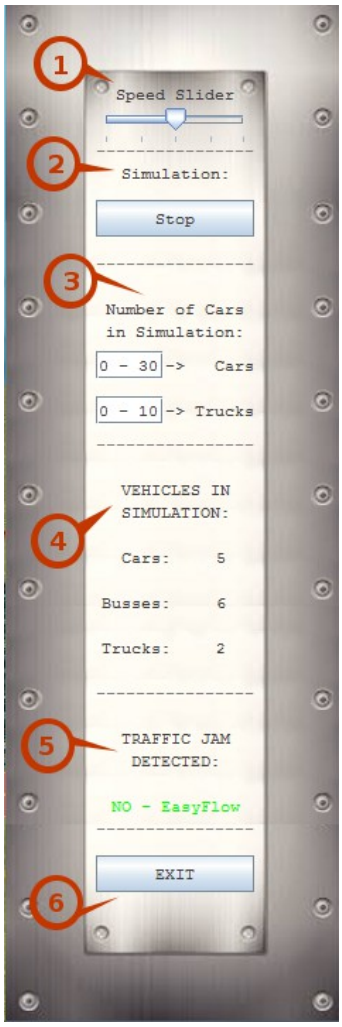
Die genaue Lage der Verkehrszeichen und Ampeln und die Voreinstellung der Ampelfarbe.

3. Autos und Busse

Autos und Busse, die unmittelbar nach dem Start in der Simulation aufscheinen und losfahren, sind in city.xml definiert.

Der Zugriff auf das XML-Dokument wurde mit der Komponente JDOM realisiert. JDOM bietet eine einfache Möglichkeit, XML-Dokumente über eine Java-API zu nutzen. JDOM wurde in das Eclipse Projekt aufgenommen und wird für den Ablauf von EasyFlow benötigt.

DAS CONTROL PANEL



1 - Speed Slider

Möglichkeit zur Änderung der Simulationsgeschwindigkeit in 5 Stufen. Damit kann die Zeit pro Simulationsschritt eingestellt werden. Der Wertebereich geht von 180ms (rechter Anschlag) bis 900ms (linker Anschlag).

2 - Simulation Stop / Start

Hält die Simulation an / startet sie wieder.

3 - Number of Cars in Simulation

Hier kann der User die Gesamtanzahl der PassengerCars und Trucks in der Simulation einstellen. Busse sind immer dieselbe Anzahl in der Simulation. Ist die hier eingestellte Gesamtanzahl höher als die derzeit in der Simulation befindlichen Fahrzeuge, so werden an den Rändern (EntryTiles) solange neue Fahrzeuge des gewünschten Typs generiert, bis die Gesamtanzahl erreicht ist.

4 - Vehicles in Simulation

Liefert einen schnellen Überblick über die Anzahl nach Fahrzeugtypen, die sich gerade in der Simulation befinden.

5 - Traffic Jam detected

Hierüber kann schnell festgestellt werden, ob es irgendwo in der Stadt gerade einen Verkehrsstau gibt, an dessen Behebung gerade gearbeitet wird oder ob der Verkehrsfluss ungehindert ist.

6 - Exit

FUNKTIONALITÄTEN DER SIMULATION

Die Verkehrslichtsignalanlagen

Die VLSA oder im Weiteren kurz Ampeln genannt, dienen zur Steuerung des Verkehrsflusses an einer 4-direktionalen Kreuzung. Wenn die Verkehrsteilnehmer eine solche Kreuzung erreichen, erkennen sie zuerst das Existieren einer Ampel und registrieren sodann, ob die Ampel auf rot oder grün steht.
Eine Kreuzung besteht aus 4 getrennten Ampeln, die aber voneinander abhängig sind und daher nur gemeinsam manipuliert werden können.
Rot: Gibt den Verkehrsteilnehmern bekannt, dass sie stehenbleiben müssen, die Fahrzeuge warten also solange, bis der Zustand der Ampel sich ändert
Grün: Gibt den Verkehrsteilnehmern bekannt, dass es ihnen erlaubt ist, die Kreuzung WENN MÖGLICH zu passieren. “Wenn möglich” deswegen, weil die Verkehrsteilnehmer dann entscheiden müssen, ob sie Rechtsabbiegen, Geradeausfahren oder Linksabbiegen müssen, und dies wiederum den Vorrangregeln laut StVO folgt. Die nähere Beschreibung der programmierten Bedingungen sind den betreffenden Kapiteln über die Abbiegevorgänge selbst zu entnehmen. Der Verkehrsteilnehmer führt den weiteren Bewegungsvorgang nach Einhalten einer kurzen realistischen Reaktionszeit (2 Controller-Ticks) durch.

Funktionalität VLSA: Umschalten des Signals in festgelegten Intervallen

Bei freiem Verkehrsfluss folgt das automatische Umschalten des Ampelsignals einem vorgegebenen Intervall.
Die technische Realisierung dieses Vorgangs wurde mithilfe eines inkrementierenden Counters gelöst.

Funktionalität VLSA: Umschalten des Signals bei Staudetektion

Wird ein Stau erkannt, schaltet das Ampelsignal intelligent um, um den zähen

Verkehrsfluss zu entlasten. Näheres hierzu ist dem Kapitel Stauerkennung zu entnehmen

Funktionalität VLSA: Umschalten des Signals manuell durch den Benutzer

Der Benutzer kann jederzeit in die Simulation eingreifen, um die Ampel zu manipulieren. Per rechtem Mausklick auf die gewünschte Ampel öffnet sich ein Pop-Up-Menü, über das bequem die Ampelphase gewechselt werden kann.
Die technische Realisierung beinhaltet ein Zurücksetzen des Intervall-Counters (die vom Benutzer eingeleitete Intervallphase kann also ganz bis zu ihrem “natürlichen” Ende durchlaufen). Eine Ampelanlage besteht aus 4 Einzelsignalen. Bei Mausklick wird ermittelt, welche der Ampeln ausgewählt wurde, und die 3 mit ihr gekoppelten Ampeln werden ebenfalls umgeschalten.

Stauerkennung

In der Simulation wird zu jedem Zeitschritt abgefragt, ob an einer oder mehreren Ampelkreuzungen ein Stau entstanden ist, indem jede Ampel die Zellen in ihre Richtung auf erhöhtes Verkehrsaufkommen (3 Fahrzeuge oder mehr) untersucht und darauf wie im Weitere beschrieben reagiert.
Der Benutzer wird auf zweierlei Arten auf ein Stauereignis aufmerksam gemacht: Einerseits durch die Statusanzeige im ControlPanel, die generell darüber informiert, ob der Verkehrsfluss in der Stadt im grünen Bereich ist oder irgendwo gerade an einer Behebung gearbeitet wird. Diese Information im ControlPanel ist nur auf grün und “EasyFlow”, wenn kein einziger Stau in der Stadt erkannt wurde. Andererseits wird auch jeder Stau einzeln bekannt gegeben, indem die exakte Stelle, an der er auftritt, einen optischen Alarm gibt, nämlich ein orangefarbiges Overlay über den verstopften Strassenzug.
Wie schon in Kapitel “Verkehrslichtsignalanlagen” beschrieben, reagiert die Ampel, bei der der Stau aufgetreten ist, automatisch auf den Stau, indem sie intelligent umschaltet und damit eine Auflösung des zähen Verkehrsaufkommens herbeiführt.

Verkehrszeichen

In der Simulation existieren 3 verschiedene Arten von Verkehrszeichen, und alle davon unterstützen die Verkehrsteilnehmer dabei, die StVO einhalten zu können.

Vorrang geben:

Dieses Verkehrszeichen befindet sich an Kreuzungen, die nicht von einer Ampel geregelt werden und geben den Verkehrsteilnehmern bekannt, dass der kreuzende Verkehrszug Vorrang hat. Sobald die Verkehrsteilnehmer erkennen, dass ein Fahrzeug auf der bevorrangten Strasse nahe ist, warten sie darauf, bis dieses die

Kreuzung passiert hat, bevor sie erneut nach links und rechts schauen. Sie fahren erst los, wenn die Kreuzung frei ist.

Einbahnen:

Diese Strassenzüge dürfen nur in die vom Verkehrszeichen angegebene Richtung befahren werden

Einfahrt verboten:

Dieses Verkehrszeichen befindet sich am Ende einer Einbahn und signalisiert den Verkehrsteilnehmern, dass sie hier nicht einbiegen dürfen

Baustellen / Hindernisse

Der Benutzer kann jederzeit bequem per Rechtsklick über das sich öffnende Pop-Up-Menü eine Baustelle in der Simulation platzieren, auf die das Verkehrsgeschehen reagieren soll.

Das Platzieren einer Baustelle ist nicht überall gestattet, sondern nur, wo ein ungefährliches Passieren des Hindernisses für die Verkehrsteilnehmer auch möglich ist.

Die technische Realisierung besteht aus einer Abfrage diverser Eigenschaften der unmittelbaren Umgebung des Mausklicks des Benutzers. Die Zellen um den Mausklick herum werden abgefragt, ob sich Bedingungen dort befinden, die ein Platzieren eines Hindernisses nicht erlauben. Diese Bedingungen sind:

- Ampelkreuzung
- Kreuzung mit Verkehrszeichenregelung
- unübersichtliche Kurve
- andere Baustelle, die sich zu nahe befindet
- Bushaltestelle
- Einbahn
- Exit / Entry Position (Stadtrand)
- Oder wenn es sich bei der Zelle, auf die geklickt wurde, gar nicht um eine Strasse handelt (Gebäude, Fluss, Parks usw)

Wurden keine hindernden Bedingungen gefunden, so öffnet sich erwähntes Pop-Up-Menü für den Benutzer, und er kann sich entscheiden, ob er eine Baustelle setzen will, auf die sämtliche Verkehrsteilnehmer mit einer programmierten Bewegungsabfolge reagieren, dem “Ausweichen”, natürlich nur nachdem sie erkannt haben, dass die Gegenfahrbahn und auch der Platz hinter der Baustelle frei für einen Ausweichvorgang ist.

Als zweiter Menüpunkt wird das Setzen eines neuen Fahrzeuges in die Simulation angeboten, diese Funktion wird im Kapitel “Einspielung / Entfernen von Verkehrsteilnehmern” beschrieben.

Eine Baustelle kann jederzeit wieder aus der Simulation entfernt werden, indem der

Benutzer einfach wieder per Rechtsklick den entsprechenden Menüpunkt “Entfernen” wählt.

Verschiedene Fahrzeugtypen

Es wurden 3 verschiedene Fahrzeugtypen in die Simulation eingebaut.

PassengerCars:

Die PassengerCars sind das Herzstück der Simulation und repräsentieren die statistische Mehrzahl der Verkehrsteilnehmer.
Anfangs befinden sich 5 PassengerCars in der Simulation, die per XML Import eingespielt werden. Der Benutzer kann allerdings mithilfe des Eingabefelds im ControlPanel jederzeit die Gesamtanzahl der PassengerCars ändern, die er sich in der Simulation wünscht. Daraufhin werden solange auf den freien EntryTiles, den Eingängen, neue PassengerCars generiert (mit zufälligen Eigenschaften), bis die gewünschte Gesamtanzahl erreicht ist. Näheres dazu ist im Kapitel “ControlPanel” beschrieben.
Darüberhinaus können weitere PassengerCars auch manuell per Rechtsklick durch den Benutzer in die Simulation platziert werden - Näheres hierzu im Kapitel “Einspielung / Entfernung der Verkehrsteilnehmer”.
PassengerCars kommen in vier unterschiedlichen Farben vor, und der zugehörige Bitmap Sprite belegt genau eine 16x16 Pixel Zelle. Je Bewegungsrichtung wird ein neues Label für die PassengerCar Bitmap gezeichnet und eingefügt

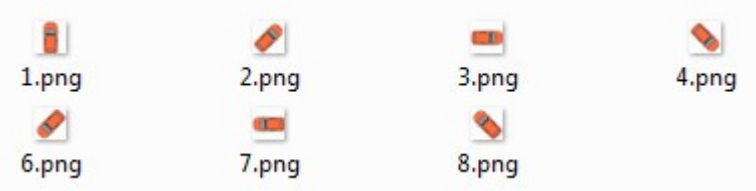


Abbildung: Auszug der PassengerCar Sprites

Busse:

Es befinden sich zu jeder Zeit 6 Busse in der Simulation, die per XML Import in die Bushaltestellen eingespielt werden, und diese nur verlassen, wenn die Fahrbahn frei ist. Ursprünglich war geplant, dass jeder Bus sich nach einer speziellen Route durch die Simulation bewegt, von Haltestelle zu Haltestelle. Die aktuelle Realisierung beinhaltet nun aber die Erhaltung der Gesamtanzahl der Busse in der Simulation - verlässt ein Bus die Stadt, wird ein anderer generiert.
Die Bus Bitmaps umfassen 2x2 Zellen in der Simulation, also 32x32 Pixel. Für jede Bewegungsrichtung muss also der Anchor bestimmt werden, wo das Label platziert werden darf.

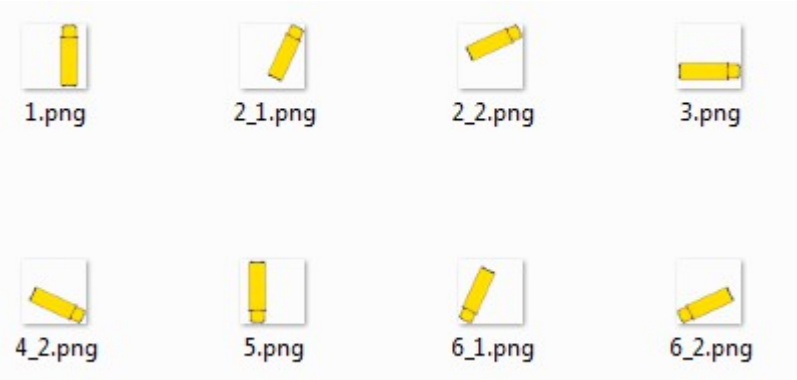


Abbildung: Auszug der Bus Sprites

Trucks:

Ebenso wie bei den PassengerCars kann auch bei den Trucks der Benutzer bestimmen, wieviele er gerne gesamt in der Simulation hätte (bis zu 10) - Näheres hierzu im Kapitel “ControlPanel”.
Anfangs befinden sich 2 Trucks in der Simulation, die per XML Import eingespielt werden und sich dann entsprechen der StVO durch das Verkehrsgeschehen bewegen.
Da auch die Truck Bitmaps 2x2 Zellen belegen, also 32x32 Pixel, musste auch hier wieder für jede Bewegungsrichtung einzeln definiert werden, wo der Anchor für das Label sein muss.
Trucks bewegen sich nach dem OldDirection-NewDirection Prinzip. Vor jeder Richtungsänderung wird die derzeitige Richtung als “OldDirection” abgespeichert und erst dann die “NewDirection” vergeben - und damit wird die Bitmap vollautomatisch erzeugt, da sich das Vorderende des Trucks in die NewDirection dreht, das Hinterende des Trucks aber in die OldDirection gerichtet ist.

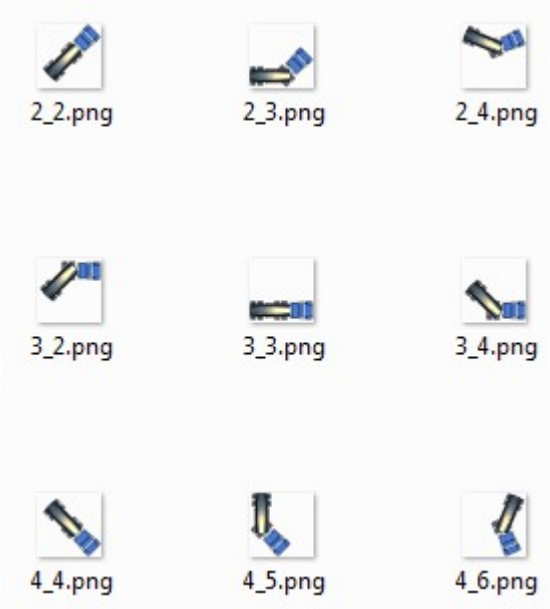


Abbildung: Auszug der Truck Sprites

Einspielung / Entfernen der Verkehrsteilnehmer

Hinzufügen von Verkehrsteilnehmern:

Ursprungszustand: Der Ursprungszustand wird über den XML Import bestimmt. Anfangs werden 5 PassengerCars, 6 Busse und 2 Trucks in die Simulation eingespielt.

Automatisches Hinzufügen: Das automatische Hinzufügen von Verkehrsteilnehmern passiert über einen zufälligen Generierungs-Automatismus über die EntryTiles (Stadtrand). Ist die gewünschte Maximalanzahl eines KFZ-Typs noch nicht erreicht, so wird abgefragt, welche EntryTiles frei sind und darauf werden solange neue Fahrzeuge generiert, bis die Maximalanzahl erreicht ist.
Für die Typen PassengerCar und Bus kann der Benutzer selbst die Maximalanzahl über das Textfenster im ControlPanel bestimmen - mehr hierzu im Kapitel “ControlPanel”

Manuelles Einspielen: Der Benutzer kann darüberhinaus auch jederzeit nach Belieben und bequem per Rechtsklick direkt in die Simulation platzieren, indem er den entsprechenden Punkt in dem Pop-Up-Menü auswählt. Hierfür ist keine Begrenzung vorgesehen, nur das automatische Generieren von KFZ wird angehalten, wenn die Maximalanzahl schon erreicht oder überschritten ist.

Entfernen von Verkehrsteilnehmern:

Die Verkehrsteilnehmer finden ihr natürliches Ende indem sie die Simulation per Befahren eines ExitTiles verlassen.

Die Bewegungslogik

Vorausschauends Fahren:

Vor Einleitung eines Abbiege- oder Kreuzungsvorgangs wird geprüft, ob Konfliktpotentiale mit anderen Verkehrsteilnehmern bestehen. So werden die betreffenden Zellen untersucht, ob sich auf ihnen entgegenkommende oder ebenfalls abbiegende Fahrzeuge befinden. Auch wird kein Überholvorgang eingeleitet, wenn Autos entgegenkommen oder die beiden Zellen hinter dem Hindernis nicht frei sind.

Abbiegeverhalten an Kreuzungen:

Das Abbiegeverhalten an Kreuzungen folgt der StVo. Handelt es sich um eine ampelgeregelte Kreuzung halten sich die Fahrzeuge an die Ampelphasen. Rechtsabbiegen und Geradeausfahren hat bei grün Vorrang, die Linksabbieger müssen sich allerdings nach dem Gegenverkehr richten. So wird der

Linksabbiegevorgang zwar eingeleitet und das Fahrzeug fährt bis in die Mitte der Kreuzung, wartet dann allerdings erst darauf, dass der entgegenkommende Verkehr seine Manöver durchgeführt hat, bevor der Linksabbiegevorgang fertig abgeschlossen wird.
Handelt es sich um eine nicht-ampelgeregelte Kreuzung, so zeigt das Vorrangschild die Bevorzugung des Verkehrs durch die StVo an. Das Fahrzeug, das vor so einem Schild steht, bleibt stehen und wartet solange, bis der Rechts- / Linksverkehr eine Zeitlücke für das gewünschte Fahrmanöver frei lässt. Die darauf folgenden Geradeaus- Rechtsabbiege- Linksabbiegevorgänge folgen dann wiederum den selben, zuvor beschriebenen Regeln die auch für die ampelgesteuerten Kreuzungen gelten.

Umfahren von Hindernissen:

Auch hier wird vor Einleiten des Manövers auf den entgegenkommenden Verkehr geachtet bzw die beiden Zellen hinter dem Hinderniss untersucht. Mehr hierzu im Kapitel “Baustellen/Hindernisse”

Der Kreisverkehr:

Beim Einfahren des Kreisverkehrs wird per Vorrangregelung auf Konfliktpotentiale mit anderen Verkehrsteilnehmern, die sich bereits im Kreisverkehr befinden (=Linkskommende), geachtet. Es kann nur nach rechts in den Kreisverkehr eingefahren werden.
Bei jeder Ausfahrt gibt es die Möglichkeit für das Fahrzeug, im Kreisverkehr zu bleiben (Geradeausfahren) oder aus dem Kreisverkehr auszufahren (= Rechts abbiegen). Beides wird wiederum natürlich nur dann durchgeführt, wenn der Weg frei ist.

Abbildungsverzeichnis