Aus:
VDI-Berichte Nr. 1745 (2003). Simulation und Simulatoren - Mobilität virtuell gestalten. Düsseldorf: VDI-Verlag
Original kann beim Verlag bestellt werden.

Fahrsimulator-Datenbasen mit dynamisch veränderbaren Straßennetzwerken

Dipl.-Inform. A. Kaussner, IZVW, Universität Würzburg
Dipl.-Inform. C. Mark, IZVW, Universität Würzburg
Dipl.-Inform. M. Grein, IZVW, Universität Würzburg
Prof. Dr. H.-P. Krüger, IZVW, Universität Würzburg
Prof. Dr. H. Noltemeier, Lehrstuhl für Informatik I, Universität Würzburg

Zusammenfassung

Für die Experimente, die am IZVW Fahrsimulator durchgeführt werden, wurde eine neuartige Architektur für Datenbasen entwickelt. Das Straßennetzwerk und die es umgebende Landschaft ist auf einer topologischen Ebene definiert. Hieraus wird während der Simulation innerhalb des Sichtbarkeitsbereichs des Fahrers eine geometrische Repräsentation berechnet. Diese Methode erleichtert den Entwurf reproduzierbarer Szenarios. Das Straßennetzwerk kann eine nahezu beliebige Ausdehnung haben und während der Simulation – unmerklich für den Fahrer – verändert werden. Die Veränderungen können interaktiv vom Versuchsleiter oder automatisch in Abhängigkeit von bestimmten an die Softwaremodule der Simulation gestellten Bedingungen ausgelöst werden.

Abstract

For the various demands of the experiments carried out in the IZVW driving simulator a new concept for databases has been developed. The road network and the surrounding landscape is modeled in a topological way. During simulation, the geometrical representation is computed in a small area of visibility around the driver. The topological model facilitates the design of reproducible scenarios. The extent of the road network is almost unlimited and it can be changed without the driver noticing it. This modifications can be controlled either interactively by the researcher or automatically by the system. In the latter case, modifications of the network can be controlled by all system variables of the simulator.

Einleitung

Grundlage eines Versuchs in einem Fahrsimulator bilden auf die jeweilige Fragestellung zugeschnittene Verkehrssituationen, in denen der Fahrer bestimmte Aufgaben ausführen muss. Diese Szenarien spielen sich an bestimmten Stellen des simulierten Straßennetzwerks ab. Das Straßennetzwerk ist in einem Software-Modul namens „Datenbasis“ abgelegt. Die Datenbasis verwaltet zwei miteinander korrelierte Informationen (vgl. z.B. [2], [5], [10]). Zum einen ist die grafische Repräsentation festgelegt, aus der die Bilder generiert werden, die der Fahrer z.B. über ein Projektionssystem angezeigt bekommt. Hierzu werden die Straße, die umgebende Landschaft sowie die Objekte, die sich auf der Straße (Pfosten, Verkehrsschilder, etc.) und in der Landschaft (Bäume, Häuser etc.) befinden, in Polygone unterteilt und mit Texturen gefüllt. Zum anderen ist in der Datenbasis die logische Struktur des Straßennetzwerks gespeichert. Dazu gehört beispielsweise der Verlauf von Fahrspuren, deren Breite und Richtung. Die logische Repräsentation des Straßennetzwerks hat eine zentrale Bedeutung innerhalb der Simulation: Verkehrssituationen werden in ihrem Kontext definiert, zahlreiche andere Software-Module (z.B. die physikalische Simulation des Fahrzeugs, die Simulation anderer Verkehrsteilnehmer und Assistenzsysteme) werden von ihr mit Informationen versorgt (vgl. [6], [11]).

Die übliche Architektur von solchen Datenbasen repräsentiert einen Ausschnitt aus der realen oder einer fiktiven Welt. Sie sind global geometrisch konsistent, d.h. ihr Straßennetzwerk kann im Ganzen als maßstabsgetreue Karte gezeichnet werden. Sie haben jedoch einige Nachteile:

- Aufgrund der Datenmengen ist die Länge des befahrbaren Straßennetzes sehr beschränkt. Selbst in umfangreichen Datenbasen können dadurch z.B. Autobahnfahrten bei hohen Geschwindigkeiten nur wenige Minuten dauern. Fragestellungen, die sich mit Problemen der Fahrer bei sehr langen Autofahrten beschäftigen, sind damit also nur schwer zu behandeln.
- Verkehrssituationen beziehen sich immer auf bestimmte Teile des Straßennetzwerks. Wenn für einen Versuch eine feste Abfolge von Situationen wichtig ist, muss der Fahrer genau instruiert werden, welche Route er durch die Datenbasis zu fahren hat. Biegt er an einer Kreuzung versehentlich falsch ab, muss der Versuch in der Regel abgebrochen werden.
- Die Abfolge der Verkehrssituationen hängt lediglich von der Route ab, die der Fahrer wählt. Situationen haben aber oft Vorbedingungen an den Zustand des Fahrers (z.B. den Grad seiner Aufmerksamkeit) oder an den Zustand bestimmter Software-Module (z.B. die aktuelle Geschwindigkeit oder den Abstand zu einem vorausfahrenden Fahrzeug). Solche Abhängigkeiten können

bei der Reihenfolge, in der Streckenabschnitte mit bestimmten Situationen auftreten, nicht berücksichtigt werden.

Am IZVW wurde ein neues Konzept zur Modellierung des Straßennetzwerks in Datenbasen entwickelt. Das Straßennetzwerk ist topologisch repräsentiert und nur noch lokal geometrisch konsistent. Eine maßstabsgetreue Karte kann nur in einem engen Bereich um den Fahrer, seinen Sichtbarkeitsbereich, angefertigt werden. Dieses Konzept beseitigt die genannten Einschränkungen: Das Straßennetzwerk kann eine nahezu beliebige Ausdehnung haben und es kann sich während der Simulation außerhalb des Sichtbarkeitsbereichs, also unmerklich für den Fahrer, verändern. Durch diese Veränderungen können Fehler, die der Fahrer bei der Wahl seiner Route macht, abgefangen werden. Zudem können sie von Eingaben des Versuchsleiters oder von den Messwerten, die in der Simulation erhoben werden, ausgelöst werden.

In diesem Artikel wird die Architektur der IZVW Datenbasis vorgestellt. Im nächsten Abschnitt „Exemplarische Fragestellung“ werden zwei Verkehrssituationen aus einer aktuellen Untersuchung des IZVW eingeführt, die im folgenden als Beispiele dienen. Der Abschnitt „Entwurf“ beschreibt, wie Anwender auf ihre Fragestellung zugeschnittene Datenbasen selbst entwerfen können. Im Abschnitt „Modellierung“ wird die interne Repräsentation des Straßennetzwerks beschrieben, aus der während der Simulation lokal geometrisch konsistente Teile im Sichtbarkeitsbereich des Fahrers erzeugt werden (Abschnitt „Simulation“).

Exemplarische Fragestellung

Beim Fahren mit Assistenzsystemen muss der Fahrer den Überblick über das System, die Fahraufgabe und die Verkehrsumwelt behalten. Ob dies immer möglich ist, wurde im Rahmen des Projekts EMPHASIS am IZVW untersucht (vgl. [4]). In einem Fahrparcours werden Situationen dargeboten, bei denen der Fahrer aufgrund richtigen Verhaltens Situationsbewusstsein beweisen kann. Er wird regelrecht in Fallen gelockt, in denen er entscheiden muss, ob die Nutzung eines bestimmten Assistenzsystems angemessen ist oder nicht.

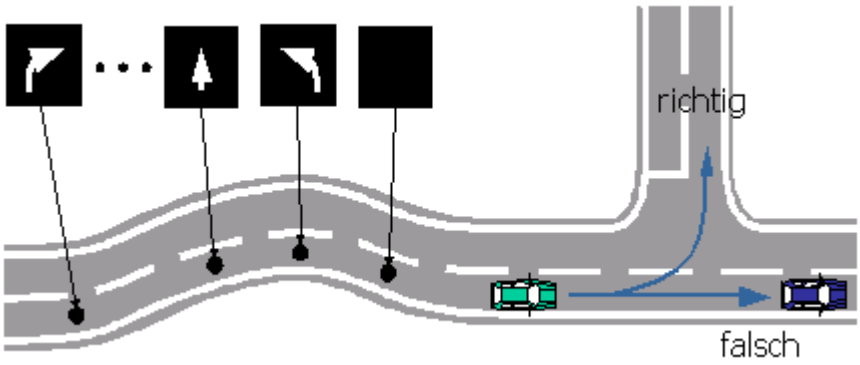

**Abbildung 1: Navigationsaufgabe /
Scenario „navigation“**

Der Parcours besteht insgesamt aus 28 Fahraufgaben, die in einer bestimmten Reihenfolge zu absolvieren sind. Die folgenden Beispielsituationen sind diesem Parcours entnommen. Sie sind zur Untersuchung von Auswirkungen des ACC (Adaptive Cruise Control) auf das Fahrverhalten konzipiert. Es handelt sich dabei um einen erweiterten Tempomat, der durch Gas- und Bremseingriffe automatisch den Abstand zu einem Führungsfahrzeug hält. Die Reichweite des Sensors, der die vorausfahrenden Fahrzeuge detektiert, beträgt ca. 150 m. In beiden Situationen wird geprüft, ob sich der Fahrer durch das Verhalten des Führungsfahrzeugs beeinflussen lässt. Unter Umständen wird er sich durch das ACC vom Führungsfahrzeug „mitziehen“ lassen, ohne zu prüfen, ob ein anderes Verhalten angemessen gewesen wäre.

Eine der Fahraufgaben des Parcours ist die sog. „Navigationsaufgabe“. Auf einem Display in der Mittelkonsole des Mockups geben Pfeile die Richtung an, die der Fahrer wählen soll. Die Instruktion lautet, dass der letzte Pfeil bevor das Display wieder schwarz wird die richtige Richtung anzeigt. Ein Führungsfahrzeug wählt dabei manchmal die für den Testfahrer falsche Richtung (vgl. Abbildung 1). Das Straßennetzwerk ist so gehalten, dass der Testfahrer, egal in welche Richtung er fährt, immer zur nächsten Situation weitergeleitet wird.

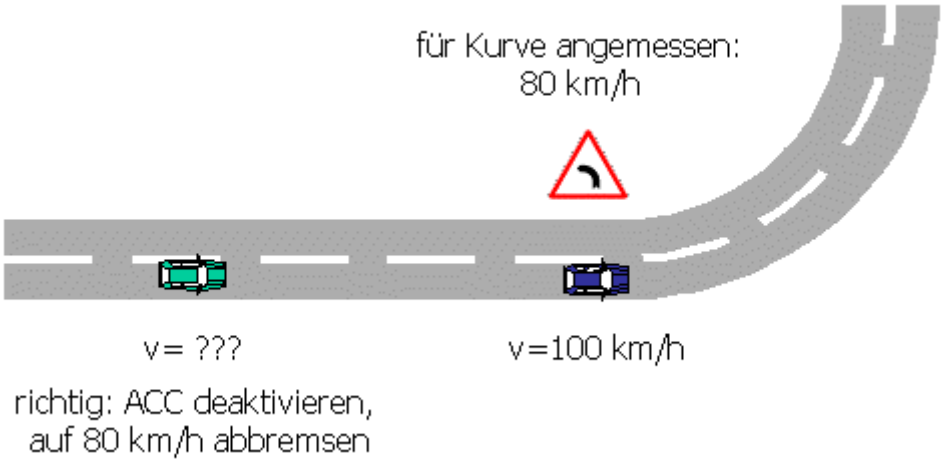

**Abbildung 2: Fahraufgabe "scharfe Kurve" /
Scenario „sharp bend“**

Bei der Fahraufgabe „Scharfe Kurve“ handelt es sich um eine sehr enge Kurve, von der der Fahrer aus einem Trainingsdurchgang weiß, dass sie nur mit ca. 80 km/h fehlerfrei durchfahren werden kann. Ein Führungsfahrzeug gibt die Kurve mit 100 km/h vor. Folgt der Testfahrer vertrauensvoll ohne das ACC abzuschalten, gerät er in Schwierigkeiten. Im Gegensatz zu den Simulatorfahrzeugen kann er im Gefahrenbereich mit dieser hohen Geschwindigkeit die Spur nicht mehr halten (vgl. Abbildung 2). Die scharfe Kurve wird erst dann dargeboten, wenn sicher gestellt ist, dass sich das Führungsfahrzeug im Sensorbereich des ACC befindet.

Entwurf

Anwender definieren die Datenbasis für einen Versuch mittels einer Skriptsprache. Die folgenden Abschnitte erläutern die schrittweise Vorgehensweise.

Karten und Module

Ein Datenbasis-Skript besteht aus einer oder mehreren Karten. Eine Karte repräsentiert die Datenbasis für einen Versuch. Beim Start der Simulation öffnet der Versuchsleiter zunächst die Datei mit dem Datenbasis-Skript und selektiert dann eine darin enthaltene Karte.
Jede Karte besteht aus Modulen. Ein Modul kapselt inhaltlich abgeschlossene Elemente eines Versuchs. Beispielsweise werden einzelne Situationen (z.B. die Navigationsaufgabe und die Situation „scharfe Kurve“) durch Module repräsentiert. Module werden außerhalb der Karten als sog. Modul-Schablonen definiert. Von diesen Schablonen werden, ähnlich wie bei Datentypen von Programmiersprachen, in den Karten Instanzen erzeugt. Die Modulinstanzen werden in den Karten über sog. Ports miteinander verknüpft. Dadurch wird der Ablauf eines Versuchs festgelegt.

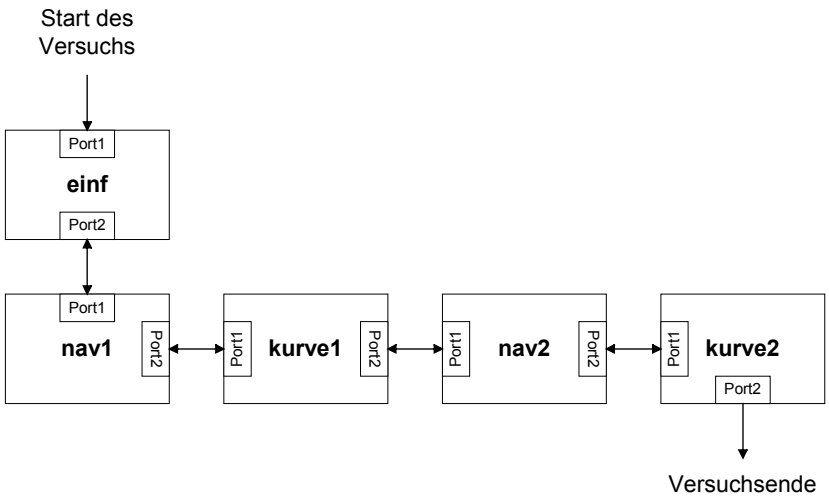

Abbildung 3: Vereinfachte Karte des Beispiel-Fahrparcours / Simplified map of the test track

Abbildung 3 zeigt eine vereinfachte Version des eingangs erwähnten Fahrparcours. Nach einer kurzen Einfahrt (Modulinstanz „einf“) gelangt der Fahrer je zweimal in die Navigationsaufgabe (Modulinstanzen „nav1“ und „nav2“) und in die scharfe Kurve (Modulinstanzen „kurve1“ und „kurve2“). Die Kapselung von Situationen in Module sowie die Instanzierung der Module aus Modul-Schablonen unterstützen die Planung und Organisation von Versuchen. Außerdem wird die Wiederverwendbarkeit von Versuchsteilen sichergestellt.

Strecken

Durch Modulschablonen werden inhaltlich abgeschlossene Teile eines Versuchs repräsentiert. Grundlage hierfür ist ein in jeder Modul-Schablone definiertes Streckennetz, das sich aus einzelnen Strecken zusammensetzt. Es gibt zwei Typen von Strecken: Als Straßen werden Strecken mit einheitlichem Querschnittsprofil ohne Abzweigungen bezeichnet. Verkehrsknoten modellieren Kreuzungen, Kreisverkehre und Übergänge zwischen Straßen mit unterschiedlichem Querschnittsprofil (z.B. Auffahrten auf Autobahnen). Diese Aufteilung ist auch in anderen Datenbasis-Architekturen üblich (vgl. z.B. [3], [5], [11]).

Die Definition von Straßen setzt sich aus folgenden Teilen zusammen:

- Lageplan: Beschreibt den Verlauf der Straße in der Ebene. Er besteht aus einer Abfolge von Geradenstücken und Kurven, die wiederum aus Übergangsbögen und Kreisbögen aufgebaut sind. Der Anwender wird dabei unterstützt, den Lageplan gemäß den RAS-L (vgl. [12]) anzulegen.

- Höhenplan: Ordnet jedem Punkt im Lageplan eine Höhe zu. Auch der Höhenplan wird RAS-L konform definiert.
- Querschnittsprofil: Beschreibt, aus wie vielen Spuren die Straße besteht, welche Breite und relative Höhe die Spuren haben, den Spurtyp (Gehsteig, Fahrbahn, begrünter Mittelstreifen) usw.
- Landschaft: Ordnet jeder Seite der Straße einen Landschaftstypen (z.B. bewirtschaftet, bewohnt, bewaldet) zu, aus dem während der Simulation automatisch ein Höhenprofil der Landschaft und zum Typ passende grafische Objekte erzeugt werden (s. nächster Abschnitt). Das Aussehen eines Landschaftstyps kann vom Anwender durch einige Parameter beeinflusst werden.
- Streckenevents: Unsichtbare Punkte auf Fahrspuren, bei deren Überfahren der Zustand von Software-Modulen der Simulation verändert wird. Das Umschalten der Navigationshinweise bei der Navigationsaufgabe erfolgt durch Streckenevents.
- Objekte: Einzelne spezielle grafische Objekte (wie z.B. Verkehrsschilder) werden auf oder an der Straße platziert.

Verkehrsknoten haben keinen Höhenplan und sind in einem Rechteck enthalten. Zu ihrer Definition partitioniert der Anwender die Spuren in logische Bereiche, die sog. Spurzellen. Innerhalb jeder Spurzelle legt er den Verlauf der Spuren durch Zuweisung eines Typs (Geradenstück, Kurve, Kreisbogen etc.) und durch die Spezifikation von Start- und Endpunkten fest (vgl. Abbildung 4). Über relative Spurhöhen kann der Anwender z.B. Gehsteige modellieren.

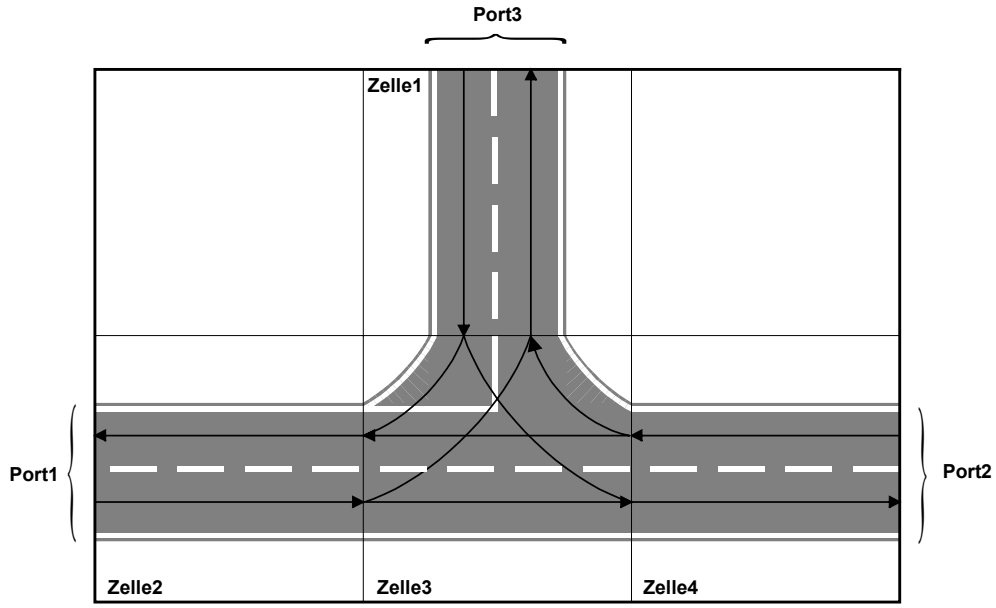

**Abbildung 4: Definition einer T-Kreuzung /
Definition of a junction**

Zusätzlich vergibt er für jede Spur Richtungsinformationen, die z.B. der simulierte Verkehr beim Durchfahren eines Verkehrsknotens nutzt. Auf jeder Seite des Rechtecks, in dem der Verkehrsknoten enthalten ist, werden Anschlüsse („Ports“) definiert. Durch die Zuweisung eines Querschnittsprofils an jeden dieser Anschlüsse wird festgelegt, welche Straßen mit dem Verkehrsknoten verbunden werden können. Wie bei den Straßen kann der Anwender Streckenevents und grafische Objekte definieren.

Streckennetz

In den Modulen erfolgt die Verknüpfung von Strecken zu einem Streckennetz unabhängig vom Lage- und Höhenplan. Betrachtet man also Lage- und Höhenplan von Strecken als deren geometrische Repräsentation, dann wird das Streckennetz auf einer topologischen Ebene beschrieben.

Jede Strecke hat Ports, über die sie mit anderen Strecken verbunden werden kann. Straßen haben zwei Ports (Port1 und Port2), die dem Anfang bzw. dem Ende der Straße zugeordnet sind. Verkehrsknotenpunkte sind rechteckige Gebiete, die auf jeder Seite (oben, unten, rechts, links) beliebig viele Ports haben können. Mit Port<i> wird der i-te Port eines Verkehrsknotenpunktes bezeichnet.

Der Anwender kann Streckennetze aufbauen, die in global geometrisch konsistenten Datenbasen nicht zu realisieren sind. So können Strecken zu Schleifen verknüpft werden, obwohl sie geometrisch, d.h. aufgrund ihres Lage- und Höhenplans, keine

Schleife bilden. Solche Schleifen werden als *topologische Schleifen* bezeichnet. Die einfachste Form besteht aus zwei Strecken, im Beispiel „s1“ und „s2“ genannt, deren Lageplan jeweils eine Gerade ist. Das Netzwerk ist in Abbildung 5 dargestellt. Auf die Auflösung der geometrischen Inkonsistenz während der Simulation wird im Abschnitt „Simulation“ eingegangen.

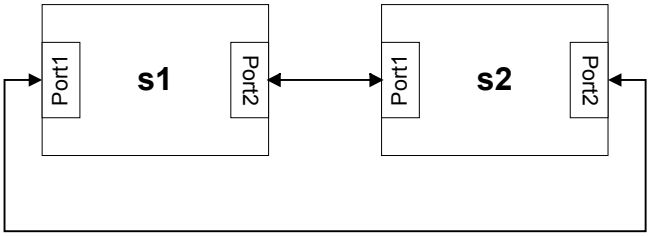

Abbildung 5: Topologische Schleife /
Topological loop

Der zweite, global geometrisch inkonsistente Typ von Streckennetzwerken heißt *topologischer Verkehrsknotenpunkt*. Eine solche Konstruktion wird beispielsweise in der Navigationsaufgabe verwendet. Der Fahrer muss aus einer Reihe von uneindeutigen Navigationshinweisen den letzten befolgen und an der T-Kreuzung links abbiegen. Wenn er fälschlicherweise gerade durch die Kreuzung fährt, soll er unmerklich an die Kreuzung zurückgeführt werden und die Aufgabe wiederholen. Das Streckennetz hierzu wird wie folgt aufgebaut (vgl. Abbildung 6): Der Fahrer kommt auf einer Straße „s0“ aus einer Situation, die der Navigationsaufgabe vorausgeht. Es folgt eine Straße „s1“, auf der der Fahrer zu der T-Kreuzung „k“ gelangt. Auf dieser Straße erhält er die Navigationshinweise. Wenn der Fahrer den letzten Hinweis richtig erkennt, biegt er in k links ab und gelangt auf die Straße „s2“, die ihn zur nächsten Situation führt. Wenn er gerade weiterfährt, kommt er auf die Straße „s3“ und von der aus wieder auf „s1“. Der topologische Verkehrsknotenpunkt liegt an Port1 der Straße „s1“.

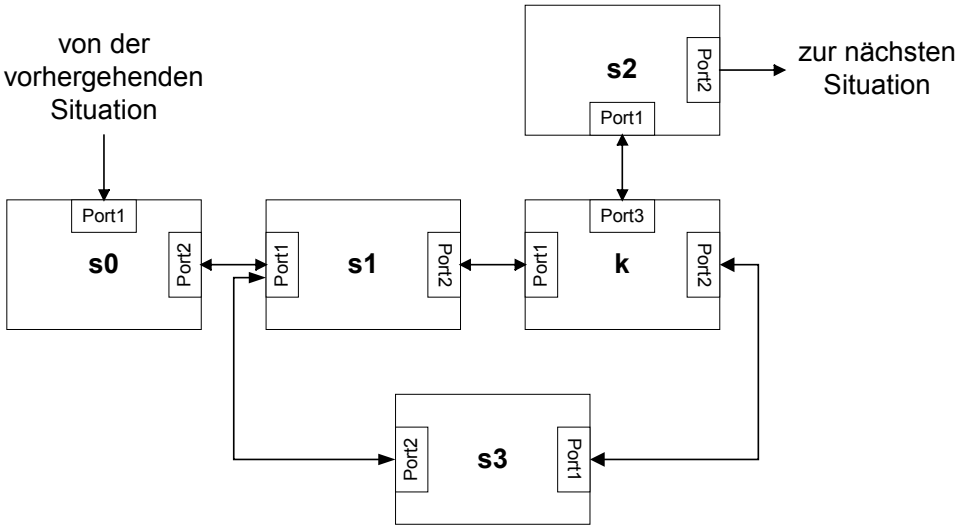

**Abbildung 6: Streckennetz der Navigationsaufgabe /
Road network for the scenario „navigation“**

Sog. bedingte Verknüpfungen werden vom Anwender beispielsweise in der Fahraufgabe „scharfe Kurve“ eingesetzt. Die enge Kurve soll erst auftauchen, wenn der Fahrer höchstens 50 Meter vom Führungsfahrzeug entfernt ist. Der Abstand des Ego-Fahrzeugs von den umliegenden Fahrzeugen wird in der Verkehrssimulation (Softwaremodul „traffic“) berechnet. Dort ist der Abstand zum nächsten vorausfahrenden Fahrzeug, das sich auf der selben Spur wie das Ego-Fahrzeug befindet, über die Variable traffic.DistSameAheadNext zugänglich. Die enge Kurve soll also auftreten, wenn die Bedingung

traffic.DistSameAheadNext <= 50

an das Softwaremodul „traffic“ erfüllt ist. Das Streckennetz für diese Fahraufgabe setzt sich aus drei Straßen zusammen (vgl. Abbildung 7): „s1“ und „s2“ bilden eine topologische Schleife, solange der Fahrer weiter als 50 m vom Führungsfahrzeug entfernt ist. Das Führungsfahrzeug wird beim erstmaligen Befahren von „s1“ in genügend großer Entfernung vom Testfahrer aufgesetzt, so dass er sein plötzliches Erscheinen nicht bemerkt. Ist der gewünschte Abstand erreicht, gelangt der Fahrer auf die Straße „kurve“, die die scharfe Kurve enthält.

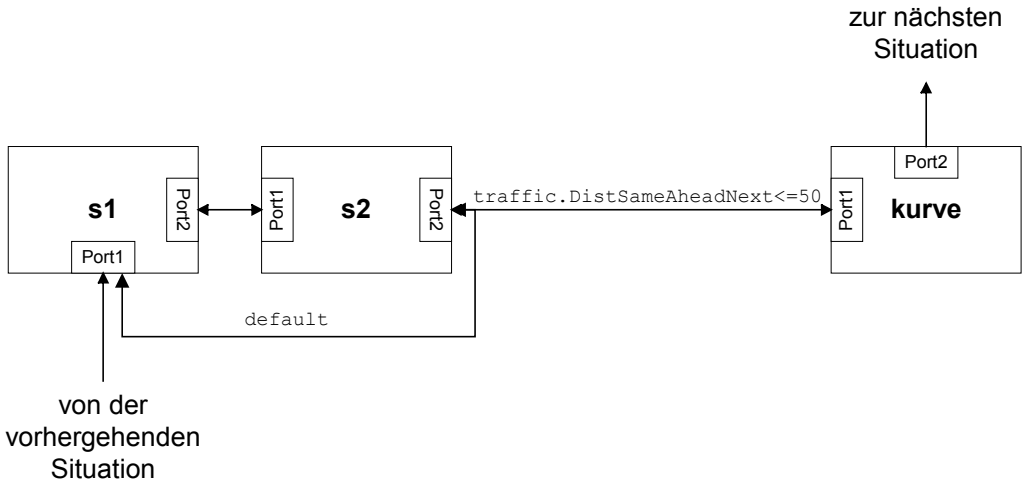

**Abbildung 7: Streckennetz der Fahraufgabe "scharfe Kurve" /
Road network for the scenario „sharp bend“**

Über bedingte Verknüpfungen kann der Versuchsleiter das Streckennetz auch während der Simulation interaktiv verändern. Die Bedingung wird dabei an den Zustand eines Kontrollelements der grafischen Benutzeroberfläche der Simulationssoftware gestellt.

Modellierung

Topologische Struktur

Die Datenbasis ist nach Methoden der objektorientierten Modellierung aufgebaut. Beim Start der Simulation wird eine in einer Skript-Datei definierte Karte in eine Datenstruktur gewandelt, die aus fünf Schichten aufgebaut ist:

- Karte: Die oberste Schicht des Modells ist die Karte. Sie besteht aus einem einzelnen Objekt vom Typ *Map*.
- Module: Module kapseln inhaltlich abgeschlossene Situationen. Wie im vorhergehenden Abschnitt dargestellt, wird der Ablauf von Situationen eines Versuchs durch die Verknüpfung von Modulen festgelegt. Deswegen bilden die Module einer Datenbasis einen Graphen MODULES.
- Strecken: Es gibt Straßen und Verkehrsknoten. Ihre Verknüpfung bildet das Straßennetzwerk einer Datenbasis. Im Modell ist eine Straße ein Objekt vom Typ *Course*, ein Verkehrsknoten ein Objekt vom Typ *Area*. Das Straßennetzwerk, das aus Objekten dieser beiden Typen besteht, ist als Graph RN modelliert.
- Spurzellen: Im vorhergehenden Abschnitt wurden Spurzellen eingeführt, um einzelne Spuren von Verkehrsknoten zu gruppieren. Im Kontext der Modellierung wird eine Spurzelle durch Objekte repräsentiert, die vom Typ *LaneCell*

abgeleitet sind. Es erweist sich als sinnvoll, auch Spuren des Typs *Course* in Spurzellen zu gruppieren und alle Spurzellen der Datenbasis zu Verknüpfen. Man erhält dadurch einen Graphen LANECELLS, der eine Verfeinerung des Graphen RN bildet.

- Spuren: Einzelne Fahrspuren von Strecken werden durch Objekte des Typs *Lane* oder davon abgeleiteter Typen repräsentiert. Auch sie werden in einem Graphen LANES verknüpft, der damit eine Verfeinerung des Graphen LANECELLS darstellt.

Die Struktur kann am Beispiel des eingangs dargestellten Fahrparcours Parcours verdeutlicht werden. Abbildung 8 zeigt für diesen Fall die einzelnen Schichten des Modells.

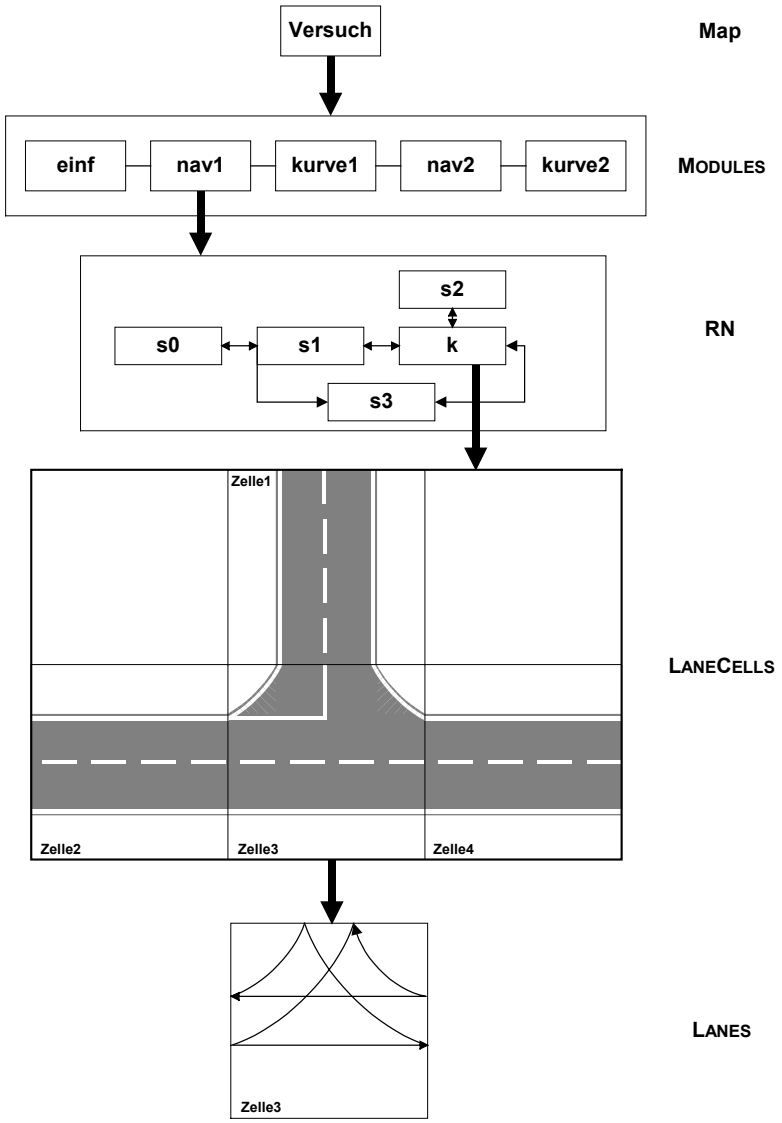

Abbildung 8: Topologische Struktur des Beispiel-Parcours / Topological structure of the test track

Die Knoten der Graphen MODULES, RN, LANECELLS und LANES sind von einem gemeinsamen Grundtyp $Node$ abgeleitet. Ein Knoten v dieses Typs besteht aus folgenden Elementen:

1. Einem Eintrag $v.Parent$ vom Typ $Node$. Dies ist der Knoten, zu dem v auf der nächst höheren Ebene gehört.
2. Einer Menge $v.ChildNodes = \{c_1,...,c_{nc}\}$ von Kindern des Knotens. In dieser Menge werden die Knoten gespeichert, die v auf der nächst tieferen Modellebene repräsentieren. Zur Abkürzung wird für die Elemente die Schreibweise $v.c_i$ verwendet.
3. Einer Menge $v.Ports = \{p_1,...,p_{np}\}$ von Ports des Knotens. Dabei ist jedes p_i vom Typ $Port$ (s.u.). Zur Abkürzung wird für einen Port eines Knotens die Schreibweise $v.p_i$ verwendet.

Die Knoten eines Graphen werden über sog. Ports verbunden. Wie in obiger Definition beschrieben, enthält jeder Knoten eine Menge von Ports. Die Ports der verschiedenen Ebenen sind Objekte des Grundtyps $Port$, der aus folgenden Elementen besteht:

1. Einer Menge $p.Edges = \{v_1.p_{i_1},...,v_{ne}.p_{ine}\}$ von Kanten, die vom Port p ausgehen. Dabei ist v_j vom Typ $Node$ und $p_{i_j} \in v_j.Ports$.
2. Einer Generalisierung $p.Gen = v.p$ des Ports, wobei v vom Typ $Node$ und p vom Typ $Port$ ist.
3. Einer Menge $p.Ref = \{v_1.p_{i_1},...,v_{nr}.p_{i_{nr}}\}$ von Verfeinerungen des Ports.

Die Verbindung zwischen den Ports verschiedener Ebenen wird durch $p.Gen$ bzw. $p.Ref$ eines Ports p hergestellt. Beispielsweise fallen mit jedem Port $lc.p_j$ einer Spurzelle lc die Ports $l_1.p_{i_1}, l_2.p_{i_2},...$ von Spuren zusammen, die an $lc.p_i$ beginnen oder enden. Folglich ist $lc.p_i.Ref = \{l_1.p_{i_1}, l_2.p_{i_2},...\}$ und umgekehrt $l_1.p_{i_1}.Gen = l_2.p_{i_2}.Gen = ... = lc.p_j$.

Für die Verknüpfung zweier verschiedener Knoten v_1, v_2 über die Ports $v_1.p_i$ und $v_2.p_j$ muss gelten:

1. $v_1.p_i \in v_2.p_j.Edges$ und $v_2.p_j \in v_1.p_i.Edges$
2. Wenn $v_1.p_i.Gen = a.p_k$ und $v_2.p_j.Gen = b.p_l$, dann sind auch a und b über die Ports $a.p_k$ und $b.p_l$ verknüpft.

3. Wenn
   $c.p_m \in v_1.p_i.Ref$, $c.Gen = v_1.p_i$ und $d.p_n \in v_2.p_j.Ref$, $d.Gen = v_2.p_j$ ist, dann sind auch c und d über die Ports $c.p_m$ und $c.p_n$ verknüpft.

Aufbauend auf die Typen $Node$ und $Port$ können jetzt die Knotentypen für die Graphen der einzelnen Ebenen des Modells definiert werden. Dabei kann jedem Knotentyp eine entsprechende Struktur aus der Datenbasisdefinition im Skript zugeordnet werden. Die Vererbungshierarchie der Knotentypen ist in Abbildung 9 als UML-Klassendiagramm dargestellt.

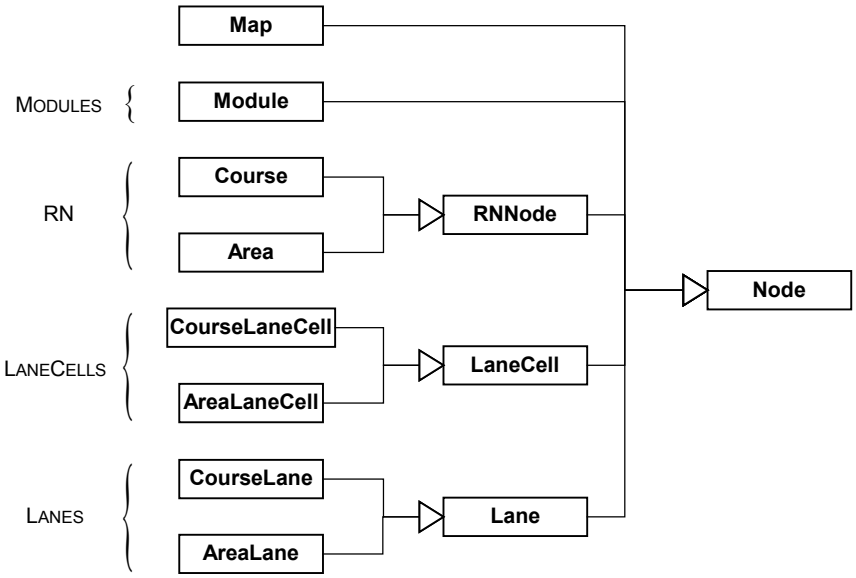

Abbildung 9: Vererbungshierarchie der Knoten des Modells /
Inheritance diagram for the nodes of the model

Jeder abgeleitete Knoten spezifiziert den Grundtyp genauer. So wird z.B. festgelegt, dass Knoten c des Typs $Course$ genau zwei Ports haben, die mit dem Beginn bzw. Ende der Straße zusammenfallen. Außerdem müssen die Elemente in $c.ChildNodes$ vom Typ $CourseLaneCell$ und $c.Parent$ vom Typ $Module$ sein (vgl. UML-Diagramm in Abbildung 10).

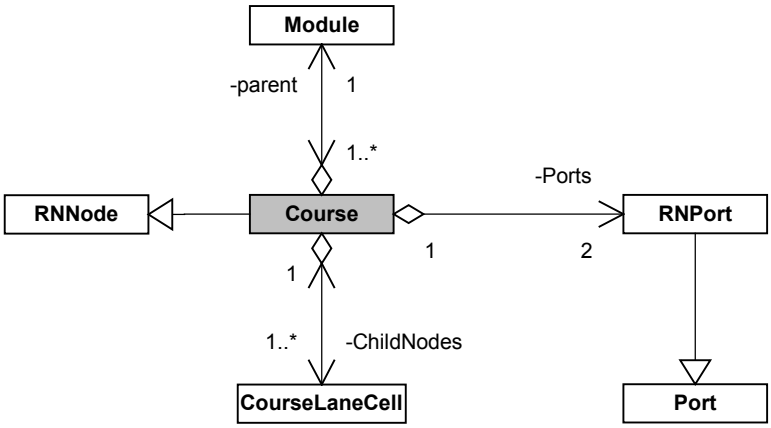

Abbildung 10: Spezifikation des Typs *Course* **/**
Specification of the type *Course*

Für die Ports der verschiedenen Knotentypen gibt es eine ähnliche Vererbungshierarchie.

Fahrspuren

Die Fahrspuren von Straßen (Typ *CourseLane*) und von Verkehrsknoten (Typ *AreaLane*) besitzen einige Gemeinsamkeiten, weswegen sie vom gemeinsamen Grundtyp *Lane* abgeleitet werden. Er enthält folgende Elemente:

1. Eine Instanz des Typs *Curve*, die den Verlauf der exakten Mitte der Fahrspur beschreibt (s.u.).
2. Eine Instanz des Typs *CrossSection*, die Querschnittsinformationen (Spurbreite, relative Höhe zu den anderen Spuren, Fahrtrichtung etc.) enthält.
3. Eine Menge *Events* von Streckenevents, die auf dieser Spur liegen.

Die Zusammenhänge sind in Abbildung 11 dargestellt.

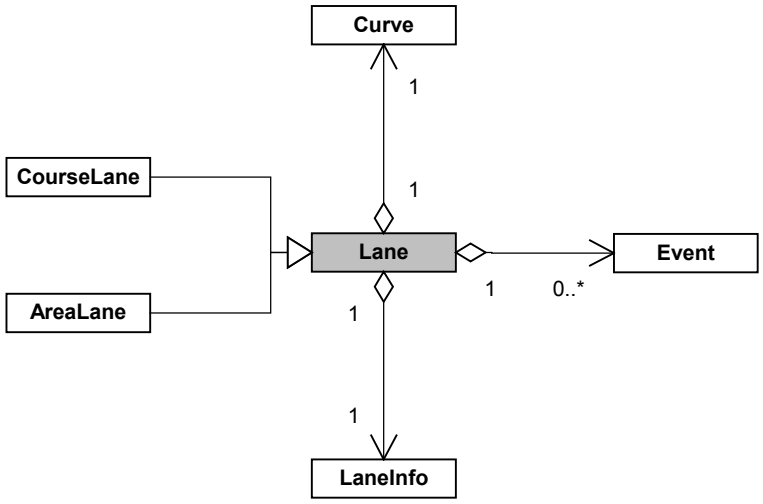

Abbildung 11: Grundtyp $Lane$ **/**
Base type $Lane$

Aufbauend auf die Arbeiten in [1] wird der Verlauf der exakten Mitte einer Fahrspur durch ebene, über die Bogenlänge parametrisierte Parameterkurven definiert. Die Parameterkurven sind durch Objekte des Typs $Curve$ repräsentiert. Dieser enthält folgende Informationen:

1. Eine Zahl $S \in IR_0^+$, die gesamte Länge der Parameterkurve.
2. Eine Abbildung $p:[0,S] \to IR^2$, die jedem Parameterwert die Koordinaten der Kurve an dieser Stelle zuordnet.
3. Eine Abbildung $\alpha:[0,S] \to IR$, die jedem Parameterwert den Tangentenwinkel der Kurve an dieser Stelle zuordnet.
4. Eine Abbildung $\kappa:[0,S] \to IR$, die jedem Parameterwert die Krümmung der Kurve an dieser Stelle zuordnet.

Der Vorteil der Repräsentation des Spurverlaufs durch Parameterkurven liegt zum einen in der effizienten Speicherung. Zum Anderen stehen Koordinaten, Tangentenwinkel und Krümmung in nahezu beliebiger Genauigkeit zur Verfügung.

Straßen

Straßen werden durch Objekte des Typs $Course$ modelliert. Der Anwender definiert Straßen durch Angabe von Lage- und Höhenplan, Landschaftstyp, Querschnittsprofil, grafischen Objekten und Streckenevents.

Der **Lageplan** von Straßen setzt sich aus einer Folge der Grundelemente Gerade, Übergangsbogen und Kreisbogen zusammen. Hieraus und aus der Tatsache, dass sich das Querschnittsprofil von Straßen nicht ändert, ergibt sich automatisch eine Einteilung in Spurzellen: Eine Spurzelle einer Straße (Typ $CourseLaneCell$) kapselt alle parallel verlaufenden Spuren eines Grundtyps. Für eine effiziente Speicherung der Spurverläufe kann wiederum die Konstanz des Querschnittsprofils ausgenutzt werden. Es reicht, den Verlauf der exakten Mitte eines Straßenabschnitts als Objekt des Typs $Curve$ in den Spurzellen als Element $refcurve$ zu speichern. Der Verlauf der Fahrspuren ergibt sich aus dem Querschnittsprofil durch eine Parallelverschiebung dieser sog. Referenzkurve.

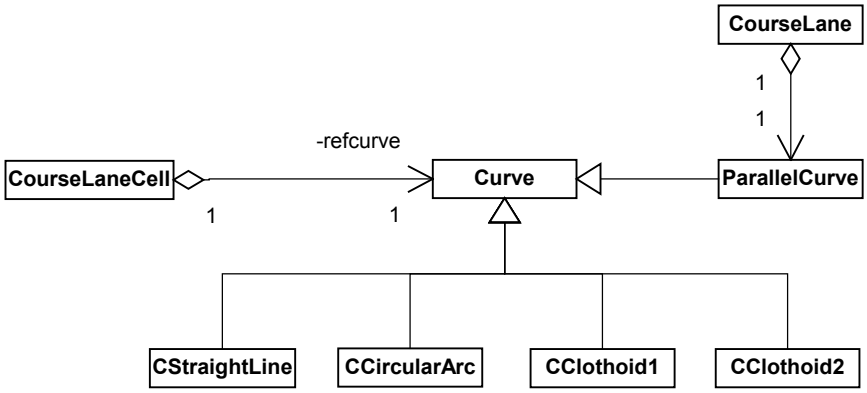

Abbildung 12: Fahrspuren von Straßen / Lanes of roads

Abbildung 12 zeigt die Zusammenhänge als UML-Diagramm. Als Typen von Referenzkurven stehen $CStraightLine$ (Geradenstücke), $CCircularArc$ (Kreisbögen), $CClothoid1$ (Übergangsbögen mit zunehmender Krümmung) und $CClothoid2$ (Übergangsbögen mit abnehmender Krümmung) zur Verfügung.

Beim **Höhenplan** schreiben die RAS-L Ausrundungen in Parabelform zwischen Segmenten mit konstanter Steigung bzw. konstantem Gefälle vor. Im Modell besteht der Höhenplan entsprechend aus einer Folge von Objekten, die den Verlauf dieser Segmente repräsentieren. Er wird - unabhängig von der Segmentierung des Lageplans – in den Objekten des Typs $Course$ gespeichert.

Zur Modellierung des **Höhenprofils der Landschaft** wird für jede Seite eines Knotens c vom Typ $Course$ eine Menge $F_1(d),...,F_n(d)$ von sog. Höhenfunktionen gespeichert. Eine Höhenfunktion $F_i(d)$ beginnt an einer bestimmten Position $s_i \in [0,S]$ auf der Straße, wobei mit S die gesamte Länge des Lageplans bezeichnet

wird. Sie spezifiziert die Höhe, auf der ein Punkt liegt, der einen Abstand d von der Straße hat (vgl. Abbildung 13).

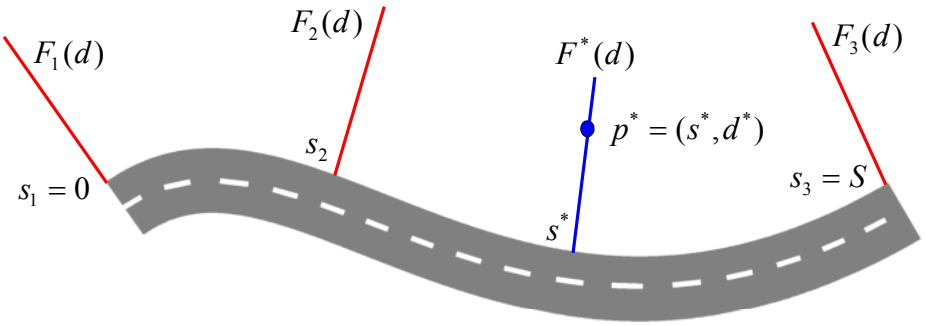

Abbildung 13: Höhenfunktionen auf der linken Seite einer Straße / Height functions on the left side of a road

Die Höhenfunktionen werden aus einfachen Funktionen (wie z.B. Sinus-Schwingungen) zusammengesetzt (vgl. [9]). Dadurch können sie einheitlich und effizient durch die Speicherung einiger „Formparameter“ (wie z.B. Amplituden und Frequenzen der Sinus-Schwingungen) repräsentiert werden. Die Werte der Formparameter werden von der automatischen Landschaftsgenerierung festgelegt, wenn die Straße in den Sichtbarkeitsbereich des Fahrers kommt (s. nächster Abschnitt). Angenommen, $p* = (s*, d*)$ ist ein Punkt, der zwischen zwei Höhenkurven F_i und F_{i+1} liegt, d.h. $s_i < s* < s_{i+1}$ (Abbildung 13). Zur Berechnung der Höhe, auf der $p*$ liegt, ist die zugehörige Höhenkurve $F*$ zu ermitteln. Hierzu müssen lediglich die Formparameter der benachbarten Höhenkurven F_i und F_{i+1} interpoliert werden. Auch hier hat die Repräsentation mittels analytischer Funktionen den Vorteil, dass Landschaftshöhen und Steigungen an jeder Stelle der Landschaft mit sehr hoher Genauigkeit ermittelt werden können.

Die **grafischen Objekte**, die von der automatischen Landschaftsgenerierung oder vom Anwender selbst auf und neben der Straße platziert werden, sind als Menge in den Objekten des Typs $CourseLaneCell$ gespeichert. Ihre Position ist in kurvilinearen Koordinaten angegeben und hat die Form (s, d). Ein Objekt mit diesen Koordinaten befindet sich im Abstand s vom Beginn der Referenzspur (entlang der Spur gemessen) und im Abstand d senkrecht zur Spur auf der Landschaftsoberfläche.

Verkehrsknotenpunkte

Die Spuren im Lageplan von Verkehrsknoten sind Objekte des Typs $AreaLane$, die wie bei den Straßen vom Typ $Lane$ abgeleitet sind.

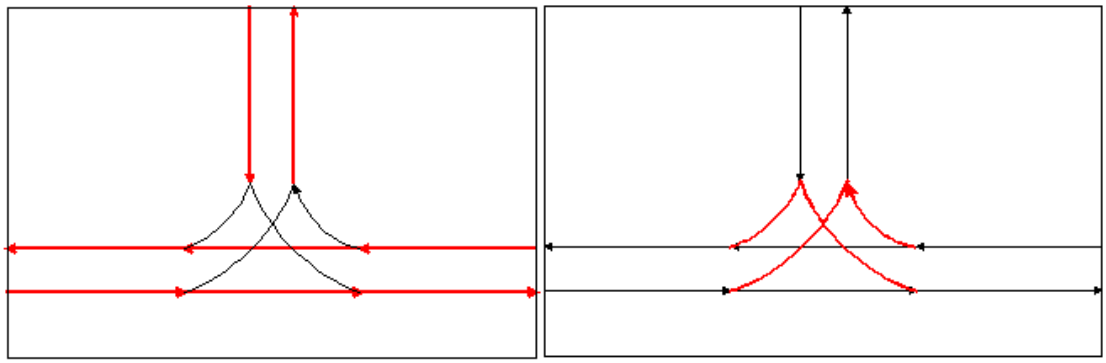

**Abbildung 14: Spurverläufe in Verkehrsknoten.
Links: Geradenstücke, rechts: einfache Kurven /
Run of lanes in junctions. Left side: straight segments,
right side: simple bends**

Für den Verlauf der Spurmitte stehen folgende Typen von Parameterkurven zur Verfügung:

- Geradenstücke (Typ *AStraightLine*) (Abbildung 14, links)
- einfache Kurven (Typ *ABezierCurve*). Objekte dieses Typs schließen immer an Geradenstücke oder Segmente von Superkurven an (Abbildung 14, rechts)
- Segmente von Superkurven (Typ *ACurveSegment*). Die Spuren einiger Kreuzungstypen können durch Geradenstücke und einfache Kurven nicht modelliert werden. Ein Beispiel ist der Kreisverkehr in Abbildung 15. Der Kreis wird deswegen als sog. Superkurve beschrieben und in den übergeordneten Objekten des Typs *Area* gespeichert. Die Segmente repräsentieren einen Ausschnitt dieser Superkurve.

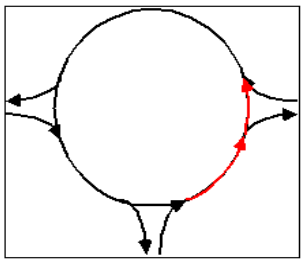

**Abbildung 15: Segmente von Superkurven /
Segments of supercurves**

Abbildung 16 zeigt die Modellierung der Spuren von Verkehrsknoten als UML-Diagramm.

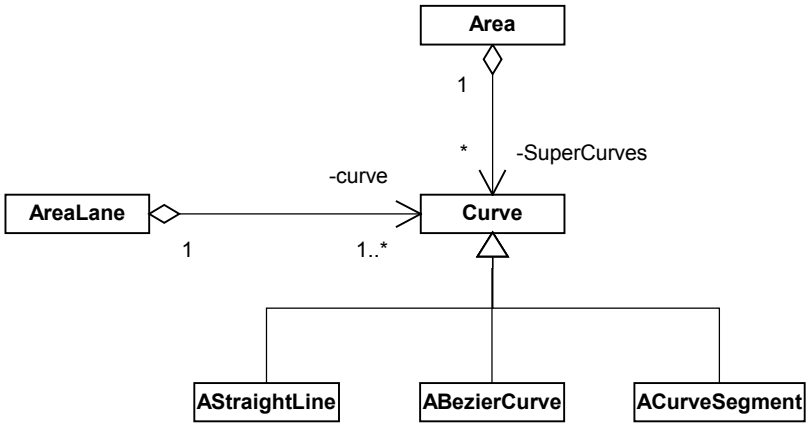

Abbildung 16: Fahrspuren von Verkehrsknoten / Lanes of junctions

Simulation

Überblick

Wie im vorhergehenden Abschnitt dargestellt, bildet das Modell des Straßennetzwerks auf jeder der Ebenen MODULES, RN, LANECELLS und LANES einen Graphen. Es handelt sich um eine topologische Repräsentation, da der Anwender durch topologische Verkehrsknotenpunkte, topologische Schleifen und bedingte Verknüpfungen Mehrdeutigkeiten bei der Abfolge von Strecken einführt. Aufgabe der sog. geometrischen Instanzierung ist es, in jedem Schritt der Simulation auf jeder Ebene einen Teilgraphen zu ermitteln, der den Sichtbarkeitsbereich des Fahrers vollständig enthält und zusätzlich geometrisch konsistent ist, d.h.:

1. Jeder Teilgraph ist zusammenhängend.
2. Die Verknüpfungsinformationen aller Ports eines Knotens eines jeden Teilgraphen sind eindeutig.
3. Die durch die Knoten der Teilgraphen repräsentierten Streckenabschnitte schließen bezüglich ihres Lage- und Höhenplans glatt aneinander, sie bilden also ein durchgängiges Streckennetz.

Sobald ein Port p eines Knotens des Typs $Course$ oder $Area$ in den Sichtbarkeitsbereich des Fahrers kommt, wird unter den Alternativen in $p.Edges$ eine eindeutige Kante ausgewählt. Über die Generalisierung $p.Gen$ des Ports kann die Verknüpfung auf der darüberliegenden Ebene der Module eindeutig festgelegt werden. Mit den Verfeinerungen $p.Ref$ können die Verbindungen auf den darunterliegenden Ebenen der Spurzellen und Spuren bestimmt werden. Die folgenden Beispiele veranschaulichen die grundsätzliche Vorgehensweise.

Ein möglicher Ablauf der Navigationsaufgabe ist in Abbildung 17 gezeigt. Die Kreise sollen den Sichtbarkeitsbereich des Fahrers darstellen. Die dunkelgrauen Strecken sind Knoten des Teilgraphen auf der Ebene RN, die hellgrauen sind im Lauf der Simulation aus dem Sichtbarkeitsbereich des Fahrers herausgefallen und liegen daher nicht im Teilgraphen. Das Visualisierungsmodul der Fahrsimulation stellt in jedem Schritt nur die Knoten dar, die in den Teilgraphen liegen. Der Fahrer nähert sich der T-Kreuzung auf der Straße s_1 (linkes Bild). Erst wenn der Port $s_1.p_2$ in den Sichtbarkeitsbereich gelangt, wird die T-Kreuzung k so transformiert, dass sie glatt an s_1 anschließt. Der Fahrer fährt in k fälschlicherweise geradeaus (mittleres Bild). Er gelangt auf die Strecke s_3, an die über eine topologische Schleife $s_1.p_1$ angeschlossen ist. Unter der Voraussetzung, dass der Lageplan von s_3 lang genug ist, ist die Straße s_1 bereits wieder aus dem Sichtbarkeitsbereich des Fahrers herausgefallen. Sobald $s_3.p_2$ in den Sichtbarkeitsbereich gerät, wird s_1 so transformiert, dass sie glatt an $s_3.p_2$ anschließt. Auch die T-Kreuzung k, zu der der Fahrer jetzt ein zweites mal über s_1 kommt, wird entsprechend transformiert, sobald $s_1.p_2$ sichtbar wird (rechtes Bild).

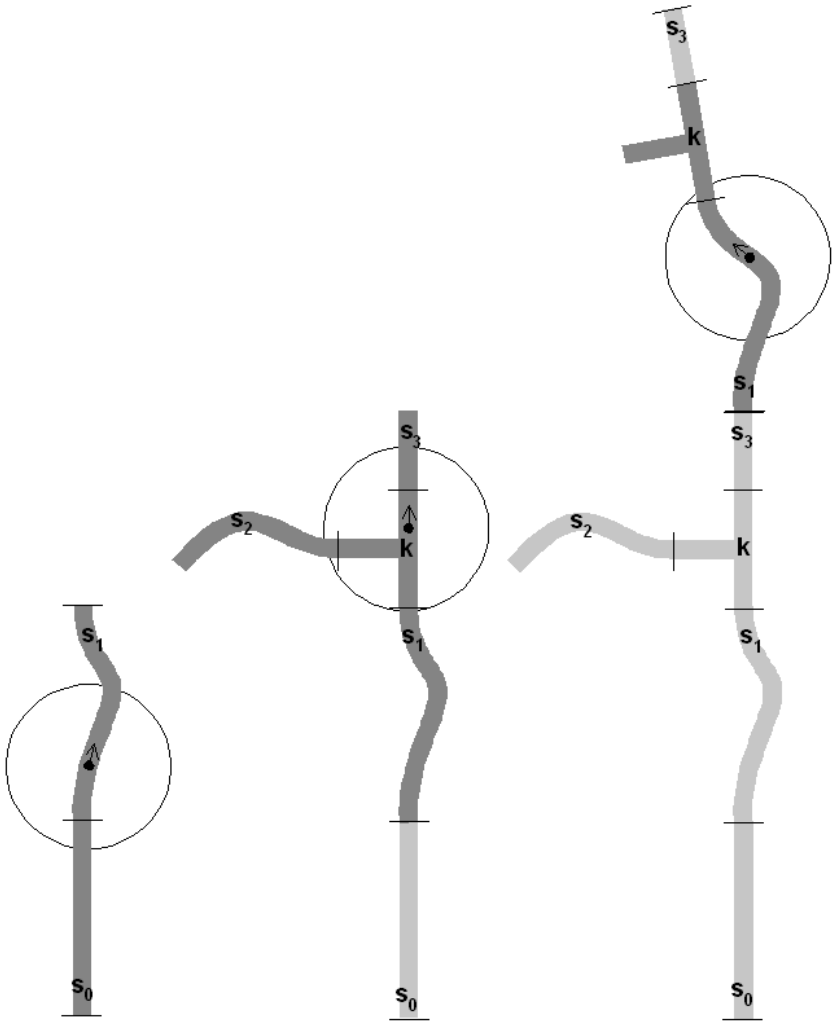

Abbildung 17: Geometrische Instanzierung der Navigationsaufgabe / Geometric instantiation of the scenario „navigation“

Abbildung 18 zeigt einen möglichen Ablauf für die Situation „scharfe Kurve“. Wieder deuten die Kreise den Sichtbarkeitsbereich des Fahrers an. Der geometrisch konsistente Teilgraph auf der Ebene RN wird von den dunkelgrauen Strecken gebildet. Im oberen Bild nähert sich der Fahrer auf den Straßen s_1 und s_2 einem vorausfahrenden Fahrzeug (weiß). Sobald der Port $s_2.p_2$ in den Sichtbarkeitsbereich kommt (mittleres Bild), wird geprüft, ob das Führungsfahrzeug bereits genügend nahe ist. Die Bedingung, die der Anwender hierfür definiert hat, lautet

traffic.DistSameAheadNext <= 50

Angenommen, diese Bedingung ist im mittleren Bild noch nicht erfüllt. Dann wird als eindeutige Verknüpfung die zwischen $s_2.p_2$ und $s_1.p_1$ gewählt. Die Strecke s_1 wird

an das Ende von s_2 transformiert und der Fahrer nähert sich weiter dem vorausfahrenden Fahrzeug an. Im unteren Bild kommt $s_2.p_2$ zum zweiten mal in den Sichtbarkeitsbereich. Jetzt ist die Bedingung erfüllt, d.h. der Fahrer ist weniger als 50 m vom Vordermann entfernt. Deswegen wird die Verknüpfung zwischen $s_2.p_2$ und $kurve.p_1$ selektiert. Der Fahrer folgt dem Vordermann, evtl. mit überhöhter Geschwindigkeit, in die scharfe Kurve.

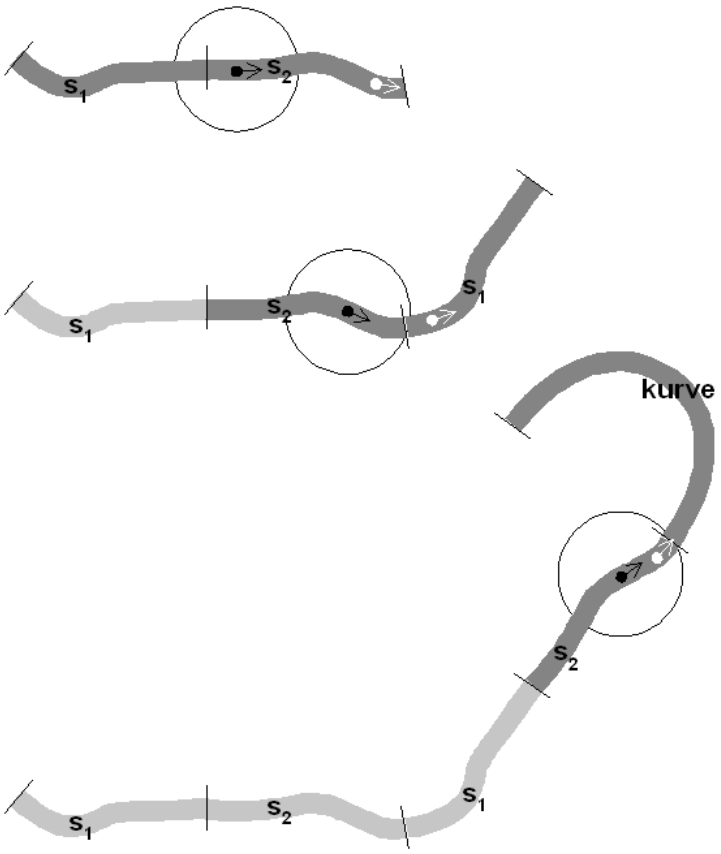

Abbildung 18: Geometrische Instanzierung der Situation "scharfe Kurve" / Geometric instantiation of the scenario „sharp bend“

Geometrische Instanzierung

Da die Knoten einer jeden Ebene vom Grundtyp *Node* abgeleitet sind, kann der Algorithmus zur Bestimmung des geometrisch konsistenten Teilgraphen als Methode von *Node* implementiert werden. Diese Methode wird *v.Update* genannt. Ausgehend vom Knoten, auf dem sich der Fahrer gerade befindet, ruft *v.Update* in jedem Simulationsschritt die Methoden der angeschlossenen Knoten auf. Dadurch wird der Graph einer Ebene rekursiv (in depth first order) traversiert. Die Rekursion endet, wenn ein Knoten erreicht wird, der sich nicht im Sichtbarkeitsbereich befindet. Dies

wird für jede Modellebene durchgeführt, beginnend mit der Ebene RN, da der Anwender das Streckennetz auf dieser Ebene definiert hat.

Um feststellen zu können, ob ein Knoten v im Vergleich zum letzten Simulationsschritt neu in den Sichtbarkeitsbereich kommt oder ob er bereits sichtbar war, bekommt er einen Zeitstempel $v.T_{Update} \in IN_0$. Beim Start der Simulation gilt stets $v.T_{Update} = 0$. Jedes mal, wenn ein Knoten bei der Traversierung seines Graphen berührt wird (wenn also seine Methode $v.Update$ aufgerufen wird) und er sichtbar ist, wird $v.T_{Update}$ auf die Nummer T des aktuellen Simulationsschritts gesetzt. Der folgende Pseudocode zeigt eine vereinfachte Version von $v.Update$.

```
Node::Update(T, ...)
{
1     // Wurde der Knoten im letzten Schritt bereits behandelt?
2     if (T_update == T) return;
3
4     if (Knoten nicht sichtbar)
5     {
6           // beim letzten mal sichtbar?
7           if (T_update == T-1)
8                 Knoten fällt aus dem Teilgraph heraus
9           else
10                Knoten könnte bald sichtbar werden
11    }
12    else
13    {
14          // noch nicht behandelt?
15          if (T_update < T-1)
16                DoGeoInst(…);
17
18          T_update = T;
19
20          // Update der angeschlossenen Knoten
21          forall (p ∈ Ports)
22          {
23                if (p ist sichtbar)
24                {
25                      Bestimme eindeutige Verknüfung
26                      Aufruf von Update der an p angeschl. Knoten
27                }
28          }
29    }
}
```

Die Informationen in den Zeilen 8 und 10 des Verfahrens werden z.B. von der Visualisierungssoftware genutzt: Die Fahrbahn, Objekte und die Landschaft werden nur

dargestellt, wenn die zugehörige Strecke auch sichtbar ist. Von Strecken, die bald sichtbar werden könnten, wird das Rendern vorbereitet.

Gerät ein Knoten in einem Simulationsschritt in den Sichtbarkeitsbereich, so sind, je nach Ebene des Graphen, in dem sich der Knoten befindet, unterschiedliche Aufgaben zu erledigen. Handelt es sich beispielsweise um eine Knoten *lc* vom Typ *CourseLaneCell*, dann muss die Referenzkurve von *lc* so transformiert werden, dass sie glatt an die Spurzelle anschließt, von der aus *lc.Update* aufgerufen wurde. Ein Knoten des Typs *Area*, der neu in den Sichtbarkeitsbereich kommt, wird als Ganzes transformiert und so glatt mit der aufrufenden Straße verbunden.

Um solche Aufgaben für die speziellen Knotentypen ausführen zu können, wird der Grundklasse *Node* eine rein virtuelle Methode *DoGeoInst* hinzugefügt. Diese Methode wird von den abgeleiteten Klassen überschrieben und jedes Mal aufgerufen, wenn ein Knoten neu in den Sichtbarkeitsbereich kommt (Zeile 16 im Algorithmus).

Landschaftsgenerierung

Bei der Definition einer Datenbasis weist der Anwender jeder Seite einer Straße einen Landschaftstypen zu und kontrolliert dessen Aussehen über einige wenige Parameter. Ein Landschaftstyp besteht aus zwei Teilen: Einer Beschreibung des Höhenprofils der Landschaft und einer Menge von 3D-Modellen, die in der Landschaft platziert werden. In der Methode *DoGeoInst* von Knoten des Typs *Course* wird aus diesen Teilen automatisch eine Landschaft generiert. Damit ist es beispielsweise möglich, der Landschaft bei jedem Durchfahren einer topologischen Schleife ein anderes Aussehen zu geben. So kann der Anwender verhindern, dass der Testfahrer die Schleife anhand landschaftlicher Merkmale erkennt.

Wie im Abschnitt „Modellierung“ dargestellt, wird das Höhenprofil einer Landschaft über Formparameter von Höhenkurven definiert. Bei jedem Aufruf von *DoGeoInst* eines Knotens vom Typ *Course* werden diese Parameter innerhalb vom Anwender definierter Grenzen zufällig gezogen und damit ein neues Höhenprofil erzeugt.

Die Objekte, die für einen bestimmten Landschaftstyp charakteristisch sind, werden in Kategorien eingeteilt. Jede Kategorie enthält eine Menge von 3D Modellen. Zusätzlich sind in jedem Landschaftstyp Constraints gespeichert, die angeben, ob Objekte einer bestimmten Kategorie die einer anderen Kategorie überlappen dürfen. Abbildung 19 zeigt die Objektkategorien und Constraints für eine einfache Version des Landschaftstyps „bewirtschaftet“.

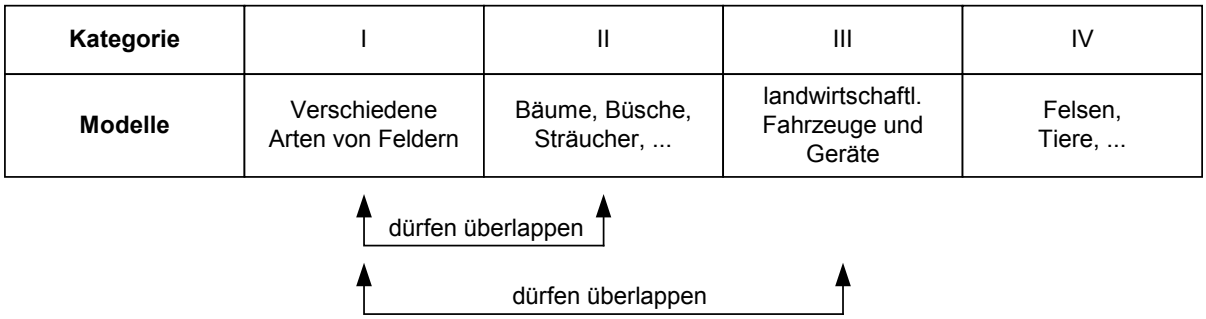

**Abbildung 19: Landschaftstyp "bewirtschaftet" /
Landscape type „farmed“**

Der Algorithmus zur Landschaftsgenerierung zieht Objekte aus jeder Kategorie und platziert sie derart, dass

- die Objekte einer Kategorie mit einer bestimmten Dichte erscheinen. Die Dichte wird bei jedem Aufruf von *DoGeoInst* aus einem Intervall gezogen, das der Anwender bei der Datenbasisdefinition vorgibt.
- die Constraints erfüllt werden. Hierzu werden R-Trees (vgl. [7]) als Methode zur räumlichen Indizierung verwendet.

Eine detaillierte Beschreibung des Verfahrens findet sich in [9]. Ein ähnlicher Algorithmus zur Verteilung der Objekte wird in [8] vorgestellt.

Zusammenfassung

Die vorgestellte Architektur für Fahrsimulator-Datenbasen erlaubt es, das Straßennetzwerk und die es umgebende Landschaft während der Simulation und für den Fahrer unmerklich zu verändern. Das Straßennetzwerk wird vom Anwender auf einer topologischen Ebene definiert. Er kann dabei die Abfolge der Streckenabschnitte von Bedingungen abhängig machen, wodurch der Entwurf reproduzierbarer Szenarios vereinfacht wird. Außerdem werden dem Anwender Mechanismen zur Verfügung gestellt, mit denen er die Freiheitsgrade, die ein Fahrer bei der Wahl der Route durch das Straßennetzwerk hat, kontrollieren kann. Aus der topologischen Repräsentation wird während der Simulation im Sichtbarkeitsbereich des Fahrers ein geometrisch konsistentes Straßennetzwerk erzeugt.

Literatur

[1] B.E. Artz: An Analytical Road Segment Terrain Database for Driving Simulation. Proc. of the Driving Simulation Conference (DSC95). Paris: 1995.

[2] A. C. Bailey: Advancement in logical road network design for the Leeds driving simulator. Proc. of the Driving Simulation Conference (DSC2000). Paris: 2000.

[3] S. Bayarri, M. Fernandez, M. Perez: Virtual reality for driving simulation. Communications of the ACM 39(5):72–76. 1996.

[4] S. Buld, S. Hoffmann, A. Kaussner, H. Tietze, I. Totzke, H.-P. Krüger: Wirkungen von Assistenz und Automation auf Fahrerzustand und Fahrsicherheit. Abschlussbericht BMBF 19 S 9812 7. Bonn: 2002.

[5] O. Carles, S. Espié: Multi-level environments modelling for road simulations. Proc. of the Driving Simulation Conference (DSC2000). Paris: 2000.

[6] M. Grein, H.-P. Krüger, H. Noltemeier: A framework for ambient Traffic in the IZVW driving simulator. Proc. of the Driving Simulation Conference (DSC2002). Paris: 2002.

[7] A. Guttman: R-TREES: A dynamic index structure for spatial searching. Proc. ACM SIGMOD. 1984.

[8] J. Hammes: Modeling of ecosystems as a data source for real-time terrain rendering. LNCS 2181. Springer. 2001.

[9] C. Mark: Generierung und Visualisierung von dynamischen Szenerien in der Fahrsimulation. Diplomarbeit an der Universität Würzburg. 2002.

[10] Y.E. Papelis: Rapid development of domain specific correlated databases using parameterized tiles. Proc. of the IMAGE Conference (pp. YE1-YE8). 1998.

[11] Y.E. Papelis, S. Bahauddin: Logical modeling of roadway environment to support real-time simulation of autonomous traffic. SIVE95: The First Workshop on Simulation and Interaction in Virtual Environments, pp. 62–71. Iowa: 1995.

[12] Richtlinien für die Anlage von Straßen, Teil: L (Linienführung). Forschungsgesellschaft für Straßen- und Verkehrswesen. Bonn – Bad Godesberg: 1995.