UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

ist
Institut für Softwaretechnik

Entwicklung eines Werkzeugs zur Konstruktion von Manövern mit grafischer Ausgabe

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers
im Studiengang Informatik

vorgelegt von
Bastian Zimmermann

Erstgutachter:	Prof. Dr. Dieter Zöbel Institut für Softwaretechnik, AG Zöbel
Zweitgutachter:	Diplom-Informatiker Christian Weyand Institut für Softwaretechnik, AG Zöbel
Betreuer:	Diplom-Informatiker Christian Weyand

Koblenz, im Juni 2011

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

.....

(Ort, Datum) (Unterschrift)

II

Zusammenfassung

Für die Planung von Wegen eines Gespanns sind komplexe Bewegungen verschiedener Bezugspunkte des Fahrzeugs zu beachten. Um die Betrachtung dieser Bewegungen zu vereinfachen, wird eine Fahrt in elementare Fahrbewegungen aufgeteilt, diese werden als Manöver bezeichnet. Ein Manöver besteht in diesem Zusammenhang aus zwei Elementen. Zum einen werden Pfade für bestimmte Bezugspunkte konstruiert, zum anderen wird das Gespann während der Manöverausführung von einem Korridor umschlossen. Die Pfade des Fahrzeugs müssen dabei fahrbar sein, das heißt, sie müssen die kinematischen Einschränkungen des Fahrzeugs beachten. Der Manöverkorridor kann als Grundlage verwendet werden, um die Kollisionsfreiheit zu garantieren. Während des Manövers verlässt kein Fahrzeugteil den Korridor. Es gibt verschiedene Manövertypen. Derzeit werden das Kurvenmanöver, das Wendemanöver und die Geradeausfahrt unterschieden. Außerdem kann ein Manöver zur Zeit mit zwei unterschiedlichen Konstruktionsmethoden erstellt werden, der konventionellen und der iterativen Methode.

In dieser Diplomarbeit wird eine Datenstruktur entworfen und implementiert, die ein Manöver konstruiert. Diese Datenstruktur wird in ein schon bestehendes Werkzeug integriert. Dabei kann der Benutzer mit der Software interagieren, um verschiedene Parameter eines Manövers zu verändern. Das Manöver wird daraufhin auf der Grundlage dieser Parameter konstruiert. Dazu gehört auch eine Visualisierung innerhalb der Software, in der die Bestandteile eines Manövers dargestellt werden können. Die Visualisierung kann in eine Bilddatei exportiert werden kann.

Abstract

Planning routes for trucks with a trailer is a complex procedure. In order to simplify this process, a route is segmented into elementary components, which represents basic motions of the considered vehicle. These elementary components are called maneuvers and are composed of two party. First, paths are constructed for certain reference points. Second, the vehicle is enclosed by a corridor during the execution of a maneuver. The paths of the vehicle have to take driveability into consideration. They must respect the kinematic constraints of the vehicle. The maneuver corridor can be used as a basis to guarantee collision-free motion planing. No part of the vehicle leaves the corridor during the maneuver. There are different types of maneuvers. Currently, the bending maneuver, the cusp maneuver and the straight line maneuver can be distinguished. In addition, a maneuver can be created with two different construction methods, the conventional and the iterative method.

In this thesis, a data structure to construct a maneuver is designed and implemented. The data structure is integrated into an already existing tool. The user can interact with the software to adjust various parameters of a maneuver. Afterwards the maneuver is generated based on these parameters. This also includes a visualization within the software, which can plot the parts of a maneuver. The visualization can be exported to an image file.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 1

1.2 Zielsetzung 2

1.3 Aufbau der Diplomarbeit 3

2 Grundlagen **5**

2.1 Beschreibung von Fahrzeugen 5

2.1.1 Einführung des Einspurmodells 5

2.1.2 Modellierung eines Zugfahrzeugs mit einachsigem Anhänger 7

2.2 Beschreibung von Fahrzeugbewegungen 7

2.2.1 Betrachtung der Fahrkurve des Zugfahrzeugs 8

2.2.2 Betrachtung der Fahrkurve des Anhängers 10

2.2.3 Innere und äußere Traktion 12

2.3 Manöver 16

2.3.1 Definition und Eigenschaften eines Manövers 16

2.3.2 Kurvenmanöver 18

2.3.2.1 Eigenschaften und Konstruktion eines Pfades nach der konventionellen Methode 19

2.3.2.2 Konstruktion eines ringförmigen Manöverkorridors nach der konventionellen Methode 24

2.3.2.3 Eigenschaften und Konstruktion eines Pfades nach der iterativen Methode 29

2.3.2.4 Konstruktion eines ringförmigen Manöverkorridors nach der iterativen Methode 32

2.3.3 Wendemanöver 35

2.3.4 Geradeausfahrt 37

2.3.5 Alternative Konstruktion eines Manöverkorridors 38

2.4 EZauto 43

2.4.1 Verwendete EZauto-Komponenten 44
2.4.1.1 Verwendung von Schnittstellen 45
2.4.1.2 Beschreibung von Fahrzeugen 45
2.4.1.3 Geometrie 46
2.4.1.4 Funktionsobjekte 46
2.4.1.5 Visualisierung des Manövers 47
2.4.2 Visualisierungswerkzeug PPFigure 47
2.5 Die Bibliothek wxWidgets 50
2.6 Verwendete Entwurfsmuster 51
2.6.1 Adapter-Entwurfsmuster 51
2.6.2 Kompositum-Entwurfsmuster 51

3 Planung der Software 53
3.1 Randbedingungen 53
3.2 Anforderungen an die Software 54
3.2.1 Funktionale Anforderungen 54
3.2.2 Nicht-funktionale Anforderungen 59
3.3 Meilensteinplan 60

4 Softwareentwurf 61
4.1 Entwurf der Benutzerschnittstelle (GUI) 63
4.1.1 Die GUI-Klassen der verschiedenen Funktionalitäten . . . 64
4.1.2 Auswahlmöglichkeiten zum Programmstart 66
4.1.3 Benutzerschnittstelle zur Konstruktion eines Manövers . . 70
4.2 Entwurf der Modelldaten 77
4.2.1 Die Hauptklasse der Modelldaten, *ManoeverDaten* 78
4.2.2 Die Daten des Gespanns in der Klasse *ManoeverFahrzeug* 81
4.2.3 Entwurf der Manöverkonstruktion 83
4.2.3.1 Konstruktion der Phasen eines Manövers 83
4.2.3.2 Konstruktion des Manöverkorridors 87
4.2.4 Visualisierung des Manövers 91
4.3 Entwurf der Darstellungskonfiguration 95
4.4 Kommunikation zwischen GUI, Modelldaten und Darstellung . . 96
4.5 Ausblick auf Entwurf und Implementierung 99

5 Vergleich verschiedener Konfigurationen 101
5.1 Visualisierungen eines Manövers mit PPFigure 101

5.2 Vergleich verschiedener Konfigurationen 104

6 Fazit **109**

6.1 Zusammenfassung der Diplomarbeit 109

6.2 Ausblick 111

A Anhang **113**

Abbildungs- und Tabellenverzeichnis

Abb. 2.1 Darstellung eines Zugfahrzeugs mit Anhänger als Einspurmodell 6
Abb. 2.2 Bestimmung des Winkels u 9
Abb. 2.3 Äußere Traktion 13
Abb. 2.4 Innere Traktion 14
Abb. 2.5 Monotonieverhalten des Einknickwinkels 15
Abb. 2.6 Winkel Γ bei Fahrtrichtung rückwärts 20
Abb. 2.7 Winkel Γ bei Fahrtrichtung vorwärts 21
Abb. 2.8 Schematische Darstellung einer Ausrichtungsänderung . . . 22
Abb. 2.9 Skizze einer Ringhülle 24
Abb. 2.10 Mittelpunkt der Anhängerachse bei einem Wendemanöver . 36
Abb. 2.11 Konstruktion einer Polygonhülle um eine Phase 38
Abb. 2.12 Aufbau der Softwarearchitektur von EZauto 44
Abb. 2.13 Darstellung eines Zugfahrzeugs mit Anhänger als Einspurmodell 48
Abb. 2.14 Layout der Software PPFigure 49
Abb. 2.15 Adapter-Entwurfsmuster in UML-Notation 51
Abb. 2.16 Kompositum-Entwurfsmuster in UML-Notation 52

Tab. 3.1 Meilensteinplan 60

Abb. 4.1 Trennung zwischen GUI, Darstellung und Modelldaten. . . . 62
Abb. 4.2 Die Hauptklassen der GUI. 63
Abb. 4.3 Klassendiagramm des GUI- Konzepts für die verschiedenen Funktionalitäten. 64
Abb. 4.4 Registerkarten für die Manöverkonstruktion. 65
Abb. 4.5 Aktivitätsdiagramm der Möglichkeiten beim Programmstart. 67
Abb. 4.6 Registerkarte beim Start des Programms. 68

Abb. 4.7 Klassendiagramm für die Benutzeroberfläche beim Start des Programms. 69
Abb. 4.8 Abstrakte Klasse *ManoeverTab* und Implementationen dieser Klasse. 72
Abb. 4.9 Module der Klasse *ManoeverTab*. 73
Abb. 4.10 Instabile Phase im Vergleich zwischen konventioneller und iterativer Methode. 74
Abb. 4.11 Die Kontrollkästchen von *FahrkurvenTab*. 75
Abb. 4.12 Die Klasse *FahrkurvenTab*. 75
Abb. 4.13 Die Klasse *KorridorTab*. 76
Abb. 4.14 Die Klasse *ZusammenfassungTab*. 77
Abb. 4.15 Die Zusammenfassung eines konventionellen Kurvenmanövers. 78
Abb. 4.16 Die abstrakte Klasse *ManoeverDaten*. 79
Abb. 4.17 Die konkreten Klassen von *ManoeverDaten*. 80
Abb. 4.18 Die Klasse *ManoeverFahrzeug* mit ihren Bestandteilen. . . . 81
Abb. 4.19 Entwurf der einzelnen Phasen eines Manövers. 84
Abb. 4.20 Pfade und Fahrkurven einer Phase. 86
Abb. 4.21 Entwurf des Konzepts für die verschiedenen Korridormethoden. 88
Abb. 4.22 Konfiguration einer Ringhülle im Vergleich zwischen konventioneller und iterativer Methode. 90
Abb. 4.23 Die Klasse *VisuWrapper* mit ihren Aggregationen 93
Abb. 4.24 Entwurf des Containers für die verschiedenen Visualisierungen. 94
Abb. 4.25 Die Klassen der Darstellungskonfiguration. 95
Abb. 4.26 Kommunikationsdiagramm für die Konstruktion eines Manövers bezogen auf eine Fahrkurve. 97
Abb. 4.27 Kommunikationsdiagramm zur Änderung der Darstellung einer Fahrkurve. 98

Abb. 5.1 Kurvenmanöver eines Gespanns nach der konventionellen Konstruktionsmethode. 102
Abb. 5.2 Kurvenmanöver mit vertauschten Lenkwinkeln. 103
Abb. 5.3 Kurvenmanöver in Fahrtrichtung vorwärts. 104
Tab. 5.1 Vergleich verschiedener Lenkwinkel der zweiten Phase bei einem LKW mit einachsigem Anhänger 105
Tab. 5.2 Vergleich verschiedener Lenkwinkel der ersten und dritten Phase 106

Tab. 5.3 Vergleich verschiedener Lenkwinkel der zweiten Phase bei einem PKW mit Anhänger 107

Abb. A.1 Die Klasse *ManoeverDaten* mit ihren Komponenten 114
Abb. A.2 Pfade der beiden Mittelpunkte vom Zugfahrzeug bei einem Kurvenmanöver. 115
Abb. A.3 Pfad der Kupplung bei einem Kurvenmanöver. 115
Abb. A.4 Pfad des Achsenmittelpunkts vom Anhänger bei einem Kurvenmanöver. 116
Abb. A.5 Fahrkurven der Räder des Zugfahrzeugs bei einem Kurvenmanöver. 116
Abb. A.6 Fahrkurven der Ecken des Zugfahrzeugs bei einem Kurvenmanöver. 117
Abb. A.7 Fahrkurven der Räder des Anhängers bei einem Kurvenmanöver. 117
Abb. A.8 Fahrkurven der Ecken des Anhängers bei einem Kurvenmanöver. 118
Abb. A.9 Fahrkurven aller Extrempunkte bei einem Kurvenmanöver. . 118

Kapitel 1

Einleitung

In dieser Einleitung wird zuerst auf die Motivation der Diplomarbeit und danach auf die Zielsetzung der Arbeit eingegangen. Anschließend wird der Aufbau der Diplomarbeit vorgestellt.

1.1 Motivation

Autonome Fahrzeugbewegungen sind keine Zukunftsvision mehr. Gerade wenn es um Fahrzeugbewegungen in der Logistik geht, stellt sich oft die ökonomische Frage nach der Effizienz. Bereiche, zu denen Menschen keinen Zutritt mehr haben und in denen sich Fahrzeuge völlig autonom bewegen sind keine Ausnahme mehr (siehe zum Beispiel [Straßmann, 2007]). Dabei spielen nicht nur die Einsparungen der Gehälter von manuell bedienten Geräten eine Rolle, auch die Vermeidung von Unfällen und die damit entstehenden Kosten durch Beschädigung und Ausfall sind von Bedeutung.

In den angesprochenen Bereichen, die zum Beispiel Lager von Häfen sein können, findet man Spezialfahrzeuge, die nur für diese Bereiche vorgesehen sind. Um die kompletten Vorgänge auf einem Betriebshof zu automatisieren, müssen aber auch die Serienfahrzeuge, die den Transport auf den öffentlichen Straßen vornehmen, so umgebaut werden, dass sie sich in bestimmten Umgebungen autonom verhalten können.

Die Arbeitsgruppe Echtzeitsysteme der Universität Koblenz-Landau beschäftigt sich mit dem autonomen, dem assistierten und dem simulierten Fahren. Bei autonomen Fahrten geht es dabei unter anderem um die Planung von Fahrten, speziell den Fahrten eines Gespanns, bestehend aus Zugfahrzeug und einachsigem

Anhänger. Um den komplexen Vorgang einer Fahrt zu beschreiben, wird diese in einfache Bausteine aufgeteilt, die zusammengesetzt die komplette Fahrt ergeben. Dieser Ansatz, der in dieser Diplomarbeit behandelt wird, bezeichnet man als Manöver. Ein Manöver ist definiert durch die Pfade bestimmter Bezugspunkte und einem Korridor, der das Fahrzeug während der Manöverausführung umschließt. Die Pfade bei einem Manöver sind immer auch fahrbar, berücksichtigen also die kinematischen Einschränkungen des Fahrzeugs. Der Korridor ist eine Fläche, die so angelegt ist, dass sich kein Teil des Fahrzeugs während des Manövers außerhalb dieser Fläche bewegt. Diese Fläche ist wichtig für die Planung, um Kollisionen ausschließen zu können.

Von der Arbeitsgruppe werden verschiedene Manövertypen unterschieden. Diese sind zur Zeit das Kurvenmanöver, das Wendemanöver und die Geradeausfahrt. Darüber hinaus gibt es zwei Konstruktionsmethoden. Bei der konventionellen Methode wird der Lenkwinkel nur im Stand geändert, die iterative Methode erweitert diesen Ansatz, so dass auch Lenkwinkeländerungen während der Fahrt möglich sind.

Bei Veröffentlichungen der Arbeitsgruppe im Bereich der Manöverkonstruktion kommt es vor, dass grafische Darstellungen eines Manövers benötigt werden. In einem vorangegangenen Projektpraktikum wurde das Werkzeug PPFigure entwickelt, mit dem es möglich ist, Fahrzeuge als Einspurmodell darzustellen und die Visualisierung eines Fahrzeugmodells in eine EPS-Datei zu exportieren. Dieses Werkzeug soll im Rahmen der Diplomarbeit so erweitert werden, dass damit auch Manöver konstruiert und visualisiert werden können.

1.2 Zielsetzung

Um dieses Werkzeug zu erweitern ist es zunächst erforderlich, die Grundlagen dieser Thematik zu erarbeiten. Diese sollen aus den Veröffentlichungen und internen Dokumentationen der Arbeitsgruppe erworben werden, um die wissenschaftlichen Grundlagen in diesem Kontext zu verstehen. Dazu gehören die kinematischen Eigenschaften eines Gespanns und die mathematische Beschreibung von Bewegungen. Auch die speziellen Konstruktionsansätze, mit denen ein Manöver erstellt werden kann, werden in diesen Dokumentationen beschrieben. Es ist auch eine Einarbeitung in die vorhandenen Software und die benötigten Softwarebibliotheken vorzunehmen.

Auf dieser Grundlage soll mit der Planung der Erweiterung dieser Software be-

gonnen werden und ein Anforderungskatalog für die Integration der neuen Funktionalität, der Generierung von Manöver-Visualisierungen, erstellt werden. Dieser Anforderungskatalog soll verschiedene Arten von Manövertypen und Konstruktionsmethoden berücksichtigen. Insbesondere soll auch auf die Konstruktionsparameter eingegangen werden, die ein Benutzer für die Berechnung eines Manövers einstellen kann.

Im nächsten Schritt soll der Entwurf der Software vorgenommen werden. Auch der Softwareentwurf soll alle Manövertypen und Konstruktionsmethoden berücksichtigen. Der Entwurf des Projektpraktikums enthält eine Trennung zwischen der Benutzerschnittstelle, den Daten und den Darstellungskonfigurationen. Diese Trennung soll auch bei der Erweiterung der Software beibehalten werden und im Entwurf berücksichtigt werden.

Die Implementierung der Software soll durch die neue Funktionalität erweitert werden. Dazu müssen die Datenstrukturen und Funktionen anhand der Modelle der Entwurfsphase implementiert werden.

Als Letztes werden die Ergebnisse des Werkzeugs dargestellt und Vergleiche verschiedener Manöverkonfigurationen erfolgen. Bei diesen Vergleichen soll insbesondere die Fläche des Manöverkorridors als Vergleichskriterium angewendet werden.

1.3 Aufbau der Diplomarbeit

In Kapitel 2 werden die Grundlagen dieser Diplomarbeit besprochen. Dazu wird als erstes auf die kinematischen Eigenschaften eines Fahrzeugs eingegangen und dabei die mathematischen Voraussetzungen für eine Bewegungsbeschreibung geschaffen. Anschließend wird die Konstruktion verschiedener Manövertypen vorgestellt. Auch die verschiedenen Konstruktionsmethoden für das Manöver und den Manöverkorridor werden dort behandelt. Danach werden die Softwarebibliotheken vorgestellt, auf denen diese Arbeit basiert. Außerdem wird der schon bestehende Teil des Werkzeugs PPFigure vorgestellt.

Nach den Grundlagen wird in Kapitel 3 das Ergebnis der Planungsphase der Softwareentwicklung vorgestellt. Dazu gehören der Anforderungskatalog, in dem die Anforderungen an die Software aufgelistet werden und der Meilensteinplan, in dem die Ergebnisse der einzelnen Phasen der Entwicklung aufgezählt sind.

Anschließend wird die Entwurfsphase in Kapitel 4 beschrieben. Die Aufteilung dieses Kapitels wird anhand der Trennung zwischen der Benutzerschnittstelle, den

Modelldaten und der Darstellungskonfiguration vorgenommen. Nachdem auf alle Teile eingegangen wurde, wird die Kommunikation zwischen den einzelnen Bestandteilen beschrieben.

In Kapitel 5 werden die Visualisierungen verschiedener Manöver vorgestellt. Anschließend werden bestimmte Konfigurationen von Manövern miteinander verglichen. Durch die Ergebnisse des Manöverkorridors, insbesondere der Fläche des Korridors, lassen sich die verschiedenen Konfigurationen miteinander vergleichen.

Am Schluss werden in Kapitel 6 die Ergebnisse dieser Diplomarbeit zusammengefasst und anschließend ein Ausblick auf die mögliche weitere Entwicklung des Werkzeugs gegeben.

Kapitel 2

Grundlagen

In diesem Kapitel werden die Eigenschaften der Fahrzeuge betrachtet. Es wird dabei zuerst auf die Mathematik der Bewegungen eines Gespanns, bestehend aus einem Zugfahrzeug und einem einachsigen Anhänger, eingegangen. Dieses Gespann wird in einem kinematischen Modell abstrahiert, um die Berechnungen zu vereinfachen. Die elementaren Bewegungen werden als Manöver bezeichnet und können als Grundlage für die Planung von Wegen betrachtet werden. Im Anschluss gibt es einen Überblick über die Softwarebibliotheken und Komponenten, auf denen die Software aufbaut.

2.1 Beschreibung von Fahrzeugen

Der erste Teil der Grundlagen behandelt das kinematische Modell eines Fahrzeugs mit den verwendeten Bezeichnern für die einzelnen Bestandteile. Das betrachtete Fahrzeug besteht aus einem Zugfahrzeug und einem einachsigen Anhänger. Zuerst wird aber auf die Modellierung eines Fahrzeugs allgemein eingegangen.

2.1.1 Einführung des Einspurmodells

Um die Bewegung eines Fahrzeugs mathematisch zu betrachten, wird dieses vereinfacht in einem Modell abgebildet. Nach [Zöbel, 2001] wird davon abgesehen, Masse und Kräfte eines Fahrzeugs und dessen Bewegung in die Berechnung fließen zu lassen. Außerdem wird das Fahrzeug in drei Schritte abstrahiert und damit ein longitudinales Modell gebildet:

1. Im ersten Schritt werden die Mehrfachachsen eines Fahrzeugs immer einzel-

ne Achse betrachtet. Es wird dazu beispielsweise eine virtuelle Achse in der Mitte einer Doppelachse modelliert.

2. Der nächste Schritt sieht vor, dass ein Rad nur in seinem Mittelpunkt die Fahrbahn berührt. Es kann also für die Berechnung dieses Mittelpunkts betrachtet werden.

3. Der dritte Schritt ist, dass das Fahrzeug als **Einspurmodell** betrachtet wird. Das daraus entstehende Fahrzeugmodell ist in Abbildung 2.1 zu sehen. Dabei werden die Räder für die Fahrzeugbewegung von oben betrachtet auf den Mittelpunkt der Achse abgebildet.

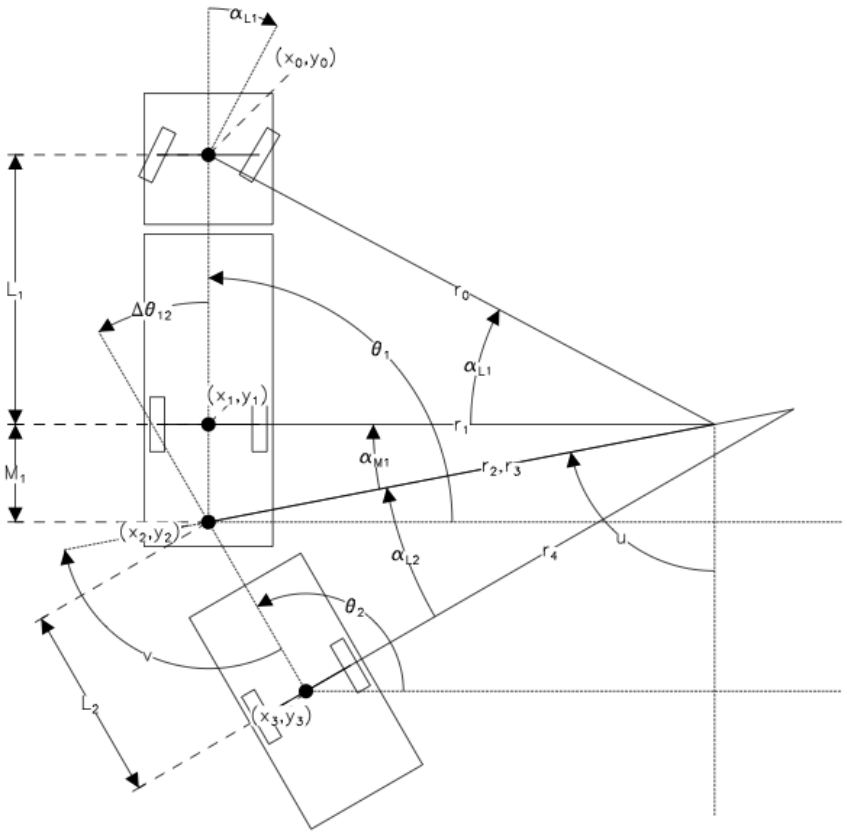


Abbildung 2.1: Darstellung eines Zugfahrzeugs mit Anhänger als Einspurmodell [Weyand & Balcerak, 2009]

2.1.2 Modellierung eines Zugfahrzeugs mit einachsigem Anhänger

Im Folgenden wird nicht mehr ein allgemeines Fahrzeug betrachtet, sondern konkret ein Gespann angenommen, das aus einem Zugfahrzeug mit einachsigem Anhänger besteht.

In der Abbildung 2.1 ist ein Zugfahrzeug mit einachsigem Anhänger zu sehen. Der Achsenmittelpunkt der Vorderachse des Zugfahrzeugs ist mit (x_0, y_0) und der Punkt der Hinterachse mit (x_1, y_1) gekennzeichnet. Die Kupplung wird mit dem Punkt (x_2, y_2) beschrieben und der Mittelpunkt der Anhängerachse mit (x_3, y_3).

Zur Beschreibung der Bewegung werden verschiedene Abstände zwischen den genannten Punkten betrachtet. Der Abstand zwischen der Vorder- und der Hinterachse wird L_1 benannt. Der Abstand zwischen Hinterachse und Anhängerkupplung nennt man M_1. Als Letztes wird die Länge zwischen der Kupplung und der Anhängerachse mit L_2 bezeichnet.

Es gibt verschiedene Winkel, die für die Berechnungen einer Bewegung benötigt werden. Bei der strukturellen Betrachtung eines Fahrzeugs ist dabei auf den Lenkwinkel, α (in der Abbildung 2.1 als α_{L_1} bezeichnet), und den Einknickwinkel zwischen Zugfahrzeug und Anhänger, $\Delta\theta$ (in der Abbildung 2.1 als $\Delta\theta_1$ bezeichnet), hinzuweisen.

Die Winkel u und v werden aus den Berechnungen der Kurvenpunkte hergeleitet und später beschrieben. Der Winkel u definiert dabei die Lage der Kupplung im Bezug auf den Ursprung in dem gewählten Koordinatensystem, wohingegen v die Ausrichtung des Anhängers bezogen auf den Winkel u beschreibt [Weyand & Balcerak, 2009].

Im Weiteren werden die oben genannten Abstände in Beziehung zueinander betrachtet und damit Winkel bestimmt, die mit α_{L_1} (entspricht dem Lenkwinkel α), α_{M_1} und α_{L_2} bezeichnet sind. Die Radien, die die Punkte verbinden und mit diesen Winkeln getrennt sind, werden mit r_0 bis r_4 bezeichnet.

2.2 Beschreibung von Fahrzeugbewegungen

Ein Zugfahrzeug mit Anhänger kann in einer elementaren Bewegung einen bestimmten Weg zurücklegen. Als Einschränkung, die innerhalb dieser Bewegung gemacht wird, bewegt sich das Zugfahrzeug mit einem unveränderten Lenkwinkel auf einer Kurve um einen festen Mittelpunkt. Für den Anhänger betrachtet man sogenannte Traktrixkurven, auch Ziehkurve genannt [Zöbel & Balcerak, 2000], um

die Fahrkurve des Anhängers zu beschreiben. Im Folgenden werden die Kurven mathematisch eingeführt.

2.2.1 Betrachtung der Fahrkurve des Zugfahrzeugs

Aus den Abmessungen des Fahrzeugs, L_1 und M_1 und dem Lenkwinkel $\alpha \neq 0°$, können die Radien und Winkel berechnet werden, die für die Bestimmung der Lage des Zugfahrzeugs nötig sind. Die Herleitungen sind dabei in [Zöbel, 2001] zu finden.

Da zwischen dem Radius r_2 und dem Abstand L_1 ein rechter Winkel besteht, ergibt sich mit dem Lenkwinkel α der Radius r_2:

$$r_2(\alpha) = \frac{L_1}{\tan(\alpha)} \tag{2.1}$$

Daraus lässt sich der Winkel α_{M_1} berechnen, wenn man den Abstand M_1 aus Abbildung 2.1 in die Berechnung einbezieht.

$$\alpha_{M_1}(\alpha) = \arctan\left(\frac{M_1}{r_2(\alpha)}\right) \tag{2.2}$$

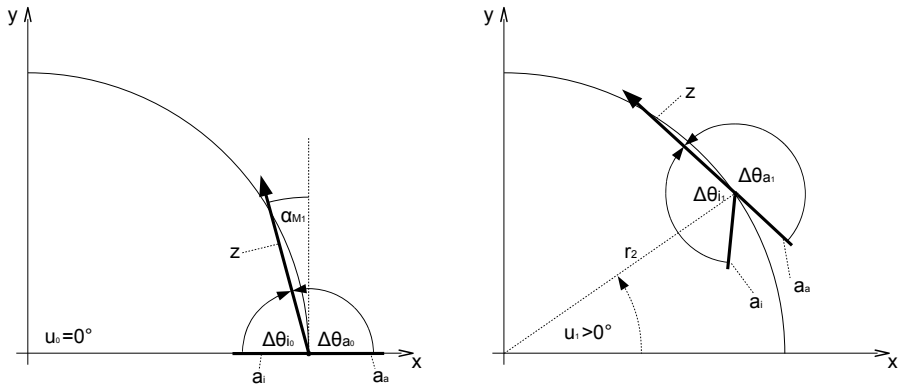
Unter Anwendung des Satzes des Pythagoras kann man nun die Radien r_1 und r_0 berechnen:

$$r_1(\alpha) = \sqrt{L_1^2 + r_2(\alpha)^2} \tag{2.3}$$

$$r_0(\alpha) = \sqrt{M_1^2 + r_2(\alpha)^2} \tag{2.4}$$

Bei der Betrachtung der Pfade des Zugfahrzeugs sind drei Punkte relevant, die Mittelpunkte der Vorder- und Hinterachse und die Kupplung. Es werden dabei immer Punkte im Koordinatensystem betrachtet, die aus einem x-Wert und einem y-Wert bestehen. Bei der Berechnung eines Punktes, (x_i, y_i), im zweidimensionalen Raum, wird ausgehend vom Koordinatenursprung der jeweilige Radius, r_i, des Punktes auf diesem Kreis in der Rechnung benötigt. Die Berechnungen für die Punkte bei beliebigen Winkeln lassen sich mit der Sinusfunktion für den y-Wert und der Kosinusfunktion für den x-Wert berechnen.

Um den ersten Punkt, die Lage der Kupplung, zu berechnen, muss zuerst genauer auf den schon genannten Winkel u eingegangen werden. In der Abbildung 2.2 ist die Ausgangssituation des Winkels u und dessen Verlauf skizziert. Die Linie



(a) Ausgangssituation: $u = 0°$ (b) Beispielsituation für $u > 0°$

Abbildung 2.2: Bestimmung des Winkels u

z soll dabei das Zugfahrzeug darstellen. Es wird davon ausgegangen, dass sich das Fahrzeug mit einem festen Lenkwinkel um den Koordinatenursprung bewegt. Der Kreisbogen stellt dabei die Lage der Kupplung während der Fahrt dar. Der Anhänger ist hier für den Fall, dass der Anhänger am Anfang innerhalb des Kreisbogens steht, der die Fahrt der Kupplung kennzeichnet, mit a_i gekennzeichnet. Im anderen Fall, bei dem der Anhänger außerhalb des Kreisbogens steht, wird der Anhänger mit a_a bezeichnet. Der Unterschied zwischen diesen beiden Fällen wird im Kapitel 2.2.3 genau beschrieben. Teil (a) der Abbildung stellt die Ausgangssituation dar. Der Anhänger liegt dabei auf der X-Achse des Koordinatensystems, folglich ist auch die X-Koordinate der Kupplung 0. Zugfahrzeug und Anhänger haben einen Einknickwinkel $\Delta\theta$ der sich vom Winkel α_{M_1} um 90° unterscheidet. In dieser Ausgangssituation hat der Winkel u_0 einen Wert von 0°.

Bewegt sich das Zugfahrzeug nun weiter um den Koordinatenursprung, verändert sich auch die Lage der Kupplung auf dem Kreisbogen. Diese Änderung der Position ist im Teil (b) der Abbildung dargestellt. Der Winkel u_1 ist nun größer als 0°. In der Abbildung ist auch der Radius der Kupplung, r_2 eingezeichnet. Betrachtet man den Winkel u und den Radius r_2, ergibt sich für die Lage der Kupplung folgende Berechnung:

$$x_2(\alpha, u) = r_2(\alpha) * \cos(u) \tag{2.5}$$

$$y_2(\alpha, u) = r_2(\alpha) * \sin(u) \tag{2.6}$$

Bei der Berechnung der Hinterachse des Zugfahrzeugs wird der Radius r_2 betrachtet. Außerdem muss zu dem gewählten Winkel u der Winkel α_{M_1} addiert werden.

$$x_1(\alpha, u) = r_2(\alpha) * \cos(u + \alpha_{M_1}(\alpha)) \tag{2.7}$$

$$y_1(\alpha, u) = r_2(\alpha) * \sin(u + \alpha_{M_1}(\alpha)) \tag{2.8}$$

Zuletzt wird die Vorderachse mit dem Radius r_1 berechnet, wobei der Winkel zusätzlich mit dem aktuellen Lenkwinkel α addiert werden muss.

$$x_0(\alpha, u) = r_1(\alpha) * \cos(u + \alpha_{M_1}(\alpha) + \alpha) \tag{2.9}$$

$$y_0(\alpha, u) = r_1(\alpha) * \sin(u + \alpha_{M_1}(\alpha) + \alpha) \tag{2.10}$$

Die Erweiterungen des Winkels kann man aus Abbildung 2.1 ablesen.

2.2.2 Betrachtung der Fahrkurve des Anhängers

Analog zur Berechnung des Zugfahrzeugs beginnt auch die Berechnung des Anhängers mit dem Radius r_4, auf dem sich der Mittelpunkt der Anhängerachse bewegt.

$$r_4(\alpha) = \sqrt{r_2(\alpha)^2 - L_2^2} \tag{2.11}$$

Daraus kann nun auch der letzte Radius, r_3, berechnet werden:

$$r_3(\alpha) = \sqrt{r_4(\alpha)^2 + L_2^2} \tag{2.12}$$

Darüber hinaus gibt es zwei weitere Winkel, die jetzt schon genannt werden können und später benötigt werden. Diese Winkel beziehen sich auf den Radius r_2, einmal im Bezug auf die Längsachse des Zugfahrzeugs, bezeichnet als wg_1 und der Andere im Bezug auf die Längsachse des Anhängers, bezeichnet als wg_2.

$$wg_1(\alpha) = \arctan\left(\frac{M_1}{r_2(\alpha)}\right) \tag{2.13}$$

$$wg_2(\alpha) = \arctan\left(\frac{L_2}{r_2(\alpha)}\right) \tag{2.14}$$

Die Punkte des Anhängers bewegen sich auf einer Traktrixkurve. Das bedeutet, dass diese ausgehend von einem bestimmten Punkt gesehen werden muss, der den Anhänger zieht, was in diesem Fall die Kupplung ist. Es gibt vier verschiedene Fälle, die laut [Weyand & Balcerak, 2009] bei einer Traktrixkurve unterschieden werden können. Abhängig davon, ob der Lenkwinkel $\alpha = 0^\circ$ oder $\alpha \neq 0^\circ$ ist, kann man jeweils die Fälle für einen aktuellen Einknickwinkel $\Delta\theta = 0^\circ$ und $\Delta\theta \neq 0^\circ$ bestimmen.

1. Der Fall $\alpha = 0^\circ$ wird nicht bei dem Kurvenmanöver oder Wendemanöver betrachtet, bei diesen Manövern ist in allen Phasen ein Lenkwinkel $\alpha \neq 0^\circ$ gefordert. Dagegen ist dieser Fall wichtig für die Geradeausfahrt.

   - Es gibt einen Spezialfall, wenn der Einknickwinkel $\Delta\theta = 0$ ist. In diesem Fall bewegen sich der Achsenmittelpunkt des Anhängers auf derselben Geraden wie die Achsenmittelpunkte des Zugfahrzeugs.
   - Im Standardfall ist der Einknickwinkel $\Delta\theta \neq 0$. Damit liegt das klassische Traktixproblem vor, das schon von Gottfried W. Leibnitz 1693 beschrieben wurde (siehe [Wünsch, 1997]).

2. Als zweites wird ein Lenkwinkel $\alpha \neq 0$ betrachtet. Dabei wird wieder der Einknickwinkel unterschieden.

   - Auch hier gibt es einen Spezialfall, in dem sich alle Punkte des Anhängers um denselben Mittelpunkt wie das Zugfahrzeug drehen. Dazu ist ein spezieller Einknickwinkel, $\Delta\theta_{circ}$ in Relation zu dem aktuellen Lenkwinkel nötig, der wie folgt berechnet werden kann:

     $$\Delta\theta_{circ}(\alpha) = (-sign(\alpha)) * (\pi - wg_1(\alpha) - wg_2(\alpha)) \tag{2.15}$$

     Umgekehrt kann auch aus einem gegebenen Einknickwinkel der Lenkwinkel für diesen Spezialfall berechnet werden. Die Berechnung dazu stammt aus [Berg & Zöbel, 2005]:

     $$\alpha(\Delta\theta_{circ}) = arctan\left(\frac{L_1 * sin(-Delta\theta_{circ})}{L_2 + M_1 * cos(\Delta\theta_{circ})}\right) \tag{2.16}$$

     Da sich während dieser Fahrt der Einknickwinkel nicht ändert, wird dieser Fall als **stabile Fahrt** beschrieben.

- Im allgemeinen Fall ist dieser Zustand nicht gegeben. In diesem, auch als **instabile Fahrt** bezeichneten Fall bewegt sich der Anhänger auf einer Traktrixkurve. Auf diesen Fall wird im Folgenden näher eingegangen.

Bei der Berechnung der Lage des Mittelpunkts des Anhängers muss, im Gegensatz zum Zugfahrzeug, auch der Winkel v berücksichtigt werden, der die Ausrichtung des Anhängers im Bezug zu dem Winkel u beschreibt. Die Formel für die Koordinatenberechnung ist laut [Weyand & Balcerak, 2009]:

$$x_3(u, v, \alpha) = r_2(\alpha) * \cos(u) + L_2 * \cos(u - v) \tag{2.17}$$

$$y_3(u, v, \alpha) = r_2(\alpha) * \sin(u) + L_2 * \sin(u - v) \tag{2.18}$$

Die Bedeutung der Winkel u und v in dieser Berechnung werden im nächsten Abschnitt erläutert.

2.2.3 Innere und äußere Traktion

Es werden zwei verschiedene Arten der Traktion betrachtet, die sich darauf beziehen, ob sich der Anhänger am Anfang außerhalb des Radius der Kupplung befindet, äußere Traktion, oder innerhalb dieses Radius, innere Traktion genannt. Diese Fälle wurden schon in Abbildung 2.2 angedeutet. Bevor auf diese Traktionsarten eingegangen wird, soll gezeigt werden, wie sich die Winkel u und v mathematisch beschreiben lassen.

Der Winkel v lässt sich in Beziehung zu dem Winkel u und dem Lenkwinkel α betrachten. Es gibt aber auch die Möglichkeit, die Beziehungen zwischen α, dem aktuellen Einknickwinkel $\Delta\theta$ und dem Winkel α_{M_1} zu v zu beschreiben, was hier auch zuerst erfolgt (vgl. [Weyand *et al.*, 2010]):

$$v(\alpha, \alpha_{M_1}, \Delta\theta) = sign(\alpha) * \frac{\pi}{2} - \alpha_{M_1}(\alpha) - \Delta\theta \tag{2.19}$$

Diese Rechnung ist notwendig, da sich die Berechnung von u auf v und den Lenkwinkel α bezieht. Die folgenden Formeln beziehen sich dabei nur auf den zweiten Fall, also einem Lenkwinkel $\alpha \neq 0°$. Eine weitere Einschränkung, die bei einem Gespann mit Zugfahrzeug und einachsigem Anhänger in der Regel gegeben ist, bezieht sich darauf, dass $r_2 > L_2$ ist. Auch die anderen Fälle können mathe-

matisch beschrieben werden und sind unter [Zöbel, 2008] zu finden. Der Winkel u lässt sich aus dem Winkel v und dem Lenkwinkel α berechnen:

$$u(v,\alpha) = \frac{L_2}{r_4(\alpha)} * ln\left(\left|\frac{(r_2(\alpha) - L_2) * \tan(v/2) + r_4(\alpha)}{(r_2(\alpha) - L_2) * \tan(v/2) - r_4(\alpha)}\right|\right) \tag{2.20}$$

Diese Formel ist nur für $|v| \neq 2 * \arctan(r_4(\alpha)/(r_2(\alpha) - L_2))$ definiert, was auf den Definitionsbereich der Logarithmusfunktion zurückzuführen ist.

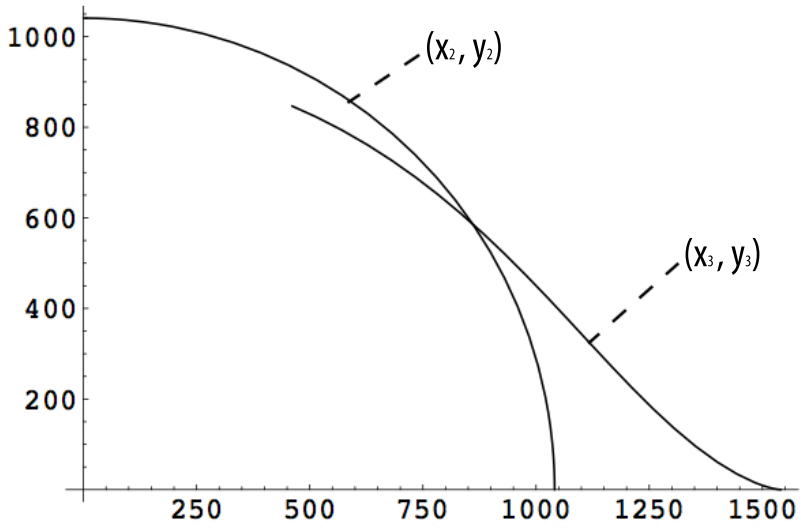


Abbildung 2.3: Äußere Traktion [Zöbel, 2008]

Davon ausgehend kann man nun v in Relation zu u setzen. Die folgenden beiden Formeln können unabhängig vom Einknickwinkel, $\Delta\theta_{circ}$ berechnet werden, der für die Formel 2.19 nötig ist. Bei der Berechnung müssen zwei Fälle betrachtet werden, die sich auf die eben getroffene Einschränkung beziehen. Der erste Fall ist, dass $|v| < 2 * \arctan(r_4(\alpha)/(r_2(\alpha) - L_2))$. In dieser Situation spricht man auch von einer **äußeren Traktion**. Der Name bezieht sich darauf, dass sich der Anhänger an der Stelle $u = 0°$ außerhalb des Kreises befindet, den das Zugfahrzeug abfährt. In Abbildung 2.3 ist die Situation zu sehen, die für diesen Fall gilt. Während sich die Kupplung (x_2, y_2) auf einem Kreisbogen um den Koordinatenursprung bewegt, liegt die Kurve des Anhängers (x_3, y_3) am Anfang außerhalb dieses Kreisbogens.

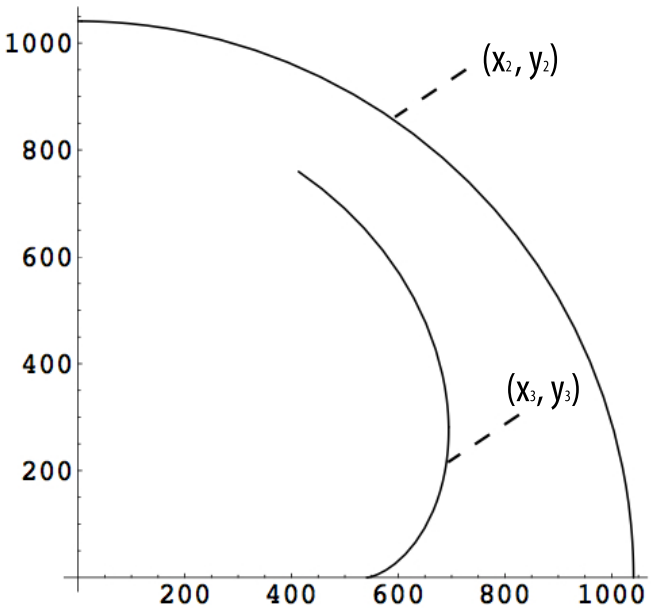


Abbildung 2.4: Innere Traktion [Zöbel, 2008]

Die Formel für diesen Fall ist nach [Weyand & Balcerak, 2009]:

$$v(u,\alpha) = 2 * \arctan\left(\frac{r_4(\alpha)}{r_2(\alpha) - L_2} * \frac{e^{(r_4(\alpha)/L_2)*u} - 1}{e^{(r_4(\alpha)/L_2)*u} + 1}\right) \tag{2.21}$$

Im zweiten Fall, der **inneren Traktion**, geht man davon aus, dass $|v| > 2 * \arctan(r_4(\alpha)/(r_2(\alpha) - L_2))$. Hierbei steht der Anhänger bei $u = 0$ innerhalb des Kreisbogens des Zugfahrzeugs. Diese Traktionsart ist in Abbildung 2.4 zu sehen. Die Kurve des Anhängers befindet sich innerhalb des Kreisbogens, auf dem sich die Kupplung bewegt. Die Berechnung dazu variiert nur im Kehrwert des letzten Bruchs der Formel und lautet:

$$v(u,\alpha) = 2 * \arctan\left(\frac{r_4(\alpha)}{r_2(\alpha) - L_2} * \frac{e^{(r_4(\alpha)/L_2)*u} + 1}{e^{(r_4(\alpha)/L_2)*u} - 1}\right) \tag{2.22}$$

Abbildung 2.5 zeigt ein Schema für die Vorwärtsfahrt, mit dem entschieden werden kann, ob es sich um eine äußere oder eine innere Traktion handelt, welche Formel zur Berechnung von v in Frage kommt. Es sei darauf hingewiesen, dass sich beide Traktionsarten im Bezug auf den Einknickwinkel $\Delta\theta$ streng monoton verhal-

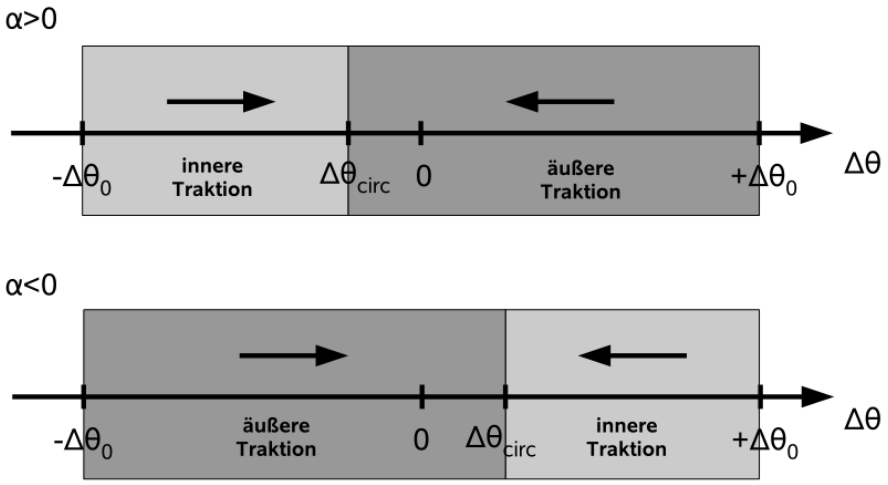


Abbildung 2.5: Monotonieverhalten des Einknickwinkels [Weyand & Balcerak, 2009]

ten und sich dem Winkel $\Delta\theta_{circ}$ beliebig nah annähern. Bei einer Rückwärtsfahrt müssen die Pfeile umgedreht werden.

In der Abbildung sieht man, dass es vier verschiedene Fälle zu beachten gibt, um die Traktionsart zu bestimmen. Diese Fälle sind abhängig von dem Vorzeichen des Lenkwinkels α, und dem Einknickwinkel für die stabile Fahrt $\Delta\theta_{circ}$. Die Grenzen des Intervalls $[-\Delta\theta_0, \Delta\theta_0]$, der als Definitionsbereich für die Formel 2.19 festgelegt ist, ergeben sich nach [Weyand *et al.*, 2010] für $v = 0$, bzw. $v = \pi$.

1. Der Lenkwinkel α ist positiv:

   (a) $-\Delta\theta_0 < \Delta\theta < \Delta\theta_{circ} < 0$: Es liegt eine innere Traktion vor.

   (b) $\Delta\theta_{circ} < \Delta\theta < \Delta\theta_0 \wedge \Delta\theta_{circ} < 0$: Es liegt eine äußere Traktion vor.

2. Der Lenkwinkel α ist negativ:

   (a) $0 < \Delta\theta_{circ} < \Delta\theta < \Delta\theta_0$: Es liegt eine innere Traktion vor.

   (b) $\Delta\theta_0 < \Delta\theta < \Delta\theta_{circ} \wedge \Delta\theta_0 > 0$: Es liegt eine äußere Traktion vor.

Die Entscheidung, welche Traktionsart vorliegt, ist von der Fahrtrichtung unabhängig. Die Fahrtrichtung gibt aber vor, ob der Einknickwinkel monoton größer oder kleiner wird. In Abbildung 2.5 ist eine Vorwärtsfahrt zu sehen. Bei dieser Fahrt wächst der Einknickwinkel in den Fällen 1.a und 2.b. In den Fällen 1.b und 2.a wird der Einknickwinkel monoton kleiner. Bei einer Rückwärtsfahrt ergibt sich

eine umgekehrte Betrachtung. Der Einknickwinkel wird in den Fällen 1.a und 2.b kleiner und wächst in den Fällen 1.b und 2.a.

Die Unterscheidung der Traktionsart wird nur bei der instabilen Fahrt benötigt, bis der Einknickwinkel $\Delta\theta_{circ}$ bzw. 0° erreicht wurden.

Als Letztes sollen die Ausrichtungen von Fahrzeug, θ_1, und Anhänger, θ_2, betrachtet werden. Diese können in Abhängigkeit des Lenkwinkels α und des Winkels u mathematisch beschrieben werden. Für das Zugfahrzeug ergibt sich damit nach [Weyand & Balcerak, 2009]:

$$\theta_1(\alpha, u) = sign(\alpha) * (\pi/2 - \alpha_{M_1}(\alpha)) + u \tag{2.23}$$

Die Ausrichtung des Anhängers wird berechnet mit:

$$\theta_2(\alpha, u) = (u - v(u, \alpha)) + \pi \tag{2.24}$$

2.3 Manöver

Der Hintergrund dieser Arbeit ist die Betrachtung von elementaren Fahrzeugbewegungen. Ziel einer derartigen Bewegung ist es, die Position des Fahrzeugs, bzw. dessen Ausrichtung am Ende einer Bewegung nach einer bestimmten Vorgabe zu ändern. Die Berechnung dieser Bewegungen und des Korridors, der die Bewegung umschließt, ermöglicht einerseits eine Planung der Fahrt, aber auch die Überwachung während der Fahrt.

2.3.1 Definition und Eigenschaften eines Manövers

Der elementare Bestandteil einer in der Arbeitsgruppe Echtzeitsysteme betrachteten Wegplanung wird als **Manöver** bezeichnet. Ein Manöver besteht immer aus den Fahrtkurven für bestimmte Punkte des Fahrzeugs und einer Fläche. Diese Fläche wird von einer Hülle umschlossen, die das Fahrzeug während der Bewegung nicht überschreitet. Manöver werden eingesetzt, um Fahrten eines Fahrzeugs zu planen, was in Bezug auf die kinematische Fahrbarkeit der Fahrkurven und zur Sicherheit für die Planung kollisionsfreier Wege eine wesentliche Rolle spielt.

Bei einem Manöver wird gefordert, dass das Fahrzeug am Anfang und am Ende der Bewegung gerade gerichtet ist. Das bedeutet, das Manöver wird an einer bestimmten Fahrzeugposition gestartet, wobei der Einknickwinkel zwischen Zugfahrzeug und Anhänger 0° beträgt. Im Zuge des Manövers kann nun eine Änderung

der Lage des Fahrzeugs beschrieben werden, wobei am Ende wieder ein Einknickwinkel von 0° gefordert wird. Innerhalb der Bewegung bewegen sich die Bezugspunkte auf den Achsen, die Kupplung, sowie weitere charakteristische Punkte auf **Pfaden**. Aus den kinematischen Beschränkungen eines Fahrzeugs lässt sich die Fahrbarkeit ableiten, die bei einem Manöver immer gewährleistet ist.

Ein weiterer wichtiger Bestandteil eines Manövers ist die Fläche, die die Bewegung des Fahrzeug während der Bewegung umschließt. Diese Fläche wird **Manöverkorridor** oder **Hülle** genannt und ist wichtig, um eine kollisionsfreie Planung der Wege vornehmen zu können [Weyand & Balcerak, 2009]. Da eine Fahrt durch technische Abweichungen in der Regel nicht genau gemäß der Planung abgefahren werden kann, kann es passieren, dass das Fahrzeug den **ausreichenden Manöverkorridor** verlässt. Dafür kann die Fläche um einen bestimmten Prozentsatz vergrößert werden. Mit diesem **sicheren Manöverkorridor** können diese Abweichungen in der Planung berücksichtigt werden.

Es können mehrere **Manövertypen** unterschieden werden. Zur Zeit werden die **Geradeausfahrt**, das **Wendemanöver** und das im Laufe dieser Arbeit visualisierte **Kurvenmanöver** betrachtet. Die Geradeausfahrt wird dabei in einer Phase beschrieben, in welcher der Lenkwinkel und Einknickwinkel 0° betragen, es wird also wirklich geradeaus gefahren. In einem Kurven- bzw. Wendemanöver ändert sich die Ausrichtung des Fahrzeugs um einen bestimmten Winkel, mit dem Unterschied, dass bei dem Wendemanöver eine Änderung der Fahrtrichtung vorgenommen wird, wobei diese beim Kurvenmanöver gleich bleibt.

Ein Manöver wird aufgeteilt in eine oder mehrere **Phasen** . Eine einzelne Phase wird im Zusammenhang dieser Arbeit definiert als eine Fahrt mit festem Lenkwinkel und einem bestimmten Einknickwinkel, der im Zuge dieser Phase erreicht werden muss und damit die Grenze einer Phase darstellt.

Eine Abstufung bei der Betrachtung eines Manövers sind **Methoden**, mit denen die Lenkwinkel während der Phasen geändert werden. In der **konventionellen** Methode geschehen Lenkwinkeländerungen an den Phasenübergängen. Dazu muss das Fahrzeug zur Änderung stehen bleiben, um in den neuen Zustand zu gelangen und dem vorgegebenen Pfad präzise zu folgen. Geschieht die Änderung während der Fahrt, wird der berechnete Weg verlassen, was ein Risiko bezüglich der Kollisionskontrolle darstellen kann. Deshalb gibt es eine Erweiterung dieser Methode, bei der der Lenkwinkel **iterativ** in kleinen Schritten verändert wird, bis der Ziellenkwinkel erreicht ist. Dabei müssen mehr Phasen beschrieben werden, als in der konventionellen Methode. Im vorigen Abschnitt wurde eine Phase als Fahrt mit fes-

tem Lenkwinkel definiert. Um diese Definition beizubehalten, werden die Phasen mit Lenkwinkeländerung in mehrere Teilphasen aufgeteilt, die auch **Iterationen** genannt werden.

Auch um einen Korridor eines Manövers zu konstruieren gibt es verschiedene Methoden. Es wird darauf eingegangen, einen Ring oder Ringausschnitt um das Manöver zu bilden. Dabei wird der minimale und der maximale Radius um einen bestimmten Punkt des Manövers gefunden und eine **Ringhülle** gebildet. Eine Alternative dazu ist es, um die einzelnen Teile eines Manövers Polygone zu bilden und diese nachher zu einer **Polygonhülle** zu vereinen.

Es wird nun auf die einzelnen Typen und Methoden eingegangen, wobei mit dem Kurvenmanöver begonnen wird.

2.3.2 Kurvenmanöver

Wenn man die Ausrichtung eines Gespanns um einen beliebigen Winkel Γ ändern möchte und dabei auch die Fahrtrichtung beibehält, handelt es sich um ein Kurvenmanöver. Dabei wird das Manöver in einer Fahrtrichtung, vorwärts oder rückwärts, durchfahren. Es gibt zwei Methoden ein Kurvenmanöver zu erzeugen. Beide Methoden beziehen sich darauf, dass es verschiedene Phasen während eines Manövers gibt. Diese Phasen sind entweder stabil oder instabil. Bei stabilen Phasen bewegen sich Zugfahrzeug und Anhänger auf konzentrischen Kreisbögen. Instabile Phasen werden dagegen benötigt, um aus dem gerade gerichteten Zustand einen stabilen Zustand zu erreichen oder umgekehrt. Bei der sogenannten konventionellen Methode geschieht eine Lenkwinkeländerung immer zwischen den Phasen. Eine Erweiterung ist die iterative Methode, bei der es Phasen gibt, in denen sich der Lenkwinkel inkrementell bis zum Ziellenkwinkel verändert und so ein Anhalten für die Lenkwinkeländerung nicht mehr nötig ist.

Als erstes wird die konventionelle Methode betrachtet. Die Bezeichner werden im Folgenden so gewählt, dass sich eine Unterscheidung der Phase am Index erkennen lässt, angefangen mit der ersten Phase und einem Index $i = 1$. Außerdem muss in manchen Betrachtungen unterschieden werden, ob auf einen Bezeichner am Anfang oder am Ende der Phase Bezug genommen wird. Der Anfang der Phase wird dann mit einem Index $i = 0$ gekennzeichnet, das Ende einer Phase mit $i = 1$. Eine Unterscheidung der Indizes ergibt sich dabei jeweils aus dem Zusammenhang.

2.3.2.1 Eigenschaften und Konstruktion eines Pfades nach der konventionellen Methode

Ein konventionelles Kurvenmanöver besteht aus drei Phasen. Dabei wird unterschieden, ob das Manöver vorwärts oder rückwärts gefahren werden soll. Im Fall der Rückwärtsfahrt ergeben sich folgende Phasen, in denen der Lenkwinkel jeweils gleich bleibt (vgl. [Weyand & Balcerak, 2009]):

1. In der ersten Phase startet das Manöver mit einem Einknickwinkel $\Delta\theta = 0°$. Das Fahrzeug bewegt sich rückwärts mit einem gegebenen Lenkwinkel α_1, bis sich der Einknickwinkel $\Delta\theta_{circ}$ (siehe 2.15) ergibt. Der zugehörige Lenkwinkel zu diesem Einknickwinkel bezieht sich dabei auf den Lenkwinkel, α_2, der in der zweiten Phase vorgegeben ist. Die Ausrichtungsänderung während dieser Phase beträgt $d\theta_1$.

2. Bei der zweiten Phase fahren Fahrzeug und Anhänger auf einer stabilen Fahrt mit dem aus der vorigen Phase erreichten Einknickwinkel $\Delta\theta_{circ}$ und dem zugehörigen Lenkwinkel α_2. Auch hier gibt es eine Ausrichtungsänderung um den Wert $d\theta_2$. Die Berechnung dieses Wertes ist ein wichtiger Teil der Manöverkonstruktion und wird im Anschluss der dritten Phase vorgestellt. Bei einer Rückwärtsfahrt ist darauf zu achten, dass das Vorzeichen des Lenkwinkels nicht dem der ersten Phase entspricht.

3. In der dritten und letzten Phase des Manövers wird das Gespann wieder gerade gerichtet. Der Betrag des Lenkwinkels, der in der zweiten Phase für eine stabile Fahrt benutzt wurde, muss dabei erhöht werden. Außerdem wird das Vorzeichen beibehalten. Der Lenkwinkel wird als α_3 bezeichnet. Die Änderung der Ausrichtung wird hierbei mit $d\theta_3$ bezeichnet. Das Fahrzeug befindet sich am Ende auf einer Geraden, die um den Winkel Γ zur Ursprungsgeraden liegt. Die Grenzen dieser Phase müssen vor der zweiten Phase berechnet werden.

Diese Reihenfolge ist nur für die Berechnung einer Rückwärtsfahrt gültig. Bei einer Vorwärtsfahrt wird diese einfach umgekehrt.

Zur Berechnung der stabilen, zweiten Phase ist es nötig, den Winkel $d\theta_2$, der in dieser Phase gefahren werden muss, zu berechnen. Dazu müssen die instabilen Phasen vorher berechnet werden. Diese sind nötig, um in einen vorgegebenen Einknickwinkel für die stabile Phase zu gelangen, oder von diesem Einknickwin-

kel wieder zu einem gerade gerichteten Fahrzeug. Die Länge dieser beiden Phasen muss mit dem gesamten gewünschten Winkel Γ verrechnet werden.

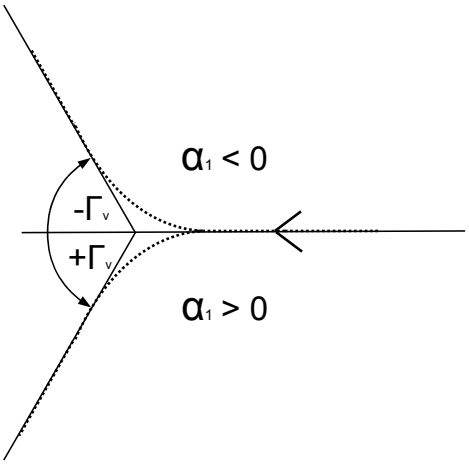


Abbildung 2.6: Winkel Γ bei Fahrtrichtung rückwärts

Der Winkel Γ kann aus verschiedenen Sichten betrachtet werden. Geht man von einer Startposition aus und will die Ausrichtung gegen den Uhrzeigersinn ändern, so ist der Winkel positiv, mit dem Uhrzeigersinn ist er negativ. Das Vorzeichen des Winkel Γ bestimmt auch das Vorzeichen der stabilen, zweiten Phase. In dieser Arbeit wurde mit dem Ansatz gearbeitet, dass der Winkel $\Gamma_v \in [0°, 360°]$ immer ohne Vorzeichen angegeben wird. Ob die Ausrichtung dann in positiver oder negativer Richtung geändert wird, ist von dem angegebenen ersten Lenkwinkel bestimmt. In Abbildung 2.6 kann man die beiden Fälle für Γ sehen. Ausgegangen wird auf der waagerechten Geraden in Richtung des Pfeiles. Bei einem negativen Lenkwinkel α_1 wird die Ausrichtung mit dem Uhrzeigersinn geändert, für den Winkel Γ gilt also $\Gamma = -\Gamma_v$. Ist der Lenkwinkel der ersten Phase $\alpha_1 > 0$, wird die Ausrichtung gegen den Uhrzeigersinn geändert, es gilt $\Gamma = +\Gamma_v$.

Wird ein Weg in Fahrtrichtung vorwärts berechnet, ändert sich das berechnete Manöver, wie in Abbildung 2.7 zu sehen ist. Auch hier startet das Fahrzeug auf der waagerechten Gerade, allerdings in die entgegengesetzte Richtung. Ist der Lenkwinkel der ersten Phase positiv, wird die Ausrichtung in einem Winkel Γ gegen den Uhrzeigersinn geändert, also ist $\Gamma = +\Gamma_v$. Im Fall, dass der Lenkwinkel α_1 negativ ist, ist auch Winkel $\Gamma = -\Gamma_v$.

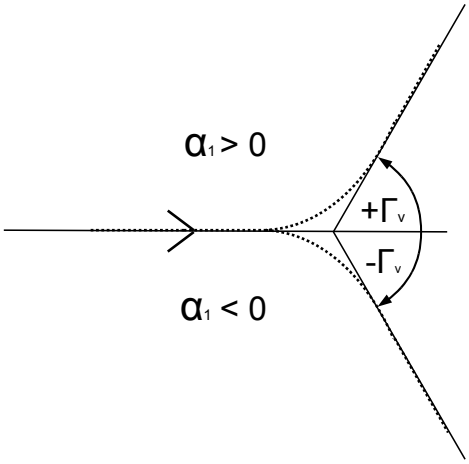


Abbildung 2.7: Winkel Γ bei Fahrtrichtung vorwärts

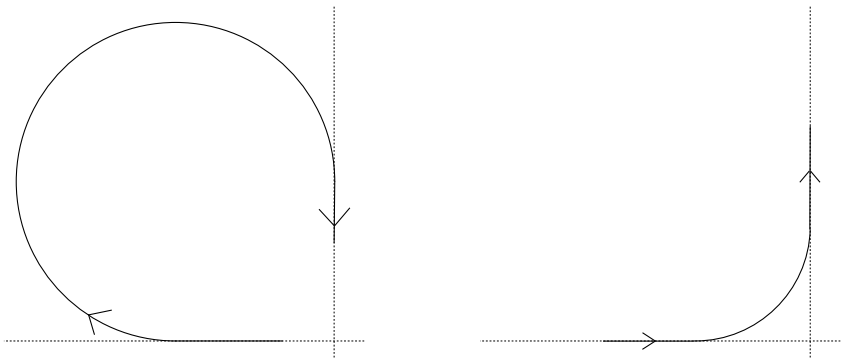
Die Unterscheidung des Winkels nach der Fahrtrichtung und dem ersten Lenkwinkel wird besonders deutlich, wenn man einen großen Winkel Γ in eine Richtung und den entsprechenden Winkel in die andere Fahrtrichtung vergleicht. In Abbildung 2.8 sind zwei Manöver zu sehen. Im Fall (a) bewegt sich das Fahrzeug rückwärts, in Fall (b) vorwärts. Betrachtet man nur die Ausrichtungen am Anfang und am Ende der Fahrten, sind die beiden Manöver gleich. Der Pfad im rechten Manöver ist allerdings deutlich länger, als der Pfad im linken Manöver, da im Fall (a) die Ausrichtung in einem grossen Winkel, hier 270°, geändert wird, im Fall (b) der Vorwärtsfahrt nur eine Ausrichtungsänderung von 90° vorgenommen wird.

Der Winkel Γ, anhand der gezeigten Fälle vorzeichenbehaftet, zusammen mit den Ausrichtungsänderungen der ersten und dritten Phase, sind die Grundlagen für die Berechnung der Ausrichtungsänderung, die in der stabilen Phase vorgenommen werden muss (vgl. [Weyand & Balcerak, 2009]):

$$d\theta_2 = \Gamma - d\theta_3 - d\theta_1 \tag{2.25}$$

Als Abschluss dieses Abschnitts soll nun eine Skizzierung der Berechnung eines Pfades für ein konventionelles Kurvenmanöver erfolgen. In dieser Auflistung geht es besonders um die Reihenfolge, die zu beachten ist:

1. Eingaben der Berechnung sind die Lenkwinkel, die in den drei Phasen gelten sollen, außerdem der Winkel Γ, um den sich die Ausrichtung des Fahrzeugs



(a) Fahrtrichtung rückwärts (b) Fahrtrichtung vorwärts

Abbildung 2.8: Schematische Darstellung einer Ausrichtungsänderung

ändern soll und die Fahrtrichtung vorwärts oder rückwärts. Schon jetzt kann eine erste Überprüfung der Werte erfolgen, ob mit diesen ein Manöver konstruiert werden kann:

(a) Die einfachste Überprüfung ist, dass die Lenkwinkel nicht 0° betragen dürfen. Eine Überprüfung des maximalen Lenkwinkels ist nicht nötig, da dieser in der Software nicht überschritten werden kann.

(b) Die weiteren Überprüfungen müssen je nach Fahrtrichtung betrachtet werden. Die Plausibilität kann aufgrund des Monotonieverhaltens, das in Abbildung 2.5 für eine Vorwärtsfahrt zu sehen ist und der damit verknüpften Eigenschaft, wie sich der Einknickwinkel, $\Delta\theta$, ändert, bestimmt werden:

   i. Bei der Rückwärtsfahrt müssen die Vorzeichen der Lenkwinkel der Phase 1 und Phase 2 unterschiedlich, die Vorzeichen der Lenkwinkel der Phase 2 und Phase 3 gleich sein. Außerdem muss der Lenkwinkel der Phase 2 vom Betrag her kleiner sein, als der der Phase 3.

   ii. Bei der Vorwärtsfahrt müssen die Vorzeichen der Lenkwinkel der Phasen 1 und 2 gleich sein, die der Phasen 2 und 3 hingegen unterschiedlich. Auch hier muss der Lenkwinkel der Phase 2 vom Betrag her kleiner sein als der Lenkwinkel der Phase 1.

2. Aus dem Lenkwinkel der zweiten Phase wird mit der Formel 2.15 der benötigte Einknickwinkel der stabilen Fahrt berechnet. Dieser Einknickwinkel

ist die Grundlage der Längen der Phasen 1 und 3, die im nächsten Schritt ermittelt werden. Der Einknickwinkel $\Delta\theta_{circ}$ ist gleichzeitig auch die Grenze der ersten Phase und der Start der letzten Phase. Die jeweils andere Grenze ist 0°, da das Fahrzeug am Anfang und am Ende gerade ausgerichtet ist.

3. Mithilfe der Formel 2.19 werden die Winkel v_0 und v_1 für die instabilen Phasen berechnet. Der Winkel v_0 beschreibt die relative Lage des Anhängers im Bezug auf den Winkel u am Anfang der Phase. Dazu wird der Starteinknickwinkel $\Delta\theta_0$ benötigt. Der Winkel v_0 gilt für das Ende der Phase. Hier wird der Endeinknickwinkel, $\Delta\theta_1$, benötigt. $\Delta\theta_0$ und $\Delta\theta_1$ sind je nach Phase der Einknickwinkel der stabilen Phase oder 0°.

4. Aus den Winkeln v_0 und v_1 und dem Lenkwinkel α lassen sich nun mit der Formel 2.20 die Werte des Winkels u_0 am Beginn und u_1 am Ende der instabilen Phasen 1 und 3 berechnen.

5. Jetzt kann mit der Formel 2.24 die Ausrichtung des Anhängers am Start und Ziel der Phase bestimmt werden. Die Differenz aus diesen Werten ergibt die Änderung der Ausrichtung $d\theta_i$ während dieser Phase.

6. Setzt man diese Werte nun in die Formeln 2.25, erhält man die Länge der Phase 2, der stabilen Fahrt, auf der sich alle Punkte des Gespanns um einen gemeinsamen Punkt drehen. Die Phasen 1 und 3 dienen also dazu, von dem gerade gerichteten Zustand in den stabilen Zustand zu gelangen bzw. vom stabilen Zustand wieder gerade gerichtet zu werden. Auch hier kann eine Überprüfung der Werte erfolgen, da die errechnete Länge größer als 0 sein muss.

7. Die Grenzen der stabilen Fahrt, die durch u-Winkel gegeben sind, lassen sich nicht mit der Rechnung 2.20 berechnen, diese ist nur für die instabile Fahrt gültig und würde im Falle der stabilen Fahrt einen unendlichen Betrag ergeben. Das liegt daran, dass die Traktrixkurve gegen die stabile Fahrt konvergiert, diesen Wert aber nie erreicht. Als Grenzen für diese Fahrt wird die vorher berechnete Länge eingesetzt, die den Winkel der Phase angibt.

8. Für die Darstellung der Punkte auf den Pfaden dienen die Formeln 2.5 bis 2.10 für das Zugfahrzeug bzw. 2.17 und 2.18 für den Anhänger. Die Kurve wird für die Visualisierung diskretisiert, wobei jeweils zwischen den berechneten Grenzen u_0 und u_1 iteriert wird. Hierzu wird auch der Winkel

$v \in [v_0, v_1]$ im Bezug auf das jeweilige $u \in [u_0, u_1]$ benötigt. Diesen Winkel kann man mit 2.21 oder 2.22 berechnen, wenn es sich um eine instabile Fahrt handelt, ansonsten mit der Formel 2.19. Die jeweilige Traktionsart kann nach dem in Abschnitt 2.2.3 dargestellten Schema bestimmt werden.

2.3.2.2 Konstruktion eines ringförmigen Manöverkorridors nach der konventionellen Methode

Der Manöverkorridor beschreibt die Fläche, die ein Fahrzeug während eines Manövers benötigt. Dabei sind verschiedene Gestalten des Korridors und damit unterschiedliche Konstruktionsmethoden denkbar. Während dieser Arbeit wird insbesondere eine ringförmige Korridorgestalt betrachtet. Diese Ringform ist als Skizze in Abbildung 2.9 zu sehen und bietet sich besonders für ein Kurvenmanöver an. Eine weitere Möglichkeit, die im Softwareentwurf berücksichtigt wird, ist eine Hülle die sich aus einer Menge einzelner, überlappender Polygone zusammensetzt. Diese Methode ist für beliebige Manövertypen anwendbar und wird in Kapitel 2.3.5 vorgestellt. Bei allen unterschiedlichen Formen soll die Fläche der Hülle möglichst klein sein. Es wird aber nicht gefordert, dass diese minimal ist, um eine effiziente Berechnung zu ermöglichen.

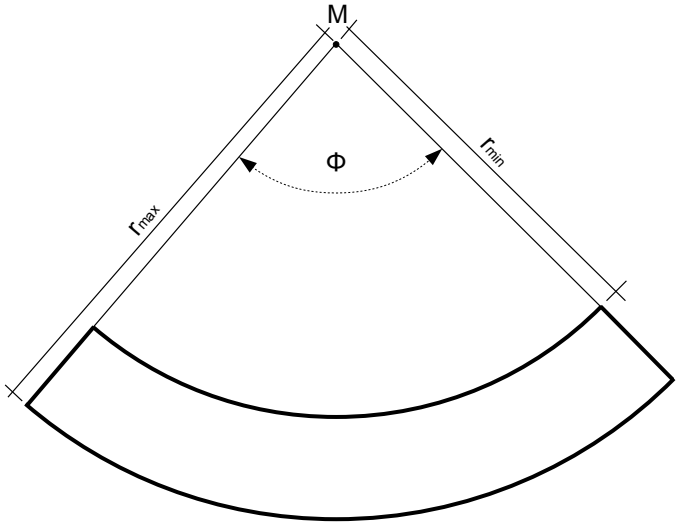


Abbildung 2.9: Skizze einer Ringhülle

Die Berechnung einer Ringhülle erfolgt, indem man zu bestimmten Zeitpunkten des Manövers bestimmte Extrempunkte betrachtet. Diese Extrempunkte können

die acht Ecken des Zugfahrzeugs und des Anhängers sein, ebenso die sechs Räder oder die Kupplung. Bei der Berechnung wird nicht die linke oder rechte Seite des Gespanns unterschieden, sondern ob sich der betrachtete Punkt auf der Seite des Fahrzeugs befindet, die dem Kurveninneren näher ist (als Nah bezeichnet) oder die weiter weg ist vom Mittelpunkt der Kurve (als Weit bezeichnet). Die Ecken des Zugfahrzeugs werden also als *zugVorneEckeNah*, *zugVorneEckeWeit*, *zugHintenEckeNah* und *zugHintenEckeWeit* bezeichnet, die Räder als *zugVorneRadNah*, *zugVorneRadWeit*, *zugHintenRadNah* und *zugHintenRadWeit*. Die vier Ecken des Anhängers heißen analog dazu *anhVorneEckeNah*, *anhVorneEckeWeit*, *anhHintenEckeNah* und *anhHintenEckeWeit*. Der Anhänger hat nur zwei Räder, nämlich *anhRadNah* und *anhRadWeit*.

Eine Ringhülle besteht aus zwei Kreisbögen, die den minimalen Radius, r_{min} und den maximalen Radius, r_{max}, eines Manövers begrenzen. Abgeschlossen wird der Teilring auch auf den Seiten durch bestimmte Punkte, die betrachtet werden müssen und die Kreisbögen in einem Winkel Φ begrenzen. Dieser Winkel wird Zentriwinkel oder Kreiswinkel genannt. Als Mittelpunkt, M, der Kreisbögen wird der Mittelpunkt der zweiten Phase angenommen. Auch die Mittelpunkte der beiden anderen Phasen sind für die Konstruktion relevant und werden als M_1, als Mittelpunkt der ersten Phase und M_3, als Mittelpunkt der dritten Phase, bezeichnet.

Die Extrempunkte, die für die Radien in Betracht kommen, müssen für jede Phase berechnet werden. Um die Berechnung möglichst effizient zu gestalten, werden für die beiden Radien der Ringhülle der verschiedenen Phasen nur die Punkte betrachtet, die auch wirklich zu dem jeweiligen Radius beitragen können. Der Kern der Berechnungen stammt aus [Weyand & Balcerak, 2009] und wurde während der Bearbeitungszeit der Diplomarbeit erweitert. Um zu bestimmen, welche Punkte jeweils für den minimalen und den maximalen Radius von Zugfahrzeug und Anhänger in Betracht kommen, werden im Folgenden die Phasen einzeln durchgegangen. Auf Grundlage dieser Punkte lassen sich Abstände berechnen, die als Kandidaten des jeweiligen Radius in Betracht kommen können. Um schließlich den minimalen Radius, r_{min} zu bestimmen, wird aus den in Frage kommenden Abständen das Minimum berechnet, analog dazu das Maximum für den maximalen Radius, r_{max}. Ohne Beschränkung der Allgemeinheit wird im Folgenden von einer Rückwärtsfahrt ausgegangen.

Phase 1

- Um den *minimalen* Radius des *Zugfahrzeugs* der ersten Phase zu bestim-

men, muss man zuerst den Abstand zwischen dem Mittelpunkt der Phase, M_1, und dem Mittelpunkt der Ringhülle, M, bestimmen. Dieser Abstand wird als h_1 bezeichnet. Danach berechnet man die Länge zwischen dem Phasenmittelpunkt, M_1, und folgenden Extrempunkten: die kurvenäußeren Räder des Zugfahrzeugs *zugVorneRadWeit*, h_v genannt, und *zugHintenRadWeit*, h_h, sowie die Kupplung, h_k. Der Abstand zu M_1 bleibt während der Phase an jeder Stelle gleich, da es sich um die Radien der Kreisbahnen für die bestimmten Punkte handelt. Von dem Abstand h_1 wird jeweils der Radius subtrahiert. Die Ergebnisse der Berechnungen sind die Kandidaten für den minimalen Radius.

- Für den *maximalen* Radius des *Zugfahrzeugs* in der ersten Phase werden die kurveninneren Ecken, *zugVorneEckeNah* und *zugHintenEckeNah*, an der Stelle u_0 betrachtet. Berechnet werden die Abstände zum Mittelpunkt der Ringhülle, M, wobei das Maximum der beiden Werte berücksichtigt wird.

- Das *Minimum* des *Anhängers* wird berechnet, indem der Abstand zwischen dem äußere Rad, *anhRadWeit*, an der Stelle u_0 und dem Mittelpunkt M berechnet wird.

- Den *maximalen* Radius des *Anhängers* erhält man, wenn man die Länge zwischen den inneren Ecken des Fahrzeugs an u_0 der Phase, *anhVorneEckeNah* und *anhVorneEckeNah*, und dem Mittelpunkt der Hülle, M, berechnet. Von den beiden Längen wird nur das Maximum weiter betrachtet.

Phase 2

In der zweiten Phase befindet sich das Fahrzeug auf einer stabilen Fahrt um den Mittelpunkt der Ringhülle, M. Betrachtet werden kann also im Folgenden eine beliebige Stelle, praktisch wird aber wieder auf die Stelle u_0 der zweiten Phase zurückgegriffen.

- Für den *minimalen* Radius von *Zugfahrzeug* und *Anhänger* während dieser Phase wird das Minimum der Längen zwischen Mittelpunkt M und dem inneren Rad von der Hinterachse des Zugfahrzeugs (*zugHintenRadNah*) bzw. dem Anhänger (*anhRadNah*) bestimmt.

- Hingegen werden beim *maximalen* Radius alle kurvenäußeren Ecken von *Zugfahrzeug* und *Anhänger*, also *zugVorneEckeWeit*, *zugHintenEckeWeit*, *an-*

hVorneEckeWeit und *anhHintenEckeWeit*, betrachtet. Das Maximum zwischen den Punkten und dem Mittelpunkt M stellt das Maximum dieser Phase dar.

Phase 3

- Den ersten Kandidaten für das *Minimum* des *Zugfahrzeugs* der dritten Phase findet man bei der Berechnung des Abstands zwischen dem hinteren kurveninneren Rad (*zugHintenRadNah*) an der Stelle u_1 und dem Mittelpunkt der Hülle, M. Darüber hinaus muss der Abstand zwischen der hinteren, inneren Ecke des Zugfahrzeugs, *zugHintenEckeNah*, an u_1 und dem Mittelpunkt der Hülle, M, betrachtet werden.

- Um den *maximalen* Radius für das *Zugfahrzeug* zu berechnen, wird als Hilfsgröße der Abstand zwischen dem Mittelpunkt der dritten Phase, M_3, und dem Mittelpunkt der Hülle, M, berechnet und als h_3 bezeichnet. Anschließend berechnet man den Abstand zwischen dem Mittelpunkt der Phase, M_3, und der vorderen äußeren Ecke (*zugVorneEckeWeit*) an der Stelle u_1, h_e genannt. Das gewünschte Maximum ist dann der Hilfsabstand h_3, addiert mit dem Abstand h_e. Außerdem wird auch der Abstand zwischen dem Mittelpunkt der Ringhülle, M, und dem Punkt der vorderen kurvenäußeren Ecke an der Stelle u_1 in die Berechnung des Maximums einbezogen.

- Das *Minimum* des *Anhängers* kann man finden, indem man zuerst die Sekanten durch die inneren Ecken, *anhVorneEckeNah* und *anhHintenEckeNah* an u_0 und u_1 bildet. Diese werden s_v, für die Sekante durch die vordere Ecke, und s_h, für die Sekante durch die hintere Ecke, genannt. Für die beiden Ecken wird danach der Abstand zwischen dem Mittelpunkt der Hülle, M, und den Sekanten, s_v und s_h, bestimmt. Da diese Phase außerdem einen Wendepunkt enthalten kann, muss auch dieser beachtet werden. Das nötige u_w für die Stelle des Einknickwinkels kann mit dem aktuellen Lenkwinkel und der Formel zur Berechnung von v mit dem Einknickwinkel, $\Delta\theta$, und α_{M_1}, Formel 2.19, ermittelt werden, wobei am Wendepunkt $\alpha_{M_1}(\alpha) = \Delta\theta$ gilt. Auch zu diesem Punkt wird der Abstand zum Mittelpunkt M berechnet. Das Minimum ergibt sich dann aus dem Minimum aller berechneten Abstände.

- Für den *maximalen* Radius des *Anhängers* berechnet man die Abstände zwischen dem Mittelpunkt der Hülle und den Extrempunkten an den kurven-

äußeren Ecken, *anhVorneEckeWeit* und *anhHintenEckeWeit* an der Stelle u_1 und ermittelt davon das Maximum.

Nach der Berechnung aller möglichen Abstände werden für den äußeren Radius der Hülle das Maximum aus allen Phasen von Zugfahrzeug und Anhänger gebildet, ebenso wird für den innere Radius das Minimum berechnet.

Mit den beiden Radien kann ein Ring gebildet werden, der im Mittelpunkt der zweiten Phase, M, seinen Ursprung hat und dessen Radien, r_{min} und r_{max}, das Manöver umschließen. Um daraus einen Teilring zu erhalten, muss ein Winkel, Φ, berechnet werden, der den Ring begrenzt. Dazu müssen wieder bestimmte Extrempunkte am Anfang der ersten Phase und am Ende der letzten Phase berücksichtigt werden. Diese Punkte sind mögliche Kandidaten der Begrenzungen. Man verbindet nun den Mittelpunkt, M, mit jeweils einem Punkt am Anfang der Phase 1, P_1, und einem Punkt am Ende der Phase 3, P_3. Daraus ergibt sich ein Winkel $\measuredangle(P_1, M, P_3)$. Folgende Punkte müssen betrachtet werden:

1. Die kurvenäußere vordere Ecke des Zugfahrzeugs *zugVorneEckeWeit* an der Stelle u_0 der ersten Phase im Bezug zu der kurvenäußeren hinteren Ecke des Anhängers *anhHintenEckeWeit* an der Stelle u_1 der letzten Phase.

2. Die kurvenäußere vordere Ecke des Zugfahrzeugs *zugVorneEckeWeit* an der Stelle u_0 der ersten Phase im Bezug zu der kurveninneren hinteren Ecke des Anhängers *anhHintenEckeNah* an der Stelle u_1 der letzten Phase.

3. Die kurveninnere vordere Ecke des Zugfahrzeugs *zugVorneEckeNah* an der Stelle u_0 der ersten Phase im Bezug zu der kurveninneren hinteren Ecke des Anhängers *anhHintenEckeNah* an der Stelle u_1 der letzten Phase.

4. Die kurveninnere vordere Ecke des Zugfahrzeugs *zugVorneEckeNah* an der Stelle u_0 der ersten Phase im Bezug zu der kurvenäußeren hinteren Ecke des Anhängers *anhHintenEckeWeit* an der Stelle u_1 der letzten Phase.

Das Maximum dieser Winkel bestimmt den Winkel des Ringausschnitts. Die bei diesem Maximum ermittelten Ecken werden als B_1 für die Begrenzung am Anfang der ersten Phase, und B_3 für die Begrenzung am Ende der letzten Phase bezeichnet. Man kann eine Gerade konstruieren, wobei gefordert ist, dass die Punkte B_1 und M auf dieser Geraden liegen. Zwischen dem minimalen und dem maximalen Radius der Ringhülle stellt diese Gerade die Begrenzung des Teilrings

auf der Seite des Manöveranfangs dar. Analog dazu wird am Ende des Manövers die Gerade betrachtet, auf denen die Punkte B_3 und M liegen.

Es kann bei einem großen Winkel $\Gamma \rightarrow 360°$ dazu kommen, dass Teile des Anhängers der Phase 3 Teile des Zugfahrzeugs der ersten Phase überlappen. Dann würde nach dem obigen Schema ein zu kleiner Radius für das Manöver berechnet werden. In diesem Fall soll daher der Ring geschlossen werden. Um diesen Fall zu identifizieren wird der Winkel Γ mit dem errechneten Winkel verglichen und der Ring vollständig konstruiert, wenn $\Gamma < \measuredangle(B_1, M, B_3)$.

Zu beachten ist, dass diese Vorgehensweise für die Bestimmung des Winkels nur für die Rückwärtsfahrt gilt, für ein vorwärts fahrendes Fahrzeug müssen die Phasen getauscht werden.

2.3.2.3 Eigenschaften und Konstruktion eines Pfades nach der iterativen Methode

Die zuvor gezeigte Methode hat den Nachteil, dass bei den Phasenübergängen eine abrupte Lenkwinkeländerung erfolgen muss. In der Praxis bedeutet dies, dass das Gespann stehen bleiben muss, um den Lenkwinkel zu ändern, wenn man präzises Fahren des Manövers voraussetzt. In einer idealen Lösung würde der Lenkwinkel kontinuierlich während der Fahrt geändert werden. Um eine planbare Lösung konstruieren zu können, gibt es die iterative Konstruktion eines Kurvenmanövers, wie sie in [Weyand & Balcerak, 2009] und [Weyand *et al.*, 2010] vorgestellt wird. Die Idee ist, den Lenkwinkel während einer Manöverphase inkrementell zu ändern. Geht man nach dieser Methode vor, gibt es nun fünf, statt drei, Phasen. Die Erweiterung der Phasen ist nötig, da jetzt nicht nur eine bestimmte Änderung des Einknickwinkels vorgenommen wird, sondern auch der Lenkwinkel während der Phasen geändert wird.

1. Auch hier startet das Fahrzeug im gerade gerichteten Zustand, d.h. mit einem Einknickwinkel $\Delta\theta = 0°$. Im Gegensatz zur konventionellen Methode wird mit einem Lenkwinkel von $\alpha = 0°$ gestartet. Während dieser Phase wird der Lenkwinkel inkrementell erhöht, bis der gewünschte Ziellenkwinkel erreicht wurde und damit die Phase beendet ist. Der erreichte Einknickwinkel während der Phase entspricht $\Delta\theta_{circ} - c$. Die Variable c beschreibt dabei, dass der gewünschte Einknickwinkel für die stabile Phase noch nicht erreicht wurde. Diese Variable ist vorzeichenbehaftet, je nachdem ob der aktuelle Einknickwinkel $\Delta\theta > \Delta\theta_{circ}$ oder $\Delta\theta < \Delta\theta_{circ}$ ist. Der aktuelle Einknickwinkel hat

hier $\Delta\theta_{circ}$ noch nicht erreicht.

2. Die zweiten Phase ist eine Übergangsphase und kürzer, als die erste Phase. Dabei muss der Einknickwinkel $\Delta\theta_{circ}$ erreicht werden. Gleichzeitig wird der Lenkwinkel geändert, um auf den diesem Einknickwinkel entsprechenden Lenkwinkel zu gelangen. Am Ende der Phase ist nun die Vorraussetzung für die stabile Fahrt geschaffen. Das heißt, das Fahrzeug hat den gewünschten Einknickwinkel der stabilen, nächsten Phase und den entsprechende Lenkwinkel nach der Formel 2.15 erreicht.

3. Die dritte Phase ist hierbei die stabile Phase. Der Lenkwinkel und der Einknickwinkel sind dabei in der aus Formel 2.15 beschriebenen Beziehung zueinander und alle Punkte bewegen sich auf konzentrischen Kreisbögen des Gespanns.

4. Die vierte Phase ist, wie die zweite Phase, eine kurze Übergangsphase. Dabei wird der Lenkwinkel inkrementell von dem Lenkwinkel der dritten Phase ab geändert, bis der Lenkwinkel der letzten Phase erreicht wird. Am Ende der Phase steht das Fahrzeug in einem Einknickwinkel von $\Delta\theta_{circ} - d$. Die Bedingungen an die Variable d sind dabei genau wie in der zweiten Phase. Die Variablen sind unterschiedlich gewählt, da die Parameter der einzelnen Phasen unabhängig voneinander sind.

5. Während der letzten Phase wird das Fahrzeug wieder gerade gerichtet. Der Lenkwinkel ändert sich dabei von dem Lenkwinkel der vierten Phase zu $0°$. Ebenso wird der Einknickwinkel während der Phase auf $0°$ geändert.

Wie bei der konventionellen Methode kann auch hier die Länge der stabilen Phase erst berechnet werden, wenn die Ausrichtungsänderungen der anderen Phasen ermittelt wurden. Um die Ausrichtungsänderung der dritten Phase zu berechnen, gilt folgende Formel:

$$d\theta_3 = \Gamma - d\theta_1 - d\theta_2 - d\theta_4 - d\theta_5 \tag{2.26}$$

Der Winkel Γ ist dabei, wie in Kapitel 2.3.2.1 gezeigt, je nach Fahrtrichtung und Wahl des ersten Lenkwinkels, vorzeichenbehaftet.

Bei der zweiten und vierten Phase ist zu beachten, dass die berechneten u- und v-Werte nur für die instabile Fahrt gelten. Das Monotonieverhalten wurde schon anhand Abbildung 2.5 gezeigt. Die Änderung des Einknickwinkels verhält sich

monoton und konvergiert während einer Vorwärtsfahrt gegen den Einknickwinkel der stabilen Fahrt, $\Delta\theta_{circ}$. Das Monotonieverhalten besagt aber auch, dass sich der Einknickwinkel zwar $\Delta\theta_{circ}$ beliebig nähert, diesen aber nie ganz erreicht. Ebenso verhält es sich für den gerade gerichteten Zustand mit $\Delta\theta_{circ} = 0°$. Die Formel 2.20 ist nicht definiert für die stabile Fahrt. Daher ist es wichtig, die Phasen nicht direkt für den stabilen Einknickwinkel zu berechnen, sondern einen kleinen Abstand zu $\Delta\theta_{circ}$ einzuhalten (siehe [Weyand *et al.*, 2010]).

Freiheitsgrade bei der Wahl der Inkremente

Es sind zwei Freiheitsgrade im Bezug auf die Inkremente zu beachten. Der erste ist die Größe der Lenkwinkeländerung $d\alpha$, mit der die Anzahl der Iterationen bestimmt wird. Umgekehrt kann auch mithilfe der Anzahl der Iterationen auf die Lenkwinkeländerung geschlossen werden. Der zweite Freiheitsgrad ist die Änderung des Einknickwinkels $d(\Delta\theta)$.

Es ist ein weiterer Ansatz denkbar, bei dem die Größe der Inkremente variabel sind. Dies wird in Abhängigkeit dazu gesetzt, ob sich der Einknickwinkel nahe des stabilen Einknickwinkels befindet oder nicht. In der Nähe des stabilen Einknickwinkels sollen die Inkremente dabei kleiner werden, wobei sie in weiterer Entfernung von diesem Wert größer sein können. Dieser Ansatz wird gegenüber den festen Inkrementen favorisiert.

Betrachtung des Monotonieverhaltens

Im Kapitel 2.2.3 wurde schon darauf eingegangen, dass die Änderung des Einknickwinkels bei einer vorwärts gerichteten Bewegung mit einem festen Lenkwinkel monoton ist und gegen den Einknickwinkel der stabilen Fahrt ($\Delta\theta_{circ}$) konvergiert. Mit diesem Verhalten wird bei der konventionellen Methode die gewünschte Entwicklung während der Phasen garantiert. Da sich die Lenkwinkel bei Betrachtung der iterativen Methode auch während der Phasen ändert, muss noch genauer auf das Monotonieverhalten der Einknickwinkeländerung eingegangen werden.

Wenn man eine Traktrixkurve betrachtet, beeinflussen verschiedene Kriterien das Monotonieverhalten der Einknickwinkeländerung eines Fahrzeugs. Als erstes sind die Vorzeichen der Anfangs- und Endlenkwinkel einer Phase zu nennen. Man muss Bedingungen beachten, die an den Einknickwinkel geknüpft sind, bezogen auf den Start und das Ende einer Phase. Diese Bedingungen für den Starteinknickwinkel, $\Delta\theta_0$, beziehen sich darauf, ob der Einknickwinkel positiv oder negativ ist, und ob er kleiner oder größer als der Einknickwinkel der stabilen Fahrt,

$\Delta\theta_{circ}$ ist. Auch für den Endeinknickwinkel, $\Delta\theta_1$, kann der Vergleich mit dem stabilen Einknickwinkel relevant sein. Außerdem kommen hier die Einknickwinkel in Betracht, die sich bei einem Winkel $u = 0°$ ergeben. Diese Winkel sind in Abbildung 2.2 im Fall (a) zu sehen und sind abhängig von der Traktionsart. Bei einer inneren Traktion wird der Winkel $\Delta\theta_{i_0}$ betrachtet, bei einer äußeren Traktion der Winkel $\Delta\theta_{i_0}$. Die Art, wie sich der Einknickwinkel ändern soll, wird durch das Vorzeichen von $d(\Delta\theta)$ vorgegeben und spielt eine weitere Rolle für die Geschwindigkeit des Monotonieverhaltens. Der letzte Punkt ist die Fahrtrichtung der Phase. Die Traktionsart ist, wie schon vorher gezeigt, kein freier Parameter, sondern ergibt sich aus den vorher genannten Faktoren.

Aus diesen Kriterien werden in [Weyand & Balcerak, 2009] alle möglichen Fälle betrachtet und daraus zwei wichtige Schlussfolgerungen für das Verhalten gezogen:

1. Wenn sich das Vorzeichen des Lenkwinkels während einer Phase nicht ändert und die Traktionsart gleich bleibt, verhält sich die Änderung des Einknickwinkels monoton.

2. Wenn sich das Vorzeichen des Lenkwinkels und die Traktionsart ändern, verhält sich die Änderung des Einknickwinkels ebenfalls monoton.

2.3.2.4 Konstruktion eines ringförmigen Manöverkorridors nach der iterativen Methode

In Kapitel 2.3.2.2 wurde die Konstruktion eines ringförmigen Korridors bei der konventionellen Methode vorgestellt. Diese bezieht sich auf Extrempunkte, die bei den Phasen unterschiedlich betrachtet werden. Auch bei der iterativen Methode wird so vorgegangen, um den Ring zu konstruieren. Jedoch ändern sich durch den anderen Manöveraufbau die Konstruktionsschritte für den Korridor. Es gibt insbesondere mehrere mögliche Kandidaten für die beiden Radien r_{min} und r_{max}, da Wendepunkte auftreten können, die in der letzten Phase untersucht werden müssen. Als Mittelpunkt des Rings wird dabei wieder der Mittelpunkt der Kurven der stabilen, dritten Phase verwendet.

Die Konstruktion ändert sich im Bezug auf das konventionelle Manöver. Das Vorgehen ist in [Weyand & Balcerak, 2009] zu finden und wird im Folgenden für die verschiedenen Phasen vorgestellt.

Phase 1 und 2

Während den ersten beiden Phasen gibt es eine Unterscheidung der Grenzen zwischen dem Zugfahrzeug und dem Anhänger. Während bei dem Anhänger die definierten Phasen betrachtet werden können, unterteilt man die ersten beiden Phasen des Zugfahrzeugs in zwei Bereiche, die für jeden Kandidaten betrachtet werden müssen. Diese Bereiche teilen sich auf in die Lenkwinkeländerung von $0°$ nach α_{max} und wieder zurück nach $0°$, was den ersten Bereich der Phasen darstellt. Die Änderung von $0°$ nach α_{circ}, dem Lenkwinkel, der zur stabilen Fahrt benötigt wird, ist der zweite Bereich. Die Stellen u_i, an denen die Kandidaten berechnet werden müssen, sind auch auf diese Bereiche bezogen, wobei u_0 den Startwert der ersten Iteration und u_1 den Endwert der letzten Iteration des Bereiches darstellen.

1. Der *minimale* Radius des ersten Bereichs des *Zugfahrzeugs* der ersten und zweiten Phase muss dadurch bestimmt werden, dass man die *kurvenäußeren* Eckpunkte, nämlich *zugVorneEckeWeit* und *zugHintenEckeWeit* betrachtet. Hier werden die Tangenten an u_0 und u_1 gelegt, wodurch ein Schnittpunkt entsteht. Der Schnittpunkt zwischen den Tangenten der vorderen Ecke wird S_v genannt, der Schnittpunkt der hinteren Ecke S_h. Jetzt bestimmt man jeweils eine Strecke zwischen den Schnittpunkten, S_v, bzw. S_h, und den Kurvenpunkten der beiden Ecken an der Stelle u_1, P_v für die vordere Ecke und P_h für die hintere Ecke. Von den Strecken $\overline{S_vP_v}$, bzw. $\overline{S_hP_h}$ wird nun der Abstand zum Mittelpunkt M des Rings betrachtet. Das Minimum der Abstände stellt den minimalen Radius dieses Bereichs dar.

   Im zweiten Bereich werden Sekanten durch u_0 und u_1 an den Kurven der kurveninneren Räder - *zugVorneRadNah* und *zugHintenRadNah* - gelegt und dann die Abstände der Sekanten zum Mittelpunkt M betrachtet. Auch hier wird das Minimum der Abstände berücksichtigt.

2. Um den *maximalen* Radius des *Zugfahrzeugs* für den ersten Bereich zu bestimmen, muss eine Betrachtung der inneren Räder (*zugVorneRadNah*, bzw. *zugHintenRadNah*) und der *Kupplung* erfolgen. Als Erstes legt man Sekanten durch die Kurven an den Stellen u_0 und u_1. Als Maximum dieses Bereiches sind dann jeweils die Abstände zwischen den Sekanten und dem Mittelpunkt der Ringhülle M zu betrachten.

   Der zweite Bereich ist bestimmt durch die äußeren Ecken des Zugfahrzeugs, *zugVorneEckeWeit* und *zugHintenEckeWeit*, und der Kupplung. Dazu werden für jeden dieser Bezugspunkte Tangenten durch u_0 und u_1 dieses Bereichs

gelegt. Für jeden Bezugspunkt entsteht ein Schnittpunkt der beiden zugehörigen Tangenten, der mit dem Mittelpunkt, M, verbunden wird. Diese Abstände bilden weitere Kandidaten für das Maximum.

3. Auch der *Anhänger* kann Kandidaten für das *Minimum* der beiden ersten Phasen haben. Für diese werden an den Stellen u_0 und u_1 Sekanten durch die Kurven des inneren Rads *anhRadNah* gelegt und anschließend der Abstand der Sekanten zum Mittelpunkt M betrachtet. Die Aufteilung in Bereiche ist für den Anhänger nicht nötig, bei der Betrachtung des Anhängers ist u_0 der Anfangswinkel der ersten Phase und u_1 der Endwinkel der zweiten Phase.

4. Der *maximale* Radius des *Anhängers* errechnet sich, indem man Tangenten an die Stellen u_0 und u_1 der kurvenäußeren Eckpunkte des Anhängers, *anhVorneEckeWeit* und *anhHintenEckeWeit* legt, die Schnittpunkte bestimmt und das Maximum der beiden Abstände zum Mittelpunkt M ermittelt.

Phase 3

Bei dieser Phase handelt es sich um die stabile Fahrt. Hierbei bewegen sich alle Punkte um den Mittelpunkt der Ringhülle M. Es gibt hier keinen Unterschied zu der stabilen, zweiten Phase des konventionellen Manövers, daher können auch die Minima und Maxima der Ringhülle nach der auf Seite 26 beschriebenen Vorgehensweise ermittelt werden.

Phasen 4 und 5

Im letzten Teil des Manövers kann es am Ende der fünften Phase zu mehreren Wendepunkten kommen, die betrachtet werden müssen. Trotzdem kann innerhalb eines Iterationsschritts, genau wie bei der konventionellen Methode, nur ein Wendepunkt entstehen, da sich während einer Iteration der Lenkwinkel nicht ändert. Wenn die folgende Ungleichung erfüllt ist, gibt es einen Wendepunkt an der Stelle u_w:

$$|u_0| \leq |u_w| \leq |u_1| \tag{2.27}$$

Der Winkel $u_w = u(v_w)$ kann analog zum konventionellen Manöver berechnet werden. Das benötigte v_w erhält man mit der Formel 2.19. Dabei gilt $\Delta\theta = -\alpha_{M_1}$.

Die einzelnen Kandidaten der vierten und fünften Phase können folgendermaßen ermittelt werden:

1. Das *Minimum* des *Zugfahrzeugs* wird von den kurveninneren Rädern, *zugVorneRadNah* und *zugHintenRadNah* bestimmt. Hier werden wieder die Sekanten für die beiden Kurven durch u_0 und u_1 gelegt und der Abstand zum Mittelpunkt der Ringhülle berechnet.

2. Das *Maximum* des *Zugfahrzeugs* in den letzten beiden Phasen hat keinen Einfluss auf das absolute Maximum des Manövers.

3. Bei der Berechnung des *minimalen* Radius des *Anhängers* wird die Kurve des inneren Rades (*anhRadNah*) betrachtet. Als erstes wird eine Sekante durch die Stellen u_0 und u_1 gelegt und der Abstand zum Mittelpunkt berechnet. Darüber hinaus gibt es mindestens einen Wendepunkt, der betrachtet werden muss. Von allen Wendepunkten dieser Kurve muss der Abstand zum Mittelpunkt bestimmt und bei der Auswahl des Minimums berücksichtigt werden.

4. Für das *Maximum* des *Anhängers* werden die inneren Eckpunkte betrachtet. An der Stelle u_1 wird jeweils der Abstand zum Mittelpunkt bestimmt und das Maximum davon ermittelt.

Die Hülle wird gegebenenfalls wieder durch einen bestimmten Winkel begrenzt. Dieser berechnet sich aus bestimmten Kandidaten der ersten und der letzten Phase. Das Vorgehen ist dabei genauso wie bei der konventionellen Methode und kann in Kapitel 2.3.2.2 nachgelesen werden.

2.3.3 Wendemanöver

Während eines Wendemanövers wird die Ausrichtung eines Fahrzeugs um einen bestimmten Winkel geändert. Der Vorteil eines Wendemanövers ist, dass für das Manöver eine kleinere Fläche benötigt wird, weil es genau eine Fahrtrichtungsänderung während des Manövers gibt. Verschiedene Ausprägungen eines Wendemanövers können in Erwägung gezogen werden.

Die Ansätze dazu sind in [Weyand & Balcerak, 2010] zu finden. Hier wird daraus eine Methode beschrieben, in der fünf Phasen für das Manöver benötigt werden.

Die Bedingungen, die an ein Kurvenmanöver gestellt werden, nämlich die gerade gerichtete Start- und Endausrichtung, gelten auch für das Wendemanöver. Die fünf Phasen dieses Typs werden hier anhand des konventionellen Ansatzes erläutert. Diese Phasen sind anhand der Betrachtung des Pfads für den Mittelpunkt der

Anhängerachse in Abbildung 2.10 zu sehen. Die instabilen Phasen sind dicker gezeichnet, als die stabilen Phasen. Bei der Bezeichnung der Phasen wird dabei davon ausgegangen, dass zuerst eine Bewegung in Fahrtrichtung rückwärts erfolgt. Eine Vorwärtsfahrt am Anfang des Manövers muss in umgekehrter Reihenfolge betrachtet werden. Es ergeben sich folgende Phasen:

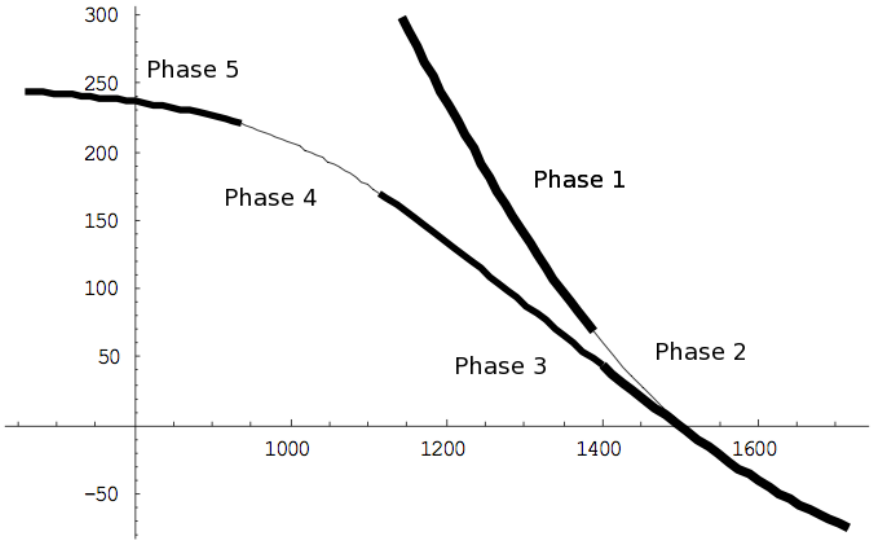


Abbildung 2.10: Mittelpunkt der Anhängerachse bei einem Wendemanöver [Weyand & Balcerak, 2010]

1. Die erste Phase beginnt mit der Fahrtrichtung rückwärts. Hierbei geht es darum, aus dem gerade gerichteten Zustand mit dem Einknickwinkel $\Delta\theta = 0°$ in den Einknickwinkel der stabilen Fahrt zu gelangen. Dazu muss der Lenkwinkel, α_1, ein anderes Vorzeichen haben, als der Lenkwinkel der zweiten Phase, α_2. Ist der Einknickwinkel der berechnete stabile Einknickwinkel der nächsten Phase, $\Delta\theta_2$, erreicht, wird zu Phase 2 übergegangen. Während dieser Phase wird eine Ausrichtungsänderung von $d\theta_1$ vorgenommen.

2. In der zweiten Phase bewegen sich Fahrzeug und Anhänger um den gleichen Mittelpunkt. Es handelt sich dabei also um die stabile Fahrt. Der Einknickwinkel bleibt während dieser Phase unverändert und die Ausrichtungsänderung, $d\theta_2$, die während dieser Phase erfolgt, kann analog zum Kurvenmanöver erst berechnet werden, wenn alle instabilen Phasen berechnet wurden.

3. Die dritte Phase beginnt mit einem Fahrtrichtungswechsel. Hierbei gibt es

eine instabile Fahrt, bei weclher der Einknickwinkel auf die stabile Fahrt der nächsten Phase geändert wird und sich während der Phase um $d\theta_3$ ändert. Der Einknickwinkel ändert sich dabei über den Nullpunkt hinaus und erreicht schließlich $\Delta\theta_4$, was dem stabilen Einknickwinkel der vierten Phase entspricht.

4. Für den zuvor erreichten Einknickwinkel, $\Delta\theta_4$, wird der Lenkwinkel, α_4, so angepasst, dass in dieser Phase eine stabile Fahrt in Vorwärtsrichtung gefahren wird. Auch hier kann die Ausrichtungsänderung, $d\theta_4$, erst nach den Ausrichtungsänderungen der instabilen Phase berechnet werden.

5. Die letzte, instabile Phase richtet das Fahrzeug wieder gerade. Der Lenkwinkel dieser Phase muss sich im Vorzeichen von der vorigen Phase unterscheiden. Der Einknickwinkel ändert sich von $\Delta\theta_4$ zu $0°$, die Ausrichtungsänderung wird als $d\theta_5$ bezeichnet.

Auch hier müssen die instabilen Phasen, die Phasen 1, 3 und 5, zuerst berechnet werden, um die Länge der stabilen Phasen zu berechnen. Anders als bei dem Kurvenmanöver gibt es aber zwei stabile Phasen, die zweite und die vierte. Die Berechnung der Länge erfolgt, analog zur Formel 2.25, mit:

$$d\theta_2 + d\theta_4 = \Gamma - d\theta 1 - d\theta 3 - d\theta 5 \tag{2.28}$$

Die Länge des Pfades, die benötigt wird, um eine Einknickwinkeländerung vorzunehmen, hängt von der Art der Änderung und der Fahrtrichtung ab. Bei einer Vorwärtsfahrt ist der Pfad deutlich länger, wenn man den Einknickwinkel der stabilen Fahrt erreichen will, als wenn man das Gespann gerade richten möchte (wenn man von einer vom Betrag her gleichen Einknickwinkeländerung $d\theta$ ausgeht). Die Rückwärtsfahrt verhält sich komplementär. Daher sollte in diesem Beispiel die erste stabile Phase länger sein als die zweite. Dieses Vorgehen führt zu einer kleineren Manöverfläche.

2.3.4 Geradeausfahrt

Als letzter Manövertyp wird hier auf die Geradeausfahrt eingegangen. Bei diesem Typ gibt es nur eine Phase, in der der Einknickwinkel und der Lenkwinkel während der gesamten Fahrt $0°$ betragen. Daraus ergibt sich eine veränderte Position am Ende des Manövers mit gleicher Ausrichtung.

Mathematisch ergibt sich dabei für die Kurve des Anhängers der Spezialfall $\alpha = 0°$ für $\Delta\theta = 0°$, der in Kapitel 2.2.2 erläutert wird.

2.3.5 Alternative Konstruktion eines Manöverkorridors

Im Kapitel 2.3.2 wurde auf die Erstellung eines ringförmigen Manöverkorridors eingegangen. Es gibt noch weitere Ansätze, die eine effektivere Fläche, aber auch eine allgemeinere Anwendung erlauben, zum Beispiel die Betrachtung einer Polygonhülle, die in [Weyand, 2010] vorgestellt wird.

Im Gegensatz zur Konstruktion als Ringausschnitt werden für jede Phase Polygonhüllen gebildet und anschließend aus der Vereinigung der einzelnen Polygonen der Phasen ein Manöverkorridor gebildet. Eine Phase wird in diesem Abschnitt nicht nur dadurch definiert, dass Lenkwinkel und Fahrtrichtung gleich bleiben, es wird auch die Änderung des Einknickwinkels betrachtet. Der Einknickwinkel ändert sich nicht über die Nullstellung hinweg.

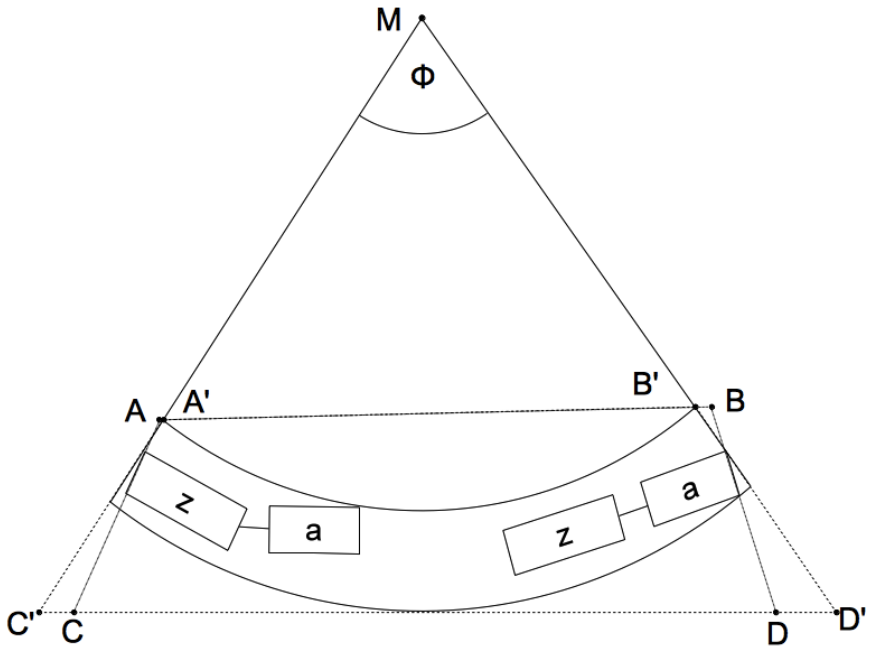


Abbildung 2.11: Konstruktion einer Polygonhülle um eine Phase

Im ersten Schritt wird für jede Phase ein Trapez T' mit den Eckpunkten A, B', C' und D' berechnet (vgl. Abbildung 2.11), bei dem sich die Fahrkurven aller Fahrzeugpunkte während dieser Phase innerhalb dieses Trapezes befinden.

Der erste Punkt, der für die Konstruktion benötigt wird, ist der Mittelpunkt M, um den sich das Zugfahrzeug während dieser Phase bewegt. Danach wird analog zur Ringhülle ein minimaler und ein maximaler Radius bestimmt. Aus den Radien könnten auch Kreisbögen erstellt werden. Der Winkel Φ zwischen den beiden Kreisbögen wird dabei genauso bestimmt, wie es bei der Konstruktion einer Ringhülle in Kapitel 2.3.2.2 beschrieben wurde. Es werden bei dieser Konstruktion zwei Punkte gefunden, die in Relation zum Mittelpunkt betrachtet den maximalen Winkel der Phase beschreiben. Der Punkt am Anfang der Phase wird dabei als R_0 bezeichnet, der Punkt am Ende der Phase R_1.

Es müssen für die minimalen und maximalen Radien nicht alle Extrempunkte berechnet werden, sondern nur die Kandidaten, die in Frage kommen. Für die Berechnung werden die Stellen u_{min} und u_{max} benötigt, denen entweder u_0 oder u_1 zugewiesen wird. Die Variablen u_{min} und u_{max} beziehen sich dabei auf die beiden Enden des betrachteten Kurvenausschnitts. Die Stelle u_{min} stellt dabei das Ende mit dem minimalen Abstand zum Mittelpunkt M dar. Im Gegensatz dazu ist der Abstand an der Stelle u_{max} maximal. Um zu bestimmen, welcher Wert minimal und welcher maximal ist, muss zwischen der inneren und der äußeren Traktion unterschieden werden. Dabei gilt für die innere Traktion:

- $|u_0| < |u_1| : u_{min} = u_0, u_{max} = u_1$
- $|u_0| > |u_1| : u_{min} = u_1, u_{max} = u_0$

Bei der äußeren Traktion gilt umgekehrt:

- $|u_0| < |u_1| : u_{min} = u_1, u_{max} = u_0$
- $|u_0| > |u_1| : u_{min} = u_0, u_{max} = u_1$

Eine weitere Unterscheidung ist, ob eine Phase konvex oder konkav ist. Konvex ist sie, wenn Lenkwinkel und der Einknickwinkel während der Phase ein gleiches Vorzeichen haben, anderenfalls ist sie konkav. Bei beiden Phasentypen werden wieder bestimmte Kandidaten für den minimalen und den maximalen Radius berechnet und anschließend das Minimum, bzw. Maximum bestimmt. Die Phasentypen unterscheiden sich im Hinblick auf die Kandidaten, die betrachtet werden müssen.

Die Punkte des Zugfahrzeugs ändern während einer Phase nicht den Abstand zum Mittelpunkt. Bei dem Anhänger ändern sich jedoch die Abstände im Fall der instabilen Fahrt. Dabei wird ein weiterer Winkel benötigt, der zwischen der Kupplung und den Punkten zum Mittelpunkt der Achse und zum hinteren Eckpunkt gespannt wird. Dieser wird β genannt und wie folgt berechnet:

$$\beta = \arctan\left(\frac{ba}{bah + L_2}\right) \tag{2.29}$$

Die Variable ba beschreibt dabei die Hälfte der Breite des Anhängers und bah den Abstand zwischen der Achse des Anhängers und der hinteren Kante, also den hintere Überhang.

Für eine konvexe Phase müssen folgende Kandidaten betrachtet werden:

- Der *minimale* Radius des *Zugfahrzeugs* wird bestimmt, indem man den Abstand zwischen dem Mittelpunkt M und dem kurveninneren Hinterrad (*zugHintenRadNah*) an der Stelle u_{min} berechnet.

- Für den *Anhänger* kommen folgende Punkte in Frage: Die beiden kurveninneren Ecken, *anhVorneEckeNah* und *anhHintenEckeNah* und das kurveninnere Rad *anhRadNah*. Von diesen wird der Abstand zum Mittelpunkt an der Stelle u_{min} berechnet.

  Bei einer inneren Traktion müssen weitere Berechnungen vorgenommen werden, da sich die Eckpunkte des Anhängers während einer Phase im Bezug auf den Mittelpunkt nicht monoton verhalten. Bei der konvexen Phase wird überprüft, ob der Einknickwinkel $\Delta\theta_n = -sign(\alpha)*\Pi/2-\alpha_{M_1}(\alpha)-sign(\alpha)*\beta$ überschritten wird. In diesem Fall wird folgende Berechnung als Minimumkandidat betrachtet, wobei $|MK|$ der Abstand zwischen dem Mittelpunkt und der Kupplung ist:

  $$\text{Min}_{pos} = \sqrt{(L_2 + bah)^2 + (ba)^2} - |MK| \tag{2.30}$$

- Der *maximalen* Radius des *Zugfahrzeugs* wird ermittelt, indem die Abstände zwischen den kurvenäußeren Ecken (*zugHintenEckeWeit* und *zugVorneEckeWeit*), bzw. der Kupplung und dem Mittelpunkt an der Stelle u_{max} berechnet werden.

- Um den *maximalen* Radius für den *Anhänger* zu bestimmen, werden die Abstände zwischen den kurvenäußeren Eckpunkten (*anhHintenEckeWeit* und

anhVorneEckeWeit) und dem Mittelpunkt an der Stelle u_{max} berechnet. Unter der Annahme, dass $r_2(\alpha) > L_2$, müssen die kurveninneren Eckpunkte nicht betrachtet werden.

Im Fall einer konkaven Phase ändern sich die Kandidaten wie folgt:

- Für das *Minimum* des Zugfahrzeugs berechnet man den Abstand zwischen dem kurveninneren Hinterrad an der Stelle u_{min}, *zugHintenRadNah*, und dem Mittelpunkt M.
- Der *Anhänger* hat als Kandidaten für das *Minimum* die Abstände zwischen dem Mittelpunkt und den vorderen Ecken, *anhVorneEckeNah* und *anhVorneEckeWeit*, an der Stelle u_{min}.
- Für das *Maximum* muss nur der *Anhänger* betrachtet werden: Einmal an der Stelle u_{max} die hinteren Eckpunkte, *anhHintenEckeNah* und *anhHintenEckeWeit* und deren Abstand zum Mittelpunkt.

  Hier muss eine weitere Berechnung erfolgen, wenn die Bedingung erfüllt ist, dass $\Delta\theta_w = sign(\alpha) * \Pi/2 - \alpha_{M_1}(\alpha) - sign(\alpha) * \beta$ überschritten wird und eine äußere Traktion vorliegt. Dann muss folgender Abstand berechnet werden:

$$\text{Max}_{pos} = \sqrt{(L_2 + bah)^2 + (ba)^2} + |MK| \tag{2.31}$$

  Analog zum Minimum des Anhängers in der konvexen Phase ist auch hier $|MK|$ der Abstand zwischen Mittelpunkt und Kupplung.

Aus allen Abständen, die für den minimalen Radius in Betracht kommen, wird jetzt das Minimum gebildet, was dann r_{min} entspricht. Ebenso wird das Maximum aus den Abständen für den maximalen Radius r_{max} gebildet.

Die vier Punkte des Trapez können nun bestimmt werden. Dazu ist der Abstand, den ein Punkt zum Mittelpunkt M hat, relevant. Dieser ist im Fall des Minimums:

$$S_{min} = \frac{r_{min}}{\cos\left(\frac{\Phi}{2}\right)} \tag{2.32}$$

Für das Maximum:

$$S_{max} = \frac{r_{max}}{\cos\left(\frac{\Phi}{2}\right)} \tag{2.33}$$

- Der Punkt A' liegt auf der Geraden durch M und R_0. Der Abstand zum Mittelpunkt ist S_{min}.
- Der Punkt B' liegt auf der Geraden durch M und R_1. Der Abstand zum Mittelpunkt ist S_{min}.
- Der Punkt C' liegt auf der Geraden durch M und R_0. Der Abstand zum Mittelpunkt ist S_{max}.
- Der Punkt D' liegt auf der Geraden durch M und R_1. Der Abstand zum Mittelpunkt ist S_{max}.

Aus den Erfahrungen der Arbeitsgruppe Echtzeitsysteme hat sich ergeben, dass der Flächeninhalt des Trapezes um die Punkte A, B, C und D kleiner ist, als das zuvor ermittelte Trapez. Um dieses Trapez zu berechnen, wird an der Stelle u_0 eine Gerade, g_0 durch die vorderen Eckpunkte des Fahrzeugs gelegt. Am Ende der Phase, u_1, wird analog eine Gerade g_1 durch die hinteren Eckpunkte des Anhängers gelegt.

Auch die Punkte A' und B' werden als Punkte einer Geraden g_{AB} betrachtet. Der Schnittpunkt zwischen der Geraden g_0 und der Geraden g_{AB} ist der Punkt A des Trapezes, der Schnittpunkt der Geraden g_1 und der Geraden g_{AB} ist der Punkt B.

Für die Punkte C und D wird eine Gerade g_{CD} durch die Punkte C' und D' gelegt. Der Punkt C ist dabei der Schnittpunkt der Geraden g_0 und g_{CD}, der Punkt D ist der Schnittpunkt der Geraden g_1 und g_{CD}.

Wendet man diese Methode auf jede Phase an, so erhält man für jede Phase ein Trapez, welches diese Phase umschließt. Die Vereinigung dieser Trapeze ergibt ein Polygon, das die gesamte Fläche des Manövers umschließt. Dies ist die Polygonhülle. Die Berechnung erfolgt dabei unabhängig von Manövertyp, Methode und der Art der Phase. Sie ist also auf alle Manöver anwendbar.

2.4 EZauto

Die Arbeitsgruppe Echtzeitsysteme der Universität Koblenz-Landau beschäftigt sich mit Fragestellungen des autonomen und assistierten Fahrens mit Serienfahrzeugen. Die Arbeitsgruppe entwickelt eine Softwarerarchitektur und die zugehörige Bibliothek, die verschiedene Ziele innerhalb der Domäne *mobile Systeme* verfolgt (vgl. [Wojke, 2010]) und **EZauto** genannt wird. Zu den Zielen gehören:

- Die Beschreibung der Eigenschaften eines Fahrzeugs nach bestimmten Sichten. Damit sind zum Beispiel die kinematischen Eigenschaften oder auch die strukturelle Beschreibung gemeint.

- Nutzung von Bewegungsmodellen, die auf die Fahrzeugbeschreibungen zurückgreifen.

- Die Umgebungen von Fahrzeugen, also Fahrbahnen und Hindernisse.

- Das Steuern eines Fahrzeugs nach der Berechnung und Planung der Bewegung unter Berücksichtigung des konkreten Fahrzeugs und der Umgebung.

- Visuelle Ausgaben verschiedener berechneter Daten und Simulationen von realen Bedingungen.

Die Architektur ist in drei Schichten aufgeteilt, die in Abbildung 2.12 verdeutlicht wurden. Die erste Schicht bildet dabei das Fundament. Dort sind „grundlegende domänenunspezifische Funktionalitäten“ ([Wojke, 2010]) untergebracht. Zu den Gruppen des Fundaments gehören zum Beispiel die Mathematik, die Darstellung und die Interprozesskommunikation.

Die nächste Schicht wird als Kern bezeichnet. Darin sind schon domänenspezifische Funktionalitäten enthalten, die aber noch nicht auf die spezielle Anwendung spezifiziert sind. In dieser Schicht findet man zum Beispiel die Gruppen für die Fahrzeugbeschreibung, die Wegfindung und -verfolgung, sowie Bewegungsmodelle.

In der Anwendungsentwicklung wird auf das Fundament und den Kern zugegriffen. Alle allgemeinen Funktionalitäten werden in die unteren Schichten ausgegliedert. Das können zum Beispiel Simulatoren, ein automatisierter Betriebshof oder ein autonomes Fahrzeug sein.

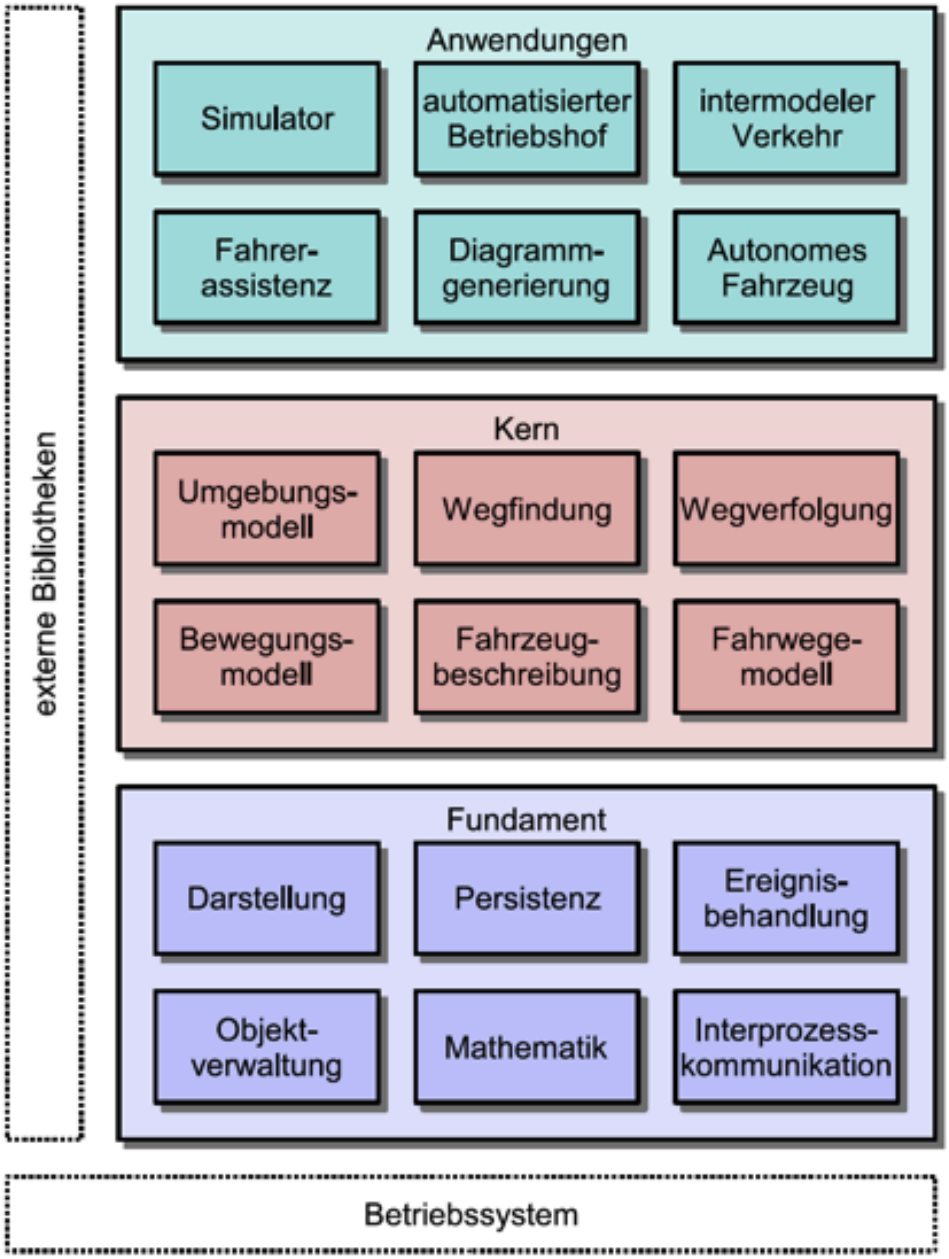


Abbildung 2.12: Aufbau der Softwarearchitektur von EZauto [Wojke, 2010]

2.4.1 Verwendete EZauto-Komponenten

Bei der Entwicklung der Anwendung dieser Arbeit wurden verschiedene Komponenten der Softwarebibliothek EZauto eingebunden. Die wichtigsten Komponen-

ten sollen dazu im folgenden beschrieben werden (vgl. auch [Wojke, 2010]).

2.4.1.1 Verwendung von Schnittstellen

Schnittstellen sind abstrakte Klassen ohne Attribute. Lediglich die Signaturen der Methoden werden deklariert. Eine Komponente kann verschiedene Schnittstellen implementieren und so unterschiedliche Sichten auf Instanzen dieser Komponente ermöglichen. Aus Sicht der jeweiligen Schnittstelle gibt es dabei aber keine Änderung in der möglichen Funktionalität der Implementierung. Semantisch beginnen Schnittstellen in der Bibliothek mit einem kleinen *s* (z.B. *sPfad*). Es existiert dabei nur eine Header-Datei, in der nur die Schnittstellen der Methoden definiert sind.

2.4.1.2 Beschreibung von Fahrzeugen

Die Fahrzeugbeschreibung ist ein Teil der Kernschicht in der Architektur. Ein Fahrzeug kann man, wie schon in Kapitel 2.1 aufgezeigt, aus mehreren Sichten beschreiben. Die strukturelle Sicht wird dabei statisch betrachtet und ist ohne Einfluss der Zeit. Dazu gehören die Fahrzeugabmessungen oder der maximal mögliche Lenkwinkel. Diese Parameter sind schon zu Beginn der Betrachtung gegeben und ändern sich nicht. Betrachtet man nun den aktuellen Lenkwinkel, beschreibt man schon den dynamischen Teil des Fahrzeugs. Dieser Teil ist nur zu einem bestimmten Zeitpunkt gültig und kann sich für ein spezifisches Fahrzeug ändern. Die Daten können auch als Funktionen der Zeit über einen Zeitraum betrachtet werden.

Ein Fahrzeug wird mit einer Kette verschiedener statischer und dynamischer Beschreibungen modelliert. Die statischen Daten werden in einer XML- oder einer Binär-Datei gespeichert. Diese kann mit der Bibliothek ausgelesen und in die verschiedenen Gespannteile, also Zugfahrzeug und evt. mehrere Anhänger, zerlegt werden. Die *Fahrzeugdaten* können in einer *Fahrzeugdatenkette* aneinander gekettet sein. In den Fahrzeugdaten befinden sich Informationen zur Position und Ausrichtung des Fahrzeugs, außerdem zu dem aktuellen Lenkwinkel, den aktuellen Einknickwinkeln oder der Geschwindigkeit, also dem dynamischen Teil der Fahrzeugsicht. Außerdem enthalten die Fahrzeugdaten die *Fahrzeugbeschreibungen*, in denen die Struktur des Fahrzeugs beschrieben wird. Dabei werden Zugfahrzeug und Anhänger getrennt betrachtet und verschiedene Fahrzeugbeschreibungen für ein Gespann angelegt. Aus den Beschreibungen kann dann eine *Fahrzeugbeschreibungskette* gebildet werden, in der alle Beschreibungen aneinander gehangen sind.

2.4.1.3 Geometrie

Im Laufe der Diplomarbeit wurde insbesondere auf die Geometrie-Gruppe in EZ-auto zurückgegriffen. Die Klasse *Punkt2D* repräsentiert Punkte oder Vektoren im zweidimensionalen Koordinatensystem. Hilfreich dabei ist die Berechnung der Länge eines Vektors (zum Beispiel für die Kandidatensuche der minimalen und maximalen Radien der Ringhülle). Für die Berechnung wie Skalarprodukt oder Vektorprodukt wurden die Operatoren überschrieben.

Die Klasse *Geometrie* enthält Methoden zur Winkelberechnung zwischen Vektoren und zur Suche nach Schnittpunkten zweier Geraden.

Mit der Klasse *Rotation2D* können Punkte im Koordinatensystem transformiert werden, um eine Aneinanderreihung der Phasen zu ermöglichen. Diese Transformation wird bei der Manöverkonstruktion an verschiedenen Stellen benötigt. Bei der Berechnung einer Phase wird diese immer im Bezug auf den Koordinatenursprung behandelt. Damit die Konstruktion als ein Manöver von aufeinanderfolgenden Phasen betrachtet wird, müssen die berechneten Punkte transformiert werden. Dazu wird eine Rotation um den Startpunkt der aktuellen Phase durchgeführt und das Ergebnis dann um einen Vektor verschoben, der Bezug nimmt auf das Ende der letzten Phase. Dadurch, dass sich die berechneten Punkte immer auf die letzte Phase beziehen, liegen diese nach der Transformation hintereinander.

2.4.1.4 Funktionsobjekte

Eine Klasse kann so modelliert werden, dass man die dazugehörenden Objekte wie mathematische Funktionen behandeln kann. Zu einer Funktion gehören die Definitionsmenge und eine Zielmenge.

In der Praxis geschieht das, indem man den Klammeroperator *()* überschreibt. Bei der Erstellung der Objekte werden Argumente übergeben, die für die Funktionsvorschrift der Zielmenge benötigt werden. Die Struktur der Funktionsvorschrift ist dabei immer gleich, die Werte der Vorschrift können sich für verschiedene Objekte unterscheiden. Die Parameter des Klammeroperators stellen die Funktionsargumente dar. Die Definitionsmenge kann mit einem Minimum und einem Maximum auf einen Definitionsbereich eingeschränkt werden.

Für die Bestimmung der Positionen zu einem bestimmten Zeitpunkt werden die Schnittstellen *sUnaerePlanareFunktion* und *sDiskretePlanareFunktion* in konkreten Klassen implementiert. Als Funktionsargument für die Berechnung ist der Winkel u notwendig, der den aktuellen Winkel der Kupplung im Bezug zum Koor-

dinatenursprung darstellt. Der Lenkwinkel α wird als Wert der Funktionsvorschrift für eine Phase festgesetzt. Zurückgegeben wird ein Objekt der Klasse Punkt2D, das die berechneten x- und y-Koordinaten der Position enthält.

Zu diesen Klassen gibt es Visualisierungen, in denen bestimmt ist, wie die Funktion visualisiert werden soll.

2.4.1.5 Visualisierung des Manövers

Im letzten Teil, der hier von EZauto vorgestellt wird, geht es um die Visualisierung. In der Gruppe *Darstellung* des Fundaments der Bibliothek gibt es dafür zwei relevante Schnittstellen. Die Schnittstelle *sZeichenflaeche* bietet Methoden an, um Linien, Polygone oder auch Text darzustellen. Auch Farbe, Linienart und Größe können hier verändert werden. Implementiert wird die Klasse für verschiedene Ausgabeformaten, wie Postscript-Dateien und eine Zeichenfläche für das hier verwendete wxWidgets.

Die zweite Schnittstelle ist *sVisualisierung*. Hier gibt es nur eine Methode, die *zeichne()* heißt. Die Zeichenfläche wird dort übergeben. Innerhalb der implementierten *zeichne()*-Methode der Visualisierung werden die einzelnen gewünschten Elemente auf der Zeichenfläche dargestellt. Dies geschieht durch aufrufen der jeweiligen Methoden in der Zeichenfläche.

2.4.2 Visualisierungswerkzeug PPFigure

Im Rahmen der Diplomarbeit wurde eine Software erweitert, die innerhalb eines Projektpraktikums entstanden ist. Zur Vorstellung des dabei entstandenen Werkzeugs wird auf die Abschlussdokumentation verwiesen, die während des Praktikums entstanden ist (vgl. [Lellmann *et al.*, 2010]).

Bei der Zielsetzung von PPFigure geht es um die Darstellung von Fahrzeugmodellen und deren Ausgabe als EPS-Datei für wissenschaftliche Ausarbeitungen. Dabei wurde die Implementierung eines Einspurmodells realisiert. Die Daten eines Fahrzeugs können aus einer XML-Datei eingelesen werden und werden in einem Einspurmodell des Fahrzeugs visualisiert. In Abbildung 2.13 wird das Ergebnis der Darstellung gezeigt.

Die dynamischen Fahrzeugdaten können dabei vom Benutzer geändert werden. So kann man zum Beispiel den Lenkwinkel des Zugfahrzeugs oder die Einknickwinkel zwischen den einzelnen Gespannteilen einstellen. Diese Änderungen werden sofort in der Darstellung übernommen. Wenn es sich um ein Zugfahrzeug

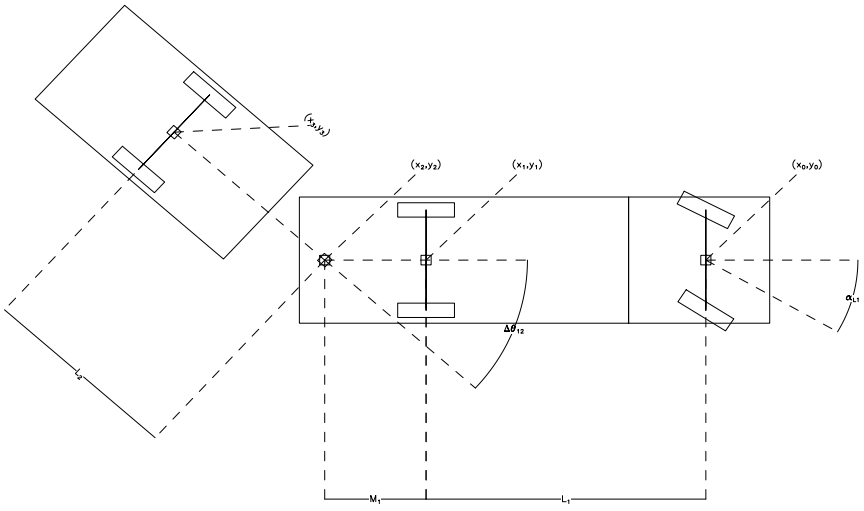


Abbildung 2.13: Darstellung eines Zugfahrzeugs mit Anhänger als Einspurmodell

mit einachsigem Anhänger handelt, ist auch die Darstellung der Winkel u und v möglich.

Die Architektur der Software wurde in dieser Arbeit übernommen. Konkret geht es dabei um die Trennung zwischen den Modelldaten, in denen alle Fahrzeugdaten und die dazu gehörenden Berechnungen enthalten sind und die GUI, die mit wxWidgets entwickelt wurde und mit der die Einstellungen durch den Benutzer vorgenommen werden können. Diese Teile werden verbunden durch die Darstellung, die für die Darstellungskonfiguration bestimmter Modellkomponenten benötigt wird. Diese Trennung wird im Kapitel 4.4 genauer besprochen.

Im Layout des Werkzeugs wurde ein Bereich für die Anzeige der Darstellung eines Einspurmodells reserviert. Im restlichen Teil können die Parametereinstel-

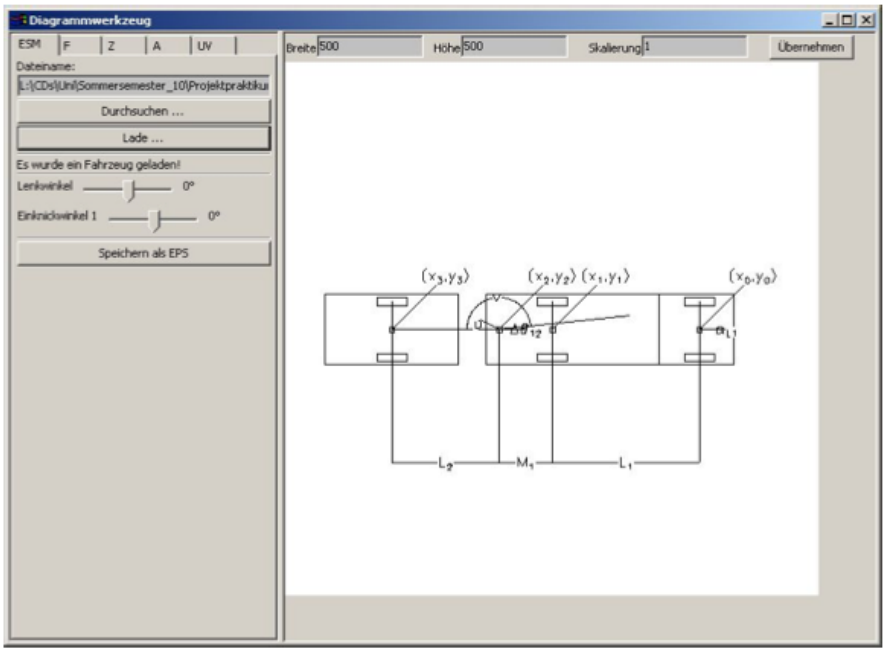


Abbildung 2.14: Layout der Software PPFigure

lungen durch den Benutzer vorgenommen werden. In Abbildung 2.14 kann man das Layout des Werkzeugs sehen, nachdem ein Fahrzeug geladen wurde.

Nach dem Laden der Fahrzeugdaten werden diese direkt angezeigt, wobei für die dynamischen Fahrzeugdaten Standardwerte übernommen werden. Außerdem werden Registerkarten, auch Tabs genannt, erstellt, um die Einstellungen zu gliedern. In der ersten Registerkarte geht es um das Einspurmodell. Hier können Lenkwinkel und Einknickwinkel für alle Anhänger geändert werden. Die zweite behandelt den Teil, der für alle Fahrzeugteile gleich ist. Hier können Offsets in x- und y-Richtung gesetzt und die Darstellung rotiert werden. Der dritte Tab behandelt das Zugfahrzeug, der vierte die Anhänger. Dabei können Teile des Fahrzeugs aus- und eingeblendet und bestimmt werden wie diese angezeigt werden. Ebenso können Beschriftungen erstellt und geändert werden. In der nächsten Registerkarte kann man Einstellungen zur Darstellung der Winkel u und v machen. Der letzte Tab dient dem Speichern der aktuellen Darstellung als EPS-Datei.

2.5 Die Bibliothek wxWidgets zur Entwicklung grafischer Benutzeroberflächen

Die Open Source-Softwarebibliothek wxWidgets ist eine Bibliothek, mit der man Desktop- und mobile Anwendungen mit grafischen Benutzerschnittstellen (GUI) entwickelt kann (vgl. [Smart *et al.*, 2006]). Die Bibliothek kann in ein C++ Projekt auf verschiedenen Plattformen eingebunden werden, zum Beispiel Windows, Linux und Mac OS X. Man kann auf viele Standardbedienelemente für die GUI-Entwicklung zurückgreifen, zum Beispiel Auswahlkasten (Checkbox) oder Schaltflächen (Button) und es ist möglich, Bilder und Grafiken zu zeichnen. Die Anfänge von wxWidgets gehen zurück in das Jahr 1992, an die Universität Edinburgh. Die Zielsetzung der Entwicklung war die Möglichkeit einer GUI für mehrere Plattformen.

Die wichtigste Klasse eines wxWidgets-Projekts ist die Application-Klasse. Diese Klasse ist abgeleitet von der Klasse *wxApp* und es existiert nur eine Instanz. Es muss eine Methode implementiert werden, in der die GUI-Objekte erzeugt werden können, zum Beispiel der Hauptframe, in dem sich alle Teile eines Programms befinden.

In diesem Frame (abgeleitet von *wxFrame*) wird das Hauptfenster erstellt, mit dem typischen Kopf eines Programms bestehend aus Programm-Icon, Name und den Kontrollbuttons für die Fenstergröße und gegebenenfalls einem Menü und einer Statuszeile. Innerhalb des Frames können weitere Objekte erzeugt werden, zum Beispiel Panels, die wiederum Objekte wie Buttons enthalten können.

Die Kommunikation zwischen den Programmelementen, Benutzer und den Aktionen, die daraufhin erfolgen sollen, werden von einem Event-Handling ermöglicht. Um das Vorgehen dabei zu verdeutlichen, wird es anhand eines Buttons beispielhaft erklärt. Dem Button wird beim Erzeugen eine ID mitgegeben. In einer Event-Tabelle werden die verschiedenen möglichen Events aufgelistet und mit dem Button verknüpft. Ebenso wird in der Tabelle angegeben, welche Funktion beim Auslösen des bestimmten Events für diesen Button aufgerufen werden soll. Jedes Kontrollobjekt kann mehrere Events auslösen. So kann ein Textfeld ein Event auslösen, wenn man etwas darin ändert oder aber erst, nachdem die Eingabe mit der Taste „Enter“ bestätigt wurde. Es kann aber auch ein Kontrollevent ausgelöst werden, wenn die maximale Anzahl an Buchstaben bei der Eingabe überschritten wurde.

2.6 Verwendete Entwurfsmuster

Im Zuge des Entwurfs wurde auf zwei Entwurfsmuster (Patterns) zurückgegriffen, die aus [Gamma *et al.*, 2004] stammen. In Entwurfsmustern werden für bestimmte, häufig auftretende Probleme in der objektorientierten Programmierung Lösungen vorgeschlagen, die im hohen Maße auf Wiederverwertbarkeit entwickelt wurden. In dieser Arbeit wurde auf die Muster Adapter und Composite zurückgegriffen.

2.6.1 Adapter-Entwurfsmuster

Das Adapter-Entwurfsmuster, auch als Wrapper bekannt, ist ein strukturelles Entwurfsmuster. Es wird dann verwendet, wenn eine Klasse nicht wiederverwendet werden kann, weil die Schnittstelle nicht mehr den Anforderungen entspricht. In dem Klassendiagramm in Abbildung 2.15 ist zu sehen, was es mit diesem Muster auf sich hat.

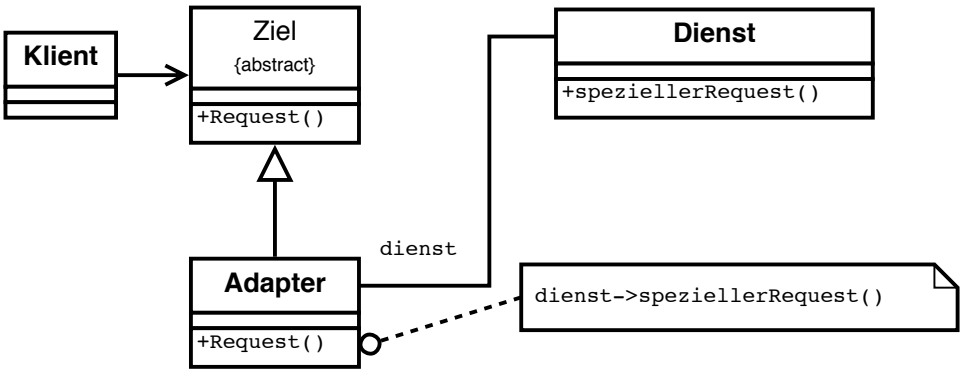


Abbildung 2.15: Adapter-Entwurfsmuster in UML-Notation [Gamma *et al.*, 2004]

Teilnehmer sind ein *Ziel*, eine Schnittstelle, die auf die Domäne spezifiziert ist, ein *Klient*, der über die Sicht der Zielschnittstelle auf den Adapter zugreift, ein *Adapter*, der die Zielschnittstelle abbildet und implementiert und einen Dienst, der mit dem Adapter eingebunden werden soll. In der Adapterklasse existieren die selben Methoden wie in der Zielklasse. Innerhalb dieser Methoden wird auf den Dienst zurückgegriffen.

2.6.2 Kompositum-Entwurfsmuster

Das Kompositum-Entwurfsmuster (engl. Composite) ist ebenfalls ein strukturelles Entwurfsmuster. Dieses wird verwendet, um nicht darauf einzugehen, ob es sich

um ein individuelles Objekt oder einen Container handelt, der mehrere Objekte dieser Art enthält. Es wird also eine Baumstruktur abgebildet, in der Objekte die Blätter darstellen, ein Kompositum mehrere Blätter und weitere Komposita enthalten kann und diese gleich behandelt werden.

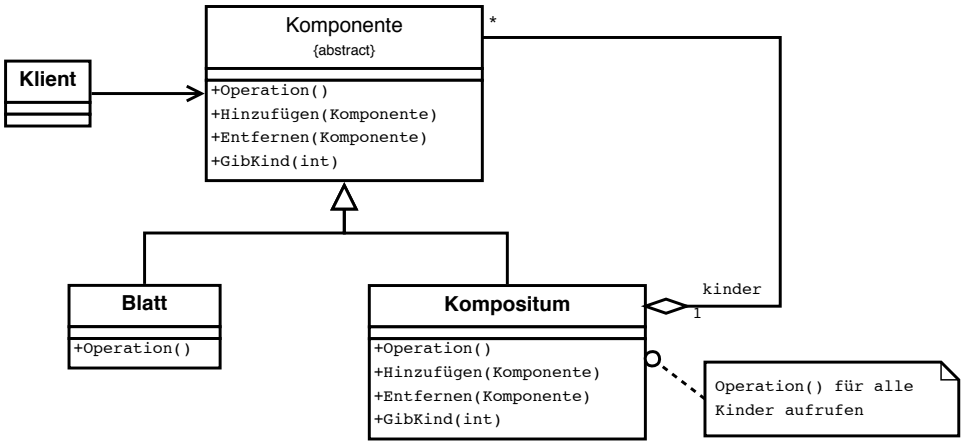


Abbildung 2.16: Kompositum-Entwurfsmuster in UML-Notation [Gamma *et al.*, 2004]

In Abbildung 2.16 ist die Struktur des Kompositum-Entwurfmusters zu sehen. Eine *Komponente* ist eine Schnittstelle, in der alle Methoden abgebildet sind, die Kompositum und Blätter benötigen. Auch die Möglichkeit weitere Kinder hinzuzufügen oder zu entfernen muss hier schon berücksichtigt werden. Diese Methoden werden in der Klasse *Kompositum* benötigt. Dort können Kinder hinzugefügt werden, die die Schnittstelle *Komponente* implementieren. Die Klasse *Blatt* repräsentiert ein individuelles Objekt der Klasse, das keine weiteren Objekte dieser Art enthält. Dort werden die eigentlichen Operationen ausgeführt. Im Kompositum müssen diese Operationen dann für alle Kinder aufgerufen werden.

Kapitel 3

Planung der Software

Die Entwicklung der Software ist an das Wasserfallmodell angelehnt. Das Wasserfallmodell war das erste Vorgehensmodell für die Softwareentwicklung. Der Name kommt daher, da sich die einzelnen Phasen, in die sich das Vorgehen aufteilt, durch Kaskaden von einer zur nächsten Phase darstellen lassen. Die einzelnen Phasen können nach [Sommerville, 2001] aufgeteilt werden in:

1. Analyse und Definition der Anforderungen
2. System- und Softwareentwurf
3. Implementierung und Komponententest
4. Integration und Systemtest
5. Betrieb und Wartung

Im Laufe der Diplomarbeit werden nur die ersten drei Punkte behandelt. Dieses Kapitel beschäftigt sich dabei mit der Analyse und der Definition der Anforderungen. Artefakte, die in diesem Abschnitt der Entwicklung entstehen, sind ein Anforderungskatalog und der Meilensteinplan.

3.1 Randbedingungen

Die Software wird mit der Programmiersprache C++ implementiert. Als Entwicklungsumgebung wird Microsoft Visual Studio 2005 benutzt. Für die Entwicklung der GUI wird die Bibliothek wxWidgets in der Version 2.8.10 verwendet.

3.2 Anforderungen an die Software

In diesem Abschnitt wird der erstellte Anforderungskatalog vorgestellt, nach dessen Grundlage die Software entwickelt wird. Die Anforderungen wurden in funktionale und nicht-funktionale Anforderungen aufgeteilt. Außerdem wird mit Pflicht und Wunsch unterschieden, ob eine Anforderung zwingend in der Software zu implementieren ist oder die Implementierung nur wünschenswert wäre. Funktionale Anforderungen beschreiben, den Leistungsumfang der Software, während nichtfunktionale Anforderungen die Eigenschaften des Produktes festlegen.

3.2.1 Funktionale Anforderungen

Pflicht

Allgemeines

1. Die Software muss so erweitert werden, dass zusätzlich zu dem Einspurmodell verschiedene Funktionalitäten integriert werden können.
2. Das Programm muss um die Funktionalität erweitert werden, Manöver verschiedener Art behandeln zu können.
3. Es muss eine Auswahl zwischen verschiedenen Manöverarten ermöglicht werden. Dazu gehören die folgenden Manöver: Geradeausfahrt, Kurvenfahrt und Wenden.
4. Es muss beachtet werden, dass verschiedene Möglichkeiten zur Konstruktion eines bestimmten Manövertyps möglich sein können, dazu gehören auch verschiedene Arten von Manöverkorridoren.
5. Es muss bei der Auswahl bestimmter Funktionalitäten, Methoden oder Manöverarten darauf geachtet werden, dass bei Bedarf auch weitere hinzugefügt werden können.
6. Es muss eine Auswahl zwischen der konventionellen Methode und dem iterativen Ansatz möglich sein.

Fahrzeug

7. Die Beschreibung eines Fahrzeugs muss importiert werden können. Dabei handelt es sich um die geometrische Beschreibung des Fahrzeugs, die im XML-Format vorliegt.

8. Der Lenkwinkel für den Einknickwinkel, mit dem sich das Fahrzeug auf einem konzentrischen Kreisbogen bewegt, muss angegeben werden können.

9. Alternativ muss der Einknickwinkel angegeben werden können, um den zugehörigen Lenkwinkel zu berechnen.

10. Der maximale Lenkwinkel des Fahrzeugs und der Einknickwinkel des Anhängers müssen aus der XML-Datei mit der Fahrzeugbeschreibung eingelesen werden.

Kurvenmanöver

11. Im Laufe dieser Arbeit muss die Funktionalität entwickelt werden, dass Kurvenmanöver konstruiert, manipuliert, angezeigt und gespeichert werden können.

12. Die Startausrichtung und Endausrichtung des Fahrzeugs müssen als Koordinaten einer Gerade eingegeben werden können.

13. Alternativ dazu sollen die Geraden auch über den Winkel bestimmt werden können, der zwischen den Geraden entsteht.

14. Die Fahrtrichtung des Fahrzeugs muss angegeben werden können.

15. Der Lenkwinkel, mit dem sich das Fahrzeug während der stabilen Fahrt bewegt muss angegeben werden können. Dabei ist auf das richtige Vorzeichen zu achten, um die Plausibilität zum Einknickwinkel zu gewährleisten.

16. Ebenso muss es möglich sein, stattdessen den Einknickwinkel anzugeben. Dann ist eine Prüfung des berechneten Lenkwinkels nötig.

Iterativer Ansatz

17. Bei der iterativen Methode muss die Größe des Inkrements, mit der sich der Lenkwinkel ändert, angegeben werden können und das Inkrement für die Änderung des Einknickwinkels.

18. Die Inkremente müssen auch variabel angegeben werden können, um eine langsame Annäherung an die stabile Fahrt zu gewährleisten.

19. Alternativ dazu muss die Anzahl der Iterationen angegeben werden können.

Manöverkorridor

20. Es muss auswählbar sein, ob ein Manöverkorridor angezeigt werden soll.

21. Zusätzlich muss ein Sicherheitskorridor eingegeben werden können, der prozentual zum Manöverkorridor ist.

22. Die Extrempunkte, die den Manöverkorridor bestimmen, müssen berechnet und angezeigt werden können. Dazu zählen auch die Kupplung und die Hinterachse des Zugfahrzeugs.

Berechnung und Darstellung

23. Folgende Punkte sind für die Darstellung relevant:

    (a) Räder des Zugfahrzeugs

    (b) Räder des Anhängers

    (c) Kupplung

    (d) Mittelpunkte der Achsen

    (e) Eckpunkte des Fahrzeugs

24. Die Fahrkurven für diese Punkte müssen berechnet und dargestellt werden können.

25. Diese Punkte sind auch für die Erstellung des Manöverkorridors notwendig und müssen dafür hervorgehoben werden können.

26. Welche dieser Punkte dargestellt werden, muss auswählbar sein.

27. Pfade sind für den Mittelpunkt der Vorderachse und der Hinterachse zu berechnen und darzustellen.

28. Es muss eine Auswahl erfolgen, ob eine Darstellung des Fahrzeugs am Anfang und Ende des Manövers und/oder an den Phasenübergängen erfolgen soll.

29. Die Darstellung muss in einem Fenster angezeigt werden.

30. Änderungen an den Parametern müssen unmittelbar in der Darstellung übernommen werden.

31. Die Darstellung muss so skaliert werden, dass Sie in dem Fenster sichtbar ist.

32. Anfangs muss die Darstellung autoskaliert werden.

33. Die Darstellung muss als EPS-Datei exportiert werden können.

34. Es muss eine Prüfung der Plausibilität erfolgen:

    (a) Der maximale Lenkwinkel darf nicht überschritten werden.

    (b) Der maximale Einknickwinkel darf nicht überschritten werden.

    (c) Es muss sich um ein Fahrzeug mit einachsigem Anhänger handeln.

    (d) Der Winkel zwischen den Geraden muss groß genug sein, um das Manöver ausführen zu können. Daher muss das Programm einen Hinweis darauf geben.

    (e) Die Monotonie muss nach den gegebenen Regeln überprüft werden.

35. Bei fehlerhafter Plausibilitätsprüfung muss eine Meldung erfolgen.

36. Die Punkte, an denen es eine Phasenänderung gibt, müssen als besondere Punkte gekennzeichnet werden können.

37. Auch die Iterationspunkte des Zugfahrzeugs müssen, wenn gewünscht, angezeigt werden können.

Wunsch

38. Die Auswahl zwischen der Anzeige eines Einspurmodells und eines Manövers soll auch während der Laufzeit möglich sein. Dabei ist das importierte Fahrzeug auf Plausibilität zu überprüfen.

39. Die jeweiligen Phasen eines Manövers sollen sich besonders hervorheben lassen.

40. Es soll auswählbar sein, ob die Geraden für die Ausrichtung am Anfang und Ende angezeigt werden.

41. Der Mittelpunkt der Kreisbögen für den Manöverkorridor und die Radien r_{min} und r_{max} sollen angezeigt werden können.
42. Es soll ein Zoom in die Darstellung des Fahrzeugs möglich sein.
43. Die Manöverkonfiguration soll als XML-Datei abgespeichert werden können.
44. Die gespeicherte XML-Datei soll importiert werden und das gespeicherte Manöver identisch konstruiert werden können.
45. Die Darstellung soll direkt gedruckt werden können.
46. Alle Konfigurationsparameter und Ergebnisse der Berechnungen sollen in einem Reiter zusammengefasst werden. Folgende Konfigurationsparameter sollen berücksichtigt werden:
    (a) Maximaler Lenkwinkel des Fahrzeugs α_{MAX}
    (b) Maximaler Einknickwinkel des Anhängers θ_{MAX}
    (c) Fahrzeugabmessungen
    (d) Winkel zwischen der Start- und der Zielgeraden Γ
    (e) Fahrtrichtung
    (f) Inkrement für Lenkwinkeländerung $d\alpha$ (iterativer Ansatz)
    (g) Inkrement für Änderung des Einknickwinkels $d(\Delta\theta)$ (iterativer Ansatz)
    (h) Bei variablen Inkrementen werden diese nicht angezeigt.

    Folgende Ergebnisse sollen berechnet und angezeigt werden können:

    (a) Maximaler Lenkwinkel während den unterschiedlichen Phasen α_{max}.
    (b) Winkel u und v, wenn diese stabil sind
    (c) Einknickwinkel am Anfang und Ende jeder Phase
    (d) Anzahl der Iterationen für jede Phase (iterativer Ansatz)
    (e) Die beiden Radien für den Manöverkorridor r_{max} und r_{min} und der Winkel zwischen den Radien σ
    (f) Breite des Manöverkorridors $r_{max} - r_{min}$
    (g) Länge des Manöverkorridors
    (h) Fläche des Manöverkorridors und die Berechnung von $r_{max}^2 - r_{min}^2$
    (i) Der Lenkwinkel α_{circ}, der auf der stabilen Fahrt benötigt wird

3.2.2 Nicht-funktionale Anforderungen

Pflicht

1. Die Programmierung muss unter Berücksichtigung der Programmierrichtlinien der AG Echtzeitsysteme erfolgen.
2. Die Software, die im Projektpraktikum entwickelt wurde, muss um die neuen Funktionen erweitert werden.
3. Änderungen des Erscheinungsbildes bezüglich der schon vorhandenen Software sind in der Dokumentation der Diplomarbeit zu begründen.
4. Winkel, die vom Benutzer eingegeben werden, müssen im Gradmaß angegeben werden. Interne Berechnungen geschehen im Bogenmaß.
5. Beim Speichern einer Datei muss der Ordner und der Dateiname auswählbar sein.
6. Beim Speichern einer Datei muss bei schon vorhandenen Dateinamen eine Abfrage erfolgen, ob diese überschrieben werden soll.
7. Die Schriftgröße in der Darstellung muss einstellbar sein.
8. Die Liniendicke der gezeichneten Linien und Pfade muss einstellbar sein.
9. Die Anzeige muss auch für eine Verwendung in schwarzweiß optimiert werden. Dazu ist die Linienart unterschiedlich darzustellen.

Wunsch

10. Das Erscheinungsbild der Software soll, soweit möglich, an das schon vorhandene Erscheinungsbild angepasst werden.

3.3 Meilensteinplan

Dieser Abschnitt beschäftigt sich mit den Meilensteinen, die während der Arbeit erreicht werden sollen. Ein Meilenstein stellt dabei das Ende einer Teilaufgabe dar. Anhand der Meilensteine kann dann der Verlauf des Projekts kontrolliert und validiert werden. Der Meilensteinplan ist in Tabelle 3.1 zu sehen.

Tabelle 3.1: Meilensteinplan

Meilenstein	**Ergebnis**
Abschluss der Planungs- und Definitionsphase	Anforderungskatalog
Abschluss der Entwurfsphase	Modelle und Klassenbeschreibungen
Abschluss der Implementierung	Software
Evaluation des konventionellen Verfahrens	Evaluationsergebnis
Schriftliche Ausarbeitung	Dokumentation

In der ersten Spalte sind die einzelnen Meilensteine definiert, die sich am Vorgehensmodell orientieren. Die zweite Spalte zeigt die Ergebnisse, die das Ende dieser Phase markieren.

Kapitel 4

Softwareentwurf

In diesem Kapitel wird die Entwurfsphase beschrieben. In dieser Phase der Softwareentwicklung wird die strukturelle Beschreibung der Software vorgenommen. Nach [Sommerville, 2001] gibt es während des Entwurfs verschiedene Iterationen. Der Entwurf startet in einer sehr abstrakten Version und wird iterativ konkretisiert. Gestartet wird mit dem Architekturentwurf der Software. Das System wird dabei in Subsysteme aufgeteilt und diese in Beziehung zueinander gesetzt. Im weiteren Verlauf werden die Subsysteme weiter verfeinert und daraus schließlich Komponenten entwickelt. Auch die Beziehungen zwischen diesen Komponenten werden dabei modelliert. Dafür werden Klassendiagramme eingesetzt. Das Verhalten wird in Aktivitäts- und Kommunikationsdiagrammen dargestellt.

Die Architektur der Software wird angelehnt an das Projektpraktikum, in dem die erste Version dieser Software entstanden ist. Ein wesentlicher Aspekt ist die Trennung zwischen GUI, Darstellung und Modelldaten, die schon in Kapitel 2.4.2 erwähnt wurde. Diese Trennung geht zurück auf das Architekturmuster *"Model View Controller"* (vgl. [Gamma *et al.*, 2004]), der *Controller*- und *View*-Teil wird allerdings nicht getrennt und von der *GUI* übernommen. Die Architektur ist in Abbildung 4.1 zu sehen, wobei in dem Diagramm schon die konkreten Klassen für ein Manöver gezeigt werden. Das Modell ist aber übertragbar auf die anderen Funktionalitäten. Die Funktionalitäten, die bisher in dem Werkzeug realisiert wurden, sind die Visualisierung eines Fahrzeugs in einem Einspurmodell und die Konstruktion eines Fahrzeugmanövers.

In dem Teil der Modelldaten befinden sich alle Daten, die für Berechnungen und Visualisierungen relevant sind. Die Daten können den Modelldaten übergeben werden, wenn sie zum Beispiel vom Benutzer vorgegeben sind. In diesem

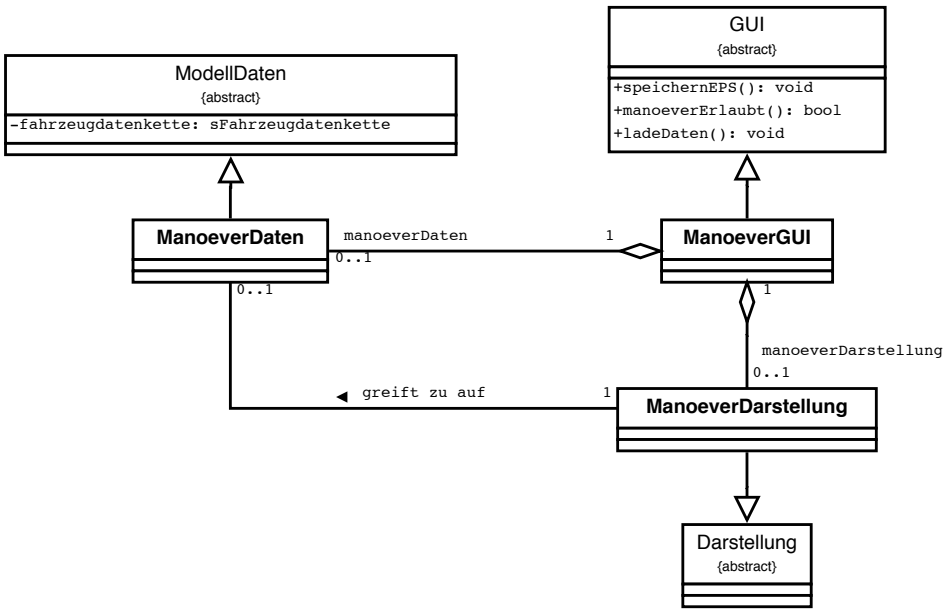


Abbildung 4.1: Trennung zwischen GUI, Darstellung und Modelldaten.

Teil finden aber auch die Berechnungen statt. Betrachtet man die Funktionalität einer Manöverkonstruktion, ist die Hauptklasse der Modelldaten die Klasse *ManoeverDaten*, die von der Klasse *Modelldaten* abgeleitet ist. Darin enthalten ist ein Attribut der Klasse *ManoeverFahrzeug*. Dieses wird von der Elternklasse geerbt und beinhaltet die Fahrzeugdatenkette, in der alle Daten des Fahrzeugs gehalten werden.

Die Darstellung befasst sich in erster Linie mit der Art der Visualisierung. Der Name wurde von der Vorgängerversion übernommen und ist eigentlich irreführend, da es sich dabei um die Darstellungskonfiguration handelt. Die eigentliche Visualisierung wird im Teil der Modelldaten vorgenommen. Für die Darstellung werden verschiedene Bestandteile der Visualisierung in Containern gruppiert, die bei der Darstellung zusammengefasst behandelt werden sollen. Innerhalb der Elemente dieser Container ist gespeichert, ob das Element angezeigt wird, welche Farbe es hat und welche Linienart verwendet werden soll. Diese Einstellungen können vom Benutzer über die GUI geändert werden, die im nächsten Abschnitt vorgestellt wird. Für ein Manöver ist die Klasse *ManoeverDarstellung* die Hauptklasse dieses Teils. Diese ist abgeleitet von der Klasse *Darstellung*.

Im GUI-Teil wird, mit Hilfe von wxWidgets, die Schnittstelle zwischen Benutzer und Software realisiert. Dieser Teil der Software agiert auch als Controller.

In Abbildung 4.1 ist daher zu sehen, dass die Klassen *ManoeverDaten* und *ManoeverDarstellung* Teile von der Klasse *ManoeverGUI* sind. Die Elternklasse von *ManoeverGUI* ist die abstrakte Klasse *GUI*. In dieser sind schon die Methoden definiert, um eine Darstellung als EPS-Datei zu speichern, die Fahrzeugdaten aus einer XML-Datei zu laden und abzufragen, ob mit dem aktuell geladenen Fahrzeug ein Manöver konstruiert werden kann. Dies ist der Fall, wenn das Gespann aus einem Zugfahrzeug mit einachsigem Anhänger besteht.

Im Folgenden wird zuerst auf die GUI des Programms eingegangen, danach werden die Modelldaten besprochen und am Schluss die Darstellung vorgestellt. In dieser Reihenfolge wurde auch während der Entwicklung vorgegangen. Im Anschluss an die Beschreibung der einzelnen Bestandteile wird detailliert auf den Zusammenhang zwischen den einzelnen Teilen eingegangen.

4.1 Entwurf der Benutzerschnittstelle (GUI)

Über die GUI kann der Benutzer mit dem Programm interagieren. Dies kann das Laden der Fahrzeugmodelle, das Auswählen der Funktionalitäten, das konfigurieren der Parameter für die Berechnungen und für die Visualisierung und die Ausgabe der Ergebnisse sein.

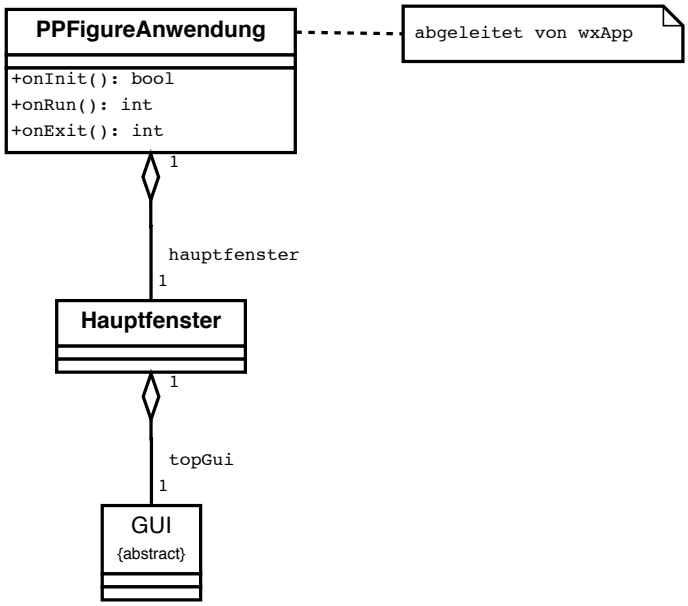


Abbildung 4.2: Die Hauptklassen der GUI.

Das Programm wird mit dem GUI-Teil gestartet. Die Hauptklassen des Programms sind in Abbildung 4.2 zu sehen. Die Klasse *PPFigureAnwendung* ist dabei von der wxWidgets-Klasse *wxApp* abgeleitet, die in Kapitel 2.5 vorgestellt wurde. Darin wird das Hauptfenster geladen und ein Objekt der Klasse *GUI* geladen. Von dieser abstrakten Klasse wurden für die verschiedenen Funktionalitäten verschiedene Klassen implementiert, die im folgenden vorgestellt werden.

4.1.1 Die GUI-Klassen der verschiedenen Funktionalitäten

Im Softwareentwurf kann zwischen zwei grundsätzlichen Funktionalitäten unterschieden werden. Dazu gehören die Darstellung eines Einspurmodells und die Konstruktion eines Manövers. Darüber hinaus ist für den Entwurf der GUI die Konfiguration, die ein Benutzer beim Start des Programms vornehmen muss, von Bedeutung und wird von den Funktionalitäten getrennt behandelt. In dem Klassendiagramm in Abbildung 4.3 ist zu sehen, dass es für jede dieser drei verschiedenen Möglichkeiten eine GUI-Klasse gibt, die von der abstrakten Klasse *GUI* abgeleitet ist.

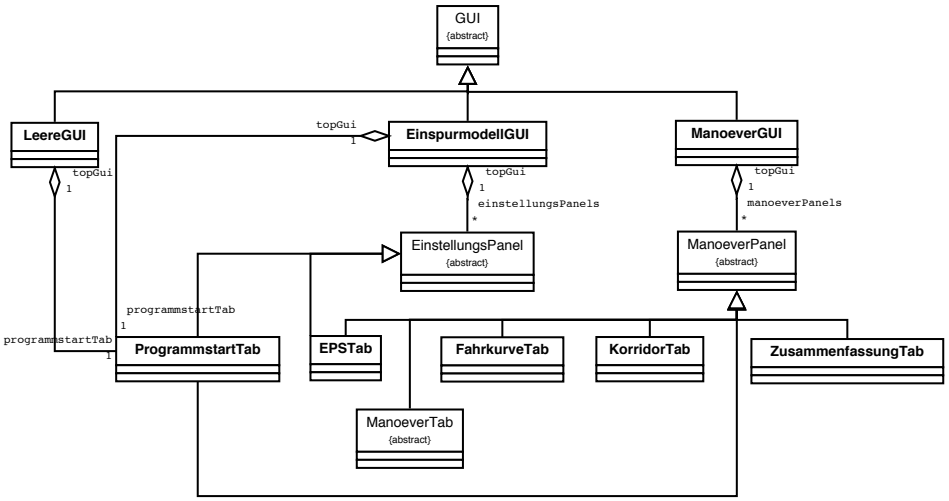


Abbildung 4.3: Klassendiagramm des GUI- Konzepts für die verschiedenen Funktionalitäten.

Jede konkrete GUI-Klasse verfügt über eine Organisation der Interaktionsmöglichkeiten in verschiedene Register. Diese Registerorganisation ist abhängig von der gewählten Funktionalität. Beim Start des Programms gibt es nur das Register *ProgrammstartTab*, in dem diese Auswahl erfolgen kann. Daraufhin werden die

in Kapitel 2.4.2 vorgestellten Register geladen, wenn man das Einspurmodell ausgewählt hat. Bei einer Manöverkonstruktion werden die Register geladen, die in Abbildung 4.4 zu sehen sind. Diese werden in Kapitel 4.1.3 genauer beschrieben.

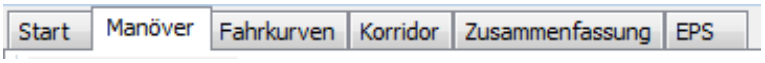


Abbildung 4.4: Registerkarten für die Manöverkonstruktion.

Der Inhalt einer Registerkarte ist von *wxPanel* abgeleitet. Für das Einspurmodell wurde dafür die abstrakte Klasse *EinstellungsPanel* entworfen. Da ein Register bei der Manöverkonstruktion nicht nur die Einstellungen, sondern auch die Anzeige von Ergebnissen enthalten kann, wurde dafür die ebenfalls abstrakte Klasse *ManoeverPanel* entworfen. Von dieser Klasse werden die konkreten Inhalte eines Registers abgeleitet. Diese sind in den Klassen *ProgrammstartTab*, *ManoeverTab*, *FahrkurvenTab*, *KorridorTab*, *ZusammenfassungTab* und *EPSTab* zu finden. Die Klassen *ProgrammstartTab* und *EPSTab* werden auch in der Klasse *EinspurmodellGUI* benötigt und sind daher auch von *EinstellungsPanel* abgeleitet. Die Endung der Namen auf *Tab* wurde gewählt, um zu verdeutlichen, dass es sich dabei um die Inhalte eines Registers handelt. Ein Panel ist dagegen sehr allgemein gehalten und kann in vielen GUI-Teilen verwendet werden. Eine Besonderheit stellt die Klasse *ManoeverTab* dar. Diese ist abstrakt und wird von Klassen implementiert, die sich auf den gewählten Manövertyp und die gewählte Konstruktionsmethode beziehen. Auf diese wird in Kapitel 4.1.3 eingegangen.

Die Registerkarte *ProgrammstartTab* ist Teil aller drei GUI-Klassen. Beim Start des Programms wird ein Objekt der Klasse *LeereGUI* erzeugt, mit dem *ProgrammstartTab* als einzige Registerkarte. Abhängig von der gewählten Funktionalität wird entweder ein Objekt der Klasse *EinspurmodellGUI* oder ein Objekt der Klasse *ManoeverGUI* erzeugt. Die Klassen des Einspurmodells wurden während des Projektpraktikums modelliert und sind in [Lellmann *et al.*, 2010] erläutert.

Beim Wechsel zwischen den GUI-Klassen muss darauf geachtet werden, dass das Objekt der aktuellen Klasse zerstört wird. Das Vorgehen dazu ist wie folgt:

- Zuerst wird aus der Klasse *topGui*, einem Objekt der Klasse *LeereGUI*, das Hauptfenster gesichert.
- Danach wird die neue GUI, je nach Funktionalität, erstellt. Bei einem Manöver wird dafür ein Objekt der Klasse *ManöverGUI* erzeugt, wobei die bisher getätigte Auswahl übergeben wird.

- Anschließend wird das alte GUI-Objekt gelöscht.

Im Folgenden wird zuerst auf die Benutzerschnittstelle zum Programmstart eingegangen und anschließend die GUI der Manöverkonstruktion besprochen.

4.1.2 Auswahlmöglichkeiten zum Programmstart

Dieser Abschnitt befasst sich mit dem Start des Programms, bei dem eine Datei mit Fahrzeugdaten geladen wird und daraufhin verschiedene, voneinander abhängige Einstellungen vom Benutzer vorgenommen werden können.

Ablauf der Auswahl beim Programmstart

Im Vergleich zur vorher entwickelten Version muss der Programmstart geändert werden. Bisher wurde nach dem Laden der Fahrzeugdatei direkt das Einspurmodell angezeigt. Das weiterentwickelte Programm soll stattdessen zu Beginn eine Auswahl ermöglichen, die laut Anforderungen folgende Alternativen bieten soll:

1. Laden des Fahrzeugs
2. Auswahl der Funktionalität: Soll ein Einspurmodell oder ein Manöver visualisiert werden.
3. Auswahl des Manövertyps: Soll ein Kurvenmanöver, ein Wendemanöver oder eine Geradeausfahrt konstruiert werden.
4. Auswahl der Methode: Soll das Manöver konventionell oder iterativ konstruiert werden.
5. Auswahl des Korridors: Soll eine ringförmige oder polygonförmige Hülle konstruiert werden.

Es wird dabei während des Auswahlprozesses sequentiell durch die einzelnen Schritte gegangen. Die Auswahl in einem Schritt beeinflusst dabei die Auswahl des nächsten Schritts. In Abbildung 4.5 ist ein Aktivitätsdiagramm zu sehen, auf dem die Möglichkeiten zum Programmstart im Bezug auf das zuvor Ausgewählte verdeutlicht sind. Die Registerkarte, die durch die Auswahl führt, ist in Abbildung 4.6 zu sehen.

Nach dem Laden der Datei kann schon bestimmt werden, ob ein Manöverkonstruktion überhaupt möglich ist. Ein Manöver kann konstruiert werden, wenn es

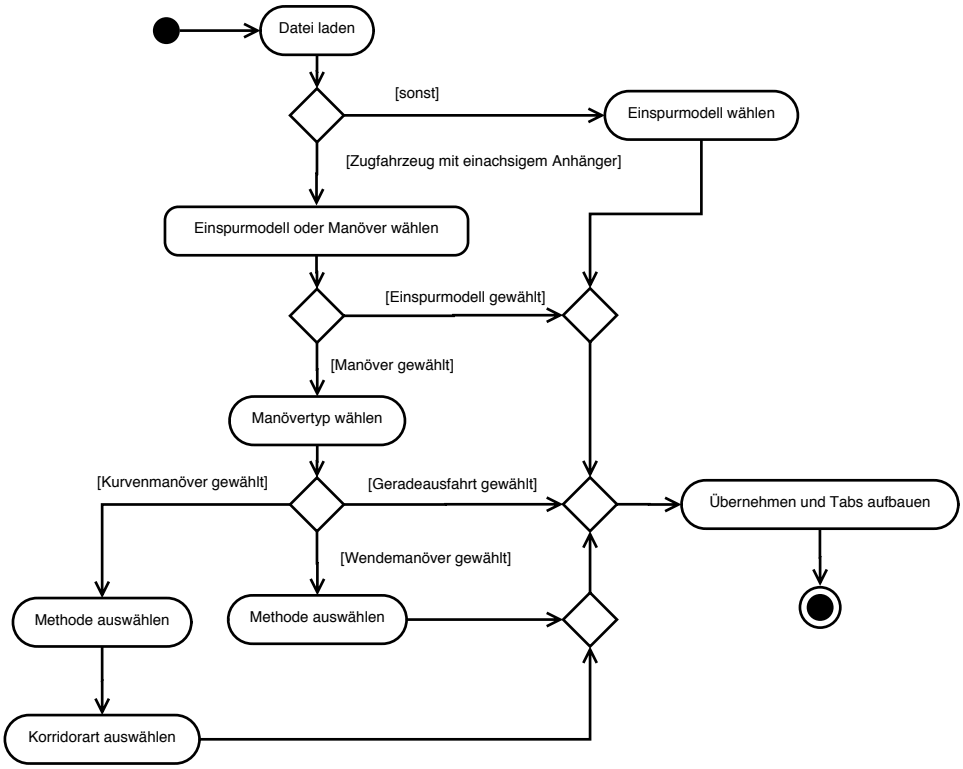


Abbildung 4.5: Aktivitätsdiagramm der Möglichkeiten beim Programmstart.

sich um ein Zugfahrzeug mit einachsigem Anhänger handelt. Anderenfalls kann nur das Einspurmodell angezeigt werden. Wählt man in diesem speziellen Fall die Funktionalität „Manöver“, kann man den Manövertyp bestimmen. Die Auswahl bietet an, ein Kurvenmanöver, ein Wendemanöver oder eine Geradeausfahrt zu konstruieren. Bei einer Geradeausfahrt ist hier die Auswahl zu Ende, da es keine verschiedenen Methoden gibt und die Hülle nur als Polygon visualisiert werden kann. Bei den anderen beiden Manövertypen kommt als nächstes die Auswahl zwischen der konventionellen und der iterativen Methode. Die Unterschiede wurden in Kapitel 2.3.2 gezeigt. Die Konstruktion der Hülle als Ring ist nur für das Kurvenmanöver möglich, daher gibt es hier zusätzlich die Auswahl zwischen einer Ringhülle und einer Polygonhülle. Bei einem Wendemanöver kann der Korridor nur aus einem Polygon bestehen. Im Anschluss kann die Auswahl bestätigt werden und das Programm die entsprechenden Registerkarten aufbauen. Dies geschieht in den GUI-Klassen.

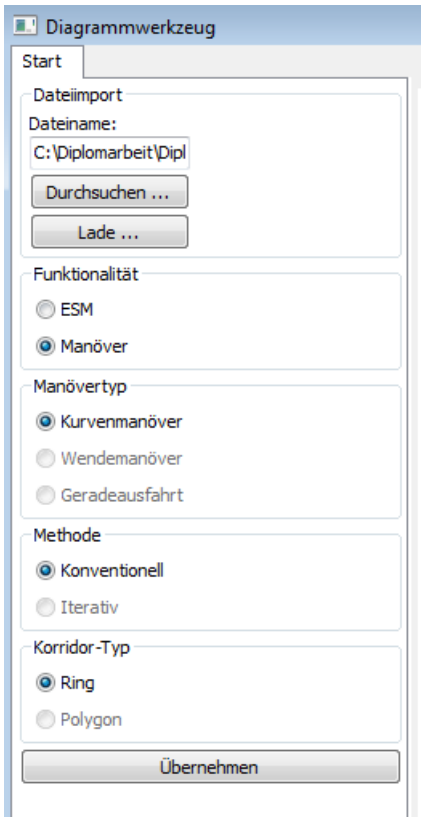


Abbildung 4.6: Registerkarte beim Start des Programms.

Klassenentwurf für die Auswahlmöglichkeiten beim Programmstart

Der Softwareentwurf für den Programmstart ist in einem Klassendiagramm in Abbildung 4.7 verdeutlicht. Beim Start wird als erstes die Klasse *LeereGUI* erzeugt. Diese ist genau wie die Klasse *ManoeverGUI* von *GUI* abgeleitet. Es wird ein Objekt der Klasse *ProgrammstartTab* erzeugt. In diesem wird sequentiell durch die Auswahlmöglichkeiten geführt. Die Bestandteile von *ProgrammstartTab* wurden in einzelnen Klassen ausgelagert. Die Namen dieser Klassen enden mit *GUI*, sind aber nicht von der abstrakten Klasse *GUI* abgeleitet.

Die Kontrollelemente für den Dateiimport sind schon bei der Erzeugung der Registerkarte zu sehen. Dafür wird ein Objekt der Klasse *DateiGUI* erzeugt. In dieser Klasse gibt es ein Element für den Dialog der Dateiauswahl und den Button zum Laden der ausgewählten Datei. Der Dialog für die Dateiauswahl wird aus der Klasse *ProgrammstartTab* in der Methode *ladeDatei()* aufgerufen. Die Auswahl der Datei ist dabei beschränkt auf XML-Dateien. Nachdem die Datei ausgewählt

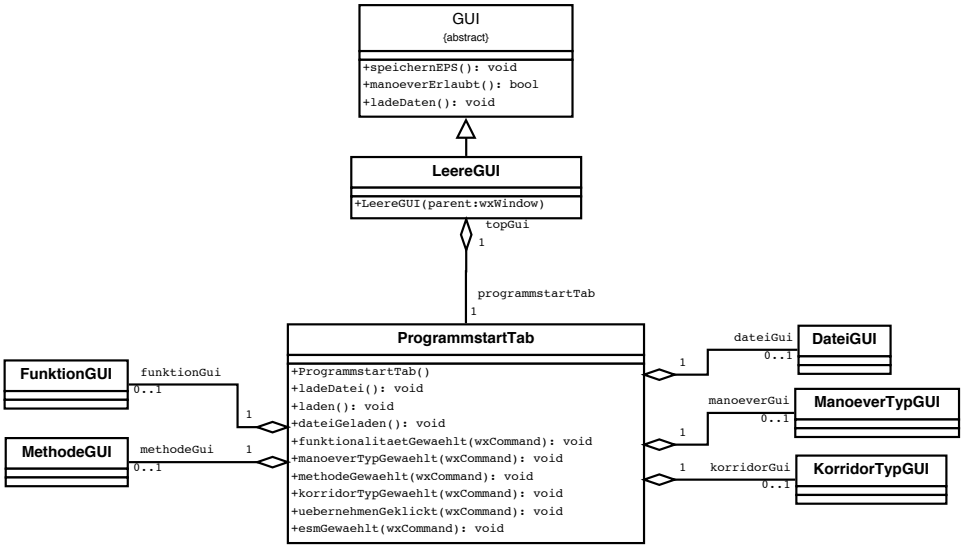


Abbildung 4.7: Klassendiagramm für die Benutzeroberfläche beim Start des Programms.

wurde, wird mit dem Klicken auf den „Lade“-Button ein Event ausgelöst, das die Methode *laden()* aufruft. Innerhalb dieser Methode wird auf die *LeereGUI* zurückgegriffen und darin die Methode *ladeDaten()* aufgerufen. Aus der ausgewählten XML-Datei wird dann die Fahrzeugdatenkette geladen.

Anhand der Fahrzeugdaten kann bestimmt werden, ob ein Manöver möglich ist oder nur das Einspurmodell angezeigt werden kann. In der Methode *dateiGeladen()* wird eine statische Box erzeugt, um die Funktionalität auszuwählen. Der Inhalt wird dabei von der Klasse *FunktionGUI* erstellt. Dem Objekt dieser Klasse wird mitgegeben, ob ein Manöver konstruiert werden kann. Dazu wird die Methode *manoeverErlaubt()* aus der Klasse *LeereGUI* aufgerufen. Kann kein Manöver erstellt werden, weil es sich nicht um ein Zugfahrzeug mit einachsigem Anhänger handelt, wird nur die Möglichkeit „Einspurmodell“ angeboten, ansonsten auch die Funktionalität „Manöver“.

In der Methode *funktionalitaetGewaehlt()* wird eine weitere Box für den Manövertyp geladen. Der Inhalt dieser Box wird in der Klasse *ManoeverTypGUI* erzeugt. Es werden Möglichkeiten für das Kurvenmanöver, das Wendemanöver und die Geradeausfahrt angeboten.

Mit der Methode *manoeverTypGewaehlt()* ist ein Event verknüpft, das den Manövertyp abfängt. Innerhalb dieser Methode muss unterschieden werden, welches Manöver ausgewählt wurde. Bei den Manövertypen Kurven- und Wendemanöver muss nun die Methode unterschieden werden, nämlich ob das Manöver konventio-

nell oder iterativ konstruiert werden soll. Die beiden Auswahlmöglichkeiten werden in der Klasse *MethodeGUI* erstellt.

Bei einem Kurvenmanöver ist es möglich, zwischen einer Ringhülle und einer Polygonhülle zu unterscheiden. Die Auswahlbox für den Korridortyp wird in der Methode *methodeGewaehlt()* erzeugt, der Inhalt der Box, also die Auswahl zwischen Ring und Polygon, kommt aus der Klasse *KorridorTypGUI*.

Auch der „Übernehmen"-Button wird mit der Event-Tabelle überwacht. Nach dem Klicken auf den Button wird die GUI für den Programmstart verlassen und die GUI für die jeweilige Funktionalität, Einspurmodell anzeigen oder Manöver konstruieren, erstellt. Der Wechsel zwischen den GUIs entspricht dem Schema, das in Kapitel 4.1.1 vorgestellt wurde.

Beim Start des Programms spielen die Modelldaten und die Darstellung noch keine Rolle. Erst beim Übergang von der Startauswahl zur Konstruktion vom Einspurmodell oder Manöver werden diese Objekte erzeugt. Als nächstes wird aber auf die Benutzerschnittstelle der Manöverkonstruktion eingegangen.

4.1.3 Benutzerschnittstelle zur Konstruktion eines Manövers

Im Folgenden wird dann davon ausgegangen, dass die Auswahl zu Beginn abgeschlossen wurde und ein beliebiges Manöver konstruiert werden soll. Dabei wird auf die einzelnen Manövertypen und Konstruktionsmethoden eingegangen, sofern sich dabei Unterschiede innerhalb der Benutzerschnittstelle ergeben.

Die Registerkarten der grafischen Benutzerschnittstelle

Die Benutzeroberfläche für die Manöverkonstruktion ist in mehrere Registerkarten (Tabs) eingeteilt. Der erste Tab ist der schon vorgestellte *ProgrammstartTab*. Dieser soll auch hier angezeigt werden, um die Auswahl während der Konstruktion ändern zu können.

Die nächste Registerkarte ist zuständig für die Einstellungen der einzelnen Manöverphasen. Hier kann für die instabilen Phasen der Lenkwinkel eingestellt werden, für die stabilen Phasen kann man zwischen dem Lenkwinkel und dem Einknickwinkel unterscheiden. Bei einer iterativen Manöverkonstruktion werden hier zudem die Einstellungen für die Inkremente gemacht. Nach der Auswahl der Manövereinstellungen beim Programmstart wird zwar das Fahrzeug mit seinen Daten schon eingelesen, aber noch kein Manöver konstruiert. Dies geschieht erst, nachdem man die Phaseneinstellungen vorgenommen hat, indem der Benutzer den

„Übernehmen“-Button anklickt. Erst dann folgen die Berechnungen, die Prüfungen der Plausibilität und die Visualisierung.

Danach folgt eine Registerkarte, in der Einstellungen zur Darstellung der Pfade und Fahrkurven gemacht werden. Hier kann für die Bezugspunkte (Mittelpunkte der Achsen, Kupplung, Ecken von Zugfahrzeug und Anhänger und Räder von Zugfahrzeug und Anhänger) eingestellt werden, ob die Fahrkurven für diese Punkte angezeigt werden sollen. Ebenso kann die Art der Darstellung ausgewählt werden.

Im nächsten Tab werden die Einstellungen zum Manöverkorridor gemacht. Die Auswahl des Korridortyps ist beim Erstellen der Tabs schon abgeschlossen, daher kann hier zur Zeit nur der Sicherheitskorridor beeinflusst werden, indem man den Prozentsatz des Sicherheitskorridors zur Manöverhülle ändert. Auch hier kann die Darstellung angepasst werden, indem man den Korridor aus- und einblenden und Farbe und Linienart ändern kann.

Das nächste Register ist erst nach der Konstruktion des Manövers interessant, da dort alle Parameter und Ergebnisse zusammengefasst werden. Die Parameter sind dabei zum Beispiel die Lenkwinkel und Einknickwinkel der einzelnen Phasen. Die Ergebnisse beziehen sich insbesondere auf den Korridor und beinhalten beispielsweise die berechnete Fläche, Länge und den Winkel. Im letzten Tab kann die Abbildung als EPS-Datei gespeichert werden.

Die Registerkarte „Manöver“

In der Registerkarte „Manöver“ werden die Einstellungen zu den einzelnen Phasen gemacht. Für die Registerkarte ist die Klasse *ManoeverTab* zuständig. Da diese Einstellungen sehr unterschiedlich sind, je nach Manövertyp und Konstruktionsmethode, ist diese Klasse abstrakt und wird von den konkreten Klassen implementiert, die Manövertyp und Konstruktionsmethode entsprechen.

In der Abbildung 4.8 ist diese Spezialisierung zu sehen. Die implementierten Klassen von *ManoeverTab* haben dabei immer die gleiche Struktur. Der Aufbau der Elemente innerhalb des Tabs wird im Konstruktor vorgenommen. Diesem müssen auch die Zeiger auf die Objekte der Klassen *FahrkurvenTab*, *KorridorTab* und *ZusammenfassungTab* übergeben werden. In der Methode *uebernehmen()* sind diese von Bedeutung, da damit angestoßen werden kann, dass die Kontrollelemente zur Konfiguration der Visualisierung aktiviert werden. Im Fall der Zusammenfassung werden Parameter und Ergebnisse erst nach dem Übernehmen der Einstellungen des Benutzers angezeigt. Es ist daher erforderlich, diese Registerkarten vor dem *ManoeverTab* zu erzeugen.

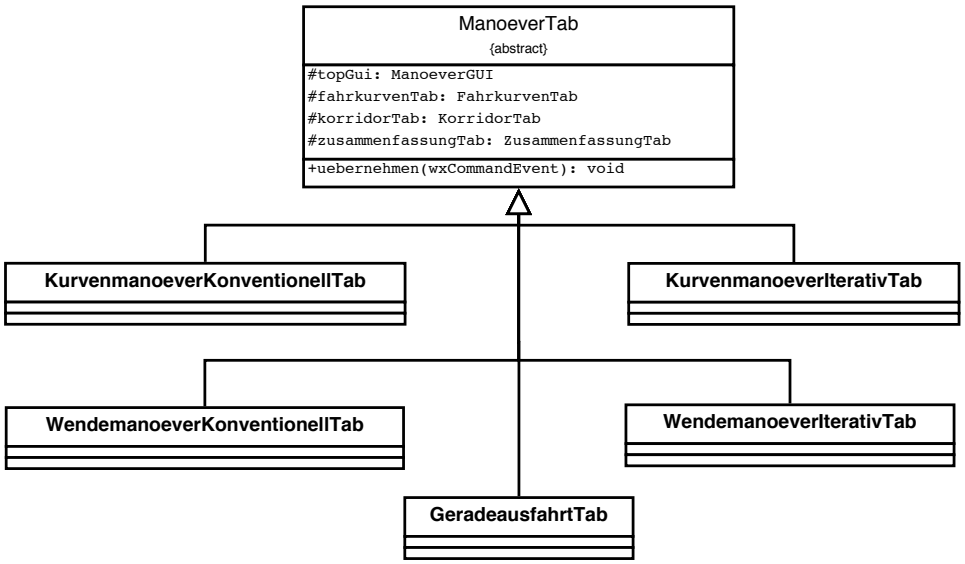


Abbildung 4.8: Abstrakte Klasse *ManoeverTab* und Implementationen dieser Klasse.

Die Unterscheidung der konkreten Klassen von der abstrakten Klasse *ManoeverTab* findet vor allem innerhalb des Aufbaus des Tabs im Konstruktor statt. Der erste Unterschied zwischen diesen Klassen ist der gewählte Manövertyp und der zweite die gewählte Methode. Aus diesen beiden Auswahlmöglichkeiten ergibt sich die Anzahl der Phasen, die in dem Manöver konstruiert werden. Bei der iterativen Methode muss der Benutzer zusätzlich zu den Lenkwinkeln bei einer instabilen Phase Angaben zu den Inkrementen machen. Da diese Unterscheidungen auch immer zu einem unterschiedlichen Aufbau des Inhalts der Registerkarte führen, werden die Tabs nicht zu den einzelnen Manövertypen zusammengefasst, sondern es wird für jeden Manövertyp und jede mögliche Methode dieses Typs eine eigene Klasse implementiert. Dieses Vorgehen ist nicht nur bei der Benutzerschnittstelle erforderlich, sondern insbesondere auch bei den Modelldaten, die in Kapitel 4.2 beschrieben werden.

Abhängig von Manövertyp und Konstruktionsmethode werden in einem *ManoeverTab* verschiedene Module erzeugt, die in einzelnen statischen Boxen untergebracht sind. Diese Module stellen jeweils eine Gruppe von Einstellungen dar, die für das Manöver getätigt werden können. Diese können innerhalb der verschiedenen Implementationen von *ManoeverTab* wiederverwendet werden, auch wenn sich die Art der Einstellung und Reihenfolge ändern können. Die Module sind in separaten Klassen definiert. Dies sind die Klassen *PolygonzugGUI*, *InstabilePha-*

seGUI, *StabilePhaseGUI* und *GeradeausfahrtGUI*. Wichtig ist, dass Veränderungen einzelner Parameter direkt in den Modelldaten übernommen werden. Dafür wird allen Modulen ein Zeiger auf das Objekt der GUI-Klasse mitgegeben, die auch Zugriff auf die Modelldaten hat. Abbildung 4.9 zeigt die Klasse *ManoeverTab* mit den möglichen Modulen. Die Methoden dieser Klassen beziehen sich auf das Abfangen bestimmter Events. Hierbei ist in allen Klassen die Methode *scroll()* dafür zuständig, die Events abzufangen, die von einem Schieberegler ausgehen. Die Identifizierung des Schiebereglers geschieht dann über IDs, die vorher gesetzt wurden. Jedem Kontrollelement das mithilfe von Event-Tabellen überwacht werden soll, wird beim Erstellen eine eindeutige ID zugeordnet. Wird ein Event ausgelöst, kann durch die Tabelle entschieden werden, welche Methode ausgeführt wird. Die ID des auslösenden Kontrollelements kann dabei der Methode übergeben werden. Somit können gleichartige Events mit derselben Methode behandelt werden und erst in dieser Methode zwischen den Kontrollelementen unterschieden werden.

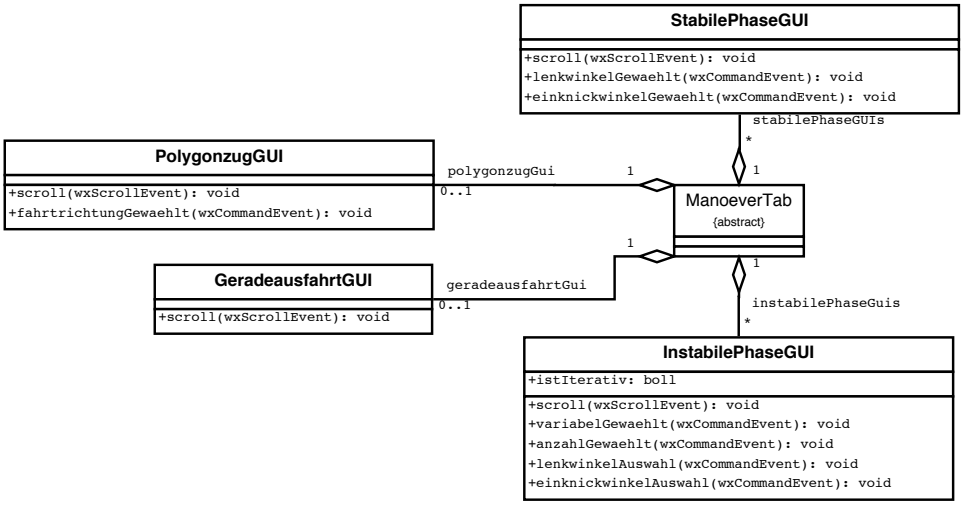


Abbildung 4.9: Module der Klasse *ManoeverTab*.

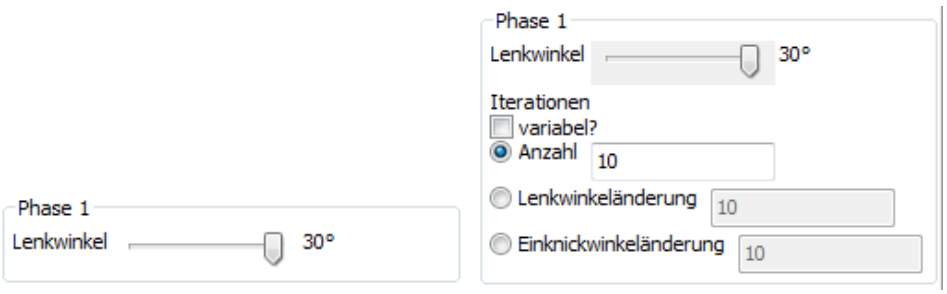
Die Einstellungen, die ein Benutzer in dieser Registerkarte vornehmen kann, beginnen für ein Kurven- und Wendemanöver mit dem Modul *PolygonzugGUI*. Darin kann die gewünschte Ausrichtungsänderung angegeben werden. Dieser Winkel kann über einen Schieberegler eingestellt werden und ist von 0° bis 360° definiert, wobei Einstellungen in 0,5° Schritten möglich sind. Die Auswirkungen der Wahl des Winkels Γ sind in Kapitel 2.3.2.1 beschrieben. Außerdem kann in dem Tab angegeben werden, ob die Fahrtrichtung vorwärts oder rückwärts ist.

Die Anzahl der Phasen und deren Anordnung sind durch die Auswahl beim

Start des Programms schon vorgegeben und können damit fest in den konkreten Klassen von *ManoeverTab* implementiert werden. Bei den Einstellungen der Phasen in einem *ManoeverTab* wird zwischen den stabilen und den instabilen Phasen unterschieden.

Für die stabile Phase ist die Klasse *StabilePhaseGUI* zuständig. In einer stabilen Phase kann man auswählen, ob man den Lenkwinkel oder den Einknickwinkel während der Phase angeben möchte. Beide können mit einem Schieberegler in 0,5°-Schritten angegeben werden. Bei der Einstellung eines Wertes wird dabei auch der entsprechend andere Wert berechnet und gesetzt.

Bei einer instabilen Phase kann nur der Lenkwinkel angegeben werden, da sich der Einknickwinkel während der Phase ändert. Dafür wird die Klasse *InstabilePhaseGUI* verwendet. Hier muss aber zwischen der konventionellen und der iterativen Konstruktionsmethode unterschieden werden. Der Unterschied zwischen den beiden Methoden ist in Abbildung 4.10 zu sehen. Bei der iterativen Methode müssen zusätzlich Angaben zu den Inkremente vorgenommen werden. Diese können über die Anzahl, die Änderung des Lenkwinkels oder die Änderung des Einknickwinkels eingestellt werden. Darüber hinaus kann angegeben werden, ob sich Inkremente variabel ändern sollen. In diesem Fall ist nur die Angabe der Anzahl der Inkremente möglich.



(a) Konventionell (b) Iterativ

Abbildung 4.10: Instabile Phase im Vergleich zwischen konventioneller und iterativer Methode.

Hat man die Konstruktion einer Geradeausfahrt ausgewählt, ist nur eine Änderung der Position ohne Ausrichtungsänderung möglich. In diesem Fall wird das Modul *GeradeausfahrtGUI* geladen. Darin kann über einen Schieberegler die Länge der Geradeausfahrt angegeben werden. Auch die Fahrtrichtung kann ausgewählt werden.

Nachdem man die Einstellungen in der Registerkarte für das Manöver über-

nommen hat, werden in den anderen Tabs die Funktionen freigeschaltet und das Manöver konstruiert.

Die Registerkarte „Fahrkurven“

In dieser Registerkarte können die Darstellungsoptionen der Fahrkurven für bestimmte Bezugspunkte geändert werden. In Abbildung 4.11 sind die Kontrollmöglichkeiten innerhalb der Registerkarte zu sehen.

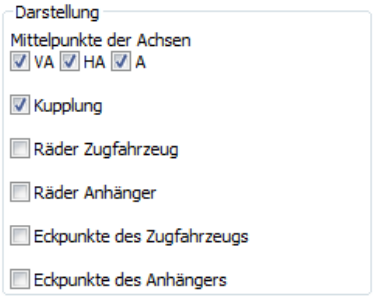


Abbildung 4.11: Die Kontrollkästchen von *FahrkurvenTab*.

Die Klasse ist in Abbildung 4.12 modelliert. In der Klasse sind drei Methoden enthalten.

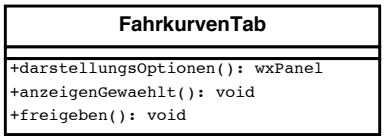


Abbildung 4.12: Die Klasse *FahrkurvenTab*.

In der Methode *darstellungsOptionen()* werden die Kontrollelemente aufgebaut, um die Darstellungsoptionen für die Fahrkurven zu setzen. Zu den Fahrkurven gehören die Pfade der Bezugspunkte von den Mittelpunkten der Achsen und der Kupplung. Außerdem können die Fahrkurven der für die Hülle zu betrachteten Extrempunkte angezeigt werden. Die Optionen für die Darstellung werden dabei zu den Eckpunkten des Zugfahrzeugs, den Eckpunkten des Anhängers, den Rädern des Zugfahrzeugs und den Rädern des Anhängers gruppiert. Die Darstellungsoptionen beinhalten, ob und wie die jeweilige Auswahl angezeigt wird. Dabei ist vorgesehen, dass Farbe und Linienart ausgewählt werden kann.

Die Methode *anzeigenGewaehlt()* ist in der Event-Tabelle hinterlegt und wird aufgerufen, wenn der Zustand des Kontrollkästchens geändert wurde. Dabei kann

abgefragt werden, ob das Kästchen markiert oder nicht markiert wurde. In der Methode kann auf die Objekte der Darstellung für die Fahrkurven, die über das GUI-Objekt verfügbar sind, zugegriffen werden. Die Darstellung wird in Kapitel 4.3 beschrieben.

Darstellungsoptionen für Fahrkurven können erst gesetzt werden, wenn ein Manöver konstruiert wurde. Daher wird aus dem *ManoeverTab* nach dem Konstruieren die Methoden *freigeben()* der anderen Tabs aufgerufen. Wenn die Methoden aufgerufen werden, werden alle Kontrollelemente dieser Register aktiviert, also für den Benutzer freigegeben.

Die Registerkarte „Korridor“

In der Registerkarte „Korridor“ können die verschiedenen Darstellungsoptionen für den Manöverkorridor ausgewählt werden. Dafür gibt es in der Klasse *KorridorTab* die Methode *darstellungsOptionen()* für die Anzeige der Kontrollelemente und *anzeigenGewaehlt()* für das Abfangen der durch den Benutzer vorgenommenen Änderungen. Die Einstellungen betreffen den Manöverkorridor und den Sicherheitskorridor.

Außerdem ist es möglich, den Manöverkorridor um einen bestimmten Wert prozentual zu vergrößern. Damit wird die sichere Hülle konstruiert, wofür weitere Methoden nötig sind, wie in Abbildung 4.13 zu sehen ist.

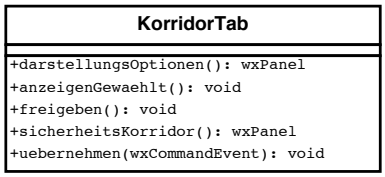


Abbildung 4.13: Die Klasse *KorridorTab*.

In der Methode *sicherheitsKorridor()* werden die Kontrollelemente erzeugt, mit der der Benutzer die prozentuale Änderung des Sicherheitskorridors einstellen kann. Die Änderung kann mit einer Bestätigung durch eine Schaltfläche übernommen werden. Diese wird mit der Methode *uebernehmen()* überwacht. Nach dem Betätigen der Schaltfläche wird der Sicherheitskorridor neu konstruiert. Danach müssen auch hier die *freigeben()*-Methoden der anderen Registerkarten aufgerufen werden.

Die Registerkarte „Zusammenfassung“

Der Inhalt dieser Registerkarte wird in der Klasse *ZusammenfassungTab* erstellt. Der Inhalt ist zweigeteilt, zum einen in die Anzeige der Einstellungen, die der Benutzer für die Konstruktion des Manövers vorgenommen hat und zum anderen in die Anzeige der Ergebnisse. In Abbildung 4.14 ist das Klassendiagramm von *ZusammenfassungTab* zu sehen.

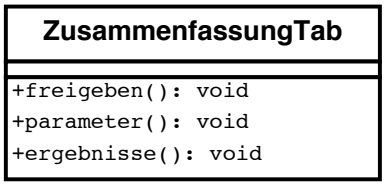


Abbildung 4.14: Die Klasse *ZusammenfassungTab*.

In der Methode *freigeben()*, die nach der Konstruktion eines Manövers aufgerufen wird, werden die Methoden *parameter()* und *ergebnisse()* aufgerufen. Die Parameter beziehen sich auf den Polygonzug, die Fahrtrichtung und auf die Einstellungen, die für die einzelnen Phasen vorgegeben werden. Die Ergebnisse werden für den berechneten Korridor angezeigt und orientieren sich dabei an den Anforderungen aus Kapitel 3.2.1. Die Zusammenfassung eines Manövers ist in Abbildung 4.15 für ein Kurvenmanöver nach konventioneller Konstruktionsmethode zu sehen.

4.2 Entwurf der Modelldaten

In den Objekten des Bereichs Modelldaten befinden sich alle Daten, die für die jeweilige Funktionalität benötigt werden. Für alle Funktionalitäten werden die Daten benötigt, die ein Fahrzeug beschreiben. Auch bei der Konstruktion eines Manövers sind die Fahrzeugdaten für die Berechnung und Visualisierung eines Manövers von Bedeutung. Die Modelldaten sind Teil der GUI, über die der Benutzer mit den Modelldaten interagieren kann. In diesem Abschnitt werden die Bestandteile der Modelldaten, die für ein Manöver erforderlich sind, vorgestellt. Dazu wird mit der Klasse *ManoeverDaten* begonnen, in der auch alle weiteren Bestandteile der Modelldaten erzeugt und verwaltet werden.

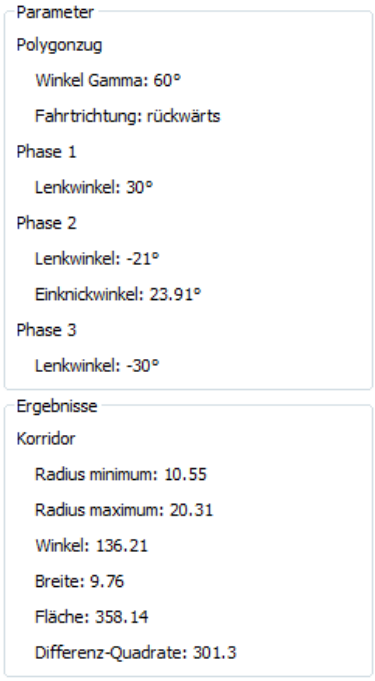


Abbildung 4.15: Die Zusammenfassung eines konventionellen Kurvenmanövers.

4.2.1 Die Hauptklasse der Modelldaten, *ManoeverDaten*

Die Klasse *ManoeverDaten* ist eine abstrakte Klasse, die direkt von der Klasse *ModellDaten* abgeleitet ist. In Abbildung 4.16 sind diese beiden Klassen zu sehen.

Von der Klasse *ModellDaten* wird die Fahrzeugdatenkette übernommen. Diese ist ein Objekt einer Klasse, die die Schnittstelle *sFahrzeugdatenkette* implementiert. Auf die Notwendigkeit, warum hier die Sicht einer Schnittstelle betrachtet wird, wird in Abschnitt 4.2.2 eingegangen.

In den *ManöverDaten* werden verschiedene Attribute verwaltet. Der Winkel Γ und die Fahrtrichtung müssen vom Benutzer vorgegeben werden und werden in den *ManoeverDaten* übernommen. Ebenso die Lenkwinkel und gegebenenfalls die Einknickwinkel für die einzelnen Phasen. Diese werden in einem Container gespeichert. So können bei der Konstruktion einer Phase die jeweiligen Winkel zugeordnet werden. Da die Phasenanzahl je nach Manövertyp und -methode unterschiedlich ist, wird auf Container zurückgegriffen, deren Längen dynamisch angepasst werden können. Auch die Attribute *translation*, *rotation*, *punktX* und *mittelpunkt* beziehen sich auf die einzelnen Phasen. In diesen Attributen befinden sich Werte, mit denen sich aus den einzelnen berechneten Phasen, mittels einer Trans-

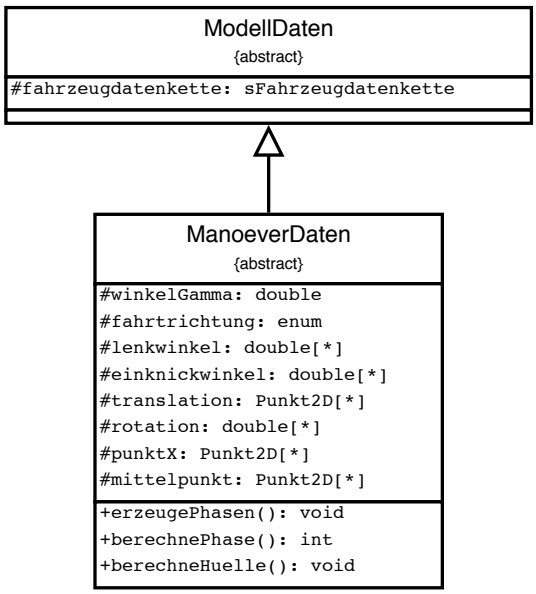


Abbildung 4.16: Die abstrakte Klasse *ManoeverDaten*.

formation, ein komplettes Manöver visualisieren lassen kann (siehe 2.4.1.3). Die einzelnen Phasen befinden sich in dem Container *phasen*, der Attribute der Schnittstelle *sPhase* enthält. Der Korridor befindet sich in einem Attribut der Schnittstelle *sHuelle*. Diese beiden Schnittstellen sind im Klassendiagramm der Manöverdaten im Anhang A.1 zu finden (es wird im Kapitel 4.2.3 genauer darauf eingegangen). Auch die Visualisierungen der Bestandteile eines Manövers werden in den Modelldaten vorgenommen. Dazu gibt es die Klasse *ManoeverVisualisierung*, von der ein Objekt existiert und dieses in der Klasse *ManoeverDaten* angelegt und verwaltet wird.

Die Methode *erzeugePhasen()* stößt die Berechnung eines Manövers an. Darin werden die einzelnen Phasen mit der Methode *berechnePhasen()* aufgerufen. In der Methode *berechneHuelle()* wird die Konstruktion des Manöverkorridors angestoßen. Da sich die Art der Berechnungen nach Manövertyp und Konstruktionsmethode unterscheidet, werden die Methoden in den konkreten Klassen, die von *ManoeverDaten* abgeleitet sind, implementiert.

In der Abbildung 4.17 sind diese Klassen zu sehen. Für ein Kurvenmanöver gibt es zwei Konstruktionsmethoden. Für eine konventionelle Konstruktion ist die Klasse *KurvenManoeverKonventionellDaten* zuständig. In der Methode *erzeugePhasen()* werden drei Phasen generiert, wobei die erste und die dritte Phase instabil sind und die zweite Phase eine stabile Phase ist. Hat der Benutzer die iterative

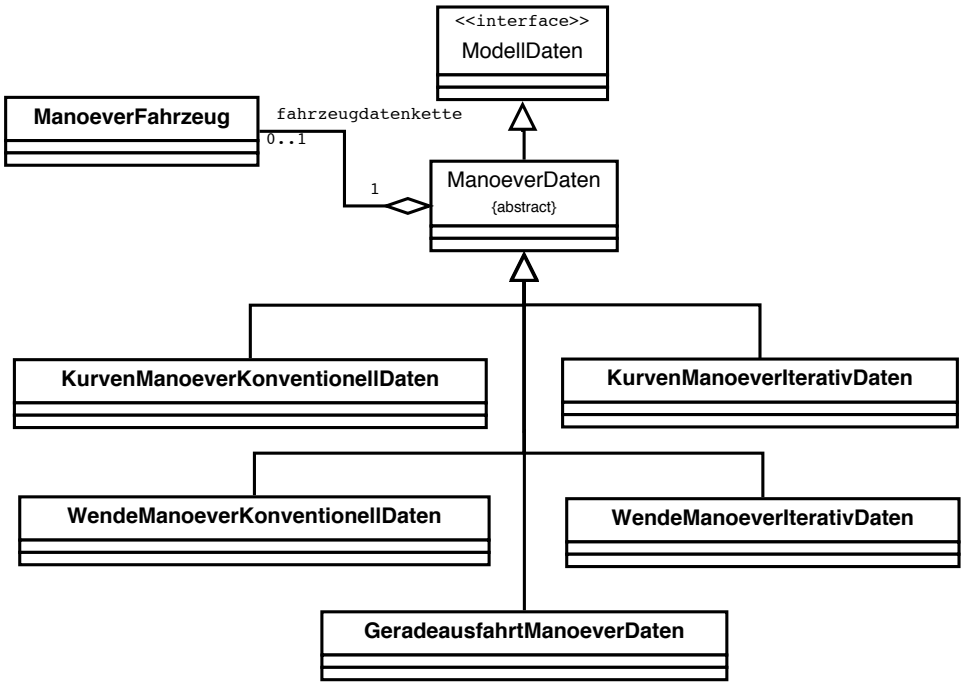


Abbildung 4.17: Die konkreten Klassen von *ManoeverDaten*.

Methode gewählt, werden fünf Phasen konstruiert. Dabei ist die dritte Phase eine stabile Phase, die anderen Phasen sind instabil. Diese Methode wird in der Klasse *KurvenManoeverIterativDaten* modelliert.

Auch für das Wendemanöver existieren zwei Konstruktionsmethoden. Die Klasse *WendeManoeverKonvetionellDaten* ist zuständig für die konventionelle Manöverkonstruktion. Bei dieser Methode gibt es fünf Phasen, wobei die zweite und vierte Phase stabil sind und es eine Fahrtrichtungsänderung während des Manövers gibt. Es gibt auch eine iterative Methode für das Wendemanöver. Für diesen Konstruktionstyp wird die Klasse *WendeManoeverIterativDaten* zuständig sein.

Als Letztes gibt es die Klasse *GeradeausfahrtManoeverDaten*, die das Manöver Geradeausfahrt realisiert. Da sich während des Manövers der Lenkwinkel nicht ändert, gibt es nur eine Phase, in der der Lenkwinkel und der Einknickwinkel 0° betragen. [1]

[1] Implementiert wurde während dieser Arbeit nur das konventionelle Kurvenmanöver. Auf die anderen Manövertypen und Konstruktionsmethoden wird nur im Entwurf eingegangen.

4.2.2 Die Daten des Gespanns in der Klasse *ManoeverFahrzeug*

Bevor auf die Konstruktion eines Manövers eingegangen wird, sollen zuerst die Daten, die das Fahrzeug beschreiben, betrachtet werden. Bei der Konstruktion eines Manövers ist dafür die Klasse *ManoeverFahrzeug* zuständig. In Abbildung 4.18 ist das Klassendiagramm von *ManoeverFahrzeug* mit den zugehörigen Bestandteilen zu sehen.

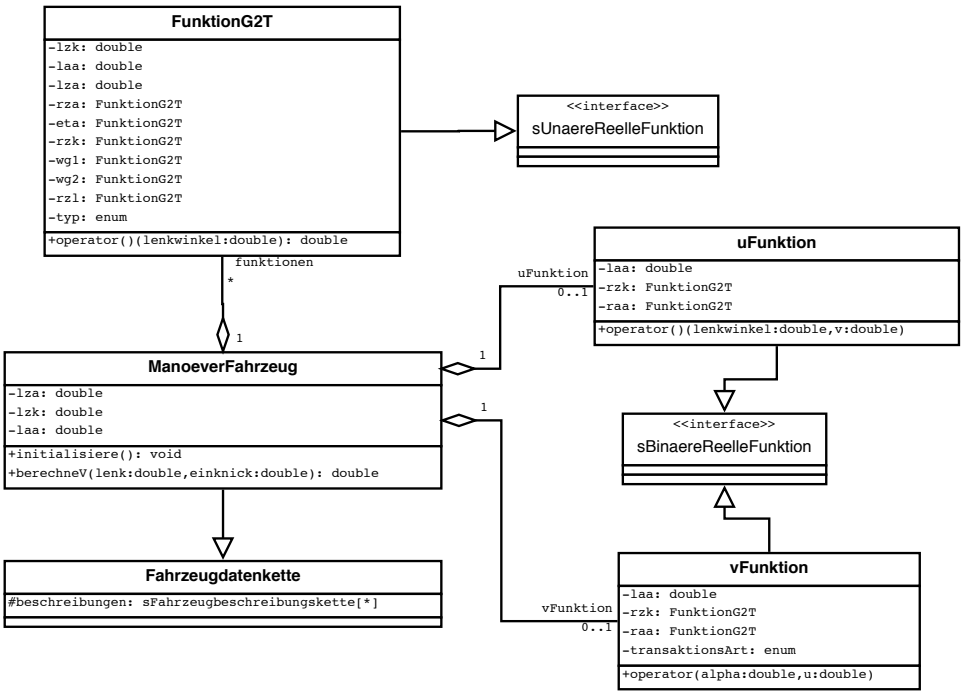


Abbildung 4.18: Die Klasse *ManoeverFahrzeug* mit ihren Bestandteilen.

Die Klasse *ManoeverFahrzeug* ist abgeleitet von der Klasse *Fahrzeugdatenkette* und implementiert genau wie die Elternklasse die Schnittstelle *sFahrzeugdatenkette*.[2] Mit der *Fahrzeugdatenkette* wird auch das Attribut *beschreibungen* geerbt, in der man die Fahrzeugbeschreibungen des Fahrzeugs findet. Der Zusammenhang zwischen der Fahrzeugdatenkette und der Fahrzeugbeschreibungskette wurde in Kapitel 2.4.1.2 erläutert.

In den Attributen der Klasse *ManoeverFahrzeug* werden die Abstände eines Fahrzeugs zwischen den Achsen und der Kupplung gespeichert. Diese Werte können direkt aus den Fahrzeugbeschreibungen berechnet werden, was in der Methode

[2] In zukünftigen Versionen wird vorgeschlagen, diese Beziehung zu ändern, um das Verhalten flexibler an neue Fahrzeuge anpassen zu können. Die Beziehung muss dann aber in allen GUI-Klassen angepasst werden.

initialisiere() geschieht. Das Attribut *lza* stellt den Abstand zwischen Vorder- und Hinterachse des Fahrzeugs, L_1, dar, *lzk* den Abstand zwischen Vorderachse und Kupplung, M_1. Das Attribut *laa* speichert den Abstand zwischen Kupplung und Anhängerachse, L_2.

In der Methode *initialisiere()* werden ebenfalls die Objekte der Klasse *FunktionG2T* erstellt. Diese Klasse stammt aus der Arbeit von Christian Weyand und ist abgeleitet von der Schnittstelle *sUnaereReelleFunktion*. Es handelt sich dabei um ein Funktionsobjekt, das in Kapitel 2.4.1.4 erläutert wurde. Die Namen für die Objekte sind in der Enumeration *typ* enthalten und beziehen sich dabei auf eine ältere Bezeichnung. Der Radius r_0 wird mit *rzl* bezeichnet, der Radius r_1 mit *rza*, der Radius r_2 mit *rzk* und der Radius r_3 mit *raa*. Auch die Winkel α_{M_1} (bezeichnet als *eta*), wg_1, wg_2, $\Delta\theta_{circ}$ können mit den Funktionen berechnet werden. Zusätzlich werden Berechnungen für Einknickwinkel bei $u = 0°$ bei den inneren und äußeren Traktionen benötigt. Diese werden als *wgu0aeT* für die äußere und *wgu0iT* für die innere Traktion bezeichnet. Für jede Funktion wird ein Funktionsobjekt angelegt. Die Berechnungen dazu wurden im Kapitel 2.1 eingeführt. Die Funktionen haben die Gemeinsamkeit, dass sie alle abhängig vom Lenkwinkel α berechnet werden. Dieser Wert wird den Funktionsobjekten als Funktionsargument übergeben. Die Konstruktion der Funktionsvorschrift ist durch die Formeln festgelegt. Die benötigten Werte werden über verschiedene Konstruktoren gesetzt, denen wiederum Funktionsobjekte der Klasse *FunktionG2T* übergeben werden können.

Des Weiteren werden auch für die Berechnungen der Winkel u und v Funktionsobjekte angelegt. Diese werden durch die Klasse *uFunktion* und *vFunktion* realisiert. Im Gegensatz zu den Berechnungen der Klasse *FunktionG2T* wird zusätzlich zu dem Lenkwinkel ein zweites Funktionsargument benötigt. Für die Berechnung von u nach der Formel 2.20 wird der Winkel v benötigt. Für die Berechnung von v nach den Formeln 2.21 für eine äußere Traktion, bzw. 2.22 für eine innere Traktion, wird zusätzlich der Winkel u benötigt. Da zwei Funktionsargumente für die Berechnung übergeben werden, wird in diesen Fällen die Schnittstelle *sBinaereReelleFunktion* implementiert.

Bei der Erzeugung des Funktionsobjekts der Klasse *uFunktion* wird der Abstand zwischen Kupplung und Anhängerachse, L_2, gespeichert. Für die Berechnung werden die Radien r_2 und r_4 benötigt, die wiederum abhängig vom Lenkwinkel sind und mit den entsprechenden Objekten der Klasse *FunktionG2T* berechnet werden können. Auch diese Objekte werden im Konstruktor übergeben.

Bei der Erzeugung des Objekts der Klasse *vFunktion* werden die gleichen Para-

meter benötigt, wie bei der Berechnung von u. Zusätzlich muss übergeben werden, ob es sich um eine äußere oder eine innere Traktion handelt, was in dem Attribut *traktionsArt* gespeichert wird.

Für die Berechnung des Winkels v gibt es eine weitere Formel, die, abhängig vom aktuellen Einknickwinkel $\Delta\theta$, dem Lenkwinkel α und dem Winkel α_{M_1} ist. Die Formel dazu ist unter 2.19 zu finden. Für diese Berechnung gibt es in der Klasse *ManoeverFahrzeug* die Methode *berechneV()*. Übergeben wird dabei der aktuelle Lenk- und der aktuelle Einknickwinkel. Der Wert α_{M_1} wird über das entsprechende Objekt von *FunktionG2T* berechnet. Diese Methode kann immer dann verwendet werden, wenn der Einknickwinkel bekannt ist. Das ist während der stabilen Fahrt der Fall, aber auch an den Grenzen einer instabilen Phase.

Das Objekt der Klasse *ManoeverFahrzeug* wird nach der Auswahl von Konstruktionstypen und -methoden mit allen Bestandteilen erzeugt. Für die Konstruktion eines Manövers werden dann die Funktionsargumente übergeben.

4.2.3 Entwurf der Manöverkonstruktion

Der Entwurf eines Manövers erfolgt unabhängig von den verschiedenen Typen und Methoden. Das Klassendiagramm mit allen Bestandteilen der Manöverdaten ist im Anhang unter Abbildung A.1 zu finden. Hier wird nun auf die einzelnen Bestandteile eingegangen. Eine Manöverkonstruktion erfolgt in zwei Schritten. Im ersten Schritt werden Fahrkurven für bestimmte Bezugspunkte konstruiert. Im zweiten Schritt wird der Korridor für das Manöver berechnet.

4.2.3.1 Konstruktion der Phasen eines Manövers

Ein Manöver wird in verschiedene Phasen aufgeteilt und besteht aus mindestens einer Phase. In Abbildung 4.19 ist der Entwurf für die Behandlung von Phasen innerhalb eines Manövers als Klassendiagramm zu sehen.

In der Klasse *ManoeverDaten* werden die verschiedenen Phasen erzeugt und verwaltet. Eine Phase ist immer abgeleitet von der abstrakten Klasse *sPhase*. Diese Klasse vererbt verschiedene Attribute. Diese sind teilweise mit einem Index ausgestattet. Der Index 0 beschreibt dabei den Anfang einer Phase und der Index 1 das Phasenende. In den Attributen *deltaTheta0* und *deltaTheta1* sind die Einknickwinkel enthalten, in *u0* und *u1* die Winkel u am Start und Ziel einer Phase und mit den Attributen *ausrichtungAnhaenger0* und *ausrichtungAnhaenger1* wird die Ausrichtung des Anhängers an den jeweiligen Grenzen gespeichert. Da eine Phase einen

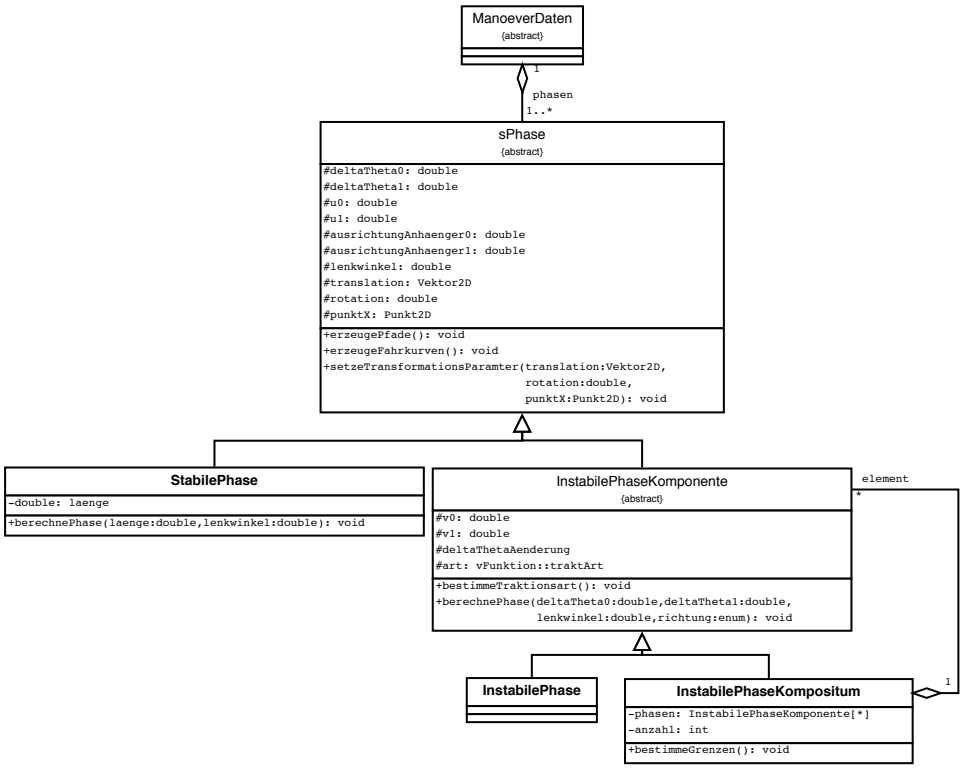


Abbildung 4.19: Entwurf der einzelnen Phasen eines Manövers.

festen Lenkwinkel hat, ist dieser im Attribut *lenkwinkel* gespeichert. In den Attributen *translation*, *rotation* und *punktX* werden Informationen gespeichert, die das Ende der letzten Phase beschreiben. Somit kann durch eine Transformation dafür gesorgt werden, dass die Manöverphasen in der richtigen Reihenfolge aneinander gereiht werden. Diese Informationen werden durch die Methode *setzeTransformationsParameter()* gesetzt.

Die Methoden *erzeugePfade()* und *erzeugePfade()* erzeugen die Objekte für Pfade und Fahrkurven. Ein Pfad stellt den Weg eines Bezugspunktes während der Phase dar. Die Bezugspunkte beziehen sich auf die Mittelpunkte der drei Achsen des Gespanns und die Kupplung. Die Fahrkurven werden für alle Extrempunkte konstruiert. Dazu zählen die Ecken und Räder von Zugfahrzeug und Anhänger, sowie die Kupplung. Diese Fahrkurven können zum einen visualisiert werden, zum anderen werden sie aber auch für die Berechnung des Korridors benötigt.

Von der abstrakten Klasse *sPhase* ist die Klasse *StabilePhase* abgeleitet. Für die Konstruktion einer stabilen Phase wird der Lenkwinkel und die Länge der Phase an die Methode *berechnePhase()* übergeben. Innerhalb der Methode werden

anhand der Länge die Objekte für die Pfade und Fahrkurven konstruiert.

Für die iterative Konstruktion wurde bereits in Kapitel 2.3.1 erläutert, dass sich der Lenkwinkel während einer instabilen Phase schrittweise ändert. In der Definition einer Phase heißt es aber, dass diese einen gleichbleibenden Lenkwinkel hat. Um das Verhalten zu modellieren, wird eine Phase daher in mehrere Teilphasen aufgeteilt. In diesen Teilphasen ist der Lenkwinkel jeweils gleichbleibend. Im Softwareentwurf wurde dazu das in Kapitel 2.6.2 Kompositum-Entwurfsmuster angewendet. Die Komponente stellt dabei die Klasse *InstabilePhaseKomponente* dar. Dies ist eine abstrakte Klasse, in der das Verhalten einer instabilen Phase definiert wird.

Zusätzlich zu den Attributen, die von der Klasse *sPhase* übernommen werden, gehört zu allen Objekten der instabilen Phase, also den Objekten die, unabhängig von der Konstruktionsart, von der abstrakten Klasse *InstabilePhaseKomponente* abgeleitet sind, der Winkel v zu Beginn und am Ende der Phase. Auch die Änderung der Ausrichtung $d\theta$ wird hier als Attribut verwaltet. Ebenso muss die Traktionsart für die Berechnung der Phase ermittelt werden. Dafür gibt es die Methode *bestimmeTraktionsart()*, die das Ergebnis in dem Attribut *art* speichert. Die Berechnung der Phase in der Methode *berechnePhase()* benötigt den Lenkwinkel und die Einknickwinkel für den Start und das Ende der Phase als Parameter. Mit den Einknickwinkeln können die Winkel u und v am Anfang und am Ende der Phase berechnet werden. Die Konstruktion eines Manövers ist für die Rückwärtsfahrt definiert, für die Fahrtrichtung vorwärts werden die Einknickwinkel in umgekehrter Reihenfolge betrachtet, daher muss auch die Richtung übergeben werden.

Von der abstrakten Komponentenklasse *InstabilePhaseKomponente* gibt es zwei Implementationen, *InstabilePhase* und *InstabilePhaseKompositum*. In der Klasse *InstabilePhase* werden die Methoden wie oben beschrieben implementiert. Für die konventionelle Konstruktion wird nur diese Klasse benötigt. Bei der iterativen Konstruktion wird zusätzlich die Klasse *InstabilePhaseKompositum* benötigt. Auch diese Klasse ist von *InstabilePhaseKomponente* abgeleitet. Die Klasse dient allerdings als Container für weitere Elemente der Klasse *InstabilePhaseKomponente*. Es gibt also für jede instabile Phase eines iterativen Manövers ein Objekt der Kompositumklasse, in dem die Objekte der Klasse *InstabilePhase* für jeweils eine Iteration konstruiert werden. Im Kompositum werden die Anzahl der Iterationen bestimmt, ebenso die Grenzen der einzelnen Iterationen. Die Methode *berechnePhase()* wird dabei iterativ für alle enthaltenen Komponenten ausgeführt.

In Abbildung 4.20 ist die Beziehung einer Phase zu den Pfaden und Fahrkurven

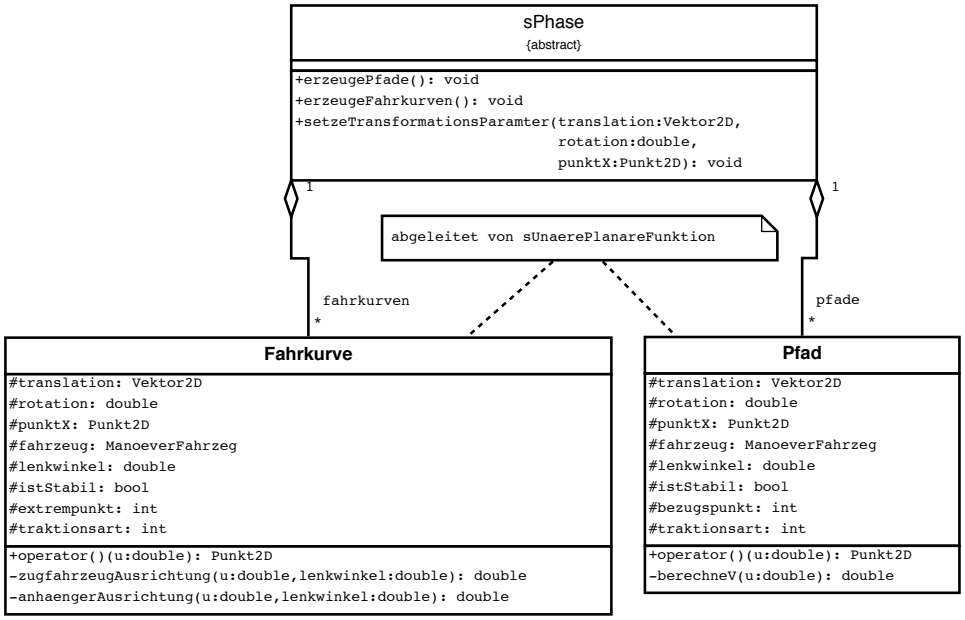


Abbildung 4.20: Pfade und Fahrkurven einer Phase.

zu sehen. Die Klassen *Pfad* und *Fahrkurve* sind als Klassen für Funktionsobjekte angelegt und erben daher von der Schnittstelle *sUnaerePlanareFunktion*. Planar bedeutet, dass die Rückgabe des Funktionsobjekts ein zweidimensionaler Punkt ist, dargestellt durch ein Objekt der Klasse *Punkt2D*.

In der Klasse *Pfad* wird die Position der Mittelpunkte der Achsen des Gespanns und die Position der Kupplung berechnet. Ein Funktionsobjekt dieser Klasse wird dabei für genau einen Bezugspunkt angelegt. Für die Achsen des Zugfahrzeugs und die Kupplung werden dabei die Formeln 2.5 bis 2.10 angewendet. Die Achsen des Anhängers werden mit den Formeln 2.17 und 2.18 berechnet. Bei der Erzeugung eines Objekts der Klasse *Pfad* wird das Objekt der Klasse *ManoeverFahrzeug* mitgegeben. Die darin enthaltenen Funktionen zur Berechnung von Radien und Winkel, werden auch für die Bestimmung der Position benötigt. Ebenso ist der Lenkwinkel der Phase bekannt und die Art der Phase (stabil oder instabil). Die Traktionsart wird benötigt um zu bestimmen, mit welcher Formel der Winkel v berechnet wird, der für die Position des Anhängers benötigt wird. Als Funktionsargument wird der Winkel u eingesetzt. Dieser Winkel beschreibt die Lage der Kupplung bezogen auf den Koordinatenursprung des gewählten Koordinatensystems. Die Kupplung bewegt sich während einer Phase auf einem Kreisbogen um diesen Ursprung. Mit dem Funktionsargument kann nun zwischen den beiden

Grenzen u_0 und u_1 einer Phase jede beliebige Position berechnet werden.

Die Klasse *Fahrkurve* ist analog zur Klasse *Pfad* aufgebaut. Auch hier wird die Position bestimmter Punkte des Fahrzeugs berechnet. Im Gegensatz zu *Pfad* werden die Extrempunkte eines Fahrzeugs betrachtet, die Auswirkungen auf den Manöverkorridor haben können. Für einige dieser Punkte ist die Ausrichtung des Zugfahrzeugs oder des Anhängers von Bedeutung. Diese können mit den Methoden *zugfahrzeugAusrichtung()* und *anhaengerAusrichtung()* bestimmt werden.

Der Rückgabetyp beider Funktionsobjekte ist die Klasse *Punkt2D*. Darin sind die x- und y-Koordinaten der Position des gewünschten Bezugspunkts an der Stelle u enthalten. Jede Phase wird im Bezug zum Koordinatenursprung gebildet. Daher ist es notwendig, die Phasen nach der Berechnung hintereinander zu setzen (siehe 2.4.1.3). Die Transformationsparameter in den Attributen *translation*, *rotation* und *punktX* beziehen sich dabei auf das Ende der vorherigen Phase und müssen vor dem Berechnen einer Position gesetzt werden. In den zurückgegebenen Ergebnissen ist diese Transformation schon berücksichtigt.

4.2.3.2 Konstruktion des Manöverkorridors

Im vorigen Abschnitt wurde bei den Fahrkurven schon darauf hingewiesen, dass diese für die Berechnung eines Korridors benötigt werden. Bei der Konstruktion eines Korridors werden Daten für die einzelnen Phasen berechnet. Diese Daten, die auf eine Phase begrenzt sind, werden als Korridor-Konfiguration bezeichnet. Der Manöverkorridor selbst gilt aber für das gesamte Manöver. In diesem Abschnitt wird der Entwurf für die in den Grundlagen vorgestellten Methoden der Konstruktion eines ringförmigen Korridors und der Konstruktion eines Korridors als Polygonhülle gezeigt.

In Abbildung 4.21 ist der Entwurf der Klassen zu sehen. Die Hülle ist Teil der Klasse *ManoeverDaten* und wird durch ein Objekt einer Klasse repräsentiert, die von der abstrakten Klasse *sHuelle* abgeleitet ist. Bei der Konstruktion eines Manövers hat der Benutzer sich schon zu Beginn des Programms auf eine Konstruktionsmethode für den Korridor festgelegt. Unabhängig von der Methode sind die Attribute für den Flächeninhalt, der Breite des Korridors und dem Wert der als prozentualer Anteil des Sicherheitskorridors angegeben wird. Auch die Objekte der verschiedenen Phasen des Manövers sind für die Berechnung notwendig und Teil der Klasse *sHuelle*.

Von der abstrakten Klasse gibt es zwei konkrete Implementationen, die Klasse *RingHuelle* für die Konstruktion eines ringförmigen Korridors (siehe Kapitel

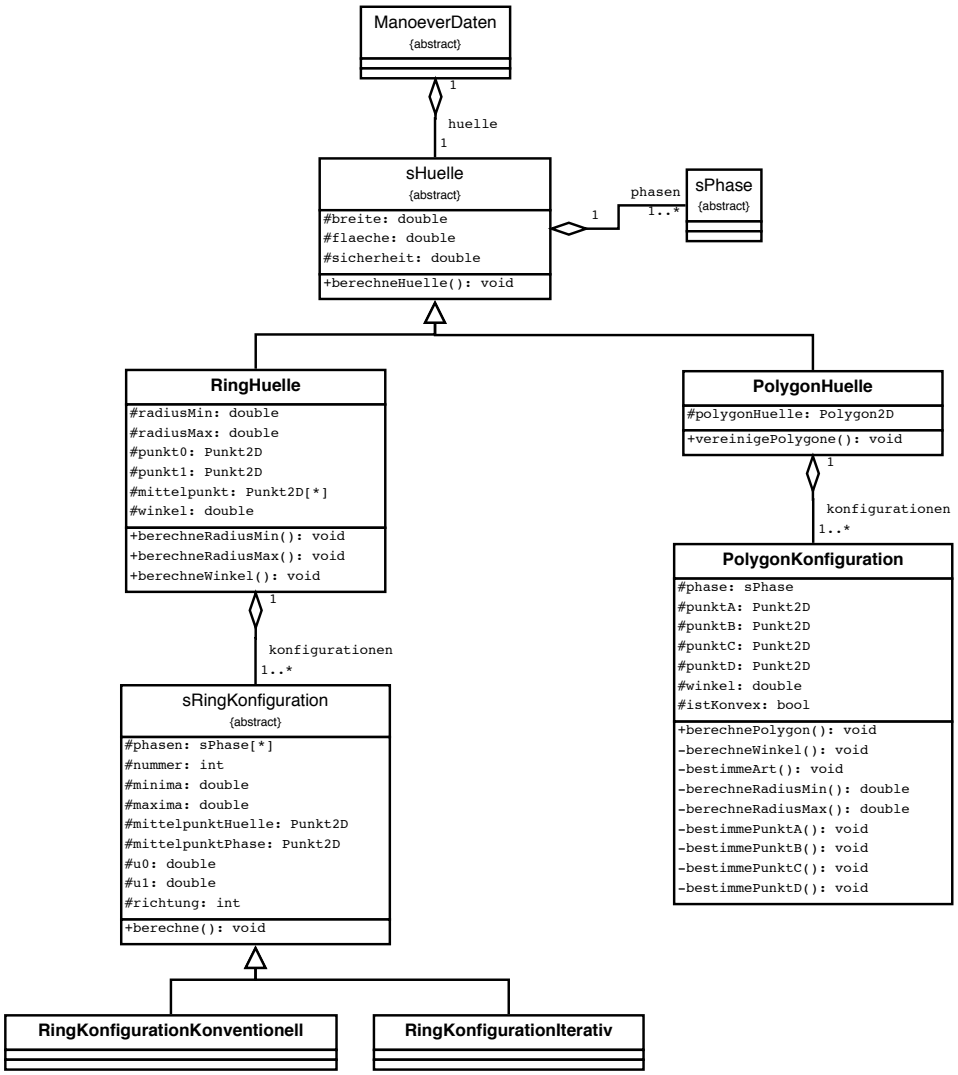


Abbildung 4.21: Entwurf des Konzepts für die verschiedenen Korridormethoden.

2.3.2.2 und 2.3.2.4) und die Klasse *PolygonHuelle* für die Konstruktion als Vereinigung von Polygonen (siehe Kapitel 2.3.5). [3] Bei beiden Klassen wird die Berechnung mit der Methode *berechneHuelle()* angestoßen.

Konstruktion einer Ringhülle

Die Klasse *RingHuelle* hat mehrere zusätzliche Attribute. In den Attributen *radiusMin* und *radiusMax* findet man den berechneten minimalen und maximalen Radius der Ringhülle. In *punkt0* wird der Extrempunkt gespeichert, der den An-

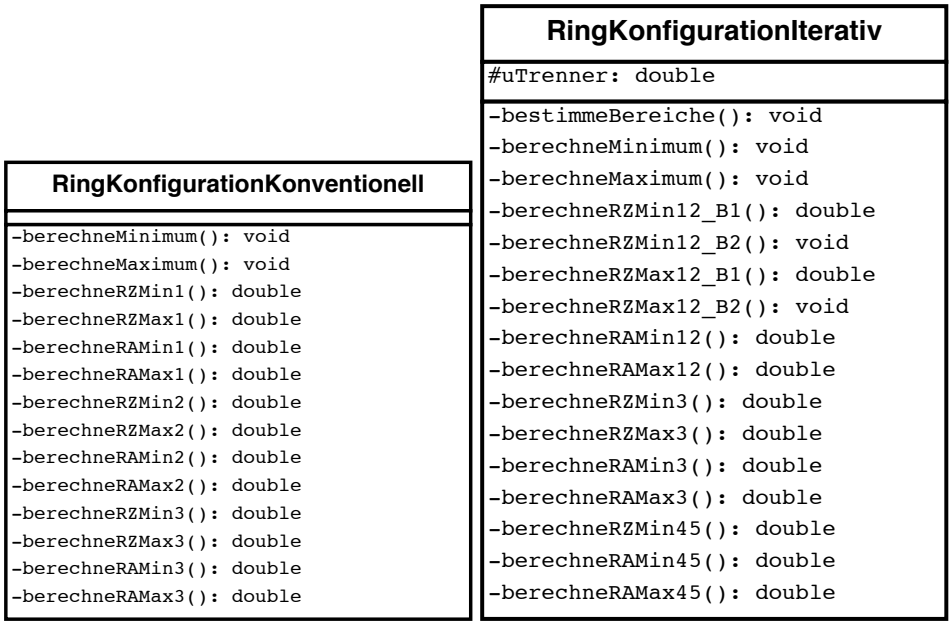
[3] Die Polygonhülle wurde im Rahmen der Diplomarbeit nur im Entwurf berücksichtigt.

fang der Ringhülle bei Phase 1 begrenzt, *punkt1* ist der Punkt der Begrenzung in der letzten Phase. Zwischen diesen Punkten und dem Mittelpunkt der Hülle kann der Winkel der Kreisbögen mit der Methode *berechneWinkel()* bestimmt werden. Der berechnete Wert wird im Attribut *winkel* gespeichert. Die Mittelpunkte der einzelnen Phasen sind im Container *mittelpunkte* enthalten. Dabei wird von den schon transformierten Punkte ausgegangen.

Um die beiden Radien der Kreisbögen zu bestimmen, wird für jeden Radius durch die einzelnen Phasen iteriert und dabei jeweils der minimale und maximale Abstand berechnet. Für diese Aufgabe gibt es für jede Phase (im Fall der iterativen Methode ggf. auch für mehrere Phasen) ein Objekt einer Klasse, die die abstrakten Klasse *sRingKonfiguration* implementiert. In dieser abstrakten Klasse werden für die Phasen die einzelnen Kandidaten nach den Methoden bestimmt, die in den Grundlagen vorgestellt wurden. Dem Objekt wird dafür die Phase übergeben, wobei insbesondere die Funktionsobjekte der Klasse *Fahrkurve* von Bedeutung sind. Darin können an den beschriebenen Stellen einer Phase für bestimmte Extrempunkte die Positionen berechnet werden. Mit diesen Positionen können die Abstände der jeweiligen Extrempunkte zu den Mittelpunkten für die Radien berechnet werden. Die Ringkonfiguration muss dafür wissen, um welche Phase es sich handelt, was im Attribut *nummer* gespeichert wird. Auch die Fahrtrichtung wird dem Objekt übergeben, ebenso die beiden Grenzen, die mit dem Winkel u am Anfang und am Ende der Phase bestimmt werden. Für die Berechnung der Abstände ist der Mittelpunkt der Hülle notwendig, bei einigen Kandidaten zusätzlich der Mittelpunkt der jeweiligen Phase. Mit der Methode *berechne()* werden für die Phase die Kandidaten berechnet und anschließend das gefundene Minimum und Maximum in den Attributen gespeichert.

Eine Ringhülle kann nur für das Kurvenmanöver konstruiert werden. Bei der Konstruktion muss aber zwischen der konventionellen und der iterativen Konstruktionsmethode unterschieden werden.

In Abbildung 4.22 ist der Unterschied der beiden Klassen *RingKonfiguration-Konventionell* und *RingKonfigurationIterativ* zu sehen. Bei der konventionellen Methode wird nach der in Kapitel 2.3.2.2 beschriebenen Konstruktion vorgegangen. Das Minimum und Maximum wird dabei jeweils für eine Phase berechnet. Bei der Berechnung eines Kandidaten für den minimalen Radius wird für die Phase, die in dem Objekt der Klasse *RingKonfigurationKonventionell* behandelt wird, der minimale Abstand des Zugfahrzeugs und des Anhängers berechnet und anschließend bestimmt, welcher Wert der kleinere ist. Nur dieser wird in dem Attribut *minimum*



(a) Konventionelle Methode (b) Iterative Methode

Abbildung 4.22: Konfiguration einer Ringhülle im Vergleich zwischen konventioneller und iterativer Methode.

(aus der abstrakten Klasse *sRingKonfiguration*) gespeichert. Analog wird für den Kandidaten des maximalen Radius vorgegangen. Nachdem alle Phasen berechnet wurden, wird das Minimum und das Maximum aller Phasen bestimmt und in dem Objekt der Klasse *RingHuelle* gespeichert.

In der Klasse *RingKonfigurationIterativ* wird die Ringkonstruktion von iterativen Manöverphasen vorgenommen. Ein Unterschied bei der iterativen Konstruktionsmethode ist, dass Phasen zusammen betrachtet und eventuell in phasenunabhängige Bereiche eingeteilt werden. Für diese Bereiche wird mit der Methode *bestimmeBereiche()* die Grenze zwischen den Bereichen gesucht und in dem Attribut *uTrenner* gespeichert. Das Vorgehen bei der Suche nach den Abständen richtet sich dabei nach der in Kapitel 2.3.2.4 beschriebenenKonstruktion.

Konstruktion einer Polygonhülle

Die Konstruktion des Manöverkorridors als Polygonhülle wurde in Kapitel 2.3.5 vorgestellt. Der Softwareentwurf kann dem Klassendiagramm in Abbildung 4.21 entnommen werden. Bei dieser Konstruktionsmethode wird zuerst um jede Phase ein Trapez gelegt, das das Manöver während der Phase umschließt. Die Berech-

nung dieses Trapezes wird in der Klasse *PolygonKonfiguration* vorgenommen. In dem Objekt dieser Klasse ist die zu berechnende Phase als Objekt von *sPhase* enthalten. Mit der Methode *bestimmeArt()* wird überprüft, ob die Phase konvex oder konkav ist. Das Ergebnis wird in dem Attribut *istKonvex* gespeichert. Die Suche nach dem minimalen und dem maximalen Radius ist abhängig von der Art der Phase. Die Methode *berechneRadiusMin()* berechnet dabei alle Kandidaten für das Minimum und gibt schließlich den minimalen Radius der Phase zurück. Die Methode *berechneRadiusMax()* führt die Berechnung des maximalen Radius durch. In der Methode *berechneWinkel()* wird der maximale Winkel der Phase, bezogen auf den Mittelpunkt und die Grenzen am Anfang und am Ende der Phase, berechnet.

Diese Berechnungen sind notwendig, um die Punkte A, B, C und D des Trapezes zu finden. Die Methoden werden in der Methode *berechnePolygon()* aufgerufen. Anschließend werden mit den gleichnamigen Methoden die einzelnen Punkte bestimmt und in den jeweiligen Attributen gespeichert.

Die Objekte der Klasse *PolygonKonfiguration* werden in der Klasse *PolygonHuelle* erzeugt. Dort werden auch die Berechnungen für jede Phase einzeln vorgenommen. Um die Polygonhülle des gesamten Manövers zu bilden wird die Methode *vereinigePolygone()* aufgerufen. Dort wird mithilfe der Bibliothek EZauto die Vereinigung über alle Polygone gebildet. Das Ergebnis der Vereinigung ist wieder ein Polygon, das im Attribut *polygonHuelle* gespeichert wird. Dabei handelt es sich um den Korridor des Manövers.

4.2.4 Visualisierung des Manövers

Die Visualisierung geschieht über die konkreten Objekten der abstrakten Klasse *ManoeverDaten*. Es werden dabei für die unterschiedlichen Bestandteile eines Manövers jeweils spezielle Methoden aufgerufen. Zu den Bestandteilen gehören die Fahrkurven, die Pfade und der Manöverkorridor. Zusätzlich gibt es aber auch Bestandteile, die die Manövervisualisierung verdeutlichen sollen. Dazu gehören der Polygonzug (die Geraden, auf denen sich das Fahrzeug vor und nach dem Manöver befindet), dem Fahrzeug selbst an bestimmten Stellen des Manövers und die Hervorhebung verschiedener Punkte des Manövers, zum Beispiel die Phasenübergänge. Nachdem das Manöver konstruiert wurde, wird auch die Visualisierung aller Bestandteile erstellt.

Für die Visualisierung verschiedener Bestandteile gibt es in der Softwarebibliothek EZauto bereits Klassen, die diese Aufgabe übernehmen. Diese beziehen sich jeweils auf die verwendeten Funktionsobjekte. Im Rahmen dieser Arbeit wurde auf

die Klasse *UnaerePlanareFunktionsvisualisierung* zurückgegriffen, die einen Kurve dadurch visualisieren kann, dass sie verschiedene Punkte miteinander verbindet. Diese Punkte können aus den Funktionsobjekten der Klassen *Pfad* und *Fahrkurve* ermittelt werden. Dazu wird ein Objekt der Klasse *DiskreteUnaerePlanareFunktion* erzeugt, die sich ebenfalls in EZauto befindet. In einer Schleife wird zwischen den beiden Grenzen einer Phase, die von dem Winkel u bestimmt sind, mit einer gegebenen Schrittweite iteriert. Bei den Funktionsobjekte wird der überladene Klammeroperator aufgerufen und als Parameter der Winkel u übergeben. Zurückgegeben wird der Punkt mit den Koordinaten für den jeweiligen Bezugspunkt. Dieser Punkt wird im Objekt für die diskrete Funktion gespeichert. Schließlich wird diese Funktion an das Objekt der Klasse *UnaerePlanareFunktionsvisualisierung* übergeben, ebenso Anfang und Ende der diskreten Funktion und die Schrittweite. Diese Klasse enthält die Methode *zeichne()*, die für die Visualisierung aufgerufen wird. Die übergebenen Punkte, die in der richtigen Reihenfolge in x-Richtung angegeben sind, werden in dieser Methode durch Polygonzüge miteinander verbunden. Durch eine geringe Schrittweite entsteht der Eindruck einer Kurve, die den jeweiligen Pfad visualisiert.

Auch für der Visualisierung eines ringförmigen Manöverkorridors werden Objekte der Klasse *UnaerePlanareFunktionsvisualisierung* angelegt. Die Werte, die in der Methode *zeichne()* als Stützstellen verwendet werden, werden dabei auch von der Klasse *DiskreteUnaerePlanareFunktion* verwaltet. Die Daten stammen aber aus keinen Funktionsobjekten. Stattdessen werden die Bestandteile der Korridorvisualisierung anhand der bei der Korridorberechnung ermittelten Daten in der Klasse *ManoeverDaten* konstruiert. Die dafür nötigen Parameter sind der minimale und der maximale Radius des Rings sowie der ermittelte Winkel des Ringausschnitts.

Für die Kreisbögen der beiden Radien werden innerhalb des ermittelten Winkels die Koordinaten einzelner Punkte auf den Kreisbögen berechnet. Es fehlen noch die seitlichen Begrenzungen des Ringausschnitts. Auch für diese werden diskrete Objekte angelegt. Am Anfang des Manövers werden dazu die beiden Anfangspunkte der Kreisbögen als Stützstellen betrachtet. Die Begrenzungen verlaufen auf einer geraden Linie zwischen den beiden Punkten. Analog dazu werden am Ende des Manövers die beiden Endpunkte der Kreisbögen betrachtet.

Für einen ringförmigen Korridor werden bei der Visualisierung bis zu vier Objekte der Klasse *UnaerePlanareFunktionsvisualisierung* angelegt. Zwei davon beinhalten die Visualisierung der beiden Radien. Besteht der Korridor nicht aus ei-

nem Teilring, sondern aus einem geschlossenen Ring (vgl. Kapitel 2.3.2.2), werden zwei Objekte für die Begrenzungen an den Seiten des Ringausschnitts angelegt.

Beim Anlegen der Visualisierungen werden keine Darstellungskonfigurationen erzeugt. Damit diese berücksichtigt werden können, wird das, in Kapitel 2.6.1 vorgestellte, Adapter-Entwurfsmuster angewendet. Der Adapter, hier *VisuWrapper* genannt, ist eine Template-Klasse und kann also für Objekte verschiedener Art erzeugt werden. In Abbildung 4.23 ist die Adapterklasse im Kontext der Manöverdaten in einem Klassendiagramm zu sehen.

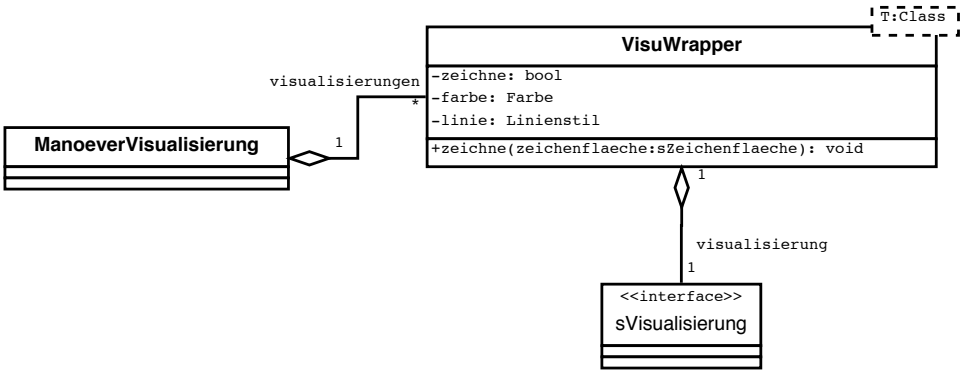


Abbildung 4.23: Die Klasse *VisuWrapper* mit ihren Aggregationen

Bei der Erstellung eines VisuWrapper-Objekts wird mit angegeben, welche Klasse betrachtet wird. Bei einem *VisuWrapper* muss es sich dabei um eine Klasse handeln, die die Schnittstelle *sVisualisierung* implementiert. Die Klasse *VisuWrapper* implementiert dabei auch wieder diese Schnittstelle, die sich insbesondere durch die Methode *zeichne()* auszeichnet. In dieser Methode wird wiederum die *zeichne()*-Methode des gekapselten Objekts aufgerufen.

Durch den Adapter können aber noch weitere Eigenschaften berücksichtigt werden, die mit den übergebenen Objekte nicht gesetzt werden können. Die erste Eigenschaft ist dabei, ob das übergebene Objekt überhaupt angezeigt wird. Dies wird im Attribut *zeichne* gesetzt. Außerdem sollen in Zukunft Farbe im Attribut *farbe* und Linienart im Attributlinie gesetzt werden können. In der *zeichne()*-Methode der Wrapper-Klasse wird zuerst überprüft, ob das Objekt angezeigt werden soll, danach wird für die Zeichenfläche die Farbe und Linienart gesetzt. Die Methode *zeichne()* des übergebenen Objekts wird nur aufgerufen, wenn das Attribut *zeichne true* ist.

Das Objekt der Klasse *UnaerePlanareFunktionsvisualisierung* wird zusammen mit Standardeigenschaften für den einzelnen Bestandteil an das Objekt der Klasse

VisuWrapper übergeben. Dieses wird nun in dem Objekt der Klasse *ManoeverVisualisierung* gespeichert. Diese Klasse dient als Container für alle Visualisierungen, die zu dem Manöver gehören und wird in Abbildung 4.24 als Klassendiagramm dargestellt.

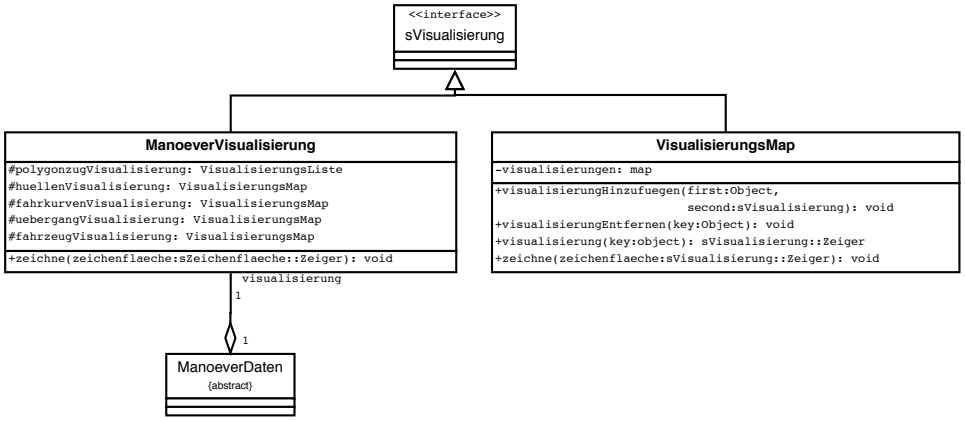


Abbildung 4.24: Entwurf des Containers für die verschiedenen Visualisierungen.

Die Klasse *VisualisierungsListe* ist schon Teil von EZauto. Darin können verschiedene Objekte der Schnittstelle *sVisualisierung* gespeichert sein. Beim Aufruf der *zeichne()*-Methode wird über alle Elemente der Liste iteriert und auch dort jeweils die Methode *zeichne()* aufgerufen. Dieser Container ist ausreichend für die Visualisierung der Geraden des Polygonzugs. Änderungen der Visualisierungskonfiguration dieses Bestandteils sollen sich immer auf alle Geraden auswirken. Für die anderen Bestandteile eines Manövers muss der Container um einen Schlüssel erweitert werden, da mehrere Objekte eines Bestandteils im Darstellungsteil gruppiert werden (siehe Kapitel 4.3). Um diese Gruppierungen vornehmen zu können, ist ein Zugriff auf bestimmte Visualisierungen nötig. Im Objekt *fahrkurvenVisualisierung* sind zum Beispiel die Visualisierungen der Fahrkurven von den Ecken des Zugfahrzeugs in einem Objekt zusammengefasst. Dieses Objekt wird mit dem Schlüssel *zugEcken* angelegt. Mit diesem Schlüssel kann auch später auf das Objekt zugegriffen werden. Dies wird in der Klasse *VisualisierungsMap* realisiert. Die Elemente dieses Containers bestehen aus einem Schlüssel und dem Objekt einer Klasse, die die Schnittstelle *sVisualisierung* implementiert. Das Objekt kann also sowohl eine einzelne Visualisierung sein, aber auch wieder ein Visualisierungscontainer. In der Klasse gibt es Methoden zum Hinzufügen neuer Elemente, zum Entfernen von Elementen oder zur Rückgabe eines Elements mit Hilfe des Schlüssels. Auch hier wird in der Methode *zeichne()* durch die einzelnen Elemente iteriert

und bei diesen die Methode aufgerufen.

Die Klasse *ManoeverVisualisierung* enthält die Visualisierung des Polygonzugs als *VisualisierungsListe*. Die folgenden Visualisierungen sind alle in Objekten des Containers *VisualisierungsMap* gespeichert: die Hüllen in *huellenVisualisierung*, die Pfade und Fahrkurven in *fahrkurvenVisualisierung*, die Übergänge der Phasen in *uebergaengeVisualisierung* und die Darstellung des Fahrzeugs an verschiedenen Stellen in *fahrzeugVisualisierung*. In der Methode *zeichne()* wird diese Methode in jedem dieser Container aufgerufen und somit die Visualisierung des kompletten Manövers nach den Benutzervorgaben vorgenommen.

4.3 Entwurf der Darstellungskonfiguration

Die Visualisierung eines Manövers wurde im vorigen Abschnitt beschrieben und ist Teil der Manöverdaten. In diesem Abschnitt geht es darum, die Visualisierung mit der Benutzerschnittstelle zu verbinden und es so dem Benutzer zu ermöglichen, die Visualisierung den eigenen Anforderungen anzupassen. Dieser Verbindungsteil wird kurz Darstellung genannt. Außerdem können in der Darstellung verschiedene Bestandteile eines Manövers gruppiert werden und damit die gleiche Darstellungskonfiguration erhalten. Die Klassen, die für ein Manöver in diesem Teil benötigt werden, sind in Abbildung 4.25 zu sehen.

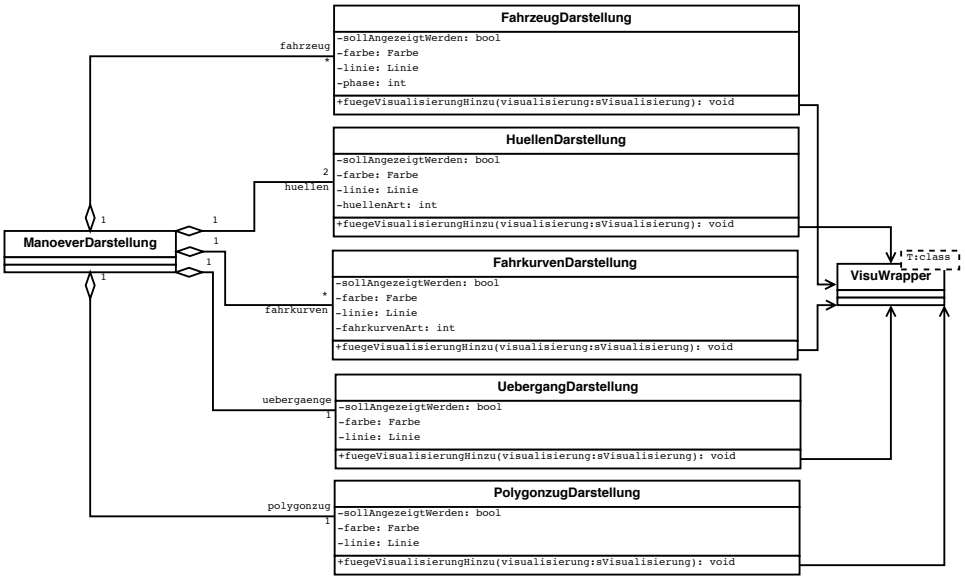


Abbildung 4.25: Die Klassen der Darstellungskonfiguration.

Im Mittelpunkt der Darstellung steht die Klasse *ManoeverDarstellung*. Dieser Klasse wird das Objekt der Klasse *ManoverDaten* übergeben, sie hat somit auch Zugriff auf die dort erstellten Visualisierungen. Für jede Visualisierungsart gibt es eine oder mehrere Komponenten in *ManoeverDarstellung*, jeweils realisiert von einer Klasse. Dazu gehören die Klassen *FahrzeugDarstellung*, *HuellenDarstellung*, *FahrkurvenDarstellung*, *UebergangDarstellung* und *PolygonzugDarstellung*. Diese Klassen haben alle die folgenden Attribute: Im Attribut *sollAngezeigtWerden* wird gespeichert, ob der jeweilige Bestandteil angezeigt werden soll, das Attribut *farbe* enthält die Farbe und das Attribut *linie* die Linie eines Objekts.

Einige Klassen sind für mehrere Objekte zuständig. Diese werden in *ManoeverDarstellung* mit einem Schlüssel angelegt, um Zugriff auf die jeweiligen Objekte zu erhalten. Bei der Klasse *FahrzeugDarstellung* handelt es sich dabei um den Phasenübergang, an denen das Fahrzeug dargestellt werden kann. Bei der Klasse *HuellenDarstellung* kann es die ausreichende und die sichere Hülle sein und in der Klasse *FahrkurvenDarstellung* die einzelnen Achsen des Gespanns, die Kupplung, die Räder vom Zugfahrzeug, die Räder vom Anhänger, die Ecken vom Zugfahrzeug und die Ecken vom Anhänger. So gruppierte Bestandteile werden immer mit den gleichen Darstellungseigenschaften angezeigt. Ein Objekt eines Bestandteils der Darstellung stellt dabei eine dieser Gruppierungen dar.

In jeder dieser Klassen gibt es die Methode *fuegeVisualisierungHinzu()*. Mit dieser Methode können einem Objekt Visualisierungen übergeben werden. Übergeben wird jeweils eine Komponente des Objekts der Klasse *ManoeverVisualisierung*. Es handelt sich bei diesen Komponenten um Objekte der Klasse *VisuWrapper*, in der die gewählten Darstellungsoptionen gesetzt werden können.

4.4 Kommunikation zwischen GUI, Modelldaten und Darstellung

Nachdem auf die einzelnen Komponenten der Architektur eingegangen wurde, soll in diesem Abschnitt die Kommunikation zwischen der GUI, den Modelldaten und der Darstellung erläutert werden. In Abbildung 4.26 ist zu sehen, wie die Kommunikation zwischen den Bestandteilen während einer Manöverkonstruktion abläuft. Bei dem Kommunikationsdiagramm wird dabei nur auf die Konstruktion und Darstellung einer Fahrkurve von einer Phase eingegangen. Das Schema lässt sich aber auf alle anderen Bestandteile eines Manövers abbilden.

Die Aufteilung im Diagramm ist so gewählt, dass die obere Reihe die Bestand-

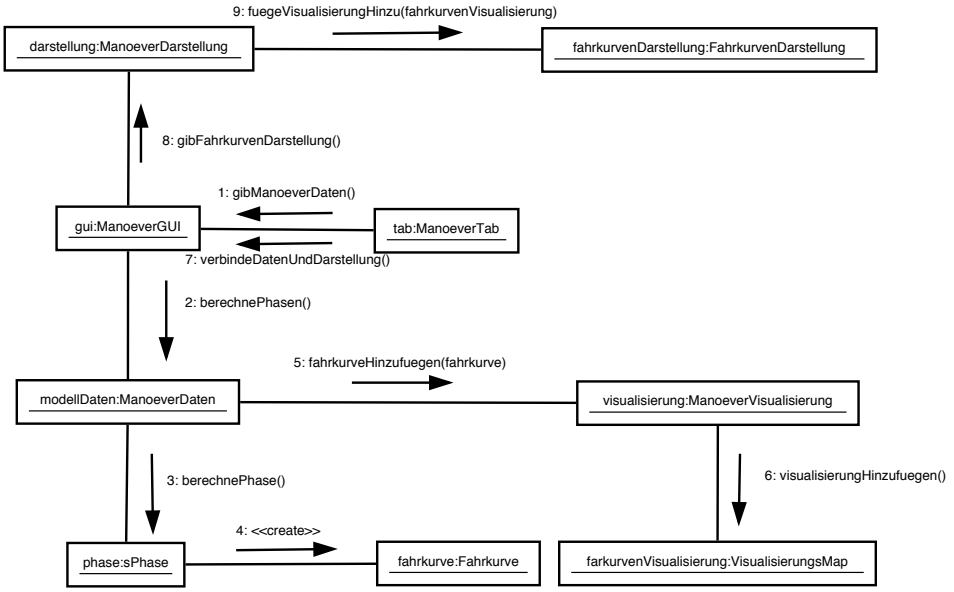


Abbildung 4.26: Kommunikationsdiagramm für die Konstruktion eines Manövers bezogen auf eine Fahrkurve.

teile des Darstellungsteils darstellen, die zweite Reihe die Bestandteile der GUI und in den unteren beiden Reihen die Teile der Manöverdaten zu sehen sind.

Die einzelnen Objekte, die in dem Diagramm dargestellt werden, sind, wenn nicht anders angegeben, schon erstellt worden. Zu dem Zeitpunkt, an dem die Kommunikation in dem Diagramm gestartet wird, sind also schon die Konstruktionsmethoden ausgewählt. Es wird die Konstruktion eines Manövers aus der Klasse *ManoeverTab* gestartet. Aus diesem Tab wird unter (1) die Methode *gibManoeverDaten()* aus der Klasse *ManoeverGUI* aufgerufen. Diese Methode gibt die aktuellen Modelldaten zurück, in diesem Fall das Objekt *modellDaten* einer Klasse, das von der abstrakten Klasse *ManoeverDaten* abgeleitet ist. In diesem Objekt wird unter (2) die Methode *berechnePhasen()* aufgerufen. Das Objekt *phase*, das die Klasse *sPhase* implementiert ist Teil von *modellDaten*, in der nun in (3) die Methode *berechnePhase()* für die Berechnung einer einzelnen Phase aufgerufen wird. In diesem Objekt werden unter (4) die Fahrkurven für diese eine Phase erstellt. Nachdem für alle Fahrkurven in *ManoeverDaten* auch die Visualisierungen erzeugt wurden, werden diese dem Objekt *visualisierung* der Klasse *ManoeverVisualisierung* übergeben. Dazu wird unter (5) die Methode *fahrkurveHinzufuegen()* aufgerufen. Der Container *fahrkurvenVisualisierung* der Klasse *FahrkurvenVisualisierung* beinhaltet alle Fahrkurven eines Manövers. Die betrachtete Fahrkurve wird in diesen Con-

tainer mit der Methode *visualisierungHinzufuegen()* unter (6) hinzugefügt.

Die Ausführung des Programms ist jetzt zurück in *ManoeverTab* und dort wird unter (7) die Methode *verbindeDatenUndDarstellung()* aus dem Objekt *gui* aufgerufen. Wie der Name schon andeutet, geht es bei dieser Methode darum, eine Verbindung zwischen den Visualisierungen der Modelldaten und der Darstellungskonfiguration zu schaffen. Das Objekt *darstellung* der Klasse *ManoeverDarstellung* ist schon Teil der *ManoeverGUI*, da dieses Objekt schon beim Anlegen der GUI erzeugt wurde. Daraus wird unter (8) die Methode *gibFahrkurvenDarstellung()* aufgerufen und damit auf das Objekt *fahrkurvenDarstellung* des Darstellungsteils zugegriffen, das für die Darstellungskonfiguration von Pfaden und Fahrkurven zuständig ist.[4] In diesem Objekt wird unter (9) mit der Methode *fuegeVisualisierungHinzu()* die Visualisierung, die in den Modelldaten angelegt wurde und sich dort in einem Container befindet, mit der Darstellung verknüpft.

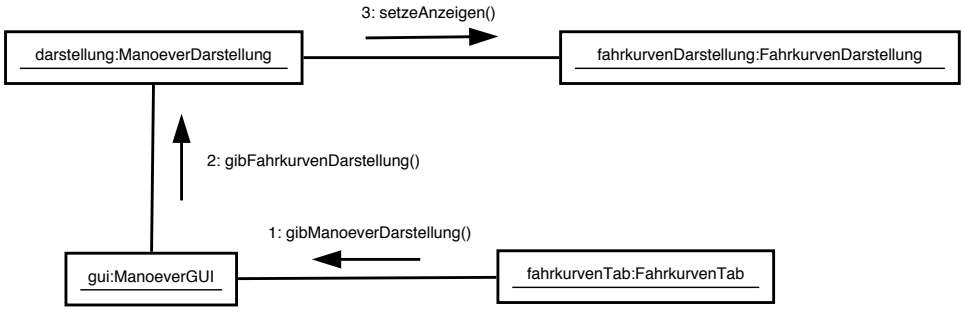


Abbildung 4.27: Kommunikationsdiagramm zur Änderung der Darstellung einer Fahrkurve.

Um die Darstellungskonfiguration aus Benutzersicht zu erläutern, wird in Abbildung 4.27 die Kommunikation gezeigt, die im Programm abläuft, wenn in der Registerkarte *FahrkurvenTab* die Einstellungen für bestimmte Fahrkurven geändert werden.

Als erstes wird unter (1) aus der Klasse *ManoeverGUI* die Methode *gibManoeverDarstellung()* aufgerufen. Damit ist ein Zugriff auf die Hauptklasse des Darstellungsteils, *ManoeverDarstellung*, möglich. In dieser Klasse wird unter (2) die Methode *gibFahrkurvenDarstellung()* auf den Bestandteil der Darstellung zugegriffen, in dem die Visualisierungen der Fahrkurven verknüpft sind. Dort kann mit der Methode *setzeAnzeigen()* unter (3) eingestellt werden, ob die Fahrkurve angezeigt wird oder nicht.

[4] Im Darstellungsteil werden Pfade und Fahrkurven nicht mehr getrennt behandelt.

4.5 Ausblick auf Entwurf und Implementierung

In diesem Abschnitt wird ein Ausblick auf die Weiterentwicklung der Software gegeben. Dabei wird insbesondere auf die Teile der Software eingegangen, die während der Bearbeitung dieser Diplomarbeit nicht implementiert werden konnten. Dies kann als Vorschlag zum Vorgehen bei der Weiterentwicklung der Software betrachtet werden.

Die fehlenden Bestandteile eines konventionellen Kurvenmanövers sind die Markierungen der Phasenübergänge und die Visualisierung der Umrisse des Fahrzeugs an bestimmten Stellen. Die Punkte der Phasenübergänge werden ermittelt, indem man die Punkte der Pfade und Fahrkurven an den beiden Grenzen einer Phase berechnet. Für die Darstellung des Fahrzeugs muss untersucht werden, welche Bestandteile aus der Visualisierung des Einspurmodells wiederverwendet werden können.

Im Anschluss an die konventionelle Methode sollte der iterative Konstruktionsansatz eingebunden werden. Dazu wurde schon eine Registerkarte angelegt, mit der die Einstellungen der iterativen Phasen vorgenommen werden. Diese sind, im Gegensatz zur konventionellen Methode, erweitert durch die Möglichkeit, die Inkremente zu bestimmen. Um die Implementation der iterativen Methode zu realisieren, sollte also bei den Modelldaten angesetzt werden. Dazu muss von der abstrakten Klasse *ManoeverDaten* die Klasse *KurvenmanoeverIterativDaten* abgeleitet werden und die jeweiligen Methoden der Klasse müssen nach der Konstruktionsvorschrift aus Kapitel 2.3.2.3 implementiert werden. Insbesondere sollte dabei auf das im Entwurf vorgestellte Kompositum-Entwurfsmuster eingegangen werden, um die Lenkwinkeländerung bei den instabilen Phasen zu berücksichtigen. Auch die Konstruktion der Ringhülle ändert sich nach der in Kapitel 2.3.2.4 vorgestellten Methode und muss in der Klasse *RingKonfigurationKurveIterativ* implementiert werden.

Der vorgestellte Entwurf der Konfigurationsklassen einer Ringhülle (vergleiche das Klassendiagramm in Abbildung 4.22 auf Seite 90) hat sich als nicht sinnvoll herausgestellt. Für jede Phase gibt es für Zugfahrzeug und Anhänger eine Methode, die den jeweiligen Kandidaten für Minimum und Maximum bestimmt. Die Objekte der Klasse *RingKonfigurationKurveKonventionell* werden aber für jede Phase angelegt. Insbesondere zeigt sich das Problem bei der iterative Konstruktionsmethode, da dort mehrere Phasen zusammen betrachtet werden müssen. Hier sollte bei der Implementierung des iterativen Manövers das Konzept so geändert

werden, dass es für die Konfiguration eines Ringes nur ein Objekt für das gesamte Manöver gibt.

Bevor die weiteren Manövertypen betrachtet werden, könnte die Alternative der Korridorkonstruktion, die Polygonhülle, implementiert werden. Diese Konstruktionsmethode wurde im Kapitel 2.3.5 genau erläutert. Der Softwareentwurf ist unter 4.2.3.2 zu finden. Da die Konstruktionsmethode des Manöverkorridors schon beim Start des Programms durch den Benutzer festgelegt wird, ist bereits beim Erzeugen von *ManoeverDaten* bekannt, welche Klasse für die Korridorkonstruktion benötigt wird.

Als Letztes sind die weiteren Manövertypen, Wendemanöver und Geradeausfahrt zu implementieren. Dabei kann das Wendemanöver sowohl konventionell, als auch iterativ konstruiert werden.

Kapitel 5

Vergleich verschiedener Manöver-Konfigurationen

In diesem Kapitel werden mit Hilfe des Werkzeugs PPFigure aus verschiedenen Konfigurationen Manöver erstellt. Als Erstes werden die Visualisierungen verschiedener Manöver vorgestellt. Danach wird auf die Auswertung des Manöverkorridors eingegangen, der sich zum Vergleichen verschiedener Einstellungen anbietet.

5.1 Visualisierungen eines Manövers mit PPFigure

In diesem Abschnitt werden die visuellen Ergebnisse vorgestellt, die mit dem Werkzeug PPFigure erstellt werden können. Dabei werden verschiedene Beispiele vorgestellt und an diesen die einzelnen visualisierten Bestandteile eines Manövers erläutert.

In Abbildung 5.1 ist ein Kurvenmanöver zu sehen, das mit der konventionellen Konstruktionsmethode erstellt wurde. Es wird eine Ausrichtungsänderung von 60° vorgegeben. Der Lenkwinkel der ersten Phase wurde mit 30° angegeben, der Lenkwinkel der zweiten Phase mit -15°, und der Lenkwinkel der dritten Phase mit -30°.

Gestartet wird auf der waagerechten Geraden des Polygonzugs. Die schwarzen Linien stellen verschiedene Pfade dar. Diese beziehen sich auf die Bezugspunkte der Achsenmittelpunkte des Gespanns und auf die Kupplung. Die Fahrtrichtung ist rückwärts. Deshalb ist der Start des Manövers auf der rechten Seite.

Um die einzelnen Pfade aus der Abbildung unterscheiden zu können, wurden

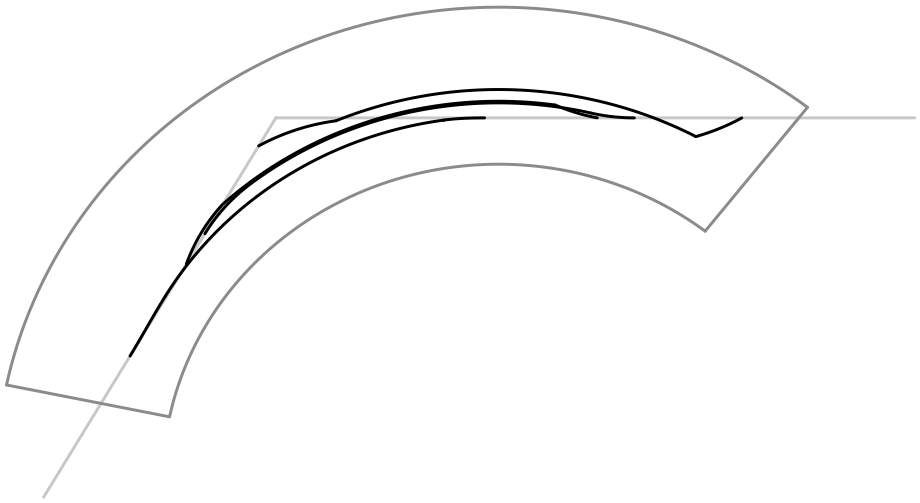


Abbildung 5.1: Kurvenmanöver eines Gespanns nach der konventionellen Konstruktionsmethode.

diese im Anhang einzeln betrachtet. In Abbildung A.2 sind die beiden Pfade der Achsenmittelpunkte des Zugfahrzeugs zu sehen. Abbildung A.3 zeigt den Pfad der Kupplung und Abbildung A.4 den Pfad des Achsenmittelpunkts des Anhängers.

In den Abbildungen wird auch der Manöverkorridor visualisiert. Dieser wurde als Ringhülle konstruiert. Für die Konstruktion des Manöverkorridors sind die Fahrkurven der Extrempunkte notwendig. Auch die Visualisierungen dieser Fahrkurven sind im Anhang zu finden. Anhand der Abbildungen wird die Konstruktion des Korridors erläutert. Diese gilt aber nur für dieses spezielle Manöver und kann bei anderen Konfigurationen und Fahrzeugen anders ausfallen. Die Fahrkurven der Räder des Zugfahrzeugs sind in Abbildung A.5 zu sehen. Die Fahrkurven für die Ecken des Zugfahrzeugs in A.6. Der maximale Radius wird von der linken vorderen Ecke des Zugfahrzeugs zu Beginn des Manövers vorgegeben. Die Begrenzung der Kreisbögen durch eine Gerade, die durch die rechte vordere Ecke des Zugfahrzeugs am Anfang des Manövers geht. Die Kupplung liefert bei dem dargestellten Manöver keinen Extrempunkt für die Begrenzungen.

Die Fahrkurven der beiden Räder des Anhängers sind in Abbildung A.7 zu sehen. Dort wurde auch der Extrempunkt für den minimalen Radius der Ringhülle berechnet. Dieser wurde in der dritten Phase bei dem rechten Rad des Anhängers ermittelt. Die Fahrkurven der Ecken des Anhängers sind in Abbildung A.8 zu se-

hen. Darin ist der Extrempunkt für die Begrenzung der Kreisbögen am Ende des Manövers zu sehen. Es handelt sich dabei um die rechte hintere Ecke des Anhängers. Der Zentriwinkel zwischen dieser Ecke und der Ecke, die die Kreisbögen am Anfang des Manövers begrenzt, ist der Winkel Φ des Manöverkorridors. In Abbildung A.9 sind alle möglichen Extrempunkte eines Manövers und der berechnete Korridor dargestellt.

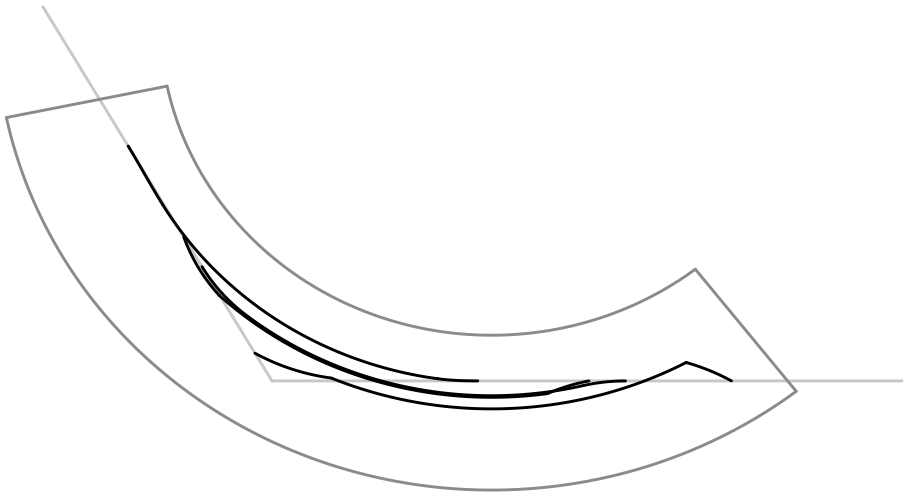


Abbildung 5.2: Kurvenmanöver mit vertauschten Lenkwinkeln.

In Abbildung 5.2 ist ebenfalls ein Manöver in Fahrtrichtung rückwärts und einer Ausrichtungsänderung von 60° dargestellt. Im Gegensatz zum ersten Beispiel wurde dabei aber mit einem negativen Lenkwinkel gestartet und die übrigen Lenkwinkel entsprechend angepasst. Der Lenkwinkel der ersten Phase beträgt nun -30°, der Lenkwinkel der zweiten Phase 15° und der Lenkwinkel der dritten Phase 30°. In der Abbildung ist zu sehen, dass der Winkel Γ ein anderes Vorzeichen hat, als in Abbildung 5.1. Während im ersten Beispiel der Winkel Γ positiv ist, ist er in diesem Fall negativ. Die Bestimmung des Vorzeichens von Γ wurde in Kapitel 2.3.2 vorgestellt.

Als letztes Beispiel für die Visualisierung soll die Fahrtrichtung geändert werden. Ein Manöver in Fahrtrichtung vorwärts kann bei der Konstruktion wie ein Manöver in Fahrtrichtung rückwärts behandelt werden. Es wird dabei nur in umgekehrter Richtung durchfahren. Ein Manöver dieser Art ist in Abbildung 5.3 zu sehen. Hierbei wurde in der ersten Phase ein negativer Lenkwinkel von -30° angegeben. In der stabilen, zweiten Phase ist der Lenkwinkel -15° und in der letzten

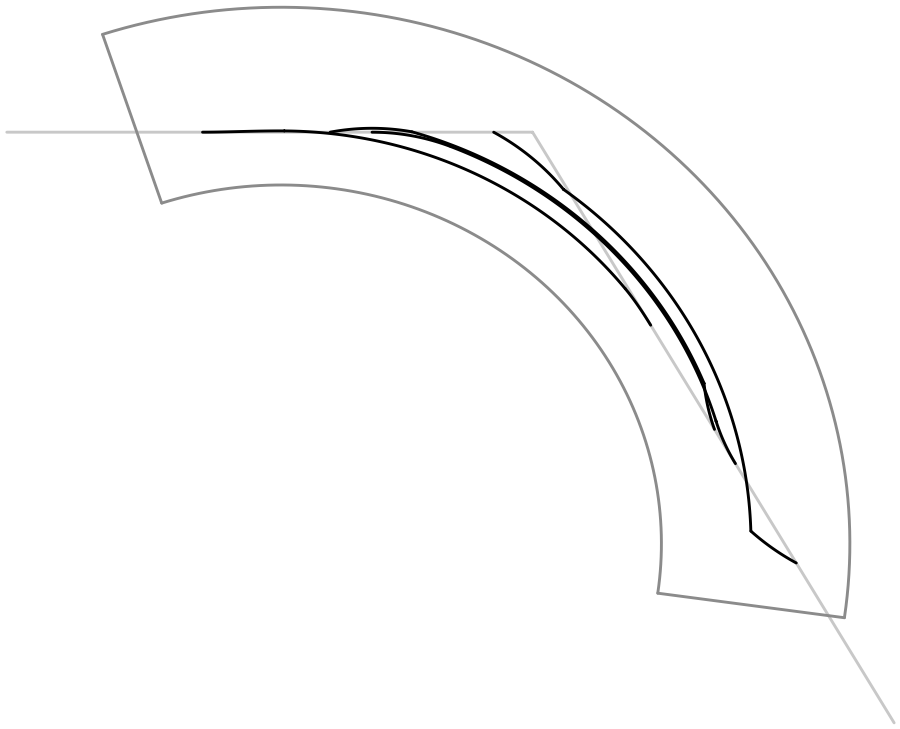


Abbildung 5.3: Kurvenmanöver in Fahrtrichtung vorwärts.

Phase ist er positiv und beträgt 30°. Dieses Manöver ist das gleiche wie in dem ersten Beispiel. Es ändert sich nur die Reihenfolge der Phasen. Die Abbildung 5.1 kann man erhalten, indem man die Abbildung 5.3 um 60° gegen den Uhrzeigersinn dreht. Damit ist die Gerade des Polygonzugs, in der das Gespann nach dem Manöver endet, in der Waagerechten.

5.2 Vergleich verschiedener Konfigurationen

Nachdem für einige Beispiele die Visualisierung vorgestellt und erläutert wurde, sollen in diesem Abschnitt die Ergebnisse des Manöverkorridors für verschiedene Konfigurationen eines konventionellen Kurvenmanövers verglichen werden. Die Ergebnisse für die Vergleiche werden dabei der Registerkarte „Zusammenfassung“ entnommen. Zu den Ergebnissen gehören der minimale Radius, r_{min}, der maximale Radius, r_{max}, der Zentriwinkel der Kreisbögen, Φ, die Breite des Korridors, b[1]

[1] Die Breite wird als Differenz der beiden Radien berechnet

und die Fläche des Korridors, F.

In den ersten Vergleichen werden reale Fahrzeugabmessungen eines LKW-Gespanns verwendet.[2] Das Zugfahrzeug hat eine Länge von 8,96m und eine Breite von 2,56m. Die Spurweite der Vorderachse beträgt 2,24m, die Spurweite der Hinterachse 1,92m. Die Spurweite der Hinterachse ist geringer, da es sich um eine Zwillingsbereifung handelt und der Mittelpunkt beider Reifen betrachtet wird. Der Anhänger hat eine Länge von 6,08m und eine Breite von 2,56m. Hierbei ist die Spurweite 2,24m. Für den Abstand L_1 ergibt sich ein Wert von 5,57m, der Abstand L_2 ist 5,84m und der Abstand M_1 ist 1,92m.

Als Erstes soll die Wirkung des Lenkwinkels der zweiten, stabilen Phase auf den Korridor untersucht werden. Aus diesem Lenkwinkel lässt sich mit der Formel 2.15 der zugehörige Einknickwinkel für eine stabile Fahrt berechnen. Als Ausrichtungsänderung wird wieder 60° gewählt. Es wurde für die erste Phase ein Lenkwinkel von 40° angegeben, für die dritte Phase ein Lenkwinkel von -40°. Die Lenkwinkel der beiden Phasen entsprechen dem maximalen Lenkwinkel des Fahrzeugs. Die Fahrtrichtung ist rückwärts.[3] Der Lenkwinkel beginnt bei -5° und der Betrag wird in 5°-Schritten erhöht. Im Bereich zwischen 15° und 20° wird für die Suche nach einem Minimum ein kleinerer Abstand gewählt.

Tabelle 5.1: Vergleich verschiedener Lenkwinkel der zweiten Phase bei einem LKW mit einachsigem Anhänger

α_2	$r_{min}\ [m]$	$r_{max}\ [m]$	Φ	$b\ [m]$	$F\ [m^2]$
-5,0°	60,00	68,10	76,18°	8,10	689,37
-10,0°	27,94	36,03	94,21°	8,10	425,73
-15,0°	17,14	25,85	113,79°	8,71	371,81
-16,0°	15,78	24,69	117,88°	8,91	370,75
-16,5°	15,16	24,16	119,95°	9,00	370,63
-17,0°	14,57	23,68	122,03°	9,10	370,76
-20,0°	11,62	21,34	134,88°	9,71	376,92
-25,0°	7,91	18,92	157,69°	11,01	406,37
-30,0°	5,20	17,54	181,64°	12,34	444,65

In Tabelle 5.1 sind die Ergebnisse für die verschiedenen Lenkwinkel in Phase

[2] Die Abmessungen stammen von dem maßstabgetreuen Modell der Arbeitsgruppe, dem Maßstab entsprechend hochgerechnet

[3] Fahrtrichtung und Lenkwinkelvorzeichen haben keine Auswirkungen auf die berechneten Werte des Korridors.

2 zu sehen. Die Auswertung bezieht sich dabei nur auf dieses Beispiel. Die beiden Radien werden dabei monoton kleiner, je größer der Betrag des Lenkwinkels wird. Der Winkel Φ wird monoton größer. Bis zu einem Lenkwinkel der zweiten Phase von ca. 10° bleibt die Breite des Korridors konstant und wächst bei größeren Lenkwinkeln. Die Fläche besitzt in der Tabelle ein Minimum bei einem Lenkwinkel von -16,5°.

Im nächsten Vergleich werden unterschiedliche Lenkwinkel der stabilen Phasen betrachtet. Dazu werden für die erste und die dritte Phase vom Betrag her gleiche Lenkwinkel gewählt. Der Lenkwinkel der zweiten Phase wird dabei auf den Lenkwinkel -16,5° gesetzt, bei dem im vorherigen Vergleich die minimale Fläche ermittelt wurde. Der Lenkwinkel der ersten Phase startet bei dem maximalen Lenkwinkel von -40°. Entsprechend ist der Lenkwinkel der dritten Phase 40°. Die Ausrichtungsänderung von 60° wird dabei beibehalten, ebenso die Fahrtrichtung rückwärts.

Tabelle 5.2: Vergleich verschiedener Lenkwinkel der ersten und dritten Phase

$\lvert\alpha_1\rvert$ und $\lvert\alpha_3\rvert$	$r_{min}\ [m]$	$r_{max}\ [m]$	Φ	$b\ [m]$	$F\ [m^2]$
40°	15,16	24,16	119,95°	9,00	370,63
35°	15,34	24,26	121,23°	8,92	373,65
30°	15,34	24,38	122,94°	9,05	385,28
25°	15,33	24,54	125,41°	9,21	401,81
20°	15,34	24,76	129,49°	9,42	427,09
17°	15,35	24,95	133,95°	9,60	452,04

In der Tabelle 5.2 ist das Ergebnis für unterschiedliche Lenkwinkel der beiden instabilen Phasen zu sehen. Darin kann man erkennen, dass das Minimum der Fläche bei den vom Betrag her maximalen Lenkwinkeln erreicht wird. Lenkwinkel der instabilen Phasen müssen vom Betrag her größer sein, als der Lenkwinkel der stabilen Phase. Daher werden in der Tabelle keine Lenkwinkel betrachtet, die kleiner als 17° sind.

Als Letztes soll der zweite Lenkwinkel eines anderen Gespanns verglichen werden. Dazu wurden die Abmessungen eines Kleinwagens mit einem Anhänger modelliert. Das Fahrzeug hat eine Länge von 3,756m und eine Breite von 1,69m. Die vordere Spurweite beträgt 1,46m und die hintere Spurweite 1,47m. Der Anhänger hat eine Länge von 2,50m und eine Breite von 1,25m. Die Spurweite beträgt 1,25m. Für den Abstand L_1 ergibt sich ein Wert von 2,38m, L_2 ist 1,80m und M_1

ist 1,00m.

Bei dem Vergleich wird analog zum ersten Beispiel vorgegangen. Der Lenkwinkel der ersten Phase beträgt -40° und der Lenkwinkel der dritten Phase beträgt 40°. Die Fahrtrichtung ist rückwärts und die Ausrichtungsänderung beträgt 60°. Gestartet wird auch hier mit einem Lenkwinkel der zweiten Phase von -5°.

Tabelle 5.3: Vergleich verschiedener Lenkwinkel der zweiten Phase bei einem PKW mit Anhänger

α_2	$r_{min}\ [m]$	$r_{max}\ [m]$	Φ	$b\ [m]$	$F\ [m^2]$
-5,0°	24,93	29,81	76,20°	4,88	177,60
-10,0°	11,22	16,1	94,80°	4,88	110,29
-15,0°	6,60	11,56	115,35°	4,96	90,67
-20,0°	4,26	9,50	137,11°	5,24	86,31
-24.5°	2,94	8,42	157,09°	5,48	85,39
-25,0°	2,83	8,33	159,31°	5,50	85,38
-25,5°	2,71	8,24	161,52°	5,53	85,39
-30,0°	1,84	7,60	181,01°	5,57	85,83
-35,0°	1,12	7,12	200,69°	5,99	86,46

Das Ergebnis der verschiedenen Lenkwinkel ist in Tabelle 5.3 zu sehen. Auch hier werden die beiden Radien r_{min} und r_{max} monoton kleiner, wohingegen der Winkel Φ monoton steigt. Auch die Breite des Manöverkorridors, b, wächst ab einem Lenkwinkel von 15°. Für die Fläche ergibt sich in dieser Tabelle ein Minimum bei einem Lenkwinkel der zweiten Phase von -25°.

Kapitel 6

Fazit

Im letzten Kapitel dieser Diplomarbeit soll ein Fazit gezogen werden. Dazu werden zuerst die Ergebnisse dieser Arbeit zusammengefasst. Anschließend wird in einem Ausblick auf die Weiterentwicklung der Software und die Möglichkeiten der Nutzung des Werkzeugs eingegangen.

6.1 Zusammenfassung der Diplomarbeit

In einem vorangegangenen Projektpraktikum wurde das Werkzeug PPFigure erstellt. Mit Hilfe dieses Werkzeugs können die Daten eines Fahrzeugs aus einer Datei eingelesen und daraus automatisch ein Einspurmodell erzeugt werden. Die dynamischen Daten des Fahrzeugs können dabei vom Benutzer konfiguriert werden. Innerhalb der Software wird eine Visualisierung des Einspurmodells angezeigt, diese kann in eine externe Datei exportiert werden.

Diese Diplomarbeit beschäftigt sich mit der Weiterentwicklung von PPFigure. Dabei ist das Werkzeug mit einer neuen Funktionalität erweitert worden, der Konstruktion von Manövern. Im Zuge dieser Arbeit wurden als Erstes die Grundlagen eines Manövers aus den Veröffentlichungen der Arbeitsgruppe Echtzeitsysteme erarbeitet. Begonnen wurde mit der Modellierung eines Gespanns, bestehend aus Zugfahrzeug und einachsigem Anhänger. Anschließend wurde die mathematische Beschreibung von Fahrzeugbewegungen erläutert. Betrachtet wurden die Fahrkurven von Bezugspunkten des Zugfahrzeugs und des Anhängers. Unter Einschränkungen, die für die betrachteten Gespannarten in der Regel zutreffen, wurden zwei verschiedene Traktionsarten eingeführt, die innere und die äußere Traktion.

Es gibt verschiedene Manövertypen. In der Arbeitsgruppe werden Kurven-

manöver, Wendemanöver und Geradeausfahrten unterschieden. Diese Typen wurden vorgestellt und bei dem Kurvenmanöver zwei Konstruktionsmethoden erläutert. Die konventionelle Methode erlaubt Lenkwinkeländerungen nur zwischen den Phasen, die Lenkung muss also sprunghaft geändert werden. Um ein genaues Abfahren des Pfades zu gewährleisten, muss das Fahrzeug für die Lenkwinkeländerung anhalten. Die iterative Methode erweitert diesen Ansatz, so dass der Lenkwinkel in kleinen Schritten während der Fahrt geändert wird. Auch für die Konstruktion eines Manöverkorridors wurden zwei Konstruktionsmethoden erläutert. Die Ringhülle kann nur bei einem Kurvenmanöver konstruiert werden, als Alternative bietet sich die Konstruktion als Menge von Polygonen an, die das Fahrzeug jeweils während einer Phase umschließen.

Es wurde ein Einblick in die verwendeten Software-Bibliotheken, EZauto und wxWidgets, gegeben und die Version des Werkzeugs PPFigure vor der Erweiterung durch diese Arbeit vorgestellt.

Anschließend wurde ein Anforderungskatalog für die Software erstellt. Dieser diente als Grundlage für den Softwareentwurf. Im Entwurf wurde dabei die Architektur der Software aus dem vorangegangenen die im Projektpraktikum beibehalten. Diese sieht eine Trennung zwischen GUI, Modelldaten und Darstellung vor. Es wurde zuerst der Entwurf der GUI vorgenommen. Dazu musste die bestehende GUI der neuen Funktionalität angepasst werden. Beim Starten des Programms erfolgt ein Dialog, mit dem die Funktionalität ausgewählt werden kann. Außerdem können dabei Manövertyp, Konstruktionsmethode und Korridortyp ausgewählt werden. Für die Konfiguration eines Manövers wurden verschiedene Registerkarten entworfen. Bei dem Entwurf wurde auf die unterschiedlichen Typen und Methoden eingegangen und diese berücksichtigt.

Auch für den Entwurf der Manöver-Datenstruktur mussten diese Unterschiede beachtet werden. Eine abstrakte Klasse stellt dafür die Attribute und Methoden zur Verfügung, die in allen Manövern wiederverwendet werden können. Von dieser Klasse werden für unterschiedliche Typen und Methoden jeweils konkrete Klassen abgeleitet, die das jeweilige Manöver nach dem in den Grundlagen vorgestellten Vorgehen konstruieren.

Von den vorgestellten Manövertypen und Konstruktionsmethoden wurde während dieser Arbeit die Konstruktion des konventionelle Kurvenmanöver implementiert. Es werden dabei die Pfade der Achsenmittelpunkte und der Kupplung konstruiert. Ebenso können die Fahrkurven der anderen Extrempunkte angezeigt werden. Der Manöverkorridor wird als Ring konstruiert.

Die Benutzerschnittstelle verfügt über ein Register „Zusammenfassung“ in dem unter anderem die Ergebnisse des Manöverkorridors angezeigt werden. Dazu gehören der minimale und der maximale Radius, die Breite, der Zentriwinkel der Kreisbögen Φ und insbesondere die Fläche des Manöverkorridors. Anhand dieser Werte wurden verschiedene Manöver-Konfigurationen miteinander verglichen.

6.2 Ausblick

Für die Weiterentwicklung des Werkzeugs PPFigure gibt es verschiedene Ansatzmöglichkeiten. Es konnten noch nicht alle Bestandteile des konventionellen Kurvenmanövers implementiert werden. Die noch fehlenden Bestandteile sind die Markierung der Phasenübergänge und die Visualisierung des Fahrzeugs an bestimmten Stellen während des Manövers. Auch die Implementierung der iterativen Konstruktionsmethode, der weiteren Manövertypen, Wendemanöver und Geradeausfahrt und der Polygonhülle muss noch vorgenommen werden.

Das Wendemanöver kann auch mit der iterativen Methode konstruiert werden. Hierfür muss eine Konstruktionsvorschrift erarbeitet werden. Bei der iterativen Konstruktion wurde in Kapitel 2.3.2.3 erwähnt, dass ein variabler Ansatz zur Bestimmung der Größe der Inkremente möglich ist und favorisiert wird. Auch dieser Ansatz muss noch ausgearbeitet werden.

Mit Hilfe des Werkzeugs PPFigure können Darstellungen von Manövern in eine EPS-Datei exportiert werden. Diese EPS-Dateien können in Veröffentlichungen und Präsentationen verwendet werden. Hierbei kann auf bestimmte einzelne Bestandteile eines Manövers eingegangen werden, wie den Abbildungen im Anhang zu entnehmen ist.

Es ist auch möglich das Werkzeug zum Vergleich verschiedener Konfigurationen anzuwenden. Insbesondere können mit diesen Werten auch verschiedene Manövertypen und Konstruktionsmethoden miteinander verglichen werden. Nach der Implementierung der iterativen Methode bietet es sich an, diese mit der konventionellen Methode zu vergleichen. Bei gleichen Bedingungen, wie Lenkwinkel und Ausrichtungsänderung, kann anhand der Fläche des Korridors eine Evaluation des iterativen Manövers vorgenommen werden. Die Ergebnisse des konventionellen Manövers dienen dabei als Vergleichswerte. Es ist auch denkbar, dass für die gleiche Ausrichtungsänderung die minimalen Flächen von Kurven- und Wendemanöver miteinander verglichen werden.

Anhang A

Anhang

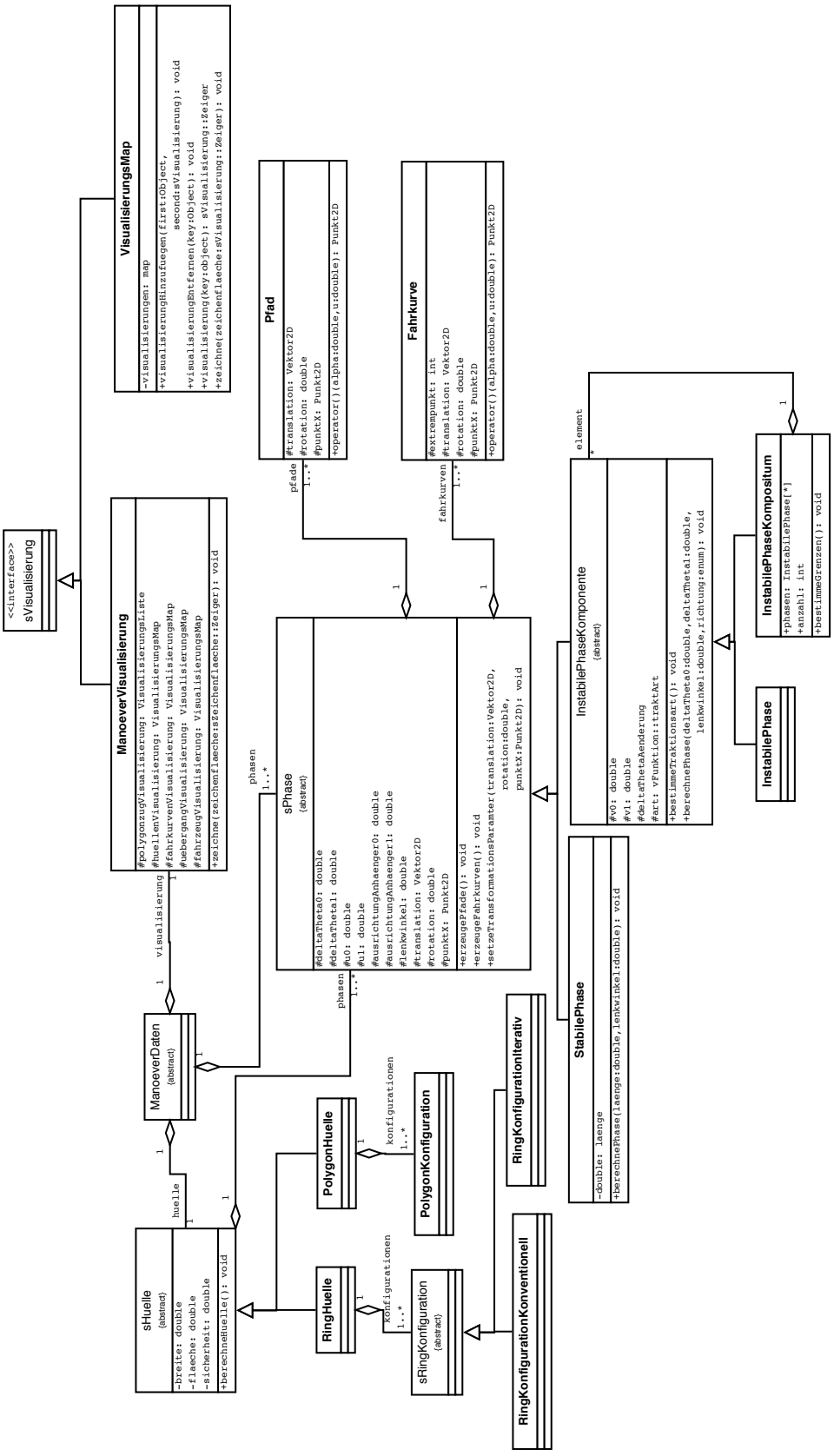


Abbildung A.1: Die Klasse *ManoeverDaten* mit ihren Komponenten

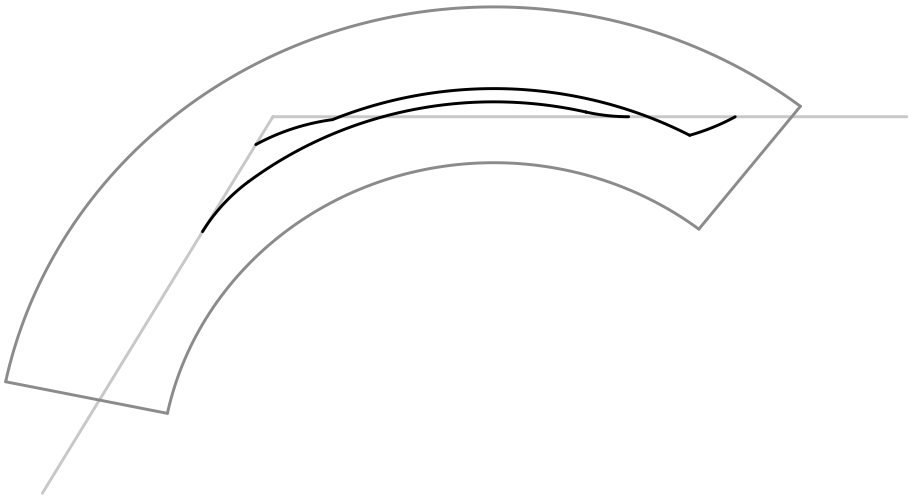


Abbildung A.2: Pfade der beiden Mittelpunkte vom Zugfahrzeug bei einem Kurvenmanöver.

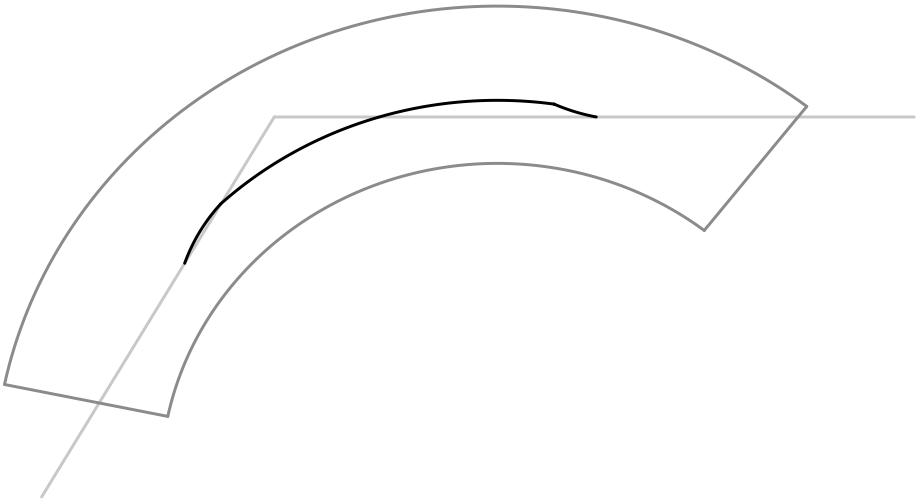


Abbildung A.3: Pfad der Kupplung bei einem Kurvenmanöver.

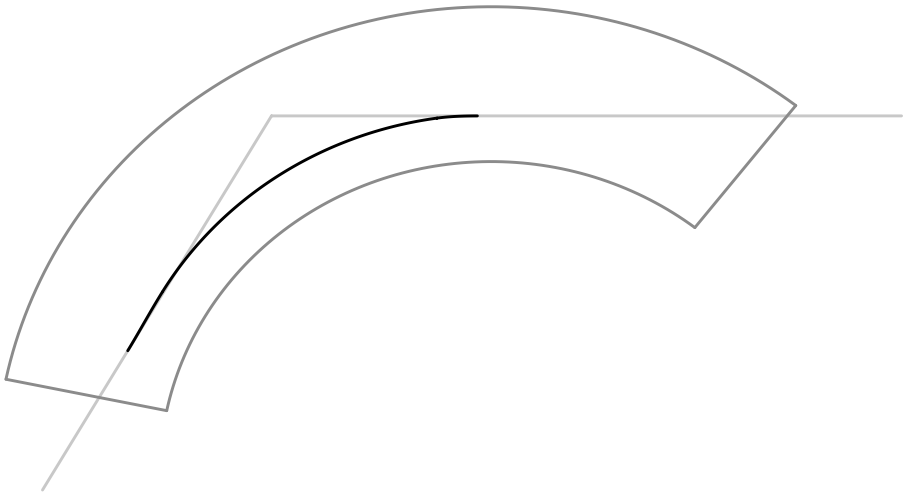


Abbildung A.4: Pfad des Achsenmittelpunkts vom Anhänger bei einem Kurvenmanöver.

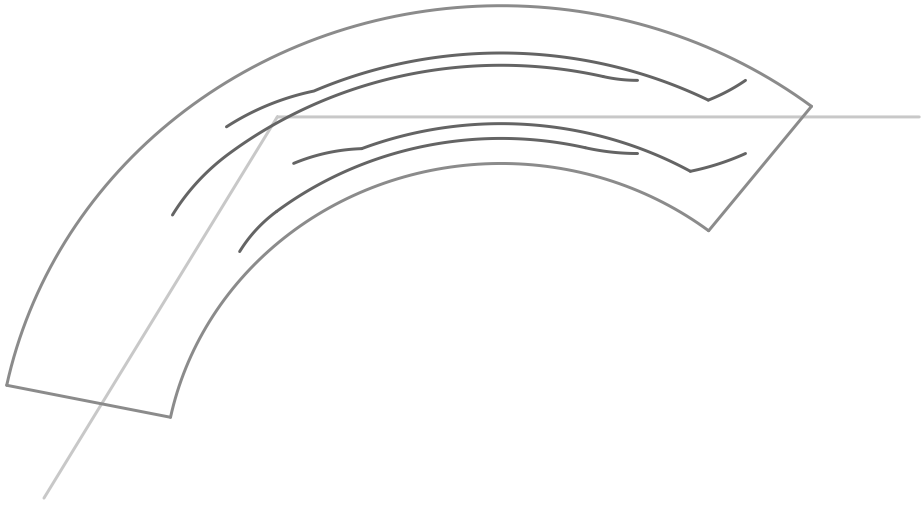


Abbildung A.5: Fahrkurven der Räder des Zugfahrzeugs bei einem Kurvenmanöver.

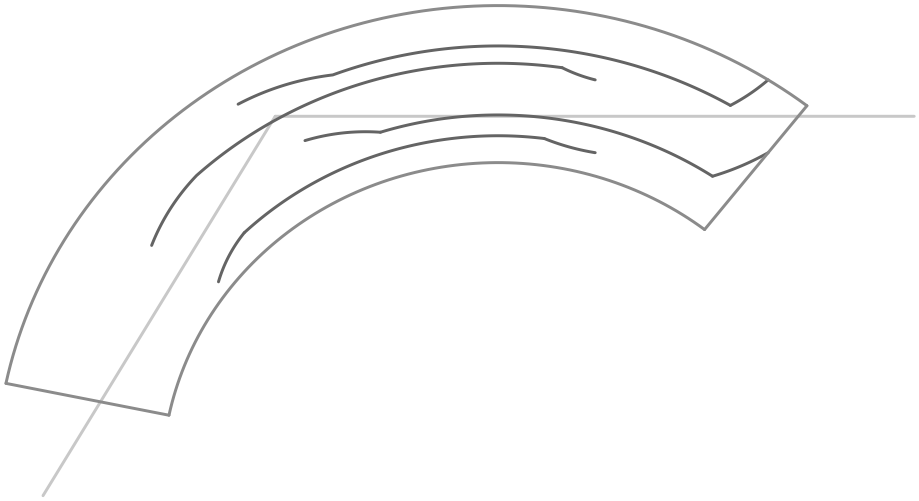


Abbildung A.6: Fahrkurven der Ecken des Zugfahrzeugs bei einem Kurvenmanöver.

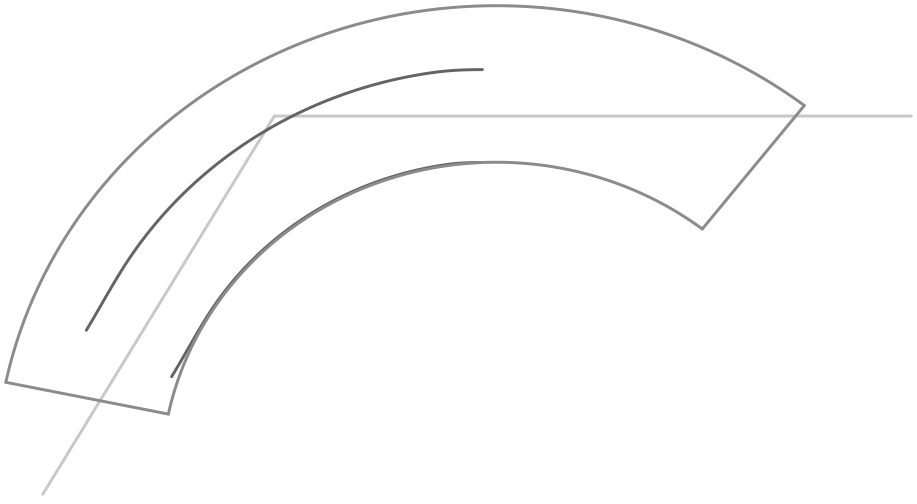


Abbildung A.7: Fahrkurven der Räder des Anhängers bei einem Kurvenmanöver.

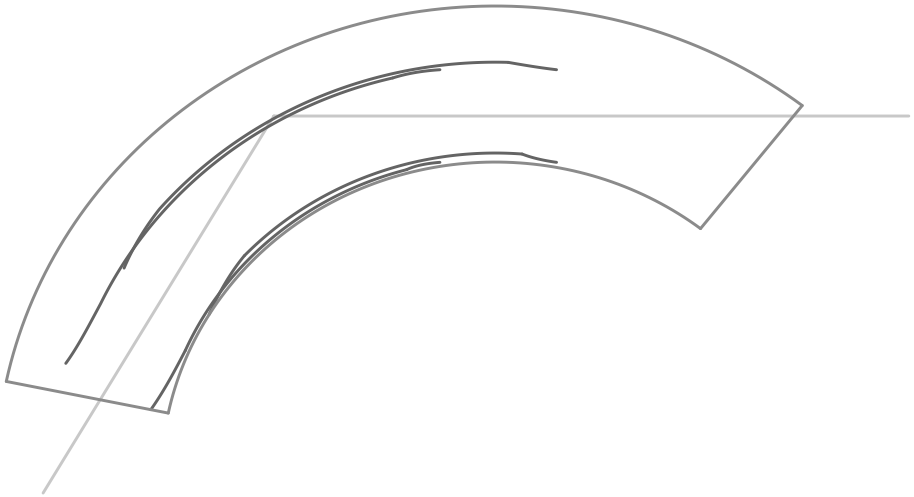


Abbildung A.8: Fahrkurven der Ecken des Anhängers bei einem Kurvenmanöver.

Abbildung A.9: Fahrkurven aller Extrempunkte bei einem Kurvenmanöver.

Literaturverzeichnis

[Berg & Zöbel, 2005] Berg, U. & Zöbel, D. (2005). Haptische Lenkassistenz zur Unterstützung der Rückwärtsfahrt von Fahrzeugen mit einachsigen Anhängern. in *6. Berliner Werkstatt Mensch-Maschine-Systeme: Zustandserkennung und Systemgestalung* Berlin.

[Gamma *et al.*, 2004] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2004). *Design Patterns - Elements of Reusable Object-Oriented Software* (Addison-Wesley: Amsterdam).

[Lellmann *et al.*, 2010] Lellmann, M., Eiserloh, C., & Schnitzler, T. (2010). Entwickung eines Werkzeugs zur Bildgenerierung von kinematischen Modellen. unpublished.

[Smart *et al.*, 2006] Smart, J., Hock, K., & Csomor, S. (2006). *Cross-Platform GUI Programming with wxWidgets* (Pearson Education: Upper Saddle River).

[Sommerville, 2001] Sommerville, I. (2001). *Software Engineering* (Pearson Studium: München). 6. edit.

[Straßmann, 2007] Straßmann, B. (2007). Der digitale Copilot. Die ZEIT.

[Weyand, 2010] Weyand, C. (2010). Allgemeiner Ansatz für polygonförmige Korridorflächen. unpublished.

[Weyand & Balcerak, 2009] Weyand, C. & Balcerak, E. (2009). Iterative Konstruktion von Manövern mit inkrementeller Lenkwinkeländerung. unpublished.

[Weyand & Balcerak, 2010] Weyand, C. & Balcerak, E. (2010). Alternative Konstruktionsansätze für Manöver des Typs Wenden. unpublished.

[Weyand *et al.*, 2010] Weyand, C. *et al.* (2010). Iterative generation of smoth maneuvers for articulated vehicles.

[Wojke, 2010] Wojke, P. (2010). EZauto - Softwarearchitektur für Anwendung des assistierten oder autonomen Fahrens. unpublished.

[Wünsch, 1997] Wünsch, V. (1997). *Differentialgeometrie: Kurven und Flächen* (Teubner: Stuttgart).

[Zöbel, 2001] Zöbel, D. (2001). Mathematical modeling of the kinematics of vehicles. in *Mathematical Modeling of Technical Processes* (Hrubina, K., ed.) pp. 178–200.

[Zöbel, 2008] Zöbel, D. (2008). Kinematic modeling of moving vehicles (ezauto). unpublished.

[Zöbel & Balcerak, 2000] Zöbel, D. & Balcerak, E. (2000). Präzise Fahrmanöver für Fahrzeuge im Gespann. Automobile Mobile Systeme (AMS2000).