LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Diplomarbeit

3D-Visualisierung von Bordnetzkommunikation im Fahrzeug

Fabian Hennecke
fabian.hennecke@web.de

Bearbeitungszeitraum:	15. 07. 2008 bis 15. 01. 2009
Betreuer:	Dipl. Medieninformatiker Michael Sedlmair
Verantw. Hochschullehrer:	Prof. Dr. Andreas Butz

Zusammenfassung

Diese Diplomarbeit befasst sich mit einer 3D-Visualisierung im Bereich der Analye von Bordnetzkommunikation im Kraftfahrzeug. Bisherige Analyse-Programme nutzen zur Darstellung der Kommunikationsvorgänge und -zusammenhänge zweidimensionale Darstellungen.
In Zusammenarbeit mit BMW sind bereits zwei Prototypen einer 3D-Visualisierung entstanden, die jedoch bis jetzt nicht ausreichend evaluiert wurden. Teil dieser Arbeit ist daher die Evaluation dieser Prototypen, um die Ergebnisse anschließend in die Aufstellung einer Konzeptsammlung einfließen zu lassen. Diese Sammlung wird Experten des Fachgebiets Analyse/Diagnose im Rahmen eines User-centered-approachs zur Bewertung vorgelegt, bevor aus den einzelnen Ideen dieser Sammlung das endgültige Visualisierungskonzept erstellt wird.
Dieses Konzept wird im Anschluss daran implementiert. Teil dieser Implementierung ist die Realisierung einer Anbinung an andere Analyse-/Diagnose-Visualisierungen zur Bereitstellung der 3D-Visualisierung in einem Multi Coordinated View (MCV) System.
Den Abschluss der Arbeit bildet eine Evaluation des implementierten Prototyps und der Möglichkeiten der Intergration einer 3D-Visualisierung in ein MCV-System.

Abstract

This diploma thesis deals with a 3D-visualization in the field of an analysis of in-car communication processes. State of the art analysis-tools only use a two-dimensional approach to visualize communication processes and relationships.
Two different prototyps, both using a treedimensional visualization, were already developed in cooperation with BMW. Because these prototyps were never evaluated adequately, the first part of this thesis will be the evaluation of both tools. The findigs of the evaluation can be used to create a collection of different ideas for a proper 3D-visualization. The collection will be presented to analysis-experts working at BMW, using a user centered approach to create a final concept for a 3D-visualization.
Afterwards this concept will be implemented with a possible connectivity to other visualization-tools, so that a 3D-view can be used in a Multi Coordinated View (MCV) system.
At the end of this thesis the implemented prototype will be evaluated to be able to estimate the possibilities of an 3D-visualization integrated into an MCV system.

Aufgabenstellung

Aus dem automobilen Bereich der technisch-wissenschaftlichen Visualisierung ist bekannt, dass 3D-Fahrzeugmodelle oft dazu verwendet werden um Messdaten, wie beispielsweise Daten aus dem Windkanal, Crashtestergebnisse oder Verschmut-zungsinformationen, grafisch darzustellen. Hierbei steht v. A. der räumliche Bezug der Daten im Vordergrund. Im Rahmen ihrer Studienabschlussarbeit soll der Einsatz von 3d-Modellen zur Visualisierung von Kommunikationsprozessen im Automobil untersucht werden.
Dazu soll zunächst ein interaktives, virtuelles 3D Fahrzeugmodell implementiert werden, in dem der Nutzer einerseits die Möglichkeit erhalten soll wie mit einem realen Fahrzeug zu interagieren, also z. B. den Fensterheber zu betätigen. Andererseits sollen mechanische Reaktionen - in unserem Beispiel also die Bewegung des Fensters - im Modell realitätsnah dargestellt werden.
Aufbauend auf der Applikation sollen verschiedene Anwendungs- und Einsatzgebiete der 3D-Visualiserung von Kommunikationsprozessen aufgezeigt werden, wie z. B. die zeitgesteuerte farbliche Markierung von Fahrzeugkomponenten wie Steuergeräten oder Kabelsträngen. Für eine Analyse soll ein adäquates Anwendungsgebiet ausgewählt, prototypisch realisiert sowie abschließend evaluiert werden.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, 14. Januar 2009

. . . . . .

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Im Laufe der letzten Jahre wurde das Bordnetz, das sämtliche elektronischen und elektrischen Komponenten eines Kraftfahrzeugs umfasst, durch eine steigende Anzahl von verbauten Steuergeräten und Bus-Systemen immer komplizierter. Dadurch stieg nicht nur die Komplexität der Bordnetz-Topologie, sondern auch die Kommunikation zwischen den einzelnen Steuergeräten hat erheblich zugenommen. Bei aktuellen Fahrzeugen werden zum Teil deutlich mehr als insgesamt 14.000 Nachrichten pro Sekunde über die unterschiedlichen Bus-Systeme verschickt (für weitere Details siehe [23] und [24]). Nimmt man nun eine zweistündige Testfahrt an, während der die gesamte Bordnetz-Kommunikation für die spätere Analyse aufgezeichnet wird, kommt eine erhebliche Datenmenge zusammen, pro aufgezeichneter Stunde sind dies ca. 2GB. Diese Datenmenge muss dann für die Analyse und Diagnose von Problemen aufbereitet und entsprechend dargestellt werden.
Eine einfache und viel genutzte Form der Daten-Wiedergabe ist sicherlich die Darstellung der hexadezimalen Nachrichten- und Signal-Inhalte. Allerdings sind die Informationen in dieser Form auch sehr schwer zu lesen bzw. zu überblicken. Aus diesem Grund werden in aktuellen Analyse- und Diagnose-Tools weitere (2-dimensionale) Darstellungsformen angeboten, um zuerst einen Überblick über die Daten zu ermöglichen und auf Wunsch anschließend tiefer in die Daten eintauchen zu können (siehe Abbildung 1.1).
Ein Beispiel für eine andere Darstellungsform ist die zweidimensionale Darstellung als Wertegraph. Hierbei wird ein Werteverlauf über eine gewisse Zeit in einem Koordinatensystem dargestellt. Dies ist eine einfache und übersichtliche Form der Visualisierung. Auf diese Weise lassen sich Wertesprünge oder Abweichungen vom üblichen oder zu erwartenden Verlauf gut erkennen. Diese Zeitpunkte können dann als Einstiegspunkte in die Analyse des auftretenden Problems genutzt werden.

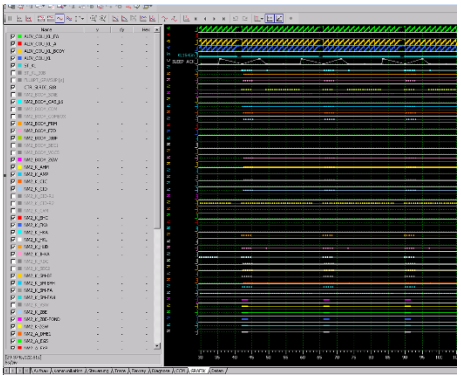


Abbildung 1.1: Aktuell genutzte 2D-Visualisierung von Bordnetz-Kommunikation (CANoe [35])

Eine weitere Möglichkeit der Darstellung ist die Netzwerk-Topologie. Hierbei ist der Aufbau des Bordnetzes in einer 2-dimensionalen Grafik dargestellt. Die Steuergeräte sind dabei über die jeweiligen Bus-Systeme mit einander verbunden und die einzelnen Zustände oder Auslastungen der Steuergeräte können - z. B. über eine farbliche Kodierung - visualisiert werden. Zusätzlich existiert in einer solchen Ansicht die Möglichkeit, sich bestimmte Nachrichten oder Signale in einer Tabelle anzeigen zu lassen, um so die jeweils aktuellen Werte oder Inhalte im Blick zu haben.
Allen bisherigen Visualisierungsansätzen ist gemeinsam, dass sie lediglich 2-dimensionale Darstellungsformen verwenden und somit den Zustand des dreidimensionalen Objekts „Fahrzeug“ nicht vollständig wiedergeben können bzw. sich dabei immer auf einen speziellen Teil des Ganzen

beschränken (z. B. Topologie, einzelne Werte, ...). Ein dreidimensionaler Visualisierungsansatz als Ergänzung zu bestehenden Visualisierungen bietet völlig neue Möglichkeiten einen Überblick über den aktuellen Fahrzeugzustand und im Speziellen auch über den Zustand einzelner Bauteile zu bekommen.
Der Ansatz dieser Diplomarbeit ist, eine solche dreidimensionale Sicht auf die gesammelten Fahrzeugdaten zu entwickeln, die die Vorteile der dreidimensionalen Darstellung für die Analyse und Diagnose der Kommunikationsdaten nutzbar macht. Ein großer Vorteil einer solchen Ansicht ist die Möglichkeit, den jeweiligen Fahrzeugzustand anhand eines 3D-Modells darzustellen und somit die Auswertung vorhandener Daten zu erleichtern, da der Nutzer bestimmte Umgebungsvariablen direkt vom Modell ablesen kann. Ein weiterer Vorteil besteht darin, dass sich der Nutzer frei im Raum bewegen und somit das Gesamtsystem aus unterschiedlichen Blickwinkeln betrachten kann. So werden individuelle und problembezogene Sichten bei der Betrachtung des 3D-Modells ermöglicht.
Allerdings kann auch eine einzelne 3D-Ansicht keine alleinige Gesamtübersicht bieten, die dem Benutzer in jedem Fall das gewünschte Maß an Details zur Verfügung stellt. Daher ist der Ansatz dieser Diplomarbeit vielmehr, eine 3D-Visualisierung als Ergänzung zu vorhandenen 2D-Visualisierungsformen zu erstellen. Dabei soll zu jeder Zeit ein guter Gesamtüberblick über das Fahrzeug und dessen Status möglich sein und bei Bedarf auch verschiedene Bordnetz-Komponenten angemessen dargestellt werden.

1.2 Ziel der Diplomarbeit

Das Ziel dieser Diplomarbeit ist die Entwicklung einer *3D-Ansicht zur Visualisierung von Bordnetzkommunikation im Fahrzeug*. Anschließend soll außerdem eine prototypische Anbindung an bereits bestehende Programme, die als Datenquelle für die 3D-Ansicht dienen können und mit denen diese in Kombination genutzt werden kann, umgesetzt werden.
Diese 3D-Ansicht soll also eine einfach zu handhabende Schnittstelle besitzen, die ohne weitere Probleme von anderen Programmen oder Tools genutzt werden kann. Sie soll unter anderem Methodenaufrufe wie z. B. `openDoorFL` in mehreren Versionen bieten, die unterschiedliche Parameter akzeptieren und somit ein Höchstmaß an Flexibilität bieten. Außerdem soll die Schnittstelle gut zu erweitern sein, falls die vorhandenen Methoden nicht ausreichen oder völlig neue Funktionen zum Modell hinzugefügt werden.
Für eine abschließende Evaluation des entwickelten Prototyps ist es auch erforderlich, eine Anbindung an ein vorhandenes Analyse-Tool umzusetzen. Damit soll der spätere Einsatz einer 3D-Ansicht im Verbund mit bisherigen Visualisierungen untersucht werden können, um dadurch entsprechend treffende Evaluationsergebnisse zu erhalten. Die Ergebnisse der Studie sollen einen Rückschluss auf den Nutzen einer 3D-Ansicht im Bereich der Analyse von Bordnetzkommunikation oder auch der Fehlerdiagnose ermöglichen.

2 Related Work

Dieses Kapitel gliedert sich in drei Teile. Im ersten Teil werden vorhandene Ansätze zur 2D- und 3D-Visualisierung im Bereich der Analyse und Diagnose vorgestellt, darunter zwei bereits vorhandene 3D-Visualisierungen, die bei BMW entstanden sind. Im Anschluss daran werden Projekte aus unterschiedlichen wissenschaftlichen und alltäglichen Bereichen vorgestellt, um einen Überblick über das gesamte Themengebiet der 3D-Visualisierung zu vermitteln. Im letzten Abschnitt dieses Kapitels wird auf allgemeine Aspekte bei der Verwendung von 3D zur Daten-Visualisierung und der Verwendung von 3D-Grafik eingegangen. Dabei werden Probleme behandelt, die bei der Verwendung einer 3D-Welt am Computer auftreten könnnen, wie sie gelöst werden können und welche weiteren Aspekte zu beachten sind. Diese Punkte sind die Grundlage für die späteren Konzeptentscheidungen.

2.1 2D-Visualisierungen im Bereich Analyse/Diagnose

2.1.1 CANalyzer und CANoe

CANalyzer und *CANoe* sind Analysetools der Firma VECTOR [55] zur Analyse und Diagnose von verteilten Systemen. Die Kommunikation über die unterschiedlichen Bus-Systeme in einem Kraftfahrzeug (CAN, LIN, MOST, ...) lassen sich mit diesen Programmen einfach durchführen.

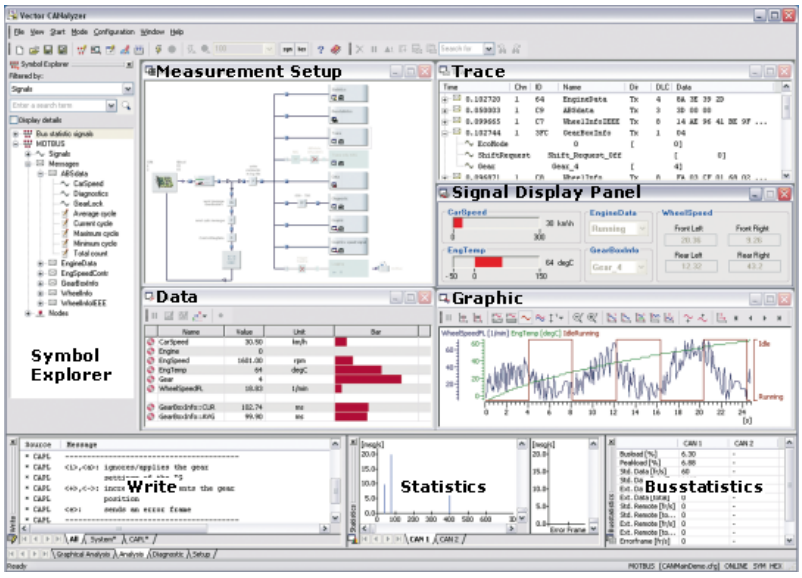


Abbildung 2.1: Hauptansicht von CANalyzer mit verschiedenen Visualisierungen desselben Datensatzes[55].

Zu diesem Zweck werden die vorhandenen Daten in verschiedenen Fenstern auf unterschiedliche Art und Weise aufbereitet und visualisiert. Wie in Abbildung 2.1 zu sehen ist, gibt es im CANalzyer sowohl die Darstellung in Textform, als auch unterschiedliche grafische Darstellungen. CANoe analysiert ebenfalls die Bordnetzkommunikation, beherrscht aber auch die automatisierte Durchführung von Tests. Es unterstützt außerdem die Entwicklung und Analyse von Netzwerken und Steuergeräten.
Im Bereich der Analyse werden diese Tools mit sogenannten „Trace-Daten“ betrieben. Diese Daten werden während einer Testfahrt mit einem Fahrzeug im Bordnetz mitgeschnitten und können in diesen Tools anschließend analysiert werden.

2.1.2 Autobahn-Visualisierung

Das Konzept der zweidimensionalen *Autobahn-Visualisierung* wurde von Benjamin Kunze während seiner Diplomarbeit [13] bei BMW entwickelt. Im Rahmen des Konzepts werden aufgezeichnete Bordnetzdaten zeitabhängig visualisiert. Die Zeit wird dabei durch einen horziontalen Zeitstrahl dargestellt.

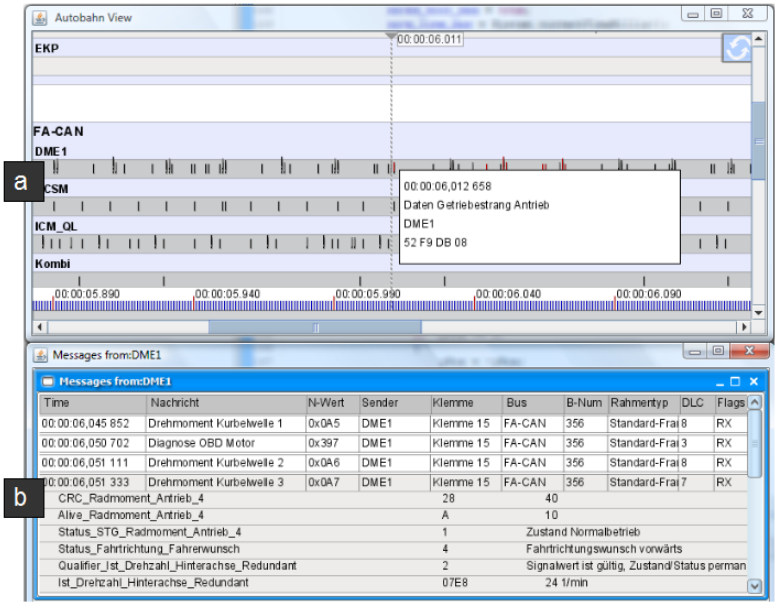


Abbildung 2.2: Autobahn-Visualisierung: (a) Hover-Information zu einer einzelnen Nachricht im Zeitleistenfenster, (b) Tabellenansicht von mehreren Nachrichten [13].

Jedes Steuergerät, das in dem visualisierten Datenfluss vorkommt, wird durch eine horizontal verlaufende Spur oberhalb des Zeitstrahls repräsentiert. Wurden zu einem Zeitpunkt von einem Steuergerät Nachrichten verschickt, wird dies über Rechtecke in der Spur des Steuergeräts zum entsprechenden Zeitpunkt auf dem Zeitstrahl visualisiert (siehe Abbildung 2.2 (a)). Bei mehreren Nachrichten werden die Rechtecke nebeneinander gezeichnet, so dass der Nutzer erkennen kann, ob eine oder mehrere Nachrichten verschickt wurden.
Ausgehend von diesem Gesamtüberblick kann in die Ansicht hineingezoomt werden, um über eine integrierte Tooltip-Funktion genauere Informationen über einzelne Nachrichten zu bekommen. Mit einem Auswahlrahmen, den man aufspannen kann, können diese Details auch in einem eigenen Fenster in einer Tabelle miteinander verglichen werden (siehe Abbildung 2.2 (b)).
Über den Zeitstrahl ist eine sehr einfache Navigation entlang der Zeitachse möglich. Durch die Möglichkeit in die Daten hineinzoomen zu können, um sich weitere Details anzeigen zu lassen, ist es auch möglich, schnell und einfach durch die angezeigten Kommunikationsdaten zu browsen.

2.2 3D-Visualisierungen im Bereich Analyse/Diagnose

Im Bereich der Analyse und Diagnose von Bordnetzkommunikation wurden bei BMW bereits zwei verschiedene Ansätze zur 3D-Visualisierung erstellt. In den beiden folgenden Unterkapiteln werden die jeweiligen Ansätze vorgestellt und genauer beschrieben. Dabei wird sowohl auf die Funktionsweise wie auch kurz auf die genutzte Technik eingegangen.

2.2.1 Konzept zur 3D-Visualisierung von Fahrzeugbordnetzarchitekturen

In den Jahren 2006/2007 wurde an der TU München in Zusammenarbeit mit BMW die Diplomarbeit zum Thema „Konzept zur 3D-Visualisierung von Fahrzeugbordnetzarchitekturen“ von M. Mozdari bearbeitet [17]. Ziel dieser Arbeit war die Erstellung einer sinnvollen 3D-Visualisierung von Bordnetzkommunikation und den daraus resultierenden Abhängigkeiten zwischen den einzelnen Steuergeräten und Bus-Systemen. Zu diesem Zweck wurde die BMW-Bordnetzdatenbank als Datengrundlage genutzt und eine passende Schnittstelle definiert. Die Bordnetzdatenbank umfasst Informationen zu allen verbauten Steuergeräten und Elementen innerhalb des Bordnetzes. Darin gespeichert sind sowohl logische Informationen zur Kommunikation als auch konkrete Informationen wie der Verbauort einer Komponente. Mit Hilfe dieser Schnittstelle war es möglich die, in der Bordnetzdatenbank hinterlegten, Informationen zu den Steuergeräten und Bussen für eine 3D-Visualisierung zu nutzen. Der Fokus lag dabei auf den Abhängigkeiten der verschickten Nachrichten, d.h. welche Nachricht wurde zuerst verschickt und was passierte danach aufgrund eben jener Nachricht.
Nach Festlegung einer Schnittstelle und Bearbeitung eines entsprechend nutzbaren 3D-Modells, konnten die Steuergeräte und Busse im Modell zur Programmlaufzeit platziert und der Nachrichtenfluss visualisiert werden. Die Visualisierung der Nachrichten wird mit Hilfe von animierten Rechtecken realisiert. Wird beispielsweise eine Nachricht von einem Steuergerät abgeschickt, erscheint im entsprechenden Steuergeräte-Objekt ein Rechteck. Dieses Rechteck bewegt sich nun über das dargestellte Bordnetz zum jeweiligen Empfänger. Auf der Seitenfläche des Rechtecks sind dabei Informationen zum Inhalt der Nachricht dargestellt (siehe Abbildung 2.3). Per Selektion einer Nachricht werden zusätzliche Informationen zu den einzelnen Nachrichten oder Steuergeräten in einem Tooltip eingeblendet. Unabhängig von den gesendeten Nachrichten findet am 3D-Modell keine Statusvisualisierung statt, d.h. das 3D-Fahrzeug dient ausschließlich zur Veranschaulichung des Einbauorts der Steuergeräte.

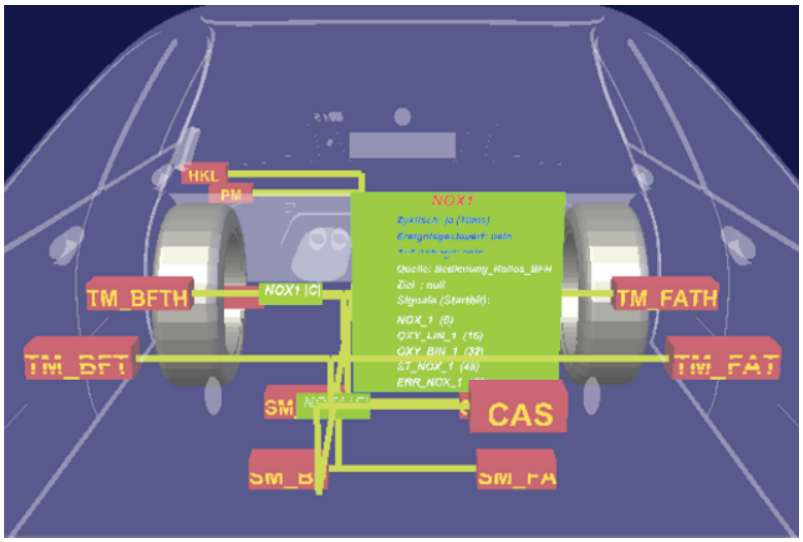


Abbildung 2.3: Nachrichtenfluss im 3D-Modell [17].

Die technische Grundlage des Programms bildet Java3D, das Fahrzeugmodell wurde in 3ds Max bearbeitet und im Anschluss mit dem CyberVRML97-Loader [37] eingeladen. Darüber hinaus wurden die eigentlichen Bordnetzdaten mit JAXB (Java Architecture for XML Bindigs) [44] eingebunden und verarbeitet. Zur Steuerung des Programms dient ausschließlich die Maus, mit der ein Selektionsmenü mit Baumstruktur und das 3D-Modell selbst (Rotation, Zoom, Point & Click)

zu steuern ist.
Der Schwerpunkt dieses Entwurfs liegt klar auf den logischen Zusammenhängen der Kommunikation im Bordnetz. Faktoren wie die Darstellung der mechanischen Fahrzeugreaktionen, die Visualisierung unterschiedlicher Bordnetzzustände und die Möglichkeit einer Darstellung in Echtzeit sind nicht in den Konzeptentwurf eingeflossen, waren aber auch nicht zentraler Bestandteil der Aufgabenstellung.

2.2.2 3D-Visualisierung von Studenten der Purdue University

Es handelt sich hierbei um ein Projekt einer Gruppe Studenten, das im Rahmen einer Übung an der Purdue University in Kollaboration mit BMW entstanden ist. Ziel dieses Projekts war es herauszufinden, ob eine interaktive virtuelle Darstellungsform des automobilen Bordnetzes eine bessere Diskussionsgrundlage zur Problemlösung bietet als die herkömmlichen Datenbanken [20].
Der entwickelte Prototyp verbindet die Visualisierung von Bordnetzkommunikation und die entsprechende Anzeige der mechanischen Reaktion des Fahrzeugs anhand einiger Beispiele. Der Benutzer kann über einige Tasten Aktionen im Fahrzeug auslösen - z. B. die Fenster herunterfahren lassen oder die Scheinwerfer aktivieren. Bevor die Aktion umgesetzt wird, ist im Bordnetz der Nachrichtenfluss und die daraus resultierende Bordnetzbelastung in animierter Form dargestellt. Der Inhalt der Nachricht selbst wird zwar nicht angezeigt, sehr wohl aber die Richtung der Nachricht, d.h. es ist zu erkennen welches Steuergerät die Nachricht abgeschickt hat. Dies wird durch einen leuchtenden Punkt, der sich auf dem Bus-Netz bewegt, visualisiert. Aktive Busse werden in dieser Phase farblich hervorgehoben (hellgrün, statt dunkelgrün) und Steuergeräte entsprechend ihrer Auslastung (Anzahl der zu verarbeitenden Nachrichten) eingefärbt. Die Auslastung wird zusätzlich durch 2D-Anzeigen im oberen rechten Bildschirmbereich mit mehreren Skalen dargestellt. Dabei existiert pro integriertem Steuergerät eine eigene Anzeige mit dem Symbol des jeweiligen Elements und dessen Namen. Darunter befindet sich eine farblich kodierte Skala, auf der ein Balken den aktuellen Auslastungszustand anzeigt. Somit sind die Informationen über die Auslastung der Steuergeräte nicht nur im 3D-Modell visualisiert.

Abbildung 2.4: oben: Steuerungsanzeige und 2D-Skalen, mittig: 3D-Modell zur Statusvisualisierung [20]

Das Programms wurde mit C und OpenGL implementiert. Die 3D-Modelle des Fahrzeugs und des nicht-realistischen Bordnetzes liegen einzeln vor und werden erst im Programmcode zusammengeführt.

Es ist erkennbar, dass der Prototyp der Purdue University auf eine andere Weise als das bereits vorgestellte Tool von M. Mozdari versucht, die Bordnetzkommunikation im Fahrzeug aufzubereiten und zu visualisieren. Dabei wird nicht auf den eigentlichen Nachrichteninhalt eingegangen, sondern es stehen die mechanischen Reaktionen des Fahrzeugs und die Darstellung der Bus-Aktivitäten bzw. Steuergeräteauslastungen im Vordergrund. Dennoch ist eine Visualisierung in Echtzeit auch hier nur eingeschränkt denkbar, denn die Andeutung des Nachrichtenflusses durch leuchtende Punkte wäre in einer Echtzeitdarstellung nicht mehr zu erkennen.

2.3 3D-Visualisierung

Bei dreidimensionaler Informationsvisualisierung werden zwei große Bereiche unterschieden werden - die abstrakte Darstellung von Informationen bei der Informationsvisualisierung (InfoVis) und die realitätsnahe Darstellung von Informationen bei der wissenschaftlichen Visualisierung (SciVis). Dabei ist beiden Gebieten die dreidimensionale Präsentation von Informationen gemein, aber die jeweils dargestellten Informationen haben sehr unterschiedliche Anforderungen an die Art der Präsentation und Informationskodierung. So findet in der Informationsvisualisierung eine Darstellung von nicht-räumlichen bzw. abstrakten Daten statt, die dem Betrachter ein schnelles Verstehen der Daten ermöglicht. Die wissenschaftliche Visualisierung versucht hingegen räumliche Daten, wie z. B. Bilder eines Computertomographen so darzustellen, dass die wesentlichen Informationen dieser Daten zu erkennen sind [27].
Ein wichtiger Bereich in dem die wissenschaftliche 3D-Visualisierung von Informationen heutzutage genutzt wird, ist die Aufbereitung und Darstellung medizinischer Daten. Mögliche Einsatzgebiete für eine medizinische 3D-Visualisierung sind unter anderem computergestützte Anatomieausbildungen und Therapieplanungen. Am Beispiel der Anatomieausbildung lässt sich gut festmachen, dass die Darstellungen im medizinischen Bereich gewissen Anforderungen genügen müssen. Denn gerade in der Medizin, z. B. bei der angesprochenen Anatomieausbildung, müssen komplexe räumliche Verhältnisse dargestellt werden, die sich z. B. durch „konkave und verzweigende Objekte“ [22] ergeben.

Abbildung 2.5: Medizinisch relevante Objekte haben oft komplexe Strukturen [22]

Da es im medizinischen Bereich nicht immer möglich ist, vorhandene Objekte zur besseren Darstellung in ihrer Form, Größe oder Position zu verändern, müssen hier entsprechende Maßnahmen getroffen werden, um eine effektive Arbeit mit den komplexen 3D-Daten zu ermöglichen.
Ein anderer Bereich, in dem die Darstellung von Informationen im 3D-Raum genutzt wird, ist die Verarbeitung und Darstellung geo-wissenschaftlicher Informationen mit Hilfe dreidimensionaler

Modelle. Eine recht simple Form der Visualisierung ist dabei die Nachbildung der realen Welt, wie sie zum Beispiel bei *Google Earth* [40] oder *Virtual Earth* [56] von Microsoft vorgenommen wird. Dort kann der Nutzer um einen virtuellen Globus navigieren, herein- und herauszoomen und stellenweise sehr detaillierte Satellitenaufnahmen der Erdoberfläche betrachten. Seit einiger Zeit wird das Programm jedoch auch um 3D-Modelle erweitert. Dabei werden in vielen größeren Städten mittlerweile Modelle von großen oder bekannten Gebäuden in die 3D-Welt eingebaut und, wie in Abbildung 2.6, mittlerweile sogar mit korrekten Texturen versehen.

Abbildung 2.6: Blick auf New York in Microsofts VirtualEarth [56]

Abgesehen von dieser eher trivialen und wenig komplexen Visualisierung mit Hilfe eines Globus, wird eine 3D-Visualisierung zum Beispiel bei *EarthSeaTrilogy* [51] dazu genutzt, um den Küstenverlauf Europas über die Zeit darzustellen. Durch verschiedene Interaktionsmöglichkeiten kann man die Veränderungen und Entwicklungen des Küstenverlaufs schnell sichtbar machen. Abbildung 2.7 vermittelt einen Eindruck dieser Visualisierung.

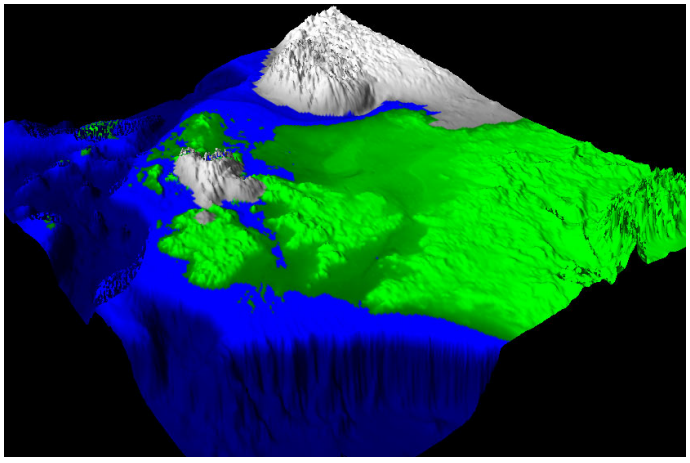


Abbildung 2.7: Screenshot des Tools EarthSeaTrilogy, das den Küstenverlauf Europas in 3D visualisiert [51]

Weiterhin können mit Hilfe eines Globus auch statistische Daten, die ein einzelnes oder mehrere

Länder betreffen (z. B. Warenströme eines Landes), dargestellt werden (siehe Abbildung 2.8) und gleichzeitig sind sowohl die Größe der Warenströme, als auch Quellen- und Zielländer auf einen Blick erkennbar.

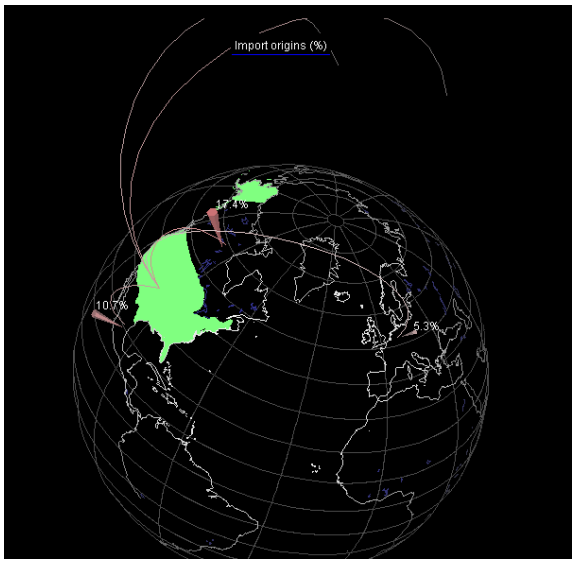


Abbildung 2.8: Screenshot des Java-Tools Global-i von Infomagnet [39]

Mithilfe von 3D-Visualisierungen lässt sich auch eine Vielzahl von verschiedenen Daten in einem einzigen Modell unterbringen und dennoch übersichtlich darstellen. Ein Beispiel hierfür ist *LibViz*, ein Programm, welches sich mit der Visualisierung von Messwerten in ihrem räumlichen Kontext innerhalb eines Gebäudes, speziell einer alten Bibliothek, beschäftigt [25]. Im konkreten Fall geht es um Messwerte, wie Luftfeuchte, Staub u. ä., in der „Old Library“ im Trinity College in Dublin. Anstelle einer 2D-Visualisierung der Messwerte, werden diese innerhalb eines 3D-Modells des Gebäudes an den entsprechenden Messpunkten visualisiert, siehe Abbildung 2.9.

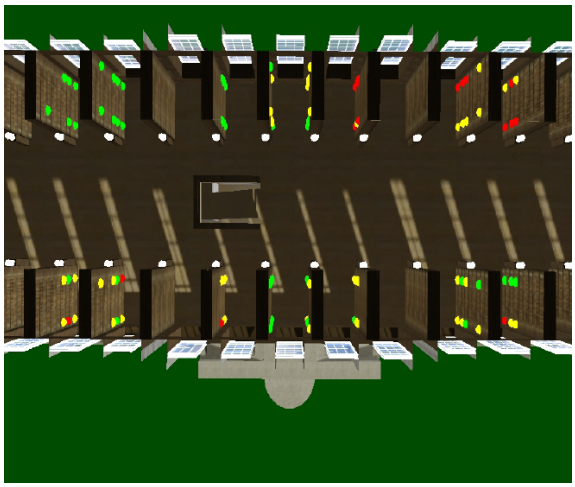


Abbildung 2.9: Screenshot der LibViz Visualisierung

Der Nutzer kann durch das Modell navigieren und hat die Möglichkeit die Messwerte der einzelnen Stationen zu betrachten. Durch die räumliche Darstellung und Positionierung der ange-

zeigten Messwerte innerhalb des Modells, können Zusammenhänge zwischen unterschiedlichen Messwerten erkannt werden. Außerdem ist es möglich, die genauen Positionen der Messstationen zu erkennen, ohne weitere Pläne oder Informationen heranziehen zu müssen.
Abgesehen von reiner Visualisierung lassen sich 3D-Szenen zusätzlich als Benutzeroberfläche einsetzen. Ein Beispiel dafür ist der Ansatz von Husoy und Skourup [8]. Sie nutzen das 3D-Modell einer industriellen Anlage sowohl zur Visualisierung von Informationen als auch zur direkten Steuerung der Anlage (siehe Abbildung 2.10). Sie zeigen anhand des 3D-Modells dieselben Daten an wie mit herkömmlichen Benutzeroberflächen, jedoch werden die Daten anders kodiert und durch das Verhalten des 3D-Modells dargestellt. Durch die Möglichkeit des direkten Eingreifens im Modell für den Benutzer, wird dieser entlastet, da alle Informationen und Steuerungsmöglichkeiten direkt in einer Oberfläche integriert sind.

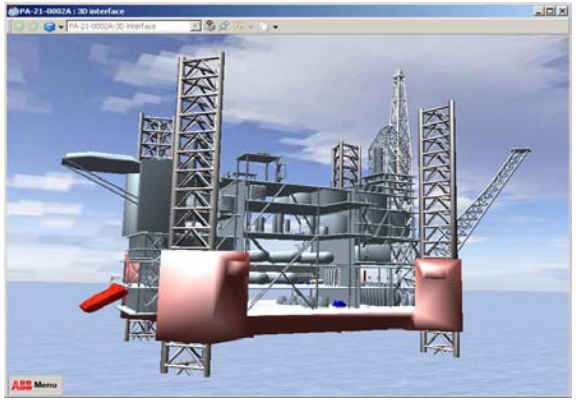


Abbildung 2.10: Beispiel der Benutzeroberfläche nach Husoy und Skourup [8]

Die bisher gezeigten Ansätze nutzen zur Darstellung einer 3D-Welt die zweidimensionale Darstellungsfläche eines Bildschirms. Einen etwas anderen und sehr auf die Wirkung und Umsetzung der 3D-Visualisierung zielenden Weg geht das Projekt „Full immersive virtual environment CAVE in chemistry education“ von Maria Limniou et al. [15]. Grundlage dieser Arbeit ist eine sogenannte CAVE - eine Umgebung, in der eine nahezu 360°Projektion von dreidimensionalen Bildern stattfinden kann (siehe Abbildung 2.11).

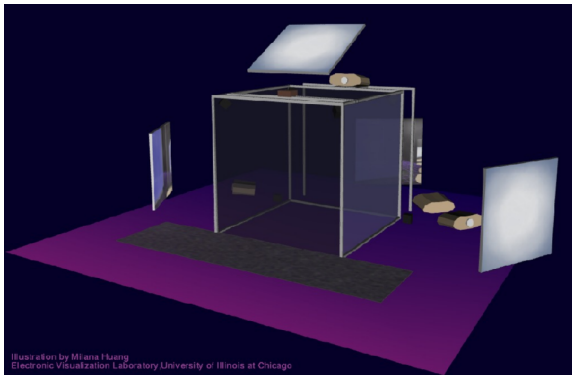


Abbildung 2.11: Schematischer Aufbau einer CAVE-Umgebung [52]

Die räumliche Darstellung beschränkt sich hierbei also nicht auf eine beschränkte Fläche im Blickfeld des Nutzers, sondern umgibt diesen fast vollständig. Limniou et al. nutzen eine solche CAVE,

um in einer Studie zu prüfen, ob eine aufwändige 3D-Projektion beim Verständnis von chemischen Molekülen eine größere Hilfestellung bietet als eine Darstellung auf einem Desktop-Bildschirm. Die Ergebnisse haben gezeigt, dass es für Studenten einfacher ist, sich mit Hilfe der CAVE-Umgebung das Verständnis für neue Moleküle und deren Aufbau anzueignenen, als auf herkömmliche Weise am Desktop-Monitor. Ein weiterer Effekt, der durch die Studie gezeigt wurde, ist die Tatsache, dass die CAVE-Nutzer enthusiastischer gearbeitet haben und einen weitaus größeren Grad der Immersion erfahren haben, als die Desktop-Nutzer[15]. Die Begründung ist, dass die Teilnehmer den Eindruck hatten, die Moleküle seien als reale Objekte vor ihnen und nicht lediglich Abbildungen auf einem Bildschirm. Hier zeigt sich das große Potential, welches die Immersion durch 3D-Welten bietet. Der Nutzer sieht nicht länger nur zu, sondern fühlt sich als Teil der 3D-Umgebung und nimmt die dargestellten Objekte entsprechend anders wahr.

2.4 Multi Coordinated View Systeme

Multi Coordinated View Systeme (MCV-Systeme) werden dazu genutzt, hochkomplexe und mehrdimensionale Informationen zu visualisieren [19]. Dabei werden unterschiedliche Visualisierungstechniken genutzt, um verschiedene Sichten auf ein und denselben Datensatz zu bieten. Durch die verschiedenen Visualisierungen wird es möglich, Zusammenhänge zwischen Daten aufzuzeigen, die mit einer einzelnen Ansicht nicht zu erkennen sind [1].

2.5 Allgemeine Regeln zur Verwendung von 3D für Visualisierungen

In diesem Kapitel werden wesentliche Punkte, die bei einer 3D-Visualisierung zu beachten sind, aufgeführt und näher beschrieben. Diese Regeln sind die Grundlage für die spätere Implementierung des Prototyps und werden im weiteren Verlauf der Arbeit immer wieder eine Rolle spielen. Zur besseren Übersicht sind die einzelnen Punkte in Kategorien zusammengefasst.

2.5.1 Verdeckungen im 3D-Raum

Ein wesentliches Problem bei einer 3D-Visualisierung ist die mögliche Verdeckung von Objekten durch andere Objekte. Denn durch Verdeckungen wird das Sichtfeld in die 3D-Welt eingeschränkt und somit die Möglichkeit, Dinge in der 3D-Welt wahrzunehmen. Je nach Position im dreidimensionalen Raum und Sichtwinkel auf ein Objekt sind eventuell andere Objekte im Vordergrund und verbergen so einen Teil des eigentlichen Zielobjekts.

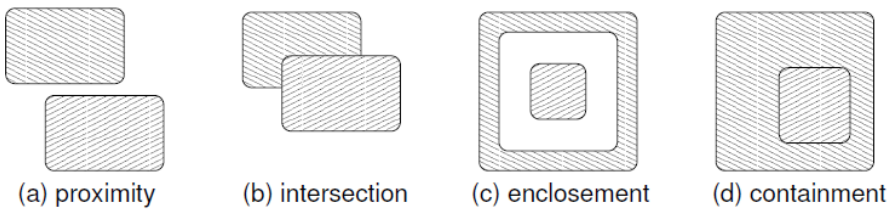


Abbildung 2.12: Objektinteraktionen nach [6], die Verdeckungen verursachen können

Elmqvist [6] definiert in diesem Zusammenhang den Parameter „Object Interaction“ und beschreibt damit die möglichen Zustände, in denen sich zwei Objekte im 3D-Raum zueinander befinden können:

none Es existiert keine Verbindung zwischen den Objekten (eigentlich nur realistisch bei Einzelobjekten)

proximity Die Objekte befinden sich nahe beieinander ohne sich jedoch zu überschneiden; je nach Sichtwinkel kann es zu Verdeckungen kommen

intersection Objekte überschneiden sich und somit kommt es zwangsläufig zu Verdeckungen

enclosement Ein oder mehrere Objekte umschließen ein anderes Objekt ohne es zu beinhalten und verdecken es dadurch

containment Ein Objekt beinhaltet ein anderes Objekt vollständig und verdeckt es somit aus jedem Blickwinkel

Da ein Fahrzeug ein sehr komplexes 3D-Modell besitzt, wird es zwangsläufig zu Verdeckungen kommen. Das bedeutet, dass eventuell wichtige oder relevante Informationen für den Benutzer unsichtbar werden. Es muss also eine geeignete Methode angeboten werden, dass der Nutzer, falls wichtige Informationen aus einem Blickwinkel verdeckt sind, die für ihn relevanten Informationen sichtbar machen kann. Um dieses Ziel zu erreichen, bietet sich die (Semi-)Transparenz von Objekten an, denn Transparenz ermöglicht es, verdeckte Objekte ganz oder teilwese sichtbar zu machen. Darüber hinaus verstärkt Semi-Transparenz den Tiefeneindruck innerhalb der 3D-Szene.

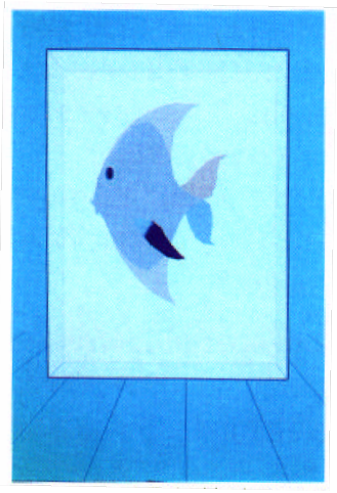


Abbildung 2.13: *Silk Cursor* mit dem Zhai [30] die Nutzung von Semi-Transparenz in 3D-Szenen evaluiert hat. Der Fisch befindet sich nur teilweise innerhalb des semitransparenten 3D-Cursors.

Wie Zhai [30] bereits 1994 mit seiner Untersuchung zum *Silk Cursor* festgestellt hat, lässt sich durch Semi-Transparenz sehr einfach erkennen, ob ein Objekt vor, hinter oder in einem anderen Objekt ist. Im Rahmen eines so komplexen Modells wie einem Fahrzeug, sollte daher unbedingt Transparenz genutzt werden, um sowohl den Tiefeneindruck, als auch die Minimierung von Verdeckungen zu unterstützen.

2.5.2 Hervorhebungstechniken im 3D-Raum

Ein weiterer Punkt, der zu beachten ist, ergibt sich aus der Aufgabenstellung: mechanische Reaktionen am Fahrzeug-Modell sollen sichtbar sein. Unabhängig von eventuellen Verdeckungen ist es nicht immer möglich zu garantieren, dass eine Aktion allein durch ihre Bewegung (Tür öffnet sich) oder Farbveränderung (Blinker blinkt) ausreichend auffällt, um vom Benutzer auch im peripheren

Blickfeld wahrgenommen zu werden. Somit muss eine geeignete Lösung gefunden werden, die es ermöglicht, Aktionen bzw. an Aktionen beteiligte Objekte hervorzuheben, zur schnellen Erfassung durch den Nutzer.
Grundlage hierfür ist die sogenannte präattentive Wahrnehmung des Menschen. Diese Wahrnehmung dafür sorgt, dass in einer großen Menge von Objekten einzelne Objekte, die sich in wesentlichen Faktoren vom Rest der Menge unterscheiden (z. B. Form, Farbe), sehr schnell identifiziert werden können.

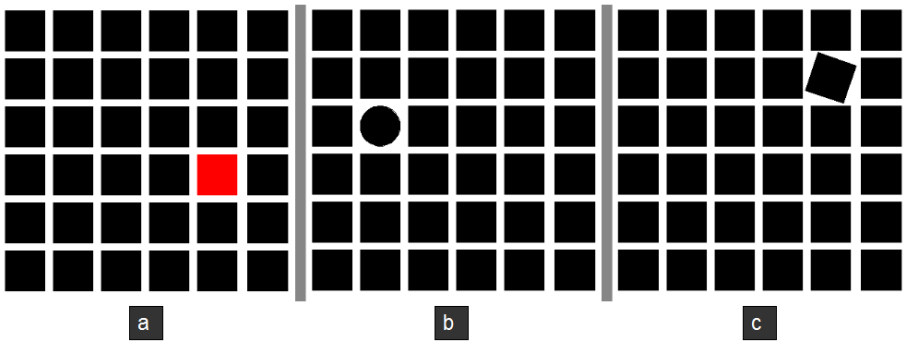


Abbildung 2.14: Unterschiedliche Faktoren können präattentiv erfasst werden: Farbe, Form, Orientierung

Hierzu führten Healey et. al [7] im Jahr 1996 eine Studie durch, in der getestet wurde, welche Faktoren eines Objekts ausreichend sind, um präattentiv Veränderungen bzw. Unterschiede wahrzunehmen. Dabei zeigten sie, dass es mehrere Faktoren gibt, die man zur Informationskodierung nutzen kann, ohne auf präattentive Wahrnehmung bzw. Verarbeitung der Informationen verzichten zu müssen. In einer Reihe von Experimenten belegten sie, dass zum Beispiel die Helligkeit und Orientierung eines Objekts solche Faktoren sind. Abbildung 2.14 verdeutlicht den Effekt der präattentiven Wahrnehmung mit Hilfe dieser Faktoren.
Abgesehen von der *Helligkeit eines Objekts* kann zum Beispiel auch die *Farbe eines Objekts* dazu genutzt werden, einen Pop-Out-Effekt zu erzeugen [28]. Farbliche Unterschiede können vom Menschen ebenfalls sehr schnell wahrgenommen und ohne bewusstes Nachdenken verarbeitet werden. Chipman [50] hat 1996 eine Aufstellung erarbeitet, in der er Eigenschaften von Objekten auflistet, die bereits als präattentive Faktoren identifiziert wurden.

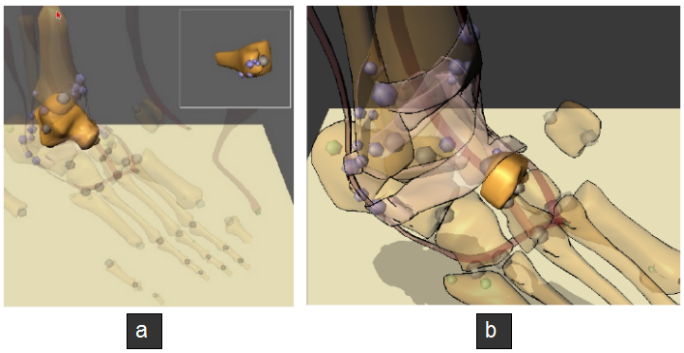


Abbildung 2.15: (a) Hervorhebung durch Transparenz; (b) zusätzlich zur Transparenz Anzeige von Konturlinien [22]

Neben den dort gelisteten Faktoren ist auch eine Hervorhebung mittels *Transparenz* möglich. Dabei werden alle unwichtigen Elemente der Szene transparent dargestellt und lediglich der Teil, der von Interesse für den Nutzer ist, wird undurchsichtig gerendert. Wie in Abbildung 2.15 zu erkennen ist, stechen die undurchsichtigen Teile der Szene sofort ins Auge - verwendet man bei den transparenten Objekten zusätzlich noch Konturlinien, werden auch deren Formen gut sichtbar, ohne jedoch dadurch die wichtigen Objekte zu verdecken. Es bleiben also mehr Informationen für den Nutzer sichtbar, ohne zu viele andere Infomationen dadurch zu verdecken.

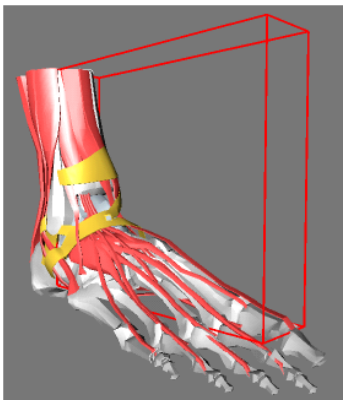


Abbildung 2.16: Hervorhebung eines Muskels und einer Sehne mittels Bounding Box [22]

Abgesehen von direkter Kodierung, gibt es noch weitere Möglichkeiten der Hervorhebung in einer 3D-Szene. Eine davon ist die Darstellung der sogenannten *Bounding Volumes*. Diese sehr einfachen Körper (Boxen, Sphären, etc.) sind die jeweils kleinstmöglichen Körper der jeweiligen Form, die ein gegebenes 3D-Objekt im Raum umspannen. Allerdings ist diese Art der Hervorhebung durchaus problematisch, wenn große Objekte damit hervorgehoben werden sollen. Dann ist nicht immer zweifelsfrei zu erkennen, welcher Teil der Gesamtszene hervorgehoben sein soll, wie in Abbildung 2.16 zu erkennen. Dort wäre auch denkbar, dass nicht nur Muskel und Sehne, sondern auch ein Teil des Knochens hervorgehoben werden soll, da die BoundingBox ein entsprechend großes Gebiet umschließt.

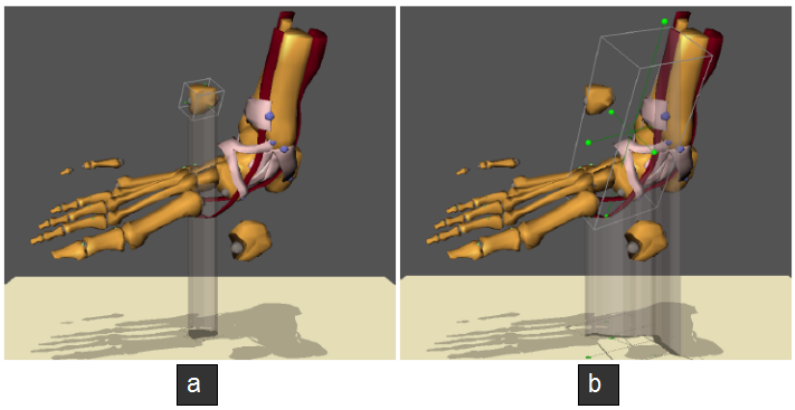


Abbildung 2.17: Hervorhebung eines Knochens durch Schattenwurf und Schattenkörper [22]

Preim [22] stellt noch zwei weitere Arten der Hervorhebung vor, die von ihm zwar speziell im medizinischen Bereich präsentiert werden, aber auch auf andere 3D-Szenen angewendet werden

können. Eine der beiden Techniken ist der gezielte Einsatz von *Schattenwurf und Schattenkörpern*. Dabei erhalten die wichtigen Objekte einen dunkleren Schatten und sind durch einen sogenannten Schattenkörper - eine Art Säule, die vom Objekt zum Boden führt und nicht vollkommen durchsichtg ist - mit dem Schatten verbunden. So wird das Auge des Betrachters vom dunkleren Schatten, da er sich von seiner Umgebung abhebt (siehe Abbildung 2.14 links), über den Schattenkörper zum eigentlichen Objekt geführt. Hierbei gibt es allerdings Probleme, wenn das Objekt eine längliche Form hat, da dann der Schatten entsprechend klein ausfällt und schwerer zu entdecken ist. Abbildung 2.17 zeigt dabei auf der linken Seite das Prinzip der Hervorhebung durch Schattenwurf mit einem kompakten Objekt, bei dem der Schatten gut sichtbar ist. Auf der rechten Seite kann man erkennen, dass diese Technik bei länglichen Objekten eher weniger geeignet ist.
Eine andere Hervorhebungstechnik, die in [22] vorgestellt wird, ist die Verwendung von *Fadenkreuzen*, genauer gesagt von einem Fadenkreuz-Overlay. Bei dieser Technik wird die Position des Mittelpunktes des hervorzuhebenden Objekts berechnet und dieser Punkt bildet den Mittelpunkt eines zweidimensionalen Fadenkreuzes, das als 2D-Overlay über dem dargestellten 3D-Modell zu sehen ist. Da die Linien des Fadenkreuzes bis zum Bildrand reichen, ist es für den Benutzer relativ einfach anhand der Linien den angepeilten Mittelpunkt des Objekts und somit auch das Objekt selbst zu finden. Schwierig wird diese relativ einfache Methode zur Hervorhebung jedoch, wenn das eigentliche Objekte von anderen Objekten verdeckt ist, denn dann kann der Nutzer nicht zweifelsfrei erkennen welches Objekt gemeint ist. Ideal wäre also eine Kombination verschiedener - bereits vorgestellter - Methoden, um eine adäquate Betonung des Objekts erreichen zu können. In Abbildung 2.18 werden mehrere Hervorhebungstechniken genutzt.

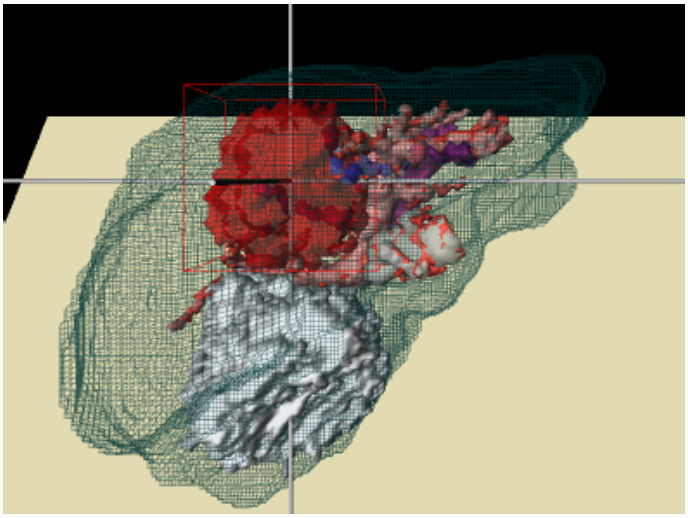


Abbildung 2.18: Hervorhebung eines Lebertumors durch Bounding Box, Einfärbung und Darstellung eines Fadenkreuzes. [22]

Die bisherigen Techniken zur Herausstellung einzelner Teile eines Modells lassen das Gesamtmodell von seiner Positionierung stets unangetastet. Es existieren aber auch Ansätze, die das Modell bzw. Teile des Modells so verschieben, dass die gewünschten Objekte besser sichtbar sind. Elmquvist [5] hat 2005 den *BalloonProbe* Ansatz vorgestellt. Er geht dabei von einer Menge von Objekten aus, die im 3D-Raum verteilt sind und sich dabei gegenseitig auch so verdecken, dass die Sicht auf innen gelegene Teile der 3D-Welt versperrt ist. Soll nun ein Objekt im Inneren hervorgehoben werden, würde aufgrund der Verdeckung keiner der bisher vorgestellten Ansätze wirklich helfen, außer die Veränderung der Transparenz aller umliegenden bzw. verdeckenden Modellteile. Ist dies aber aufgrund von z. B. farblichen Kodierungen oder ähnlichem nicht ohne weiteres mög-

lich, muss zu anderen Mitteln, wie der BaloonProbe gegriffen werden. Bei dieser Technik wird um das ausgewählte Objekt eine Art imagnäres Kraftfeld erzeugt, das die umliegenden Objekte vom Zentrum des Feldes - also dem hervorzuhebenden Objekt - nach außen „drückt“ (Abbildung 2.19).

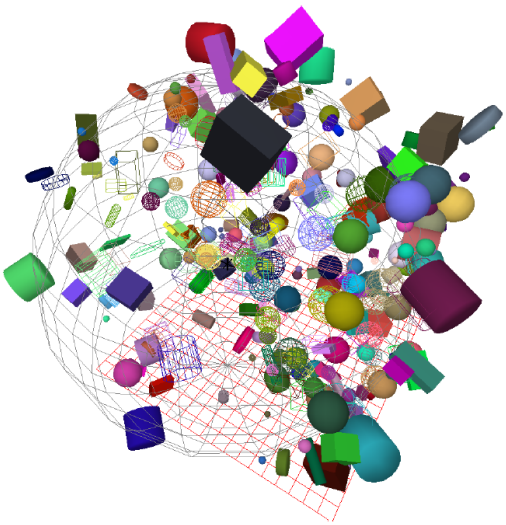


Abbildung 2.19: Funktionsprinzip der BaloonProbe Technik [5].

Dadurch werden die Verdeckungen des Zielobjekts verringert und es wird besser sichtbar, allerdings werden die räumlichen Gegebenheiten des Modells verändert, d.h. es befinden sich nicht mehr alle Objekte an ihrem ursprünglichen Platz in der 3D-Welt. An den Ursprungsstandorten werden lediglich Gitterlinien-Modelle der eigentlichen Objekte dargestellt.

Hervorhebungstechnik	Farbe	Trans-parenz	Bounding Volumes	Schatten-wurf	Faden-kreuz	Balloon-Probe
präattentiv	+	+	+	+	+	-
verdeckungsauflösend	-	+	-	-	-	+
eigenschaftserhaltend	-	-	+	+	+	+
positionserhaltend	+	+	+	+	+	-

Tabelle 2.1: Merkmale der verschiedenen Hervorhebungstechniken

Der Katalog der hier vorgestellten Hervorhebungstechniken ist selbstverständlich nicht vollständig und zeigt auch nur einen kleinen Teil der für diese Arbeit relevanten Möglichkeiten auf. Dennoch machen die angesprochenen Techniken deutlich, dass sie sich auch auf unterschiedliche Art und Weise kombinieren lassen. Durch eine Kombination mehrerer Hervorhebungstechniken können Elemente einer 3D-Szene zum Teil weit besser hervorgehoben werden, als wenn dies mit einer einzigen Technik realisiert wird. Es wird aber auch deutlich, dass es sich beim Gebiet der Hervorhebung von Objekten im 3D-Raum um ein komplexes Problem handelt. Denn die nutzbaren Hevorhebungstechniken hängen auch stark vom jeweiligen Einsatzgebiet des 3D-Modells ab und welche Priorität z. B. die Positionierung der Elemente innerhalb des Gesamtmodells einnimmt.

2.5.3 Kameramodelle

Ein weiterer Punkt, der bei einer 3D-Visualisierung und insgesamt beim Umgang mit 3D-Darstellungen nicht vernachlässigt werden darf, ist die Steuerung des Programms bzw. der Kamera. Denn die Kamera stellt das Sichtfenster des Benutzers in die 3D-Welt dar und ist dafür verantwortlich, wie er die dargestellte Szene wahrnimmt. Dabei sollte das Kamerasystem dem Nutzer möglichst keine Einschränkungen aufzwingen, die ihn bei einer effizienten Nutzung des Programms behindern. Solche Einschränkungen können Sichtwinkel sein, die nicht erreicht werden können oder eine zu umständliche Steuerung. Um zu gewährleisten, dass eben genau solche Fehler nicht gemacht werden, muss zunächst untersucht werden, welche Arten der Kamerapositionierung und -steuerung es gibt. Im Bereich der Computerspiele, die in der heutigen Zeit nur noch selten ohne aufwändige 3D-Grafik auskommen, haben sich drei, für diese Arbeit relevante, Kamerapositionierungen herauskristallisiert. Christie et al. [3] haben diese folgendermaßen kategorisiert und zugleich die Merkmale und Besonderheiten der einzelnen Verfahren zusammengefasst:

First person camera Der Nutzer sieht die 3D-Welt durch die Augen eines imaginären Avatars und hat direkten Einfluss auf Blickrichtung und Bewegung der Kamera.

Third person camera Bei der Third person camera sieht der Nutzer von hinten auf seinen Avatar in der 3D-Welt und steuert diesen. Meist lässt sich dabei die Kamera unabhängig vom Avatar drehen und neigen, aber bei Bewegungen des Avatars folgt ihm die Kamera.

Action replay camera Diese Art der Kameraführung versucht durch unterschiedliche Blickwinkel aussagekräftige Bilder zu schaffen, aber dadurch hat der Nutzer keinen wirklichen Einfluss auf die gezeigten Bilder.

Abbildung 2.20: (a) First person camera [38]; (b) Third person camera [53]; (c) Action replay camera [36]

Im wesentlichen kann man zusätzlich zu den genannten Positionierungen noch zwei Arten der Kamerasteuerung unterscheiden: eine automatisierte Kamera und eine völlig frei konfigurierbare und bewegliche Kamera. Die automatische Kameraführung hat den Vorteil, dass der Nutzer bei der Einschätzung und Orientierung innerhalb der 3D-Szene unterstützt wird. Nnadi et al. [18] haben einen selbstentwickelten Prototyp getestet, bei dem die Kamera hinter dem Avatar des Benutzers in der 3D-Welt schwebte, d.h. es wurde eine Third person camera verwendet (siehe Abbildung 2.20). Betritt der Avatar eine neue Szene wird die Kamera automatisch so gesteuert, dass wichtige Objekte der Szene in den Fokus des Nutzers gerückt werden. Die These war, dass das Erkennen wichtiger Objekte dadurch vereinfacht wird und der Nutzer so bei der Einschätzung der neuen Umgebung unterstützt wird. Die Ergebnisse der Studie deuten daraufhin, dass diese Unterstützung des Nutzers durch die automatische Kamera einen positiven Einfluss auf das Erkennen von wichtigen Objekten und dem Zurechtfinden innerhalb der Szene hat. Auch wenn

die Ergebnisse nicht sonderlich eindeutig sind, so gibt es dennoch einen geringen, aber statistisch signifikanten, Unterschied zugunsten der automatisierten Kamera.
Falls es in der 3D-Szene keinen Avatar des Nutzers gibt, sondern die Kamera in Form einer First person camera verwendet wird, lässt sich auch ein in [21] genutzter Ansatz verwenden. Dabei wird die Kamera stets automatisch positioniert unter der Zielsetzung, möglichst viele Eckpunkte des gewünschten Objekts sichtbar zu machen. Es wird also angenommen, dass das Objekt immer dann am besten zu sehen bzw. zu erkennen ist, wenn eine möglichst hohe Zahl an Eckpunkten zu sehen ist. Dieser Ansatz hat aber nach Preim [21] das Problem, dass die Kamera zum Teil in sehr unnatürliche Positionen gebracht wird. Der Blickwinkel sei zwar unter dem Gesichtspunkt, eine maximale Anzahl an Eckpunkten darzustellen, optimal gewählt, sei aber für den Nutzer gewöhnungsbedürftig, da dieser sich aus dem teilweise ungewohnten Blickwinkel heraus neu orientieren müsse.

2.5.4 Textdarstellung in 3D

Ein Problem, das bei einer 3D-Informationsvisualisierung zu beachten ist, ist die Darstellung von Text-Informationen in der Visualisierung. In einer 3D-Szene kann Text auf zwei unterschiedliche Arten präsentiert werden. Eine Möglichkeit ist die *parallele Ausrichtung zur Kamera*, d.h. der Text ist dem Betrachter immer zugewandt - unabhängig von der sonstigen Kameraposition und -rotation. Diese Art der Darstellung ist häufig bei Computerspielen anzutreffen. Dort werden auf diese Weise Informationen präsentiert, die ansonsten nicht in der Spielwelt dargestellt werden können. Ein Beispiel hierfür ist die Anzeige der „Lebenspunkte“ in vielen Ego-Shootern oder eine Ressourcenanzeige bei diversen Strategiespielen (siehe Abbildung 2.21). Während die Darstellung von Informationen auf diese Art in Computerspielen die Immersion stört und daher so wenig wie möglich genutzt wird, stellt sie für die 3D-Viualisierung eine gute Möglichkeit dar, dem Nutzer Informationen gut lesbar zu präsentieren.

Abbildung 2.21: (a) Anzeige mit HUD-Elementen stören die Immersion; (b) gute Immersion ohne HUD-Elemente [57]

Abgesehen von dieser Art der Darstellung kann man einen Text auch direkt *auf die Seite eines Objekts rendern*. Dadurch wird der Text ein Teil der Gesamtszene und ist dadurch natürlich aufgrund von Verdeckungen (vgl. Abbildung 2.12) nicht immer lesbar. Larson et. al haben 2000 die erste Untersuchung zu diesem Thema unternommen [14]. Das Ziel der Untersuchung war es, herauszufinden, inwiefern die rotierte Darstellung von Text einen Einfluss auf die Lesbarkeit im Allgemeinen und die Lesegeschwindigkeit in Bezug auf die verwendete Schriftgröße hat. Zu

diesem Zweck wurden Probanden vorgerenderte Bitmaps (siehe Abbildung 2.22) gezeigt, auf denen der dargestellte Text jeweils in einem bestimmten Winkel rotiert wurde.

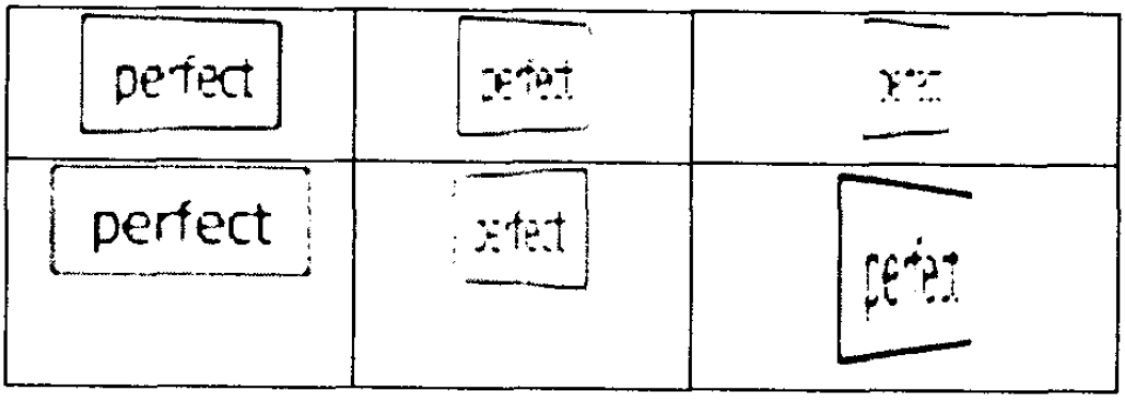


Abbildung 2.22: Einige der verwendeten Text-Bausteine [14]

Insgesamt gab es 15 Bilder (7 links gedreht, 7 rechts gedreht, 1 parallel) und es wurde die Zeit gemessen, die benötigt wurde, um den dargestellten Text zu lesen. Die Studie hat gezeigt, dass ein rotierter Text schwieriger zu lesen ist als ein parallel zur Bildschirmfläche dargestellter Text. Zwar lässt sich dem Effekt mit einer größeren Schriftart bis zu einem gewissen Grad entgegenwirken, aber der Effekt selbst bleibt in abgeschwächter Form erhalten. Während geringe Winkel nur wenig Einfluss auf die Lesbarkeit und Lesegeschwindigkeit hatten, zeigte sich bei größeren Winkeln (>60°) eine deutlich verlängerte Lesezeit.

2.5.5 Zusammenfassung

Insgesamt wurden in diesem Kapitel teils völlig unterschiedliche Ansätze der 3D-Visualisierung oder Techniken im Umgang mit 3D-Darstellungen aufgezeigt. Diese Ansätze können jedoch nur einen kleinen Teil dessen wiedergeben, was auf dem Feld der dreidimensionalen Grafik aktuell entwickelt wird. Daher sollte das Kapitel vor allem dazu dienen, einen breitgefächerten Eindruck zum Thema 3D zu vermitteln und sich nicht in Detailfragen zu verlieren. Dabei wurden gezielt Themen ausgesucht, die zu erwartende Probleme bei der Entwicklung des Visualisierungsprototyps behandeln, wie zum Beispiel Verdeckungen. Das Kapitel stellt viele Ansätze vor und demonstriert die jeweilige Umsetzung anhand von Bildern.
Aber gerade im Hinblick auf umfangreiche Hardware-Installationen, die bei einer 3D-Darstellung von Informationen hilfreich sein können (z. B. die CAVE-Umgebung [15] (siehe Abbildung 2.11)), scheint es noch ein enormes Forschungs- und Einsatzpotential zu geben, das momentan noch nicht überschaubar ist. Gerade im Bereich der Analyse und Diagnose sind die Einsatzmöglichkeiten solcher Installationen bis jetzt nicht in Betracht gezogen worden.

3 Anforderungsanalyse

Dieses Kapitel beschreibt die Evaluation der beiden bereits vorhandenen 3D-Prototypen, da sie in diesem Zusammenhang noch nicht evaluiert wurden. Im ersten Abschnitt wird daher zuerst das Ziel der qualitativen Evaluation bzw. die zu prüfende Hypothese aufgestellt und aufgrund des bisher vorgestellten Related Works begründet. Anschließend wird genauer auf einzelne Aspekte eingegangen, die während der Tests herausgefunden werden sollten. Im zweiten Abschnitt des Kapitels werden die Ergebnisse der Studie vorgestellt und besprochen. Den Abschluss bildet eine Abschätzung darüber, welche Schlüsse diese Ergebnisse für den neuen Prototyp zulassen.

3.1 Ziele der Evaluation

Da zum Thema 3D-Visualiserung im Bereich der Analyse und Diagnose zu Beginn dieser Diplomarbeit bereits zwei Prototypen vorlagen, die jedoch nicht oder nicht ausreichend evaluiert sind, ist der erste Schritt bei der Entwicklung eines neuen Konzepts die Evaluation der vorhandenen Konzepte. Durch eine ausführliche Evaluation mit Hilfe von Analyse-/Diagnose-Experten bei BMW sollen bestehende Probleme der Visualisierungen im Bereich der Analyse und Diagnose gefunden werden. Durch die Einbeziehung von Experten sollte sichergestellt werden, dass die Ergebnisse praxisbezogen sind und es darüberhinaus auch nützliche Anregungen geben kann in Bezug auf völlig neue Ideen der Visualisierung. Die befragten Experten sind mit dem Themengebiet Bordnetzkommunikation vertraut und kennen mindestens eine zweidimensionale Visualisierung aus den aktuell bei BMW genutzten Analyse-/Diagnose-Programmen. Sie können also gut beurteilen, welche Visualisierungsformen einen Mehrwert für die Analyse und Diagnose bedeuten. Daher können gute Lösungsansätze der vorhandenen Prototypen erkannt und für diese Arbeit entsprechend angepasst und optimiert werden.
Im Rahmen eines User-centered-approach wurde entschieden, eine qualitative Evaluation mit Hilfe von Leitfadeninterviews durchzuführen. Der Unterschied zur quantitativen Studie besteht dabei vor allem darin, dass es deutlich weniger Teilnehmer gibt, aber die Einzelgespräche mit diesen intensiver geführt werden können. Es stehen dabei weniger Statistiken im Vordergrund, sondern viel mehr persönliche Kommentare und Meinungen der einzelnen Probanden. So war es auch möglich, abgesehen von Antworten auf konkret gestellte Fragen, Rückmeldungen aufzunehmen und in der späteren Auswertung zu berücksichtigen. Kuckartz fasst dies in seinem Buch zur qualitativen Evaluation zusammen, indem er feststellt, dass man qualitativen Methoden „eine größere Offenheit und eine Berücksichtigung der Perspektive der Beteiligten“ [12] zuschreibt - und genau diese Perspektiven der Zielgruppe der späteren Anwendung sollen in den neuen Prototyp einfließen. Zu diesem Zweck wurde entschieden, eine qualitative Evaluation durchzuführen.
Nach der Festlegung auf eine qualitative Studie mit Experten von BMW wurden noch die Ziele der Evaluation festgelegt. Ein wesentlicher Punkt, der überprüft werden sollte, war die Frage, ob eine 3D-Ansicht allein schon als Visualisierung genügen kann oder ob die Informationen zu umfangreich sind, um sie verständlich in 3D darstellen zu können. Sollte dies der Fall sein, muss über Möglichkeiten nachgedacht werden, wie die 3D-Ansicht eine sinnvolle Ergänzung zu bereits vorhandenen Visualisierungen und Programmen darstellen kann. Weiterführend muss dann festgestellt werden, welche Informationen sich generell in 3D darstellen und visualisieren lassen und welchen Wert sie in dieser Form für den Benutzer haben. Wie bereits in Kapitel 2 gezeigt, gibt es vielfältige Möglichkeiten eine dritte Dimension zur Visualisierung von Informationen zu nutzen. Daher ist im Laufe der Gespräche auch darauf einzugehen inwiefern die gezeigten Visualisierungen eine Ergänzung zu vorhandenen Tools darstellen können oder ob es noch andere Möglichkeiten gibt, bei denen 3D einen Mehrwert bedeuten würde.
Ein wichtiger Faktor eines 3D-Modells ist auch seine Detailtreue. Aufgrund der sehr unterschiedlich ausfallenden Fahrzeug-Detailtreue der beiden zu testenden Tools (siehe Abbildung 3.1) wurde Wert darauf gelegt, bei der Studie Aussagen zu diesem Thema zu erhalten. Dabei war von beson-

derer Bedeutung wie viele Fahrzeugdetails sichtbar sein sollten, um dadurch auch möglichst viele Informationen unterbringen zu können, ohne jedoch das Modell aufgrund zu vieler Details unübersichtlich zu machen.

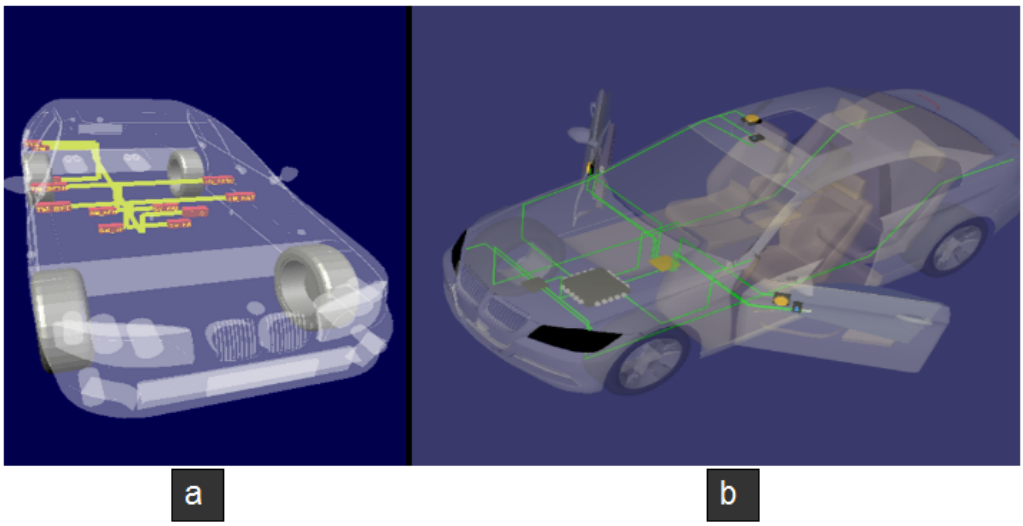


Abbildung 3.1: Vergleich der beiden Prototypen hinsichtlich des Fahrzeug-Detailgrads: (a) Prototyp von Mozdari [17], (b) Prototyp der Purdue University [20].

Im Hinblick auf ein übersichtliches Modell, das den Benutzer nicht mit einer Vielzahl überflüssiger Details belastet, ist dies ein wichtiger Faktor bei der späteren Konzepterstellung. Eine vergeichbare Fragestellung gab es im Bezug auf die Bordnetzdarstellung, da es auch hier unterschiedliche Abstufungen der Darstellungsqualität geben kann. Zum einen wäre ein einfacher Kabelbaum denkbar, der auch durchaus durch den Fahrzeuginnenraum gespannt wird, zum anderen wäre es auch möglich, den Kabelbaum im 3D-Modell sehr realitätsnah zu gestalten.
Aufgrund der vorgestellten Arbeiten zum Thema 3D und den obengenannten Überlegungen war es möglich vor der Evaluation mehrere Hypothesen aufzustellen, die durch die Ergebnisse des Leitfadeninterviews überprüft werden sollten. Die erste Hypothese besagt, dass eine 3D-Fahrzeugmodell gut geeignet ist, um mechanische Fahrzeugzustände (Tür geöffnet, Abblendlicht an, ...) auf eine einfache Art zu visualisieren und dem Betrachter schnell zu vermitteln. Die zweite Hypothese unterstellt die gute Visualisierbarkeit von Bordnetz-Daten (Auslastungen, Nachrichten\Sekunde, ...) in einem 3D-Modell im Bereich der Analyse und Diagnose. Die abschließende dritte Hypothese besagt, dass die 3D-Ansicht allein nicht in der Lage ist alle anfallenden Daten ausreichend präzise darzustellen. Erst in Kombination mit weiteren 2D-Visualisierungen im Rahmen eines sogenannten *Multi-Coordinated-View-Systems (MCV-System)* kann ein echter Mehrwert für die Analyse und Diagnose entstehen. Die Besonderheit eines MCV-Systems besteht darin, dass Daten aus einer einzigen Quelle auf unterschiedliche Weisen aufbereitet und dargestellt werden. Der Nutzer erhält somit völlig unterschiedliche Sichten auf die Daten, die in Kombination zu neuen Erkenntnissen und Einsichten führen können. Nach Abschluss der Evaluation werden diese drei Hypothesen mit den Ergebnissen verglichen und sowohl die Evaluationsergebnisse, als auch der Vergleich mit den aufgestellten Hypothesen fließen in die anschließende Konzeptentwicklung ein.

3.2 Ablauf und Ergebnisse der Evaluation

Die Leitfadeninterviews, die im Rahmen der Evaluation durchgeführt wurden, fanden bei BMW mit sechs Experten der Bereiche Analyse und Diagnose von Bordnetzdaten statt. Die angesetz-

ten Einzelgespräche dauerten jeweils etwa eine Stunde. In dieser Zeit wurden den Testpersonen zuerst eine allgemeine Frage zum Thema 3D und den Möglichkeiten von 3D-Darstellungen im Analyse-/Diagnose-Bereich gestellt. Nach diesem Einstieg wurden den Teilnehmern in jeweils wechselnder Reihenfolge, um einseitige Beeinflussungen auszuschließen, die beiden Prototypen vorgestellt, deren Funktionsweise erläutert und im Anschluss daran wurden dir Programme von den Nutzern bedient. Nach jeder Benutzung wurden spezielle Eigenschaften der Prototypen anhand eines Fragebogens besprochen.

3.2.1 Pre-Befragung

Die Gespräche starteten mit einer Frage nach der allgemeinen Einschätzung der Anwendungsmöglichkeiten von 3D im Bereich der Analyse und Diagnose. Die Testpersonen wurden dabei nach ihrer Einstellung zu diesem Themas befragt und welche positiven bzw. negativen Erwartungen sie an eine 3D-Visualisierung haben. Ein wesentlicher Punkt, der fast von allen Befragten angesprochen wurde, ist die Befürchtung, dass durch 3D-Präsentationen eher das Design in den Vordergrund rückt. Viele verbinden mit der Darstellung eines Modells im dreidimensionalen Raum eine eher auf Präsentation und Effekte zielende Darstellung. Sie sind skeptisch bzgl. einer 3D-Informationsvisualisierung. Es wurde mehrfach im Verlauf dieses Interview-Abschnitts erwähnt, dass selbst aktuelle 2D-Visualisierungen zum Teil bereits hoffnungslos überladen und somit den Ansprüchen komplexer Informationen „nicht gewachsen“ sind. In diesem Zusammenhang kamen bereits Vorschläge für eine den Bedürfnissen angepasste 3D-Gestaltung auf. Die Vorschläge reichen von Präsentationen bis hin zu Schulungen, in denen das Thema Bordnetzkommunikation bearbeitet wird. Sicherlich spielte hierbei der, von den Testpersonen erwartete, „Joy-of-View“ der 3D-Ansicht eine gewisse Rolle, denn schon zu Beginn der Gespräche wurde schnell klar, dass bis auf eine Ausnahme alle Teilnehmer dem Thema 3D-Darstellung sehr positiv gegenüberstehen. Dies wurde unter anderem deutlich durch Aussagen, man erhoffe sich durch ein 3D-Modell einen „Aha-Effekt“ und man kenne die vielen Möglichkeiten von CAD-Programmen, die zum Teil nützliche Funktionen für die Analyse und Diagnose ermöglichen könnten.
Zielgerichtet auf *Vor- oder Nachteile einer 3D-Darstellung* angesprochen, wurden die Aussagen konkreter. Nachteile wurden bei einer 3D-Visualisierung vor allem im Bereich der Kommunikationsdarstellung vermutet, da wichtige Informationen nicht adäquat dargestellt werden könnten. Die deutlichste Aussage dabei war, dass eine dreidimensionale Darstellung „Unsinn für die Entwicklung von Bordnetzen“ sei. Sie sei zu komplex, um in einem einfachen Modell sinnvoll dargestellt zu werden, ohne Kompromisse eingehen zu müssen. Jedoch wurde die Idee einer 3D-Darstellung nicht nur negativ gesehen, sondern es gab auch bereits vor der Präsentation des ersten Prototyps Äußerungen zu sinnvollen 3D-Visualisierungsmöglichkeiten anhand eines dreidimensionalen Fahrzeugmodells. Bis auf eine Ausnahme sahen alle das mechanische Umgebungsmodell, also die Fähigkeit des 3D-Modells ein exaktes Abbild des Fahrzeugs, dessen Bordnetzdaten visualisiert werden, darzustellen (z. B. Tür geöffnet, Blinker aktiv, ...), als sehr gute Möglichkeit für eine 3D-Visualisierung. Auch der Einbauort von Steuergeräten wurde von den meisten als positives Merkmal einer möglichen 3D-Sicht vermutet, auch wenn hier einige Einschränkungen gemacht wurden. Da der Einbauort nicht immer von Interesse für die Analyse sei, hängt dieser positive Faktor stark vom jeweiligen Use-Case ab, in dem ein 3D-Modell zur Visualisierung genutzt werden soll.
Ähnliches wurde auch bezüglich der *Darstellung der Bus-Systeme* gesehen. Hier wurde die Möglichkeit der realisitischen Verkabelungsdarstellung als interessantes Feature hervorgehoben, da so Problemstellen (z. B. mögliche Knickstellen, Scheuerstellen, ...) bereits im Modell erkannt werden und die Fehlersuche eventuell beschleunigt werde. Außerdem wurde die Idee eingebracht, es ließe sich so die Auslastung der Bus-Systeme gut darstellen und sei dann direkt ablesbar. Auch im Rahmen der Verifikationsphase, wenn es darum geht, ob Nachrichten rechtzeitig bei den jeweiligen Steuergeräten ankommen, wurden Chancen für eine erfolgversprechende

3D-Darstellung gesehen.
Ingesamt lässt sich die erste allgemeine Befragung für den Prototyptests zusammenfassen, indem man festhält, dass die Grundeinstellung gegenüber einer 3D-Ansicht prinzipiell positiv ist. Es gibt jedoch auch einige Befürchtungen, dass bei der 3D-Visualisierung das Design in den Vordergrund rücken könnte und der zu erwartende Nutzen ausbleibt und die eigentlichen Visualisierungsziele in 3D daher nicht erreicht werden. Dennoch waren bis auf einen Befragten alle gegenüber der Idee der 3D-Informationsvisualisierung von Bordnetzkommunikation aufgeschlossen.
Nach der allgemeinen Befragung wurden den Testpersonen die beiden Prototypen vorgestellt. Um dabei Beeinflussungen oder Verfälschungen zu verhindern, wurde die Präsentationsreihenfolge immer wieder variiert. So sollte sichergestellt werden, dass die Präsentation des ersten Prototyps nicht permanent die Bewertung des zweiten Prototyps beeinflussen kann. Im Rahmen der jeweiligen Präsentation wurde den Testpersonen zuerst die Steuerung der Prototypen erklärt, bevor sie im Anschluss daran selbstständig kleinere Aufgaben mit den jeweiligen Tools erledigen sollten. Diese Aufgaben bestanden z. B. daraus, sich einzelne Nachrichten anzeigen zu lassen oder bestimmte Auslastungszustände von Steuergeräten zu erreichen und diese auch zu erkennen. Während der Tests wurden interessante Äußerungen der Personen mitgeschrieben und einige Verhaltensweisen bei der Benutzung der Prototpyen protokolliert.

3.2.2 Evaluation des Prototyps von Mozdari

Die Evaluation des Prototyps von M. Mozdari 2.2.1 begann jeweils damit, den Testpersonen die Steuerung und grobe Funktionsweise des Programms zu erklären. Dabei ging es vor allem um die Maussteuerung in Bezug auf die Kontrolle der 3D-Kamera. Die eigentlichen Funktionen des Filtermechanismus wurden nur kurz angeschnitten, um zu sehen, inwieweit diese Art der Filtermöglichkeit von Vorteil ist in Anbetracht der großen Datenmengen. Im Anschluss an diese kurze Einweisung wurden die Probanden gebeten, sich einzelne Nachrichten, die im weiteren Verlauf durch das Bordnetz zum empfangenden Steuergerät wandern, anzeigen zu lassen und wenn möglich, deren Inhalt wiederzugeben. Bevor konkrete Visualisierungseigenschaften abgefragt wurden, hatten die Testpersonen die Möglichkeit, alle Vor- und Nachteile bzw. Probleme, die sie bis zu diesem Zeitpunkt sahen, zu nennen. Hier kamen bereits viele Anmerkungen zustande, die den späteren Fragen vorgriffen und die sich je nach Testperson auch zum Teil deutlich unterschieden. An dieser Stelle wurde deutlich, dass die jeweiligen Hintergründe bzgl. Bordnetzkommunikation und dem Einsatzgebiet für Visualisierungen einen nicht unerheblichen Einfluss auf die Sichtweise für neue Visualisierungstechniken haben. Im weiteren Verlauf des Kapitels werden die ersten Aussagen der Probanden, die auf die Fragen „Was gefällt Ihnen am Prototyp?“ und „Was gefällt Ihnen nicht?“ geantwortet wurden, aufgrund der besseren Übersicht den jeweiligen späteren Fragegebieten zugeordnet. Die Aussagen werden dann ausdrücklich als solche gekennzeichnet, die von den Probanden zu Beginn des Frageteils geäußert wurden.
Die erste Frage beschäftigte sich mit der konkreten Darstellung der Bus-Systeme. Hierzu zählen sowohl die Visualisierung der Steuergeräte als auch die der Verkabelungen der Busse. Hier ist festzuhalten, dass eine gewisse Abstraktion, aber auch der realisitische Ort der Steuergeräte wichtig ist. Die meisten Personen gaben an, dass eine vollendete Detailtreue der dargestellten Verkabelung nicht nötig sei, sondern es hier zugunsten einer besseren Übersichtlichkeit wünschenswert sei, wenn die Kabelverläufe bis zu einem gewissen Grad abstrahiert werden. Die relativ abstrakte Bus-Darstellung im Tool wurde für sehr übersichtlich und daher für gut befunden. Eventuelle Anforderungen an die realistische Kabeldarstellung wurden mit Verweisen auf den jeweiligen Use-Case kommentiert. Ein weiterer Vorteil wurde darin gesehen, dass die Busse unterschiedliche farbliche Kodierungen aufwiesen, da so eine gute Unterscheidung gewährleistet ist. Im Verlauf der allgemeinen Befragung wurde zu diesem Thema besonders oft angesprochen, dass der Verbauort der Steuergeräte zumindest in etwa zu erkennen ist und diese Information im Normalfall -

in Abhängigkeit von den Use-Cases - ausreichend ist. Ähnlich wurde bereits zu Beginn krititsiert, dass die Steuergeräte lediglich durch einfache Quader repräsentiert werden und es keine visuelle Unterscheidung der einzelnen Geräte gibt, die, da sie die Orientierung im Raum für den Nutzer vereinfacht, sicherlich sinvoll wäre. Alle in diesem Abschnitt angesprochenen Eigeschaften können in Abbildung 3.2 (b) gut erkannt werden.
Der nächste Frageabschnitt ging bereits auf ein wesentliches Feature dieser Visualisierung ein - die Anzeige und Animation einzelner Nachrichten. Wie schon bei der Darstellung der Bus-Systeme kam auch bei dieser Frage besonders stark heraus, dass die Antwort vom jeweiligen Use-Case abhängt, d.h. in welchem Zusammenhang die Visualisierung eingesetzt werden soll. Ist das Einsatzgebiet die Analyse und Diagnose von Bordnetzdaten, waren sich die Testpersonen grundsätzlich einig, dass diese Art der Nachrichtendarstellung sehr unübersichtlich sei und die Nachrichten häufig auch nur schwer zu erfassen seien. Als ein Hauptproblem wurde meist die Darstellung der Textinformationen direkt im 3D-Raum auf den Texturen der Objekte genannt. Diese Information deckt sich mit den Feststellungen von Larson et al. [14]. Die Darstellung von Nachrichten wurde jedoch nicht grundsätzlich abgelehnt, wenn z. B. die Einfärbung spezieller Nachrichten möglich wäre, um so den Fokus des Benutzers auf bestimmte Nachrichtentypen lenken zu können. Aber selbst bei dieser Funktionsweise wurde gezweifelt, ob nicht eine 2D-Darstellung dieselben, wenn nicht eher sogar mehr Vorteile bei der Aufbereitung der Informationen biete. Diese Vermutungen wurden bereits zu Beginn der Befragung geäußert. Dort wurden besonders die Geschwindigkeit der Nachrichten und die Probleme mit dem 3D-Text als Faktoren genannt, die eine effektive Nutzung dieser Visualisierung verhindern. In Abbildung 2.3 erkennt man die Probleme, die der 3D-Text mit sich bringt und kann erahnen welche Verdeckungen es mit mehreren Nachrichten gleichzeitig gibt. Außerdem wurde zu Beginn schon darauf verwiesen, dass die gewählte Metapher „Nachrichten gehen aus dem Steuergerät rein und raus“ eher fraglich ist, da sie bei einigen Situationen nicht zutreffend sei.

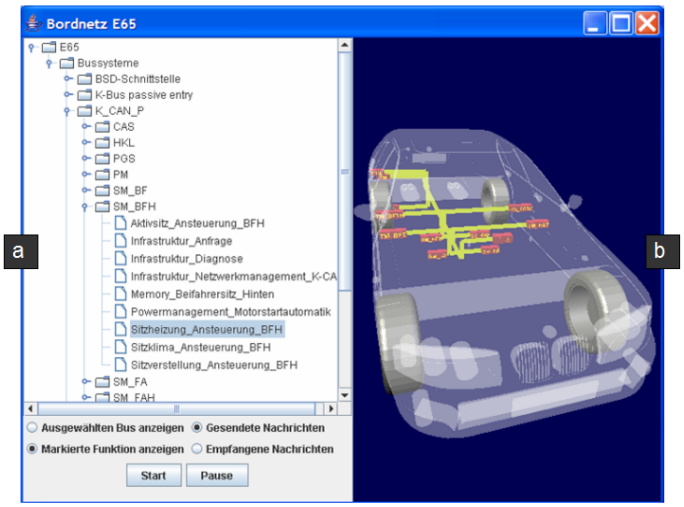


Abbildung 3.2: (a) Filteroptionen auf Grundlage der Bordnetzdatenbank, (b) 3D-Visualisierung des Nachrichtenflusses [17].

Es wurde weiterhin darauf verwiesen, dass die Zusammenhänge beim Nachrichtenfluss nicht immer zu erkennen seien. Eine Frage, die von einem Probanden gestellt wurde, war dabei: „Und woher weiß ich jetzt, ob die zweite Nachricht einfach so oder aufgrund der ersten Nachricht verschickt wurde?“. Es wird deutlich, dass komplexe Nachrichtenverläufe nicht für eine solche Darstellung geeignet scheinen - dies wurde von einem Probanden mit den Worten kommentiert,

man mache mit so einer Darstellung „mehr kaputt als gut“. Dies zeigt, dass es einen unbedingten Bedarf gibt, die Aktivität innerhalb der Steuergeräte zu visualisieren, um das Verständnis der Kommunikatioszusammenhänge zu unterstützen.
Auf die dritte Frage inwieweit die gewählte Visualisierung der Bordnetz-Topologie das Verständnis für den prinzipiellen Aufbau und die Struktur der Netze innerhalb des Fahrzeugs unterstützt, gab es unterschiedliche Rückmeldungen. Während die einen Probanden völlig mit der abstrakten Darstellung zufrieden waren, wollten andere lieber einen realistischere Verlauf der Kabelbäume im Modell. Eine Anmerkung an dieser Stelle war, dass die korrekte Darstellungsform der Bus-Topologie wichtig sei, da es im Fall eines Kabelbruchs natürlich einen Unterschied macht, ob es sich beim betroffenen Bus um z. B. eine Stern- oder eine Ringtopologie handelt. In diesem Zusammenhang wurde dann eine gewisse Abstraktion der Busse toleriert, aber insgesamt war den meisten Teilnehmern diese Art der Darstellung zu schematisch und nicht genau genug.
Eine weitere Frage, die gestellt wurde, war bereits von einigen Probanden während der Erklärung des Prototyps angesprochen worden und zielt auf den Detailgrad des Modells ab. Wie schon bei den Fragen zuvor kristallisierte sich auch hier heraus, dass die Antwort auf die Frage stark vom jeweiligen Einsatzgebiet des Tools abhängt. Von allen Teilnehmern wurde ein Trade-Off zwischen vielen Details und einer guten Übersicht erkannt. Dabei tendierten anschließend die einen zu der Feststellung, dass die gewählte Detailstufe gut gewählt sei, da auch das Bordnetz eher abstrakt dargestellt sei. Somit sei eine maximale Übersicht gewährleistet. Der andere Teil der Befragten war jedoch der Meinung das Modell biete zu wenige Informationen, da - wiederum abhängig vom Use-Case - auch mehr Informationen über das Modell wichtig sein könnten.

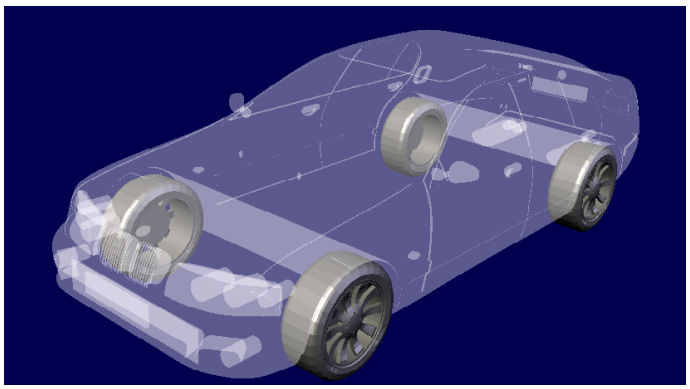


Abbildung 3.3: Je nach User-Case kann auch ein niedriger Detailgrad geeignet sein [17].

Hier wurde auch die fehlende Darstellung von mechanischen Reaktionen im Fahrzeug angesprochen und kritisiert. Zwar wurde im Allgemeinen schon zu Beginn der Tool-Vorstellung die gute Übersicht des Modells gelobt, aber wie die konkrete Nachfrage später zeigte, hat sich die Einschätzung der Personen zum Teil erheblich geändert, nachdem ein wenig mit dem Programm gearbeitet werden konnte und es bestand der Wunsch nach mehr Details.
Die letzte Frage ging auf ein herausragendes Merkmal dieses Prototyps ein. Dies ist zweifelsohne die Filterungsmöglichkeit (Abbildung 3.2 (a)) wegen der vollständig genutzten Struktur der Bordnetzdatenbank. Zu diesem Zweck wird auf der linken Seite des Programmfensters eine Baumstruktur des Bordnetzes dargestellt und man hat darüber die Möglichkeit, sich Elemente ein oder ausblenden zu lassen. Auf Nachfrage wurde die eigentliche Funktion dieses Filterungsmechanismus durchweg positiv aufgefasst. Besonders die Baumstruktur und die Topologiedarstellung wurden gelobt. Zum Teil war sogar die Rede davon, dass man eher noch mehr Filterungsoptionen benötigen würde, da die Bordnetzdatenbank entsprechend umfangreich sei. Es gab jedoch auch einige Kritikpunkte, die angesprochen wurden. Diese beschäftigten sich jedoch eher mit Fein-

heiten der Umsetzung anstatt mit dem eigentlichen Filterungskonzept. So wurde vorgeschlagen eine erweiterte farbliche Kodierung im Menü und dem Modell anzubieten, um die Verbindung zwischen beidem noch hervorzuheben. Ein anderer Vorschlag ging in die Richtung, man könne andere Symbole für die Einträge im Menü-Baum verwenden, um z. B. eine einfache Unterscheidung zwischen Steuergerät, Gateway und komplettem Bus-System zu ermöglichen. Als weitere Ergänzung wurden auch Hover-Informationen über den Menü-Einträgen vorgeschlagen, um auf diese Weise weitere Informationen im Programm unterzubringen.
Nach Abschluss der Befragung war noch Zeit für *sonstige Äußerungen und Kommentare*, die bis dahin durch keine Frage abgedeckt waren. Wie erwartet kamen in diesem Teil nicht viele Rückmeldungen, aber dennoch waren einige interessante Stellungnahmen dabei. So wurde der Filter von einigen Personen noch einmal explizit gelobt und auch die Steuerung wurde als relativ problemlos empfunden, auch wenn sie anfänglich Probleme bereitet haben sollte. Im Zusammenhang damit wurde der Vorschlag gemacht, standardmäßige Kamerapositionen anzubieten, d.h. auf Knopfdruck z. B. in eine Frontperspektive zu wechseln und diese im Anschluss noch konfigurieren zu können. Als allgemeiner Kritikpunkt wurde auch nochmal angesprochen, dass die Art der Visualisierung bei zu vielen gleichzeitigen Nachrichten nicht funktionieren kann, da dann einfach eine zu große Zahl an Informationen gleichzeitig dargestellt werden müsse. Während an dieser Stelle zu viele Informationen dargestellt werden, fehlte manchen Testpersonen die Darstellung von kumulativen Daten, wie z. B. der Auslastung der Steuergeräte, Anzahl der empfangenen Nachrichten und ähnlichen Werten. Aber hier wurde auch immer mit der Einschränkung gearbeitet, dass solche Forderungen immer vom aktuellen Use-Case abhängig sind und dabei wurden dieser Visualisierung vor allem Vorteile für Schulungen und als Lernprogramm zur Einarbeitung in die Bordnetz-Kommunikation bescheinigt. Damit wurden zum Teil die Aussagen, die vor der Benutzung gemacht wurden („...könnte gut zur Trace-Analyse eingesetzt werden...“) entkräftet bzw. abgewandelt.
Insgesamt wurde diesem Visualisierungsansatz verschiedenen Möglichkeiten attestiert einen Mehrwert im Vergleich zu vorhandenen Visualisierungen zu bieten. Allerdings wurde im Verlauf der Befragungen recht deutlich, dass die Nachrichtenvisualisierungen in Form von Objekten, die sich durch das Modell bewegen, nicht im Bereich der Analyse oder Diagnose nutzbar ist. Es treten zum einen zu viele Probleme auf bei der Lesbarkeit von Text im 3D-Raum, zum anderen würde die Visualisierung zu unübersichtlich werden unter realistischen Analyse-Bedingungen mit teilweise 14.000 Nachrichten pro Sekunde. Zwar wurde vorgeschlagen, erweiterte Filterungen zu ermöglichen, doch selbst dann wurde in Frage gestellt, ob diese Visualisierung genügend Übersicht und einen Vorteil gegenüber bisherigen Darstellungsformen bietet. Wenn man jedoch den Ideen der Testpersonen folgt, lassen sich viele Use-Cases finden, in denen diese Form der Visualisierung von Bordnetzkommunikation sehr passend eingesetzt werden könnte. So könnte das Tool durch die Darstellung einzelner Nachrichten gut dazu genutzt werden, das Verständnis für die Zusammenhänge der Kommunikationsvorgänge im Bordnetz zu fördern.

3.2.3 Evaluation von des Prototyps der Purdue University

Abgesehen von speziellen Fragen, die auf die Besonderheiten dieses Prototyps eingehen, unterschieden sich die Fragestellungen und die Vorgehensweise im Vergleich zu Kapitel 3.2.2 kaum. So wurde bei diesem Prototyp nicht nach der 3D-Textdarstellung gefragt, sondern nach den entsprechenden Herausstellungsmerkmalen dieses Projekts. Wie bereits im Abschnitt 3.2.2 werden bei dieser Auswertung die einzelnen Fragen durchgearbeitet und allgemeine Äußerungen, die vor den speziellen Fragen gemacht wurden, den entsprechendne Bereichen zugeordnet und gekennzeichnet. Doch erfolgte auch bei dieser Evaluierung zu Beginn eine kurze Einweisung in die generelle Funktionsweise, während anschließend einfache Aufgaben gelöst werden sollten, wie z. B. das Erzeugen einer Lastsituation im virtuellen Bordnetz mit anschließender Erklärung, woran die Probanden diese erkannt haben.

Die Einstiegsfrage bei diesem Prototyp ging auf die kontinuierliche Darstellung des Fahrzeugstatus - also das mechanische Umgebungsmodell - ein. Dabei waren sich alle Befragten einig, dass dies eine nützliche Visualisierung sein kann, wenn der Gesamtfahrzeugstatus eine Rolle spielt. Auch hier ist also gleich zu Beginn wieder klar geworden, dass es keine allgemeingültige Aussage gibt, ob einen Visualisierung gut oder schlecht ist, sondern immer nur in Bezug auf ein bestimmtes Einsatzgebiet. Davon abgesehen wurde hier die Möglichkeit zur Statuserkennung am Fahrzeug durchweg positiv aufgenommen. Es gab jedoch einige Verbesserungsvorschläge, wie zum Beispiel die gezielte Hervorhebung von Aktionen, denn gerade das Herauf- und Herunterfahren der Fenster im Modell wurde aufgrund der Transparenz häufig übersehen.

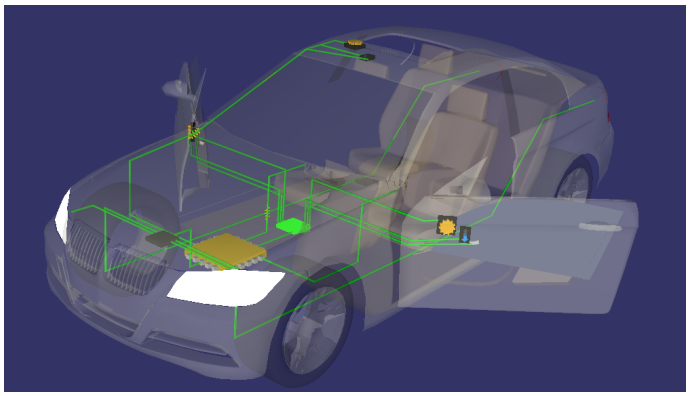


Abbildung 3.4: Anzeige der Auslastung im 3D-Modell; Aktivität einzelner Bauteile kann je nach Blickwinkel übersehen werden [20].

In diesem Zusammenhang wurde auch angesprochen, die 3D-Darstellung entweder durch eine 2D-Explosionszeichnug des Modells oder durch entsprechende Detailansichten zu ergänzen, um auch kleine Veränderungen am Fahrzeugstatus deutlich werden zu lassen.
Die nächste Frage ging auf die dargestellte Auslastung der Steuergeräte und die Aktivität der Busse ein. Im Purdue Tool werden den Steuergeräten je nach Nachrichtenanzahl gewisse Auslastungsgrade zugewiesen. Sie werden entsprechend eingefärbt und bei den Bus-Systemen wird die Aktivität ebenfalls durch Einfärbung visualisiert (siehe Abbildung 3.4). Auf die Frage, ob Ihnen diese Art der Informationsvisualisierung einen Mehrwert bringen könnte antworteten die meisten durchaus positiv. Die gewählte farbliche Kodierung wurde durchweg sofort verstanden und auch als leicht zu erkennen bewertet. Es wurde jedoch auch auf mehrere Probleme in diesem Zusammenhang aufmerksam gemacht. So existieren für Steuergeräte nicht immer Metriken, die angeben wann dieses Bauteil z. B. zu 100% ausgelastet ist oder wann es überlastet ist. Das heißt also, die Visualisierung selbst ist eine gute Idee, jedoch liegen die erforderlichen Daten dazu längst nicht immer vor. Dies ist allerdings eher ein Problem bei der späteren Umsetzung und sollte in dieser frühen Konzeptphase der Visualisierung nicht allzu kritisch bewertet werden. In diesem Zusammenhang gab es aber auch mehrfach den Wunsch, die Bus-Auslastung zu visualisieren, da sie zum Teil ein wichtiger Faktor bei der Analyse/Diagnose ist und über farbliche Kodierung, ebenso wie bei den Steuergeräten, einfach und verständlich umzusetzen ist.
Abgesehen von der direkten Darstellung im 3D-Modell wurden von den Testern auch die 2D-Anzeigen (siehe Abbildung 3.5) im oberen rechten Bereich des Fensters positiv hervorgehoben, auch wenn deren Positionierung laut einiger Aussagen verbesserungswürdig wäre. Trotz der durchweg positiven Resonanz gibt es auch hier einen wesentlichen Kritikpunkt, nämlich die mangelnde Echtzeitfähigkeit der Visualisierung. Zwar kann man eine entsprechende Umfärbung der Bauteile in Echtzeit vornehmen, allerdings werden diese Veränderungen vom menschlichen Auge kaum bis gar nicht wahrgenommen und würden somit verloren gehen.

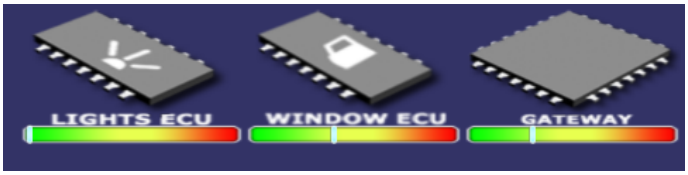


Abbildung 3.5: Anzeige der Auslastung in 2D-Elementen [20].

Wie schon in 2.2.1 wurde auch beim Programm der Purdue University eine *Frage zur Darstellung der Bordnetz-Topologie* gestellt. Der größte Unterschied zwischen beiden Prototypen ist, dass bei diesem Tool die Darstellung deutlich realitätsnäher ist, da für Steuergeräte und Gateways unterschiedliche Symbole verwendet werden und auch die Verkabelung im Modell deutlich näher an der Realität ist (siehe Abbildung 3.4). Dies wurde von den meisten Testpersonen so kommentiert, aber auch hier wurden die meisten Bemerkungen wiederum abhängig vom Use-Case der Anwendung gemacht.
Unterschiede wurden besonders bei der *Frage zur Visualisierung von Informationen der Kommunikation* (Auslastungen, Nachrichteninhalte, ...) gemacht. Für diese eignet sich laut der Probanden eher eine abstrakte Darstellung, während die realitätsnahe Darstellung der Kabelverläufe durchaus gut für Aufgaben geeignet ist, bei denen Kabellängen und eventuelle Bruchstellen an Kabeln eine wichtige Rolle spielen. Es wurde jedoch auch angemerkt, dass die entsprechenden Informationen zu den Verbauorten und Kabelverläufen aktuell sein müssen, da der Informationsgehalt der Darstellung ansonsten eher gering sei. Hierzu wurde von zwei Teilnehmern bereits vorweg bemerkt, dass eine zu realisitische Darstellung problematisch werden kann, wenn keine ausreichenden Filterungsmöglichkeiten angeboten werden. Hier wurde auf die Redewendung des „selbsttragenden Kabelbaums“ verwiesen. Dieses Zitat spielt auf die immensen Mengen an Kabeln in einem heutigen Kraftfahrzeug an. Werden diese Kabel alle realistisch dargestellt, kann die Visualisierung schnell unübersichtlich werden.

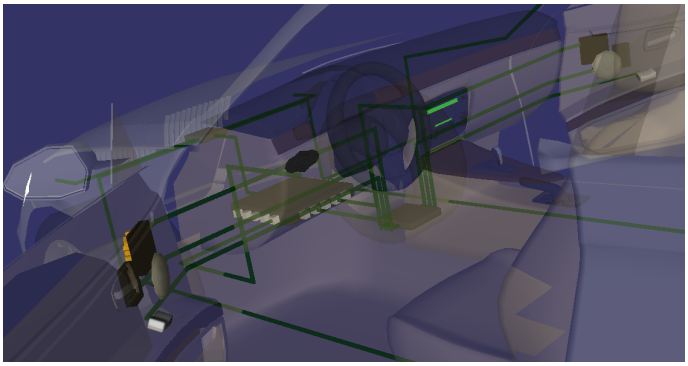


Abbildung 3.6: Der Prototyp bietet, laut Aussage der Probanden, einen guten Detailgrad, der auch noch gesteigert werden könnte [20].

Die *letzte Frage* dieser Evaluation wurde bzgl. des Modell-Detailgrads gestellt. Die einheitliche Meinung war dabei, dass der gewählte Detailgrad, wie in Abbildung 3.6, vollkommen ausreichend sei und auch durch weniger Details keinesfalls die Orientierung im Modell verloren gehen würde. Es wurden dann noch gezielt verschiedene Vorschläge zu Verfeinerung und der Abstufung von Detailstufen gemacht, z. B. dass bei einer erhöhten Zoomstufe mehr Details dargestellt werden könnten als bei einer Gesamtübersicht über das Modell. So hätte man auf Wunsch auch kleine Details im Blick, aber diese würden den Gesamtüberblick nicht unübersichtlich gestalten. Als

wichtiges Feature stellte sich an dieser Stelle die Transparenz von Komponenten heraus, wegen der es auch möglich ist, eigentlich verdeckte, Bauteile zu erkennen. Wie schon bei der Auslastung wurde auch für bestimmte mechanische Aktionen eine Auslagerung auf 2D-Elemente vorgeschlagen. Als Beispiel wurden hier die Instrumente im Cockpit genannt, welche auf kleinem Raum verschiedenste Zustände des Fahrzeugs visualisieren. Überträgt man diese Informationen auf 2D-Anzeigen innerhalb des 3D-Fensters, wären die Statusänderungen somit nicht bloß im 3D-Modell, sondern auch in den 2D-Anzeigen zu erkennen. Die Gefahr, eventuell wichtige Statusänderungen zu übersehen, kann so verringert werden. In diesem Zusammenhang wurde auch am Anfang der Befragung schon erwähnt, dass eine steigende Detaildichte zwar mehr mechanische Reaktionsdarstellungen ermöglicht, aber auch mit einer zunehmenden Verdeckungsgefahr von Veränderungen einhergeht. Wie schon mehrfach an anderer Stelle erwähnt, gibt es auch hier einen Trade-Off zwischen vielen Details und einer guten Übersicht.
Bei den *abschließenden Bemerkungen* zu diesem Prototyp hob sich vor allem der Wunsch nach einem Zusammenspiel von der 3D-Ansicht mit anderen Sichten auf die zu analysierenden Daten hervor. Mit einer solchen Zusammenarbeit wäre auch das Problem beseitigt, dass diese 3D-Darstellung keine weiteren Inhalte der Nachrichten anzeigt - denn genau dies wurde in der abschließenden Besprechung mehrfach kritisiert. Allerdings wurde dabei auch bemerkt, dass es in einer 3D-Umgebung schwierig sei, eine solche Fülle an Details adäquat darzustellen. Es wurde auch ein Vorschlag zur Erweiterung des 3D-Konzepts gemacht. Dabei wurde der Wunsch geäußert, über die Auswahl eines Fahrzeugbauteils alle mit diesem Teil verbundenen Bordnetzkomponenten angezeigt zu bekommen. Als Beispiel wurde dabei eine Fensterscheibe genannt, bei deren Selektion anschließend alle Steuergeräte hervorgehoben werden sollten, die mit den Aktionen dieser Scheibe in Verbindung gebracht werden können. Das 3D-Modell könnte an dieser Stelle also als eine Art Selektionsinterface für die Darstellung von Wirk-Ketten fungieren.

3.2.4 Abschließende Bemerkungen zu den Prototypen

Nach Abschluss der beiden Evaluationen wurden die Probanden noch gezielt nach ihrer Einschätzung bzgl. der Verwendung einer 3D-Ansicht in einem MCV-System gefragt bzw. ob sie sich eine - den gezeigten Prototypen ähnliche - Ansicht vorstellen könnten als sinnvolle Ergänzung zu bereis existierenden Programmen. Wie schon einige Male zuvor wurde auch hier auf die zum Teil völlig unterschiedlichen Use-Cases verwiesen, die eine allgemeingültige Aussage nicht möglich machen. Als Bereiche, sprich Use-Cases, in denen eine Verwendung für sinnvoll gehalten wurde, galten vor allem das Änderungsmanagement von Bordnetzkomponenten, Werkstätten und die Schulung von Personal im Bereich der Bordnetzkommunikation, denn „da wäre so eine Ansicht der Bringer“, so einer der Befragten.
Insgesamt hat die Evaluation der vorhandenen Prototypen einige aufschlussreiche Ergebnisse gebracht. Zum einen wurde sehr deutlich, dass der Nutzen von 3D-Visualisierungen stark vom jeweiligen Use-Case abhängt und es selbst dort keine eindeutig beste Visualisierung für alle Möglichkeiten gibt. Beide getesteten Prototypen wurden von den Probanden als gute Möglichkeit gesehen, in unterschiedlichen Bereichen einen Mehrwert zu vorhandenen Programmen zu bringen. Den Prototyp von M. Mozdari haben viele Testpersonen vor allem im Bereich der Präsentation und Ausbildung gesehen, aber vermuteten Schwierigkeiten, wenn auf diesem Wege aufgezeichnete Daten visualisiert werden sollen. Das Tool der Purdue University hingegen scheint sich besser für eben solche Visualisierungen zu eignen, doch auch hier gab es einige Verbesserungsvorschläge.
Die anfänglich vermutete positive Grundstimmung gegenüber einer Visualisierung in 3D hat sich durchaus bestätigt, auch wenn sich das Themengebiet als schwieriger und vielfältiger herausgestellt hat als zunächst angenommen. Andererseits haben die Gespräche auch das große Potential gezeigt, das in einer 3D-Visualisierung von Bordnetzkommunikation steckt. Wie mehrfach von den Bordnetz-Experten gesagt wurde, kann eine 3D-Ansicht eine sehr sinnvolle und nützliche Er-

weiterung zu bisherigen Visualisierungen sein.
Nach dieser Studie konnte das Ziel der Arbeit etwas konkreter formuliert werden. Wie sich mehrfach während der Gespräche gezeigt hat, wird eine Nachrichtenvisualisierung im 3D-Raum nicht richtig funktionieren - das 3D-Modell eignet sich jedoch sehr gut für die Visualisierung unterschiedlicher Fahrzeugstatus.
Der spätere Prototyp soll also ein möglichst exaktes mechanisches Umgebungsmodell des Fahrzeugs abbilden, aber dennoch auch die Möglichkeit von zusätzlichen 2D-Elementen bieten, um die 3D-Ansicht mit wichtigen Werten anreichern zu können. Dennoch hat sich während der Evaluation auch gezeigt, dass selbst mit solchen 2D-Anzeigen eine 3D-Ansicht allein nicht ausreichend sein kann, um das komplexe Feld der Bordnetzkommunikation entsprechend visualisieren zu können. Aus diesem Grund soll der spätere Prototyp mit anderen Anwendungen, die es bereits gibt oder auch mit anderen Prototypen zusammenarbeiten können, um als 3D-Visualisierung in einem MCV-System neue Möglichkeiten zu eröffnen. Um dies zu ermöglichen, soll die 3D-Ansicht über eine einfach zu erweiternde Schnittstelle ansprechbar sein. Mit einer solchen Schnittstelle wäre es problemlos möglich den Fahrzeugstatus stets aktuell zu halten.
Dabei ist es, wie die Ergebnisse vom Tool von M. Mozdari gezeigt haben, sehr wichtig, eine ausreichend große Filtermöglichkeit zu bieten und den Nutzer - wenn er dies möchte - bei der Konfiguration mehrerer Ansichten zu unterstützen. Denn abgesehen von einer Verwendung in einem MCV-System, soll diese 3D-Visualisierung selbst auch ein MCV-System bieten, d.h. unterschiedliche Ansichten auf denselben Datensatz. Dabei wird jedesmal das 3D-Fahrzeugmodell zu sehen sein. Aber jede Ansicht wird sich unabhängig voneinander konfigurieren lassen, während jedoch alle immer den aktuellen und korrekten Fahrzeugzustand repräsentieren. Aus dieser Forderung ergibt sich noch ein weiteres Problem - wenn stets der aktuelle Status zu sehen ist, hat der Nutzer keine Möglichkeit die Daten kurz vor dem aktuellen Zeitpunkt einzusehen. Es ist also zu überlegen, in welcher Form eine History-Funktion, d.h. eine Sichtbarmachung von Werten und Ereignissen der vergangenen Sekunden zu realisieren wäre.
Anhand dieser Ergebnisse und Erfahrungen ist es möglich die zuvor in Kapitel 3.1 aufgestellten Hypothesen zu überprüfen. Die *erste Hypothese*, dass der Fahrzeugstatus in einem 3D-Modell gut zu visualisieren sei, wurde durch die Ergebnisse bestätigt. Es wurde zwar deutlich, dass kleine Statusänderungen spezielle hervorgehoben werden müssen, um nicht übersehen zu werden, aber dennoch haben die Tests deutlich gezeigt, dass eine solche 3D-Visualisierung eine gute Möglichkeit ist den mechanischen Fahrzeugzustand darzustellen.
Im Gegensatz dazu ist die *zweite Hypothese* nicht vollständig bestätigt worden. Zwar wurde der erkennbare Zusammenhang zwischen Bordnetzkommunikation und mechanischem Modell gelobt, allerdings wurde bei beiden Visualisierungsarten deutlich, dass sie die gewählten 3D-Visualisierungstechniken Probleme bei der adäquaten Darstellung der Bordnetzkommunikation für den Bereich Analyse und Diagnose haben. Dies zeigt sich zum einen in der mangelnden Echtzeitfähigkeit beider Visualisierungen, aber auch darin, dass wichtige Ereignisse (Überlastung, ...) sehr einfach übersehen werden können. Daher kann die zweite Hypothese nur als teilweise bestätigt gelten und dient bei der weiteren Arbeit als Hinweis, dass in diesem Bereich verstärkt geforscht werden muss.
Die *dritte und letzte Hypothese*, eine 3D-Ansicht allein könne keine vollständige Visualisierung von Bordnetzkommunikation bieten, kann durch die erhaltenen Daten als bestätigt gesehen werden. Dazu trägt bei, dass die Bordnetzkommunikation im 3D-Modell nach Aussage der Testpersonen noch zu viele Probleme bereitet, seien es fehlende Nachrichtendetails, mangelnde Echtzeitfähigkeit oder Unübersichtlichkeit.

4 Konzeptentwicklung

Dieses Kapitel beschäftigt sich mit der Entwicklung und Bewertung von unterschiedlichen Konzepten sowohl zur Informationsvisualisierung von Bordnetzkommunikation in einem 3D-Modell, als auch mit der Darstellung mechanischer Reaktionen und der Steuerung der 3D-Umgebung. In diese Entwicklung sind verschiedene wissenschaftliche Veröffentlichungen eingeflossen, wie selbstverständlich auch die Ergebnisse der gemachten Studie aus Abschnitt 3. Die dort gewonnenen Erfahrungswerte in Bezug auf eine solche Visualisierung waren sehr hilfreich bei der Entwicklung unterschiedlicher Konzepte. Die im Verlauf dieses Kapitels vorgestellte Konzeptsammlung deckt unterschiedliche Bereiche einer solchen Visualisierung ab. Sie wurde nach Fertigstellung erneut einigen BMW Experten zur Bewertung vorgelegt. Anhand dieser Bewertungen wurde anschließend das endgültige Gesamtkonzept entwickelt (Kapitel 4.4), dessen Vorstellung den Abschluss dieses Kapitels bildet.

4.1 View-Konzepte

Zu Beginn der Konzeptvorstellung bei den jeweiligen Bewertungsgesprächen wurden drei verschiedene View-Konzepte vorgestellt. Diese stellen unterschiedliche Ansichten bzw. Ansichtsverwaltungen dar und bilden prinzipiell die Grundlage des gesamten Programmentwurfs.
Das *erste View-Konzept* ist im wesentlichen eine einzige 3D-Ansicht. Diese Sicht ist freikonfigurierbar, was die Kameraparameter und die Fahrzeugdartellung angeht. Darüber hinaus bietet sie keine Möglichkeiten, die über eine einzelne Ansicht hinausgehen würden - ist also mit den zwei getesten Prototypen vergleichbar (siehe Abbildung 4.1 (a)).
Das *zweite View-Konzept* ist hingegen ein wenig flexibler, was die Gestaltung der 3D-Ansicht angeht. Das Konzept sieht vor, dass alle Bauteile des Fahrzeugs ausgewählt und vom Modell unabhängig dargestellt werden können. Diese Möglichkeit umschließt sowohl mechanische als auch Bordnetz-Komponenten und ermöglicht somit die gezielte Selektion und hervorgehobene Darstellung von Bauteilen, um das Problem der Verdeckung von Komponenten zu umgehen (siehe Abbildung 4.1 (b)). Die auf diese Weise einzeln dargestellten Teile können frei benannt, gedreht und gezoomt werden, um den für den Nutzer besten Blick zu ermöglichen. Desweiteren sieht das Konzept vor, dass 2D-Fenster, ähnlich wie Messgeräte, an die einzelnen Bauteile angeschlossen werden können und im weiteren Verlauf dann die entsprechenden Werte visualisieren (z. B. die Fahrzeuggeschwindigkeit als Tacho, die Auslastung eines Busses als Wertegraph, ...).

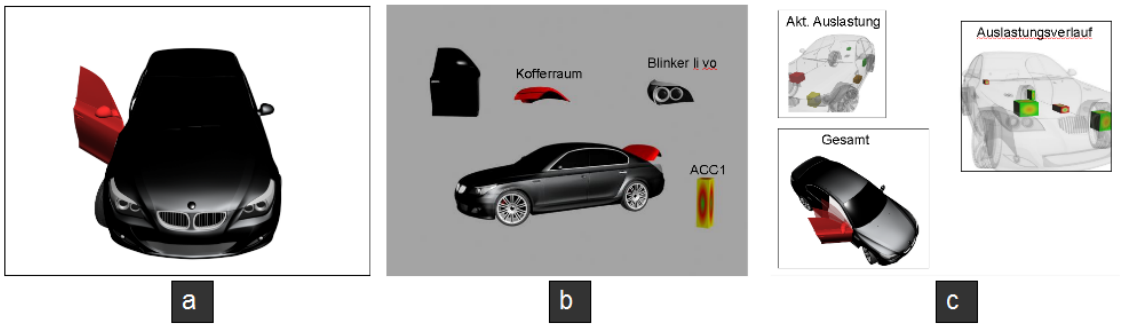


Abbildung 4.1: (a) View-Konzept 1, (b) View-Konzept 2, (c) View-Konzept 3

Das *dritte View-Konzept* ist an ein MCV-System angelehnt. Im Gegensatz zum zweiten Konzept in dem einzelne Bauteile in einer gemeinsamen Ansicht aus dem Modell herausgezogen werden konnten, können nun mehrere Ansichten generiert werden. Diese Ansichten entsprechen weitestgehend einer einzelnen Ansicht aus dem ersten View-Konzept. Darüber hinaus sind sie frei be-

nennbar und unabhängig von einander konfigurierbar. Während also zum Beispiel bei der einen Ansicht alle mechanischen Komponenten ausgeblendet wurden, um einen gute Blick auf das Bordnetz zu erhalten, kann eine andere Sicht so konfiguriert sein, dass man lediglich gewisse mechanische Komponenten sieht. Durch diese Konfigurationsmöglichkeiten sind ähnliche Ansichten wie mit dem zweiten Konzept möglich, aber auch weitaus komplexere Darstellungen, wenn man die Möglichkeiten betrachtet, die eine parallele Darstellung von mehreren 3D-Fahrzeugen bietet, wie sie in Abbildung 4.1 (c) angedeutet wird.

4.2 Visualisierungs-Konzepte

In diesem Kapitel werden die verschiedenen Konzepte zur Visualisierung von Informationen im 3D-Modell behandelt. Dabei wird zwischen speziellen Informationstypen unterschieden, um jeweils eine optimale Darstellungsform zu finden. In den folgenden Unterkapiteln werden jeweils zuerst die aufgestellten Konzepte präsentiert und die Ideen dahinter mit einigen Entwürfen verdeutlicht.

4.2.1 Auslastung

Die Visualisierung der Auslastung von Bordnetzkomponenten muss, da Textinformationen im 3D-Raum nur schwer zu platzieren und zu lesen sind (siehe Kapitel 2.5), auf eine kodierte Art und Weise geschehen. In diesem Zusammenhang bieten sich unterschiedliche Arten der Kodierung an. Denkbar ist zum Beispiel eine farbliche Kodierung (siehe Abbildung 4.2). Wie bereits erwähnt wurde, hat eine einfache farbliche Kodierung, wie bei dem Prototyp der Purdue University (Kapitel 2.2.2), jedoch den Nachteil, dass bei einer Echtzeitdarstellung der Auslastungszustände einige Zustände zwar visualisiert, aber aufgrund der Trägheit des menschlichen Auges nicht wahrgenommen werden. Um diesem Umstand Rechnug zu tragen wurde das Konzept der *radialen Farbverläufe* entwickelt. Hierbei soll ermöglicht werden, eine History, also den Verlauf eines Werts innerhalb der letzten Sekunden, direkt auf einem 3D-Modell darzustellen.

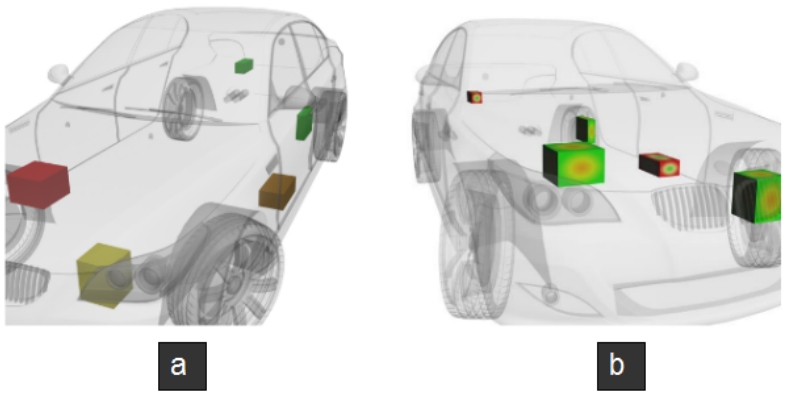


Abbildung 4.2: (a) einfache farbliche Kodierung, (b) radialer Farbverlauf mit Anzeige der Auslastung der letzten Sekunden

So soll verhindert werden, dass kurzzeitige Abweichungen vom Normwert in der 3D-Ansicht übersehen werden. Dazu wird auf den Seiten eines Steuergeräte-Modells eine Textur dargestellt, die im Mittelpunkt den aktuellen Auslastungszustand mit einem Punkt farblich kodiert darstellt. Im nächsten Moment hat das Steuergerät einen neuen Zustand erreicht und der zuletzt aktuelle Zustand bzw. dessen symbolisierende Farbe ist nun ein Stück vom Mittelpunkt nach außen gerückt

und umschließt diesen kreisförmig. Je länger ein Zeitpunkt zurückliegt, desto weiter ist die farbliche Kodierung vom Textur-Mittelpunkt entfernt. Die Zustandsanzeige eines Augenblicks wandert also vom Zentrum (*jetzt*) nach außen bis der Zustand schließlich nicht mehr angezeigt wird (z. B. nach 5 Sekunden). Abbildung 4.2 (b) zeigt ein Beispiel dieser Visualisierungstechnik und stellt Steuergeräte mit unterschiedlichen Zuständen und Farbverläufen dar. Auf diese Weise würde es gelingen den Verlauf der Steuergeräte-Auslastung oder eines ähnlichen Parameters innerhalb der 3D-Ansicht zu visualisieren, ohne auf Probleme bei der 3D-Text-Darstellung zu treffen.
Bei einem anderen Konzept, das dem singulären Einfärben sehr ähnlich ist, arbeitet man mit der *Transparenz von Objekten*. Jedoch liegt hier der Fokus besonders auf einer gesteigerten Übersichtlichkeit im Vergleich zu einer Einfärbung der Objekte. Die Idee dabei ist es, Steuergeräte und ganze Busse je nach Auslastung transparent zu gestalten, wie in Abbildung 4.3. Ist eine Komponente im Moment aktiv, so wird sie undurchsichtig gerendert. Bei länger ausbleibender Aktivität wird diese Komponente dann mehr und mehr transparent und wird schließlich gänzlich durchsichtig oder behält eine Restsichtbarkeit von z. B. 10%.

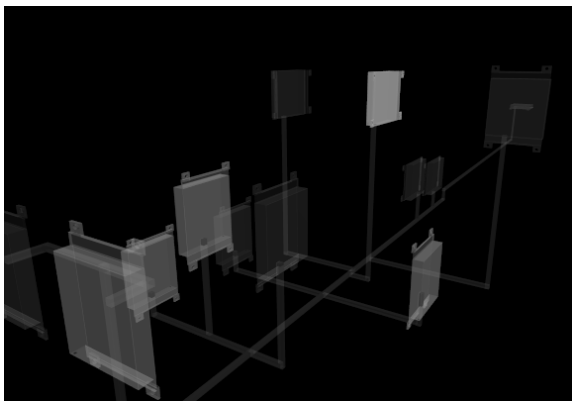


Abbildung 4.3: Steuergeräte mit unterschiedlicher Auslastung und einer Transparenz-Kodierung der Auslastung.

Wird bei der Ausblendung der Objekte eine gewisse Trägheit implementiert, d.h. wird der aktuelle Zustand nicht sofort in die Transparenz umgesetzt sondern mit einer leichten Verzögerung, könnte damit auch, ähnlich wie beim radialen Farbverlauf, für eine bestimmte Zeitspanne eine History-Funktion erstellt werden. Problematisch ist an diesem Ansatz jedoch, dass durch diese Verzögerung einzelne Spitzen in Werten nicht sichtbar würden, es sei denn diese Fälle würden abgefangen und durch eine besondere Visualisierung hervorgehoben. Insgesamt würden bei diesem Konzept die aktiven Komponenten automatisch hervorgehoben werden, da alle inaktiven Komponenten nahezu vollständig transparent wären.
Eine weitere Möglichkeit zur Visualisierung der Auslastung und vergleichbarer Parameter von Steuergeräten und Bussen besteht in der Veränderung der *Form bzw. der Größe* von Objekten. Jedes Objekt hat eine Ausgangsgröße, die den kleinstmöglichen Wert repräsentiert. Wird nun ein höherer Wert erreicht, vergrößert sich das Objekt entsprechend bis zu einer definierten Maximalgröße. Durch die Vergrößerung aktiver Objekte würden diese wiederum hervorgehoben werden und sich ggf. auch vor andere, weniger aktive, Objekte schieben.

4.2.2 Darstellung der Steuergeräte

Im Bereich der Steuergeräte- und Komponenten-Darstellung gibt es die Unterscheidung zwischen einer *realistischen Darstellung* und somit einer Nachbildung des Steuergeräte-Aussehens und der *abstrakten Darstellung*, z. B. durch einfache Quader. Während die abstrakte Darstellung noch wei-

tere Möglichkeiten offen lässt, mit welchen Formen und Körpern einzelne Objekte repräsentiert werden, gibt es bei der realistischen Darstellung keine solche Wahl. Zwar kann eine Abstufung zwischen abstrakt und realistisch vorgenommen werden, indem zum Beispiel nicht jedes Steuergerät extra modelliert wird sondern es eine Unterscheidung gibt zwischen einzelnen Typen (Steuergeräte, Gateways, ...), aber diese Typen dann alle mit dem gleichen Modell dargestellt werden.

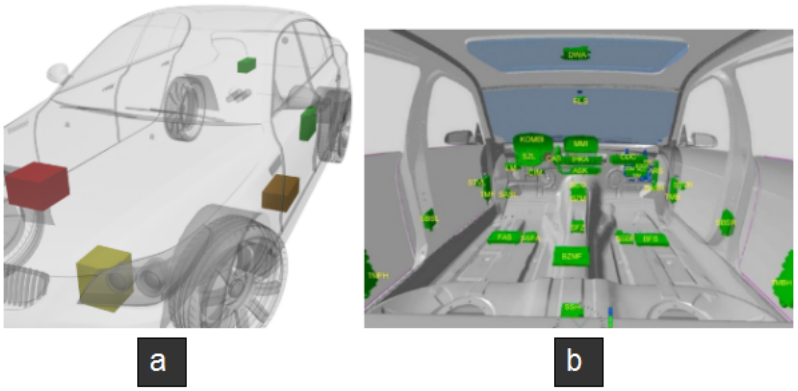


Abbildung 4.4: (a) abstrake Darstellung der Steuergeräte, (b) realistische Darstellung und Positionierung der Elemente

In Abbildung 4.4 wird außerdem auch noch der andere Faktor, der bei der Steuergeräte-Darstellung eine Rolle spielt, deutlich. Wie in der Abbildung zu erkennen ist, macht auch die Genauigkeit der Positionierung einen Unterschied aus. Dabei hat die abstrakte Positionierung (Abbildung 4.4 (a)) den Vorteil, dass der Nutzer zum einen in etwa den Verbauort abschätzen kann, aber seitens des Programms die Möglichkeit besteht, das Bordnetz und seine Komponenten übersichtlich darzustellen. Dies ist bei einer vollkommen realistischen Darstellung (Abbildung 4.4 (b)) nicht immer möglich, da häufig Bauteile eng nebeneinander verbaut sind. Ein Vorteil der realistischen Positionierung wäre allerdings die sehr genaue Darstellung des Verbauorts, wenn dieser für den Benutzer von Interesse ist.
Unabhängig vom Aussehen und der Positionierung wurde im Rahmen dieser Arbeit ein Konzept einer semantischen Lupe erstellt. Dabei dient der Mauszeiger und dessen unmittelbare Umgebung als Selektionsinstrument. Bauteile, die im 3D-Raum vom Betrachter aus gesehen hinter dieser Fläche liegen, die um den Mauszeiger herum aufgespannt wird, offenbaren dem Nutzer weitere Details.

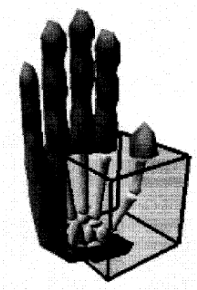


Abbildung 4.5: Funktionsweise einer „3D magic lens“ beim Zugriff auf eigentlich verdeckte Informationen [29].

Hierzu könnten detaillierte Messwerte zählen (Auslastung, empfangene Nachrichten pro Sekunde, ...), aber auch Informationen über die Bordnetztopologie und die Verteilung von Funktionen über die Steuergeräte, also welches Element welche Funktionsblöcke beherbergt. Viega et al. entwickelten in diesem Zusammenhang das Konzept der „3D magic lenses“ [29], bei dem dreidimensionale semantische Lupen dazu genutzt werden, um eigentlich verdeckte oder zusätzliche Informationen für den Benutzer sichtbar zu machen. Abbildung 4.5 veranschaulicht das Konzept mit einer dreidimensionalen Lupe, die dem Nutzer ermöglicht auf verdeckte Informationen zuzugreifen, wie in diesem Fall auf die Knochen der dargestellten Hand.

4.2.3 Fahrzeug-Verhalten

Im Bereich der Verhaltensvisualisierung im 3D-Modell gibt es mehrere Möglichkeiten mit eventuellen Aktivitäten umzugehen. Grundannahme für diese Konzepte ist, dass das Modell dazu genutzt wird, den jeweils aktuellen Fahrzeugstatus zu visualisieren, der sich aus der (aufgezeichneten) Bordnetzkommunikation ergibt. Hierzu zählen Status wie „Tür vorne links geöffnet“ oder „Abblendlicht aktiviert“, die dann entsprechend im Modell umgesetzt würden.
Der einfachste Fall in diesem Kapitel ist sicherlich die *einfache Darstellung von Animationen*, wenn Statusveränderungen eintreten. Zum Beispiel würde dann einfach die Tür aufschwingen und geöffnet stehenbleiben. Aufgrund zu befürchtender Inattentional-Blindness-Effekte, d.h. Änderungen bzw. Stimuli im Sichtfeld des Nutzers werden nicht wahrgenommen, während er sich auf eine andere Aufgabe konzentriert [16], muss überlegt werden, wie diese Änderungen hervorgehoben werden können.
Einer dieser Ansätze ist die *Hervorhebung aktiver Elemente durch eine möglichst kontrastreiche Farbgebung*. Eine aufschwingende beispielsweise rot eingefärbte Tür in einem grauen Modell wird weniger häufig übersehen werden als eine aufschwingende Tür, die dieselben Farben wie der Rest des Modells und ihrer nahen Umgebung aufweist, da die Tür dann präattentiv wahrgenommen wird (Abbildung 4.6 (a)). Ist die Animation abgeschlossen wird das Objekt wieder in seiner Ursprungsfarbe dargestellt.

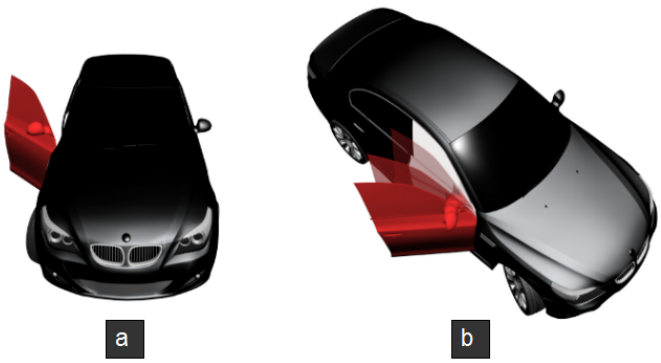


Abbildung 4.6: (a) Einfärben einer aktiven Komponente, (b) Animationssput deutet Aktivität und Richtung der Bewegung an.

Aber selbst mit einer entsprechend starken Einfärbung kann nicht vollends ausgeschlossen werden, dass einzelne Ereignisse im Modell vom Nutzer übersehen werden können. Aus diesem Grund können Darstellungsformen helfen, die eine Aktion noch sichtbar halten, obwohl sie eigentlich bereits abgeschlossen ist. Zur Realisierung dieser Idee gibt es wiederum verschiedene Möglichkeiten. Eine davon ist die Ergänzung der Einfärbung um ein *Leuchten des Objekts*. Dieses Leuchten startet mit der Animation, lässt im Laufe der Animation nach, ist aber auch nach Abschluss

der Animation noch für einen Augenblick sichtbar. Zwar würde auch eine verlängerte Einfärbung einen solchen Zweck erfüllen, doch durch das Überstrahlen der nahen Umgebung ist auch ein schwaches Leuchten noch gut zu sehen, während eine einfache Einfärbung bei Annäherung an die Ursprungsfarbe nicht immer gut sichtbar ist.
Ein anderer Ansatz, der aber dasselbe Ziel wie der gerade beschriebene Ansatz verfolgt, ist die *Darstellung von leicht transparenten, über die Zeit verblassenden Kopien der Objekte*, die entlang des Animationspfades angeordnet sind. In Abbildung 4.6 (b) ist das Prinzip zu erkennen. Die Tür hat sich geöffnet, aber auch noch kurz danach sind einige leicht transparente Darstellungen der Tür im Modell sichtbar, die dem Nutzer anzeigen, dass die Tür sich vor kurzer Zeit geöffnet hat. Als weitere Information zum bereits vorgestellten Konzept beinhaltet das Animationsspur-Konzept (Abbildung 4.6) auch noch die Richtung, in die sich ein Objekt bewegt hat. Allerdings ist das Konzept nur auf sich bewegende Obekte übertragbar und bei sehr kleinen Veränderungen eventuell nicht aussagekräftig genug.
Das letzte Konzept in diesem Bereich greift die Idee des MCV-Systems erneut auf. Hier können für Objekte, die für den Nutzer relevant sind, Trigger definiert werden, die, wenn sie ausgelöst werden, eine kleine *zusätzliche 3D-Ansicht* öffnen und die Animation des beteiligten Objekts darin darstellen. Somit wäre die Animation für den Nutzer sichtbar, ohne dass der Gesamtblickwinkel zwangsläufig auf dem entsprechenden Objekt ruhen müsste. Durch das Öffnen des neuen Fensters kann auch zusätzlich die Aufmerksamkeit des Nutzers erregt werden, wenn sich das Fester deutlich genug von der Umgebung abhebt.

4.2.4 Personeninteraktion und Sensoraktivitäten

Dieser Abschnitt beschäftigt sich mit einigen Konzepten, bei denen spezielle Ereignisse am Modell gesondert hervorgehoben werden. Hierzu können Aktionen gehören, die von einem Insassen ausgelöst werden oder auch Ereignisse, die aufgrund von Sensoremesswerten ausgelöst werden. Dieses Hervorheben soll dem Nutzer ermöglichen, die Zusammenhänge im Modell besser zu verstehen.

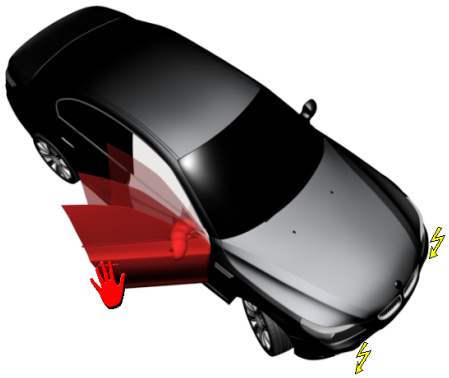


Abbildung 4.7: Mögliche Visualisierung von Fahrerinteraktion und Sensorereignissen im 3D-Modell.

Der erste Ansatz in diesem Zusammenhang ist das *Personeninteraktion*. Viele Ereignisse finden nur statt, wenn sie explizit vom (Bei-)Fahrer ausgelöst werden, z. B. das Einschalten vom Radio. Um dies dem Nutzer nun entsprechend darstellen zu können, dass von einer Person mit dem Auto interagiert wurde, sollen diese Ereignisse speziell hervorgehoben werden durch Einblenden von unterschiedlichen Symbolen. Abbildung 4.7 zeigt jeweils eine Möglichkeit für die Anzeige von Personeninteraktion.

An der Fahrzeugfront in Abbildung 4.7 ist eine Visualisierungsmöglichkeit für *Sensorereignisse* dargestellt. Diese Anzeige soll dem Nutzer vermitteln, dass durch die Messwerte der visualisierten Sensoren ein Ereignis ausgelöst haben, wie z. B. das Ertönen eines Warnsignals.
Für beide Fälle gibt es jedoch spezielle Dinge zu beachten. Die Interaktionen mit dem Fahrzeug sind nur aus den Bordnetz-Daten rekonstruierbar aufgrund des Wissens welche Schalter usw. vom Fahrer direkt bedient werden. Hier muss also eine entsprechende Abfrage stattfinden und selbst dann kann es noch zu falschen Anzeigen kommen, wenn im Bordnetz irrtümlicherweise eine Interaktion angezeigt wird, obwohl diese im aktuellen Zustad nicht möglich ist. Andererseits kann dies natürlich auch eine Art des Findens von Fehlern sein, die sonst eventuell in der Menge der Daten vorerst untergehen würden. Komplizierter hingegen ist die Anzeige von Sensorereignissen. Da die Sensoren selbst keinerlei Auswertung vornehmen, sondern ihre Daten anderen Steuergeräten zur Verfügung stellen und diese dann die Daten auswerten, ist ein direkter Rückschluss an den Sensor nicht trivial.

4.2.5 Sonifizierung

Eine weitere Konzeptidee ist die Ausweitung der Informationsdarstellung auf den Audiokanal. Bisherige Konzeptvorschläge präsentieren Informationen lediglich visuell. Zwar können so viele Informationen gut und übersichtlich dargestellt werden, aber dennoch kann es sinnvoll sein, einen weiteren Informationskanal zu öffnen und dem Nutzer ebenfalls über den auditiven Kanal eine Rückmeldung zum Fahrzeugstatus zu geben.
Hier wären zwei Möglichkeiten denkbar, die auch gemeinsam in einem Endkonzept funktionieren würden. Die erste Erweiterung um Geräusche sieht vor, dass das Programm eine *realistische Sound-Kulisse* für den Benutzer aufbaut. Dazu würden sowohl ein ggf. laufender Motor gehören, wie auch das Verschließen der Zentralverriegelung oder das Klacken eines laufenden Blinkers. Dies sind alles Geräusche, die man aus dem alltäglichen Umgang mit Fahrzeugen kennt und ohne weitere visuelle Angaben interpretieren kann. Somit könnten dem Benutzer, ohne dass er auf den Monitor schaut bzw. das 3D-Modell im Blick hat, wichtige Informationen zum Status und zum Fahrzeug vermittelt werden.
Beim zweite Ansatz werden *abstrake Warnsignale* verwendet. Hier spielen selbst definierte Grenzwerte eine wichtige Rolle, denn diese würden beim Erreichen durch das Abspielen eines entsprechenden Soundfiles deutlich gemacht werden. Denkbar sind hier etwa Auslastungswerte, die eigentlich nicht überschritten werden dürften, Bus-Lasten, die unerwartet eintreten oder auch konkrete Nachrichteninhalte, die aktuell abgeschickt wurden. In solchen Fällen könnte ein kurzer Hinweis-Ton abgespielt werden, um den Nutzer darauf aufmerksam zu machen, falls er die entsprechenden Daten gerade nicht im Blick haben sollte.
Bei der Entwicklung eines auditiven Konzepts muss jedoch auch berücksichtigt werden, dass eine zu aufdringliche Soundkulisse in einem Büroraum andere Mitarbeiter stören kann. Diesem Problem kann zwar mit der Nutzung von Kopfhörern begegnet werden, aber dennoch ergibt sich aus diesem Problemfeld, dass es nötig ist die Soundfunktionalität deaktivierbar zu machen.

4.2.6 Interaktion mit dem 3D-Modell

Die letzte Konzeptkategorie beschäftigt sich mit der direkten Interaktion des Nutzers mit dem 3D-Modell und mit der Steuerung der Kamera. Eigentlich sind dies zwar unterschiedliche Themengebiete, aber da beide im Rahmen dieser Arbeit nur am Rande bearbeitet werden sollen, werden sie an dieser Stelle unter dem Punkt Interaktion mit dem Modell zusammengelegt.
Betrachtet man die *Interaktion mit einem 3D-Modell*, so fällt auf, dass bei den bisherigen Prototypen lediglich die Kamera gesteuert wird. In einige Fällen kann es jedoch auch sinnvoll sein, direkt mit dem Modell interagieren zu können, d.h. der Nutzer greift aktiv auf das Modell zu und löst unterschiedliche Aktionen aus. Es gibt unterschiedliche Möglichkeiten, die Interaktion mit dem

Modell zu ermöglichen. Die simpelste Lösung funktioniert mit einem einfachen Point & Click Verfahren - der Nutzer wählt im 3D-Modell ein Bauteil aus, klickt es an und erhält eine Liste mit möglichen Aktionen, aus denen er eine Auswahl treffen kann. In Abbildung 4.8 wäre diese Aktion also das Schließen der Tür. Darüber hinaus gibt es auch die Idee der *Smart Objects* [10] bei denen die Objekte sozusagen wissen, was mit ihnen gemacht werden kann und sich dementsprechend dem Nutzer präsentieren. Im Beispiel mit der geöffneten Tür aus Abbildung 4.8 sähe es dann so aus, dass eine eingeblendete Hand an der Tür andeutet, die Tür zu schließen. Wählt man diese Hand aus, wird die Tür dann auch tatsächlich geschlossen. Der Bezug zur Analyse ergibt sich daraus, dass auf diese Weise visualisierbar wäre, welche Steuergeräte oder Bordnetzkomponenten mit einer Aktion in Verbindung stehen. Beim Beispiel der Tür könnte man also erkennen, welche Steuergeräte in welcher Reihenfolge beim Schließen der Tür aktiv werden.

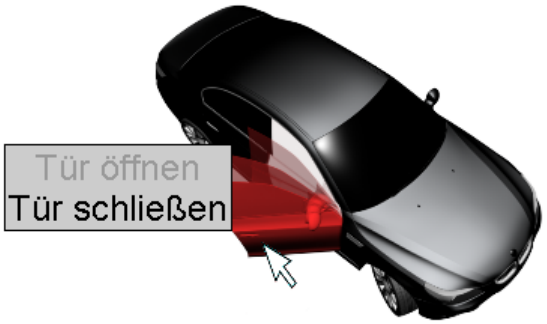


Abbildung 4.8: Auswahl einer Aktion über ein PopUp-Menü.

Der zweite Teil dieses Abschnitts befasst sich mit der *Steuerung der 3D-Kamera*. Zusätzlich zu dem genutzten Modell der beiden getesteten Prototypen (Kapitel 2.2) bei dem die Kamera um das Modell herum rotierbar und zoombar ist, gibt es noch weitere Möglichkeiten, dem Nutzer die Navigation durch die virtuelle Welt zu ermöglichen. Eine Idee wurde bereits zu Beginn dieser Ausarbeitung in Kapitel 2 als „First person camera“ beschrieben. Wichtig hierbei ist noch zu erwähnen, dass die Kamera keinerlei Einschränkungen (Schwerkraft, ...) im Raum unterliegt und sich völlig frei bewegen lässt. Dadurch werden beliebige Sichtwinkel auf das Modell eröffnet und es ist auch möglich, die Kamera an eine beliebige Stelle im Modell zu bewegen und von dort zu beobachten. Die Steuerung dieser Kamera würde mit einer Kombination aus Maus und Tastatur erfolgen, wobei die Maus zur Kamera-Rotation genutzt würde und die Tastatur zur Bewegung im Raum. Insgesamt lässt sich dieses Modell gut mit der Metapher eines Kameramanns beschreiben, der sich als nicht-sichtbarer Avatar des Nutzers durch die 3D-Szene bewegen kann.

Abbildung 4.9: Standard-Marker aus dem ARToolKit [34].

Einen völlig anderen Weg geht die *Steuerung der 3D-Kamera mithilfe von Markern* (z. B. AR-Toolkit, siehe [34]). Hierbei werden allerdings zuerst einmal erweiterte Anforderungen an die Ausstattung des benutzten Rechners gestellt, da eine Webcam benötigt wird. Mit dieser Webcam ist es dem Programm dann möglich, im Kamerabild sogenannte AR-Marker zu entdecken.
Diese Marker (siehe Abbildung 4.9) sind klar definiert und ermöglichen dem Computer das Berechnen der Markerposition und Ausrichtung im Raum. Außerdem können mehrere Marker in einem einzigen Modell für eine erhöhte Erkennungsstabilität genutzt werden.
Im konkreten Fall würde man ein reales Fahrzeugmodell mit diesen Markern ausstatten. Sobald man im Programm einen Kamerakonfigurationsmodus betritt, sucht der Rechner im Bild nach den eingestellten Markern und rotiert das 3D-Modell entsprechend der Ausrichtung der Marker zur Kamera. Dabei steht auch zur Auswahl, ob das Fahrzeugmodell vor einem einheitlich gefärbten Hintergrund oder dem aktuellen Kamerabild dargestellt werden soll, wie z. B. in Abbildung 4.10. Verlässt der Rechner den Konfigurationsmodus, z. B. auf Knopfruck, und speichert die aktuelle Rotation ab, kann der Nutzer das Modell wieder aus der Hand legen. So wäre es dem Benutzer möglich, die Kamera auf relativ natürlichem Wege mit Hilfe eines realen 3D-Objekts zu beeinflussen. Das Problem, das sich ergibt, wenn man sich in einer 3D-Welt mit Hilfe von 2D-Eingabegeräten (Tastatur und Maus) bewegen will [9], würde somit entfallen.

Abbildung 4.10: In ein Kamerabild gerendertes 3D-Fahrzeugmodell.

4.3 Konzeptbewertungen

Die im vorigen Kapitel vorgestellten Konzepte wurden nach der Zusammenstellung zusammen mit den jeweiligen Bildern erneut den Experten bei BMW vorgelegt. Im Rahmen eines weiteren Leitfadeninterviews mit sechs Mitgliedern der Zielgruppe des zu entwickelnden Programms sollten wichtige Informationen zur endgültigen Konzepterstellung und dem Design des Programms generiert werden. In diesem Abschnitt werden nun die Ergebnisse dieser Befragungen und Gespräche, die auf der Grundlage der bereits vorgestellten Konzepte geführt wurde, erläutert zusammengefasst.

4.3.1 Bewertung der View-Konzepte

Die, in den Gesprächen zuerst vorgelegten, Konzepte waren die unterschiedlichen View-Konzepte aus Kapitel 4.1. Dabei waren vor allem die Konzepte 2 und 3 die Favoriten der Befragten. Bei View-Konzept 2, also der Möglichkeit sich einzelne Komponenten aus dem Fahrzeug herausziehen zu können, vermuteten alle Teilnehmer eine gute Übersicht über die relevanten Bauteile. Bei View-Konzept 3 wurde besonders die mögliche Trennung von Elektronik und Mechanik durch

separate 3D-Anzeigefenster hervorgehoben. Insgesamt wurden beide Konzepte sehr positiv aufgenommen und überraschendweise wurde von allen Teilnehmern unabhängig voneinander eine Kombiation von beiden Konzepten vorgeschlagen. Dies würde also bedeuten, dass man sich mehrere paralelle Ansichten erstellen, sie unabhängig voneinander konfigurieren und sich innerhalb der Ansichten außerdem Bauteile aus dem Fahrzeug herausziehen kann. Die Idee eines MCV-Systems (mehrere 3D-Ansichten, separate Bauteile) innerhalb eines MCV-Systems (klassische Ansichten, 3D-Ansicht) scheint hiernach ein guter Weg zu sein, um eine 3D-Ansicht in die aktuellen Programme zu integrieren. Diese Kombination wurde insgesamt als die beste Lösung der View-Konzepte gesehen.

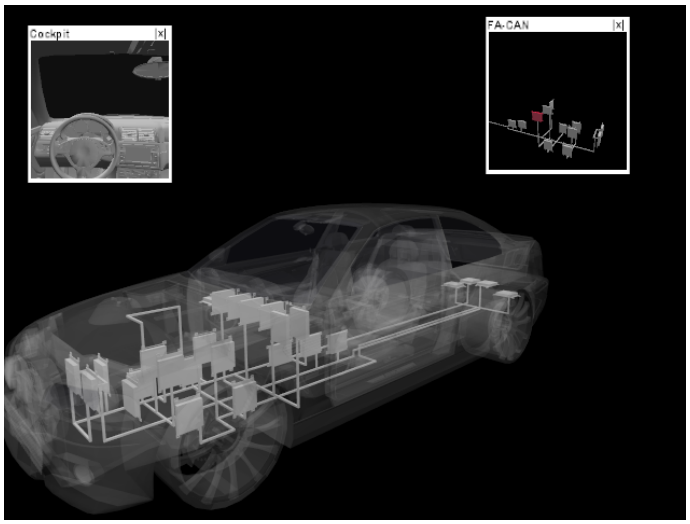


Abbildung 4.11: MCV-System in der 3D-Ansicht.

4.3.2 Bewertung der Auslastungsvisualisierungen

Im Bereich der Konzepte zur Informationsvisualisierung ist die folgende Auswertung, wie bereits die Vorstellung der Konzepte, in Abschnitte und die einzelnen Themengebiete unterteilt. Den Beginn der Auswertung machen die Visualisierungskonzepte zum Thema Auslastung und Bus-Last. Im Rahmen der Befragung wurde sehr deutlich, dass die Definition einer Auslastung im Steuerferät nicht trivial ist und auch bei der Darstellung der Bus-Last diverse Parameter beachtet werden müssen. Dies sind jedoch technische Probleme, die bei der Erstellung eines Visualisierungskonzepts in diesem Fall keine Rolle spielen sollen. Darüberhinaus gingen die Anmerkungen zur singulären Einfärbung zwar in die Richtung, dass der Status jeweils anhand der Farbe einfach abzulesen sei, aber dass die Farbkodierung auch dazu genutzt werden könne, um einzelne Baugruppen voneinander unterscheiden zu können. Hier wurde auch angesprochen, dass die Farbgestaltung in einem MCV-System zu beachten sei - ein Bauteil, das in einem Ansichtsfenster grün ist, dürfe nicht in einem anderen Ansichtsfenster rot sein. Die Einstellung zu einer singulären Einfärbung war also nicht durchweg positiv.
Die weiteren Konzepte wurden recht interessiert begutachtet und es gab umfangreiches Feedback. Vor allem die Idee des radialen Farbverlaufs wurde mehrfach hinterfragt und als „neue Idee“ gekennzeichnet, die allerdings nicht einfach zu bewerten sei, da Erfahrungswerte mit einer solchen Darstellungsform fehlen. Dennoch waren sich viele Befragte sicher, dass die so realisierte History-Funktion direkt auf den Steuergeräten eine sinnvolle Eigenschaft sei, da so kurz-zurückliegende Ereignisse im Bereich Auslastung/Bus-Last visualisiert werden könnten. Zum Teil wurde aber

auch kritisiert, dass der Nutzer bei zu häufiger Verwendung dieses Effekts visuell überlastet wird - als Beispiel wurde hier eine Situation mit mehreren Steuergeräten aufgeführt, die eine relativ stark schwankende Auslastung haben.
Ein anderes Konzept in dem bis zu einem gewissen Grad die Auslastungsdaten der letzten Sekunden dargestellt werden können, ist die Nutzung der Transparenz der Bauteile zur Visualisierung, wie in Abbildung 4.3 zu sehen. Dieses Konzept wurde zwar zum einen sehr positiv aufgenommen, aber zum anderen auch kritisch hinterfragt. Positiv aufgefallen ist dabei vor allem die zu erwartende Übersichtlichkeit, die sich je nach Situation von allein einstellen müsste. Andererseits wurde dabei auch festgestellt, dass eventuell schon kleinste Abweichungen wichtig seien und diese bei einer Transparenz-Visualisierung nicht immer zu erkennen seien, da eine direkte Skala als Vergleichsmittel fehlt.
Die letzte in diesem Kapitel vorgestellte Idee war die Veränderung von Form bzw. Größe einzelner Bauteile entsprechend ihrer Auslastung. Bei Bussen wären dies der dargestellte Kabeldurchmesser und bei Steuergeräten die Skalierung. Hier waren sich die Personen einig, dass es sich um schwierig zu erkennende Faktoren handelt. Außerdem würde ein Vorteil des 3D-Modells - die annähernd realistische Darstellung von Bordnetz und Fahrzeug - stark verzerrt werden, wenn die Steuergeräte und Busse ihr Aussehen bzw. ihre Größe dynamisch verändern würden. Ähnlich wie beim radialen Farbverlauf wurde zum Teil auch darauf verwiesen, dass diese Visualisierungstechnik bei einer Szene mit gewisser Komplexität zu unübersichtlich werden dürfte.
Insgesamt sind die unterschiedlichen Visualisierungskonzepte zum Thema Auslastung gut aufgenommen, allerdings auch zum Teil recht kritisch bewertet worden. Während der radiale Farbverlauf zwar dafür gelobt wurde, dass er eine History-Funktion in der 3D-Umgebung integriert, war man sich nicht sicher, ob diese Funktion nützlich sei, da die Erfahrungswerte mit der Art der Darstellung fehlen würden. Ebenso wurden auch die Transparenz und die Größenvariationen als Informationskodierungen eher kritisch gesehen, da bei diesen eine direkte Vergleichsmöglichkeit fehlt und kleine Veränderungen kaum zu erkennen sind. Selbst die singuläre Einfärbung von Bauteilen wurde im Rahmen eines MCV-Systems in Frage gestellt, da dort eine einheitliche Einfärbung eines Bauteils in allen Anzeigebereichen gefordert wird. Diesem Ziel würde ein singuläre Einfärben zur Auslastungsvisualisierung entgegenstehen.

4.3.3 Bewertung der Steuergeräte-Darstellung

Die Meinungen zur Darstellung der Steuergeräte und Busse ging auseinander. Während die eine Hälfte der realistischen Darstellung einen „Wiedererkennungseffekt“ zusprachen und auch die Positionierung der Steuergeräte wichtig fanden, war die andere Hälfte der Befragten eher der Meinung, zur reinen Informationsvisualisierung seien keine realen Positionierungen der Steuergeräte nötig. Als Kompromiss wurde von mehreren Teilnehmern vorgeschlagen, verschiedene Detailstufen anzubieten, so dass der Nutzer später je nach Use-Case die Wahl zwischen mehreren Ansichten hat. Bei dieser Gelegenheit wurden auch mögliche Use-Cases genauer beschrieben: die realistische Positionierung eignet sich demnach besonders für Schulungs- und Werkstatt-Zwecke, bei denen es auf den genauen Standort ankommt, während die abstrakte Positionierung bei der reinen Kommunikationsvisualisierung von Vorteil sei, da es hier auf Übersicht ankomme. Als möglichen Mittelweg schlugen die Befragten, die bereits bei der vorherigen Evaluation teilgenommen haben, vor, einen Mittelweg zwischen den Visualisierungen der getesteten Prototypen zu gehen. Das Prinzip einer semantischen Lupe wurde relativ kritisch gesehen, da die Funktionen zum Teil über mehrere Steuergeräte verteilt sind und diese Art der Visualisierung keinen wirklichen Nutzen bringen würde, zumal eine entsprechende Visualisierung in 2D übersichtlicher sei, so die Teilnehmer.

4.3.4 Bewertung des Fahrzeug-Verhaltens

Interessante Anregungen und Bewertungen gab es vor allem im Bereich des mechanischen Umgebungsmodells. Für alle Befragten schien dies der zentrale Bestandteil der 3D-Visualisierung zu sein und wurde auch durchweg gut aufgenommen, auch wenn der Nutzen - wie viele der vorgestellten Konzepte - stark vom jeweiligen Use-Case abhängig sei. Ergebnis der Konzeptvorstellung des mechanischen Umgebungsmodells war vor allem, dass es nötig sei die entsprechenden Änderungen tatsächlich hervorzuheben. Als Vergleich wurde hier mehrfach der Prototyp der Purdue Universiy herangezogen (siehe Kapitel 2.2.2), bei dem bemängelt wurde, dass einige Änderungen am Modell kaum sichtbar seien. Von den vorgestellten Hervorhebungstechniken wurde die Idee, einzelne aktive Elemente einzufärben, am besten aufgenommen. Die Darstellung von Schattenkopien „muss nicht sein“ laut einiger Teilnehmer und die Zusatzfenster, die definierte Aktionen gesondert anzeigen, seien interessant, aber sollten allenfalls optional verfügbar sein. An dieser Stelle wurde auch mehrfach auf den Konfigurationsaufwand verwiesen, der mit einer komplexen Fenster-Steuerung auf den Nutzer zukäme.

4.3.5 Bewertung der Personeninteraktion und Sensoraktivität

Die Idee, Fahreraktionen gesondert zu kennzeichnen und auch Sensorereignisse speziell hervorzuheben, wurde unterschiedlich bewertet. Viele Teilnehmer sahen in der Kennzeichnung und Hervorhebung von Fahreraktionen eine sinnvolle Erweiterung des mechanischen Umgebungsmodells, da der Nutzer weiter entlastet wird. Er müsse dann nicht mehr selbst aus den Trace-Daten herauslesen, ob der Fahrer zu einem Zeitpunkt mit dem Fahrzeug interagiert hat oder nicht. Für eine erweiterte Darstellung von aufgezeichneten Testfahrt-Daten wird hier also eine sinnvolle Neuerung gesehen. Allerdings wurde auch darauf verwiesen, dass diese Aktionen, da sie meist mit sehr kleinen Elementen stattfinden, auf jeden Fall besonders gekennzeichnet werden müssen, damit sie nicht übersehen werden. Die Darstellung der Sensordaten wurde prinzipiell auch für gut befunden, jedoch wurde durchweg zu bedenken gegeben, dass es in Verbindung mit Sensoren zum Teil sehr komplexe Systeme gäbe und es somit kaum umsetzbar bzw. realisierbar sei. Zusätzlich wurde die Frage gestellt, welche Sensoren visualisiert werden sollen und wie die Hervorhebung gestaltet werden solle, damit die Ereignisse auf jeden Fall sichtbar seien. Desweiteren wurde angemerkt, dass es keine konkrete Sensorereignisse gibt, sondern die Sensoren ständig Werte zurückliefern. Hier müssten also wiederum Grenzwerte oder Schwellenwerte definiert werden, bei denen eine visuelle Rückmeldung erfolgen soll.

4.3.6 Bewertung der Sonifizierung

Ein sehr positiv aufgenommes Konzept war die Idee der Soundrückmeldung an den Benutzer. Hier wurden von den Befragten gleich mehrere Szenarien entwickelt, in denen eine solche Rückmeldung nützlich sei und wie genau diese funktionieren könnte. Vor allem die Möglichkeit reale Sounds parallel zum 3D-Modell abzuspielen und somit auch über kleine Änderungen (z. B. Zentralverriegelung schließt) informiert zu werden, wurde sehr gut aufgenommen. Doch auch die abstrakten Warntöne bei definierten Ereignisse wurden gut aufgenommen. Grund hierfür ist, dass sie als unaufdringlicher als reale Sounds empfunden wurden. Es wäre allgemein störend, wenn Sounds nicht synchron zu den entsprechenden Modellaktionen abgespielt werden. Als Beispiel wurde unter anderem das Klacken des Blinkers genannt, dass nicht synchron zum Blinken des Blinkers läuft. Insgesamt wurde beim Thema Sound allerdings auch das Problem gesehen, dass das Arbeitsumfeld eines solchen Anwendung meist ein Großraumbüro ist. Wenn dort mehrere Rechner betrieben werden, die über eine solche Soundausgabe verfügen, kann es zu Verwirrungen kommen und andere könnten sich durch die Soundausgabe belästigt fühlen. Zwar lässt sich das Problem sehr einfach mit Kopfhörern umgehen, allerdings ist fraglich, inwieweit diese immer getragen werden und ob diese nicht ein Hindernis für teaminterne Kommunikation sind. Somit muss

Sound auf jeden Fall ein optionales Feature sein, welches ohne Probleme zu deaktivieren ist, ohne dass dem Nutzer dadurch wesentliche Funktionen verloren gehen.

4.3.7 Bewertung der Interaktion mit dem Modell

Der letzte Abschnitt der Konzeptvorstellung beschäftigte sich mit der Interaktion mit dem Modell. Hierbei sollten die Möglichkeiten und der Bedarf für einen eingeschränkten Zugriff auf das Modell abgeschätzt werden. Hier war die Rückmeldung eindeutig. Im Bereich der Analyse/Diagnose sahen die meisten keinerlei Nutzen in einer direkten Interaktion mit dem Modell. Zum einen ist in diesem Rahmen die Simulation von Aktionen sehr stressig und das genaue Timing ist nicht ohne weitere Hilfsmittel genau wiederholbar. Somit sind Ergebnisse nicht richtig reproduzierbar und es macht keinen Sinn, eine solche Möglichkeit zu implementieren. Davon unabhängig wurde auch das Problem von verdeckten Aktionen in diesem Zusammenhang angesprochen. Würde man eine Interaktion im Innenraum durch Knopfdruck auslösen ist die Reaktion außen eventuell gar nicht sichtbar. Die in diesem Zusammenhang vorgestelle AR-Kamerasteuerung wurde von allen Testpersonen als unnötig bewertet, da eine normale 3D-Kamera-Steuerung ausreiche, um eine einfache Kontrolle über die Anwendung zu ermöglichen. Als möglichen Einsatzort sahen einige jedoch Messen oder Präsentationen, bei denen Kunden oder Nicht-Experten mit einem virtuellen 3D-Modell in Kontakt kommen und so eine einfache Steuerungsmöglichkeit bekämen.

4.3.8 Fazit der Bewertungen

Insgesamt hat die Konzeptvorstellung gut funktioniert, die verschiedenen Konzepte wurden gut angenommen und es ist jeweils eine angeregte Diskussion entstanden, in deren Verlauf die Konzepte bewertet wurden und auch einige neue Ideen entstanden sind. Auf der Grundlage der Konzeptbewertungen und der zum Teil neuen Idee wurde anschließend das endgültige Konzept entwickelt, welches im nächsten Abschnitt beschrieben und mit Entwürfen vorgestellt wird.

4.4 Endgültiges Konzept

In diesem Abschnitt wird das endgültige Konzept für eine 3D-Visualisierung von Bordnetzkommunikation beschrieben. In diesem Rahmen wird auf die Bewertungen der Konzepte aus der Konzeptsammlung der vorangegangenen Kapitel zurückgegriffen. Durch die teils sehr ausführlichen Bewertungen ist es möglich die Vor- und Nachteile der einzelnen Konzeptideen abzuschätzen. Eine weitere Grundlage für einzelne Konzeptentscheidungen ist das in Kapitel 2 vorgestellte Related Work. Durch die dort gewonnen Erkenntnisse ist es möglich in Verbindung mit den Ergebnissen der Konzeptbewertungen ein endgültiges Konzept zur 3D-Visualisierung von Bordnetzkommunikation zu erstellen.

4.4.1 Verwaltung der 3D-Ansichten

Zu Beginn der Festlegung auf ein finales Konzept musste bestimmt werden, wie die Verwaltung der 3D-Ansicht funktionieren soll und ob es mehr als eine Ansicht geben wird. Aufgrund der Rückmeldungen während der Konzeptbesprechungen wurde entschieden, eine Mischung aus den View-Konzepte 2 und 3 zu realisieren. Diese Lösung wurde unabhängig von einander von allen befragten Personen vorgeschlagen und als optimal bezeichnet, da sie ein Maximum an Flexibilität und Übersicht bietet. Lediglich das Herausziehen und Platzieren von Bauteilen aus dem Fahrzeug soll nicht umgesetzt werden, stattdessen wird eine Detailansicht über weitere 3D-Fenster ermöglicht, indem alle Elemente bis auf das entsprechende Bauteil ausgeblendet werden sollen. Dies hat den Vorteil, dass sowohl normale 3D-Ansichten als auch Detailansichten von Elementen auf die gleiche Art verwaltet und bedient werden können. Somit entsteht kein Bruch im Bedienkonzept, denn dem Benutzer wird zu jeder Zeit klar dargestellt, dass es sich bei den zusätzlichen

3D-Ansichtsfenstern um Teile einer Hauptansicht handelt. Es soll also ein MCV-System (3D-Ansicht) innerhalb eines MCV-Systems (verschiedene 2D-Ansichten kombiniert mit 3D-Ansicht) entstehen. Abgesehen von dem MCV-System im 3D-Prototyp soll es eine Schnittstelle für andere Programme geben, die sich der 3D-Ansicht bedienen können. Dabei soll es möglich sein, dass auf sämtliche Teile des Modells in jeglicher Form zugegriffen wird, d.h. Transparenzen und Farben können beliebig gesetzt und Fahrzeugaktionen beliebig aufgerufen werden.

4.4.2 Auslastungsvisualisierung

Im Zusammenhang mit der bereits beschriebenen Schnittstelle für andere Programme im Rahmen eines MCV-Systems wird auf die eigenständige Auslastungsvisualisierung in diesem Prototyp verzichtet. Zwar war die Resonanz für eine entsprechende Darstellung in Form von Einfärbungen und Transparenzen relativ gut, jedoch überwiegt hier die Entscheidung das 3D-Modell als Teil eines MCV-Systems zu realisieren. Dadurch ist nicht absehbar, inwiefern eine Farbkodierung im MCV-System bereits vorhanden ist und somit ist es nicht möglich, eine eigenständige Kodierung aufzustellen ohne später eventuell in Konflikt mit anderen Programmteilen zu kommen. Um jedoch eine Visualisierung der Auslastung zu ermöglichen, kann jedes steuernde Programm von außen Fahrzeugkomponenten einfärben und so eine Auslastung oder ähnliches farblich kodiert visualisieren. Dabei kann dann sowohl auf die jeweils vorhandenen Farbkodierung Rücksicht genommen werden, als auch die Auswertung von Daten an zentraler Stelle vorgenommen werden. Zusätzlich bietet dieses Verfahren die Möglichkeit sämtliche anderen Farbkodierungen auf das 3D-Modell zu übertragen, wenn diese z. B. genutzt werden um andere Zusammenhänge zu visualisieren. Denkbar sind hier völlig verschiedene Informationen, wie z. B. eine Selektion oder eine dauerhafte Einfärbung. Dieses Linking & Brushing [11] bei dem durch das Markieren oder Hervorheben in einer Visualisierung diese Markierung bzw. Hervorhebung auch in allen anderen verfügbaren Visualisierungen angezeigt wird.

4.4.3 Darstellung von Steuergeräten und Bus-Systemen

Ein anderer Punkt, der in der Konzeptevaluation geklärt werden sollte war die Art der Darstellung von Steuergeräten und Bus-Systemen. Dabei ging es um die Klärung, ob eine realistische Darstellung nötig ist, um die Vorteile des 3D-Modells weiter zu fördern oder ob eine relativ abstrakte Darstellung nicht vielleicht besser sei. Aufgrund der starken Abhängigkeit vom jeweiligen Einsatzzweck bei dieser Frage wurde entschieden, einen Mittelweg zu gehen. Dabei wird auf eine völlig realistische Darstellung von Steuergeräten und Kabelbaum verzichtet, aber die abstrakte Darstellung ist möglichst nahe an die Realität angelehnt.

Konkret heißt das, dass die Steuergeräte nicht nur einfache Quader sind und die Kabelführung im Modell zumindest möglichst realitätsnah verlaufen soll. Dies soll einen guten Kompromiss bilden aus Übersichtlichkeit und Realismus. Beides wurde in den Gesprächen von den Beteiligten gefordert - immer in Abhängigkeit vom Use-Case des Programms. Daher wird nun versucht mit einem Mittelweg bei der Darstellung (siehe Abbildung 4.12) die meisten Einsatzgebiete abzudecken, ohne dass die Übersicht oder der Detailgrad zu sehr leiden.

4.4.4 Mechanisches Umgebungsmodell

Als Kernkomponente der 3D-Visualisierung hat sich während der Benutzerstudien sehr schnell das mechanische Umgebungsmodell herauskristallisiert. Alle befragten Analyse-/Diagnose-Experten sahen in einer 3D-Ansicht, die den aktuellen Fahrzeugstatus darstellt, eine sehr sinnvolle Erweiterung bisheriger Visualisierungskonzepte. Es ist also das Ziel dieser Arbeit, ein mechanisches Umgebungsmodell so anzubieten, dass andere Programme mühelos mit der 3D-Ansicht erweitert werden können. Dazu dient, wie bereits zu Beginn dieses Kapitels beschrieben, eine öffentliche

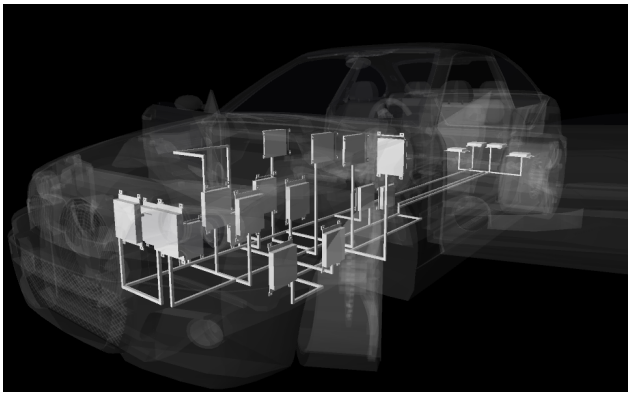


Abbildung 4.12: Ein Mittelweg zwischen realistischer Darstellung & Positionierung und einer vollständig abstrakten Darstellung.

Schnittstelle, die die diversen Funktionen des Fahrzeugs von außen steuerbar macht. Hierzu zählen einfache Vorgänge wie das Öffnen einer Tür, aber auch alle etwas komplexeren Vorgänge wie z. B. Lenken, das sich aus dem Rotieren der Vorderräder und gleichzeitigem Rotieren des Lenkrads zusammensetzt. Ein weiterer Punkt, der in diesem Zusammenhang erreicht werden soll ist die leichte Austauschbarkeit des 3D-Modells. So lässt sich die 3D-Umgebung sehr einfach an alle Fahrzeugtypen anpassen und bietet dann - wenn die Modelle den Namenskonventionen des Programms entsprechen - alle verfügbaren Funktionen des Modells an. Dies eröffnet die Möglichkeit, sich die aufgezeichneten Bordnetzdaten auch mit dem entsprechenden 3D-Modell visualisieren zu lassen.

4.4.5 2D-Anzeigeelemente

Während der Gespräche mit den Experten bei BMW wurde auch die Funktion von 2D-Anzeigeelementen in der 3D-Umgebung besprochen. Dabei zeigte sich, dass ein Interesse besteht, sich wichtige Daten in Kurzform auch in der 3D-Ansicht zugänglich zu machen. Somit ist es Teil des endgültigen Konzepts, Daten einzelner Bauteile (Auslastung, Rotationswinkel, ...) über 2D-Anzeigen sichtbar und verwaltbar zu machen. Um nicht den Überblick zu verlieren und gleichzeitig aber klarzumachen welche 2D-Elemente zu welchem 3D-Bauteil gehören, werden die 2D-Objekte mit ihren entsprechenden 3D-Gegenstücken mit einem Pfeil oder einer Linie verbunden werden. Dieses Verfahren hat sich bei Bowman et al. [2] in der Untersuchung zum Thema „New Directions in 3D User Interfaces“ als performant und gleichzeitig übersichtlich erwiesen, da die Nutzer die Anordnung der Elemente selbst bestimmen und sich so ihren eigenen Arbeitsbereich in der 3D-Umgebung schaffen können. In einer Studie verglich er dabei die Möglichkeit, die Anzeige selbst zu positionieren mit der festen Positionierung am jeweiligen 3D-Objekt. Besonders bei kleinen bzw. normalgroßen Bildschirmen war die freie Positionierung einer festen bezüglich der Geschwindigkeit bei der Benutzung deutlich überlegen. Unabhängig von der Positionierung der Elemente soll der Benutzer bei der Art der Darstellung der Informationen die Wahl zwischen zwei Möglichkeiten haben, die in Abbildung 4.13 zu sehen sind. Bei der ersten Möglichkeit (Abbildung 4.13 (a)) wird der aktuelle Wert, also beispielsweise die Auslastung eines Steuergeräts in Prozent, als entsprechender Zahlenwert dargestellt. Bei der zweiten Möglichkeit (Abbildung 4.13 (b)) handelt es sich um die Darstellung mit Hilfe eines Wertegraphen. Dieser Ansatz verfolgt das Ziel, eine gewisse Übersicht über die Werte der letzten (Milli-)Sekunden in der 3D-Ansicht zu integrieren. Um hierbei zu verhindern, dass starke Abweichungen nach oben oder unten nicht sichtbar sind, wird das Koordinatensystem anhand des kleinsten und des größten Werts immer entsprechend skaliert. Entsprechend große Änderungen müssen aber natürlich verdeutlicht werden,

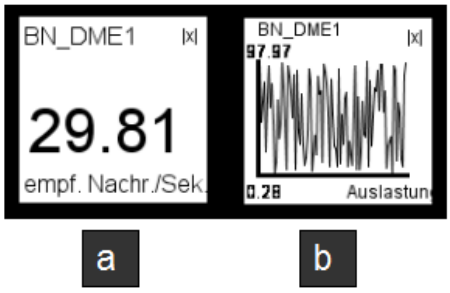


Abbildung 4.13: (a) Darstellung des aktuellen Werts, (b) Wertegraph der Auslastung.

um den Nutzer darauf aufmerksam zu machen. So sind zum einen deutliche Ausreißer nach oben oder unten gut zu erkennen und zum anderen ist eine detaillierte Sicht bei nur geringen Veränderungen möglich. Die beiden Anzeigeoptionen können individuell gewählt werden, d.h. es können parallel Zahlenwerte und Graphen von unterschiedlichen Steuergeräten dargestellt werden.

4.4.6 Auswahl von Bauteilen

Im Rahmen der Konzeptgespräche wurde mehrfach über die Auswahlmöglichkeiten von Bauteilen gesprochen. Es wurde grundsätzlich zwischen der direkten Auswahl im 3D-Fenster und der Auswahl über eine Liste unterschieden, wobei die Auswahl über eine Liste, wie das Auswahlmenü von 3ds max in Abbildung 4.14, von vielen favorisiert wurde. Sie biete die größere Übersicht und eine genauere Auswahlmöglichkeit als das direkte Anklicken und Selektieren im 3D-Modell. Zusätzlich dazu hat die Liste den Vorteil, in ein Menü integrierbar zu sein. Somit ist es möglich die Auswahl in der Liste und weitere Optionen bzgl. dieser Auswahl vorzunehmen - man hat sowohl die Auswahl und die damit gegebenen Möglichkeiten zusammen in einem Fenster im Blick.

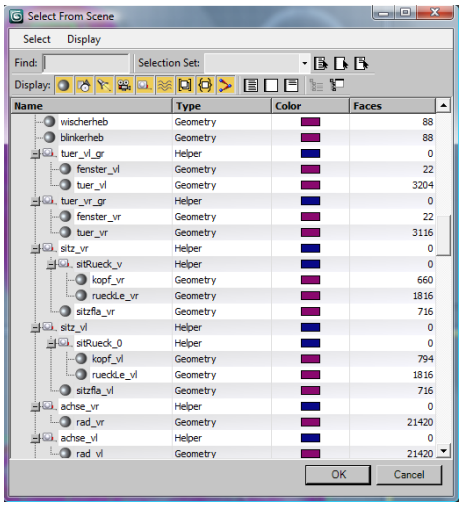


Abbildung 4.14: Selektionsmenü für Szenenobjekte von 3ds Max [31].

4.4.7 Nachrichtenvisualisierung

Es ist wichtig festzuhalten, dass im Rahmen dieses endgültigen Konzepts keine direkte Nachrichtenvisualisierung stattfinden soll. Die Tests mit der prototypischen Visualisierung von M. Moz-

dari [17] haben deutlich gezeigt, dass eine Nachrichtenvisualisierung für eine echtzeitfähige 3D-Visualisierung nicht in Frage kommen kann, da die Nachrichten schneller verschickt werden als es mit dem menschlichen Auge wahrnehmbar sind. Außerdem gibt es die Probleme mit 3D-Text-Objekten [14], somit ist eine direkte Visualisierung von Nachrichteninhalten in der 3D-Ansicht nicht Teil des endgültigen Konzepts. Denn es wäre somit zum einen nicht möglich, jede einzelne Nachricht zu erkennen und selbst wenn man die Anzahl der Nachrichten über Filter stark begrenzt, gibt es massive Probleme bei der Wahrnehmung und Lesbarkeit der Inhalte. Wie bereits zuvor in diesem Kapitel erwähnt, wird auch die Visualisierung von Daten (Auslastung, ...) auf externe Programme verlagert, um Farbkodierungsproblemen im MCV-System aus dem Weg zu gehen.

5 Umsetzung und Implementierung

Dieses Kapitel befasst sich mit der technischen Umsetzung und Realisierung des Prototyps anhand des Konzeptentwurfs aus Kapitel 4.4. Dabei wird zuerst ein Überblick über die möglichen Programme und Programmiersprachen gegeben, die im Rahmen dieser Arbeit näher betrachtet wurden und anschließend wird die Wahl der letztendlichen Programmiersprache begründet. Im Anschluss daran wird die allgemeine Programmstruktur des Prototyps vorgestellt und die genauere Funktionsweise erläutert, so dass danach auf einige Punkte des Programmcodes im Einzelnen eingegangen werden kann. Abschließend wird die prototypische Anbindung an die AutobahnView [13], die von Benjamin Kunze im Rahmen seiner Diplomarbeit bei BMW entwickelt wurde, beschrieben, die eine exemplarische Integration in ein MCV-System darstellt.
Insgesamt soll dieses Kapitel dazu dienen, die Gesamtstruktur bzw. Funktionsweise des 3D-Clients verständlich zu machen und dabei auf einige Einzelheiten genauer einzugehen. Jedoch ist es nicht das Ziel an dieser Stelle eine vollständige Dokumentation oder genaue technische Details zu liefern, da dies den Rahmen des Kapitels bei weitem sprengen würde.

5.1 Technik

Dieses Kapitel beschäftigt sich mit der *Auswahl der Implementierungstechnik für die 3D-Visualisierung*. Hierbei spielen besonders die 3D-Darstellung und die spätere Zusammenarbeit mit weiteren Programmen eine wichtige Rolle für die erste Auswahl von Techniken, aber auch für die letztendliche Entscheidung.
In Bezug auf die benötigten 3D-Grafikfeatures muss betrachtet werden welche Funktionalitäten für die Umsetzung des Endkonzepts aus Kapitel 4.4 benötigt werden. Zur Darstellung eines 3D-Fahrzeugmodells muss dieses selbstverständlich zuerst in den Speicher geladen und dargestellt werden können, daher ist die *Unterstützung häufig verwendeter 3D-Grafikformate* ein wichtiger Punkt bei der Entscheidung für eine Technik. Ein weiterer Punkt ist die *übersichtliche Erstellung und Verwaltung der vorhandenen 3D-Szene*. Ein Fahrzeug ist ein komplexes 3D-Modell mit vielen Bauteilen und von daher ist es zwingend nötig, dass diese Bauteile auf einfache Weise verwaltet werden können. Ein dritter Punkt im Bezug auf die Verwendung von 3D-Grafik ist die *Performance* der gewählten Technik. Durch die komplexe Geometrie eines Fahrzeugmodells werden erhöhte Anforderungen an die Performance der verwendeten Technik gestellt, so dass dieser Punkt ebenfalls bei der Entscheidung für eine Implementierungstechnik berücksichtigt werden muss.
Durch die Festlegung auf die Zusammenarbeit der 3D-Ansicht mit anderen Programmen im Rahmen eines MCV-Systems, muss die genutzte Technik mit anderen Anwendungen kommunizieren können. Dabei soll es keine proprietären Schnittstellen geben, sondern eine möglichst einfache Verbindung aufgebaut werden, um spätere Veränderungen zu vereinfachen. Diese Kommunikation könnte also beispielsweise über selbstdefinierte Java-Klassen laufen, wenn alle beteiligten Programme mit diesen umgehen können oder auch über eine Socket-Verbindung über die entsprechende Daten, wie z. B. Strings, ausgetauscht werden. Die folgenden Unterkapitel stellen alle betrachteten Alternativen vor und bewerten diese. Aufgrund dieser Bewertungen steht am Ende dieses Abschnitts die Entscheidung für eine der Alternativen.

5.1.1 VirTools

VirTools [54] ist eine Authoring-Umgebung für 3D-Visualisierungen (siehe Abbildung 5.2) der Firma Dassault Systèmes. Dabei stehen vor allem die Verknüpfung, Logik und das Verhalten von 3D-Objekten in einer Szene im Vordergrund, denn VirTools selbst bietet keine Möglichkeit 3D-Modelle zu erstellen. Lediglich einfache Komponenten und Objekte, wie Kameras oder Lichtquellen können in die Szene eingefügt werden. Darüber hinaus können noch Objekteigenschaften, wie das Material, verändert werden. VirTools selbst ist sehr einfach zu bedienen und erinnert durch die

einfache Erstellung einer 3D-Szene an die Programmierung von Flash-Visualisierungen mit Hilfe von Adobe Flash [33], welches ebenfalls sehr einfach zu bedienen ist.
VirTools baut die 3D-Szene anhand eines Szenegraphen (siehe Abbildung 5.1) auf und bietet dadurch eine sehr übersichtliche Strukturierung auch von komplexen Szenen. Ein Szenegraph ist ein gerichteter azyklischer Graph, der eine 3D-Umgebung charakterisiert, indem er sämtliche Objekte hierarchisch anordnet. Er hat genau einen Wurzelknoten und jeder weitere Knoten kann beliebig viele Kindknoten haben - jeder Knoten aber maximal einen Elternknoten. Dabei haben Transformationen des Elternknotens auch immer Auswirkungen auf die jeweiligen Kindknoten. Würde man im sehr simplen Beispielgraph in Abbildung 5.1 beispielsweise die Tür rotieren, so werden auch die Türkarosserie und das Fenster mitrotiert. Wird jedoch das Fenster verschoben, hat dies keinerlei Auswirkungen auf eines der sonstigen Objekte im Graph. Dieses Verfahren bietet also eine gute Möglichkeit komplexe Szenen logisch aufzubauen und dabei Eltern-Kind-Beziehungen zwischen einzelnen Objekten auszunutzen.

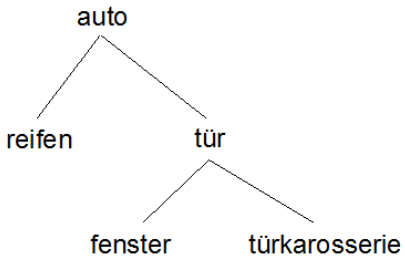


Abbildung 5.1: Beispiel für einen sehr einfachen Szenegraph.

VirTools bietet außerdem mit den sog. BuildingBlocks eine sehr einfache Möglichkeit, die Logik einer 3D-Szene zu erstellen. Dabei wird die komplette Logik einer 3D-Szene durch diese Blöcke zusammengesetzt (Abbildung 5.2 (b) und die jeweiligen Verknüpfungen der Blöcke untereinander entscheiden über die genauen Auswirkungen. So ist es innerhalb kürzester Zeit möglich, eine 3D-Welt zu erstellen, ein Objekt darin zu platzieren und diesem Objekt eine Schwerkraft-Logik zuzuweisen. Wird nun die 3D-Szene gestartet, fällt das Objekt physikalisch korrekt zu Boden, ohne diesen zu durchschlagen.

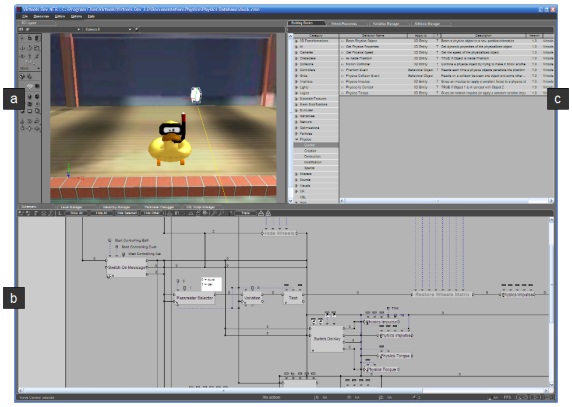


Abbildung 5.2: Benutzeroberfläche von VirTools - (a) 3D-Vorschaufenster, (b) Verknüpfung unterschiedlicher BuildingBlocks, (c) Bibliothek vordefinierter BuildingBlocks.

Dieses Verhalten ist mit wenigen BuildingBlocks zu realisieren und hält so die Komplexität einer Szene relativ gering. Außerdem bringt VirTools bereits eine Vielzahl an vordefinierten Building-Blocks mit, so dass man ggf. kaum eigene Blöcke entwickeln muss. Aufgrund der Tatsache, dass ein vollständiges mechanisches Fahrzeug-Modell relativ komplexe Geometrien aufweist, ist so eine einfache Strukturierung der Logik für die Tool-Entwicklung nützlich.
Betrachtet man sowohl die Organisation der Szene in einem Graph als auch die Verwaltung der Logik durch BuildingBlocks wird schnell klar, dass VirTools eine High-Level Lösung zur Generierung von 3D-Visualisierungen ist. Der Entwickler der 3D-Szene muss sich auch nicht um tiefgreifende Probleme mit der 3D-Engine kümmern, sondern kann sich völlig auf den Aufbau und die Logik der 3D-Welt konzentrieren. Dies macht VirTools zum einen sehr einfach zu handhaben, andererseits liegt hier auch ein großes Problem. Der Entwickler hat zwar Zugriff auf viele vorgefertigte Funktionen, jedoch hat er keinen Einfluss auf deren Implementierung bzw. Umsetzung. Sollte es zu Fehlern oder Problemen kommen, ist es nicht ohne Probleme möglich, die Fehlerquelle selbst zu beheben - man ist auf den entsprechenden Support angewiesen.
Visualisierungen, die mit VirTools erstellt wurden, können über eine Socket-Verbindung mit anderen Programmen in Verbindung treten und Daten austauschen. Die Anforderung eine VorTools-Visualisierung in ein MCV-System zu integrieren wird also erfüllt.

5.1.2 Java

Entgegen weitläufiger Vorurteile, *Java* sei zu langsam für aufwändige 3D-Umgebungen, wie z. B. Spiele, führt Andrew Davison [4] mehrere Punkte auf, die dieser Meinung widersprechen. So führt er gegen die angeblichen Performance-Probleme als Argumente auf, dass Java SE 6 bis zu 25% schneller sei als J2SE 5 und dass auch das Rendern, das mittlerweile via openGL und DirectX funktioniert, Geschwindigkeitsvorteile bringt. Als weiteren Vorteil wertet er die objektorientierte Programmierung mit Java, da sie Vorteile bei der Spiele-Entwicklung bedeute.
Es gibt für Java unterschiedliche Ansätze zur Realisierung einer 3D-Umgebung. Die erste Möglichkeit ist Java3D, eine Bibliothek mit der das Erzeugen und Verwalten von 3D-Szenen möglich ist. Ebenso wie VirTools bietet Java3D einen Szenenaufbau via Szenegraph (siehe Abbildung 5.1).

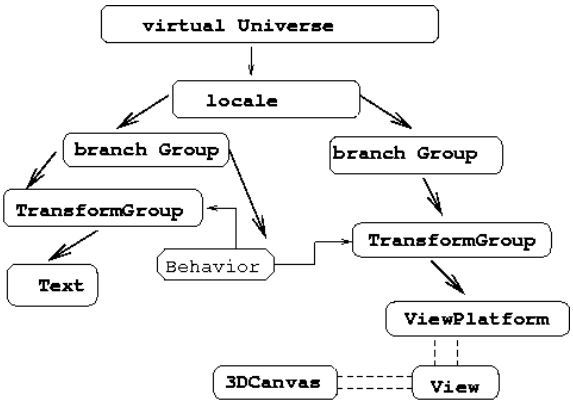


Abbildung 5.3: Schematischer Aufbau eines Programms in Java3D, in der linken Seite des Baums befinden sich virtuelle Objekte, die rechte Seite des Baums dient der Verwaltung der Kamera [42].

Allerdings ist die gesamte Entwicklung der Bibliotheken momentan eher schleppend, da das Projekt 2004 von Sun Microsystems als Open Source Projekt zur Verfügung gestellt wurde und die Entwicklung somit komplett durch die Community erfolgt. Dennoch gibt es im Internet diverse Seiten, die sich - wenn auch veraltet - mit dem Thema auseinandersetzen und einige Erweiterungen, wie z. B. Loader-Klassen für viele 3D-Formate [43], anbieten. Betrachtet man die reine

Performance von Java3D, so ist anzumerken, dass zwar diverse Grafikbefehle direkt auf der vom Betriebssystem unterstützten Hardware laufen, aber ebenso sind viele neuere Befehle und Erweiterungen noch nicht umgesetzt und implementiert. Da über Java3D kein direkter Zugriff auf die Videohardware möglich ist, können diese Features also nicht genutzt werden. Dem gegenüber steht der Vorteil eines gut strukturierten Aufbaus (siehe Abbildung 5.3), der durch die Verwendung eines 3DCanvas auch die einfache Einbindung von Java3D-Programmen in eine Applet-Umgebung ermöglicht.
Die zweite Möglichkeit Java zu 3D-Visualisierung zu nutzen stellen sog. OpenGL-Wrapper dar - ein Vertreter hiervon ist JOGL [46]. Diese Wrapper sorgen für eine direkte Umsetzung von Java Befehlen in die entsprechenden OpenGL-Befehle und machen somit OpenGL-Anweisungen unter Java zugänglich. Allerdings fällt bei dieser Variante die einfache Verwaltung der 3D-Welt via Szenegraph weg. Dafür ist die Performance im Normalfall höher als bei Java3D, da auch aktuellere Möglichkeiten der Grafikhardware angesprochen werden können und es sind diverse mächtige OpenGL-Kommandos verfügbar.
Es ist also eine Art Trade-Off zwischen Java3D und einem Wrapper, wie JOGL, erkennbar: entweder kann man eine komfortable Verwaltung der Szene mit einem Graph in Java3D nutzen und muss dafür gewisse Performance-Probleme in Kauf nehmen oder man nutzt eine effektive Umleitung der Befehle auf OpenGL, wie JOGL sie anbietet, und verzichtet aber auf den Szenegraph.

5.1.3 JMonkeyEngine

Die *JME (JMonkeyEngine)* [45] ist eine 3D-Grafikengine, die auf Java aufbaut und zuerst 2003 von Mark Powell entwickelt wurde [32]. Powell arbeitete damals im Bereich OpenGL Rendering als er auf LWJGL [47], eine Games Library für Java mit OpenGL-Unterstützung, stieß und Java als optimale Möglichkeit für die Entwicklung eigener Grafik-Tools ausmachte. Er entwickelte die ersten Versionen von JME, die mittlerweile durch eine Vielzahl von Entwicklern stetig weiterentwickelt und verbessert wird. Bei JME handelt sich um eine OpenSource 3D-Engine, die mittlerweile in 2 Versionen erhältlich (jme und jme2) ist, wobei der Unterschied vor allem neue Funktionen und Möglichkeiten bei jme2 ausmachen. Das grundlegende Gerüst der Engine ist größtenteils identisch oder sehr ähnlich. Ähnlich wie Java3D wird bei der JME ein Szenegraph zur Organisation der 3D-Szene genutzt und bietet somit auch dieselben Vorteile, wie zuvor in Kapitel 5.1.2 beschrieben. Allerdings liegt der große Unterschied zu Java3D in der Tatsache, dass die Syntax zwar stark an Java angelehnt ist, jedoch der Unterbau der Engine auf JOGL bzw. LWJGL [47] beruht. Es werden also die Vorzüge eines Szenegraphen mit der Performance eines OpenGL-Wrappers verbunden. Ein anderer Vorteil ist die immense Fülle bereits implementierter Funktionen, die dem Entwickler zur Verfügung gestellt werden. Es existieren verschiedene Grundgerüste, um ein Programm zu entwickeln. Diese Varianten fangen bei einem sehr simplen Entwurf an, der dem Entwickler viele Einstellungen der 3D-Szene abnimmt, bis hin zur Möglichkeit jeden Parameter der Engine von Hand einzustellen.

Abbildung 5.4: Einige Beispiele von Spielen, die mit der JME entwickelt wurden [45].

Abgesehen von den technischen Eigenschaften bietet die JME außerdem noch eine aktive Community im Internet, die die Engine ständig weiterentwickelt - ein Zeichen dafür ist nicht zuletzt die Downloadmöglichkeit direkt aus dem Entwickler SVN-Repository. Durch die große Zahl an Nutzern sind auch bereits eine Vielzahl an Wikis und Tutorials geschrieben worden, die den Einstieg in die Engine vereinfachen. Darüberhinaus gibt es sehr viele Testklassen, die die Benutzung von Methoden und Klassen an einigen einfachen Beispielen zeigen. Teil dieser Test-Klassen ist ein kleines Spiel, welches in mehreren Etappen bzw. Entwicklungstufen in unterschiedlichen Paketen vorliegt und von Stufe zu Stufe komplexer wird. An diesem Beispiel lassen sie somit viele Techniken an einem konkreten Projekt betrachten.
Der weitere Support und die Aktualität ist also im Vergleich zu Java3D sehr groß, jedoch ist die Engine aufgrund der Vielzahl an Möglichkeiten auch weitaus komplexer. Diese Menge an Funktionen bei gleichzeitig hoher Performance ermöglichen es, dass selbst umfangreiche Projekte, wie größere Spiele, von Community-Mitgliedern entwickelt werden. Dass die JME dabei auch in der Lage ist, ansprechende Grafik bei guter Performance darzustellen, zeigen die Screenshots in Abbildung 5.4.
Die JMonkeyEngine bietet also einge große Auswahl an Möglichkeiten und Funktionen, die dem Entwickler an die Hand gegeben werden, jedoch muss sich dieser auch entsprechend in die Konzepte der Engine einarbeiten. Gerade im Bereich der unterschiedlichen Ausgangsplattformen, die, wie beschrieben, dem Entwickler verschiedene Einstellungen von vorneherein abnehmen können, kann es am Anfang zu Verwirrungen kommen. Davon abgesehen ist gerade der gute Support, auch durch die Entwickler, im JME-Forum ein wirklicher Vorteil dieser Lösung. Durch die Vielzahl an Benutzern gibt es fast kein Problem, das nicht schon zuvor von einem anderen Entwickler gelöst wurde.

5.1.4 OpenGL

OpenGL (Open Graphics Library) [48] ist eine plattform- und programmiersprachenunabhängige Schnittstelle zur Entwicklung und Programmierung von 3D-Grafik. Die OpenGL-Befehle greifen direkt auf Bibliotheken des Betriebssystems oder Teile des Grafikkarten-Treibers zurück. Durch diese hardwarenahe Realisierung erreicht OpenGL eine sehr gute Performance und ist aufgrund der Verwaltung durch die OpenGL ARB [49], ein Konsortium von Firmen, die im Bereich der 3D-Computergrafik tätig sind, eine relativ gut entwickelte und unterstützte Schnittstelle.
Doch die angesprochene low-level Implementierung von OpenGL hat das Problem, dass Organisationstrukturen, wie ein Szenegraph, vollständig fehlen. Somit muss der Entwickler selbstständig dafür sorgen, dass die Inhalte der 3D-Szene so verwaltet werden, dass er einen reibungslosen Zugriff darauf hat.
Da das endgültige Konzept den Einsatz der 3D-Ansicht in einem MCV-System vorsieht und die Anwendungen, die den anderen Teil des MCV-Systems stellen, unter Windows laufen, käme für die OpenGL-Programmierung als Sprachen zum Beispiel C oder C++ in Frage. Zwar wäre eine Kommunikation über Sockets auch zwischen Rechnern mit unterschiedlichen Betriebssystemen möglich, doch das angestrebte MCV-System soll auf einem einzigen Rechner und ohne größeren Aufwand (z. B. virtuelle Maschine) funktionieren.
Insgesamt bietet die Nutzung von OpenGL-Bibliotheken in Verbindung mit C oder einer vergleichbaren Sprache auch bei komplexen 3D-Szenen eine sehr gute Performance. Allerdings mangelt es an komfortablen Verwaltungsstrukturen für Objekte innerhalb der 3D-Szene.

5.1.5 Entscheidung für eine Technik

Zu Beginn der Implementierungsphase wurde entschieden, Java bzw. Java3D als Programmiersprache zur Erstellung der 3D-Visualisierung zu nutzen. Dazu hat vor allem beigetragen, dass Java3D eine solide Schnittstelle bietet und offensichtlich für die Anforderungen dieser Arbeit aus-

reichend ist. Es sind Loader-Klassen für die verbreitetsten 3D-Formate vorhanden und aufgrund des Szenegraphs sind auch komplexere Geometrieveränderungen an Modellen kein großes Problem.
Ein Grund, der gegen die Nutzung von VirTools sprach, waren die relativ hohen Anschaffungskosten von VirTools. Zwar wäre die Erstellung einer aufwändigen 3D-Szene mit Hilfe dieses Tools sehr einfach möglich gewesen, aber die sehr hohen Kosten einer Einzelplatzlizenz waren ein Grund, sich dagegen zu entscheiden. Ein anderer Grund war das angesprochene Problem, dass man auf interne Programmabläufe keinen direkten Einfluss hat. Diese Gründe gaben den Ausschlag, sich gegen VirTools zu entscheiden. Die Entscheidung gegen OpenGL in Verbindung mit C bzw. einen reinen Java OpenGL-Wrapper begründet sich vor allem in der fehlenden Szenegraph-Unterstützung. Hier mag zwar die Performance sehr gut sein, aber der Verwaltungs- und Programmieraufwand dürfte aufgrund der mangelnden Organisation innerhalb der 3D-Szene recht schnell sehr groß werden. JME wurde an dieser Stelle zunächst auch ausgeschlossen, da die Engine recht umfangreich ist und einige Einarbeitungszeit benötigt - als größter offensichtlicher Pluspunkt stand an dieser Stelle lediglich die Performance, die aber aufgrund des prototypischen Charakters des zu entwickelnden Programms vorerst zu vernachlässigen ist.
Somit wurde entschieden, die Arbeit in Java3D zu programmieren. Jedoch traten nach etwa zwei Wochen massive Probleme bei der Darstellung von überlagernden Transparenzen bei Java3D auf. Die Recherche im Internet ergab nach einiger Zeit, dass es sich hierbei wohl um ein zentrales Problem von Java3D handelt, dass seit längerem bekannt ist, aber aufgrund der mangelnden Aktualität der Software noch nicht entfernt wurde. Da dieses Problem - bei mehreren transparenten Objekten verschwanden einige Objekte einfach vollständig - nicht akzeptabel für den späteren Prototyp war, musste die Implementierung mit einer der anderen Möglichkeiten durchgeführt werden. Um nicht noch einmal das Problem veralteter Software zu haben, aber weiterhin einen Szenegraph nutzen zu können, wurde entschieden die Arbeit nun mit der JME zu programmieren. Dabei konnte zumindest das bereits bearbeitete 3D-Modell weiterverwendet werden, da JME - genauso wie Java3D - über eine Loader-Klasse für 3ds-Dateien (3ds max [31]) verfügt.

5.2 Implementierung

Dieses Kapitel beschreibt die Umsetzung der 3D-Visualisierung. Dabei werden sowohl der Gesamtentwurf des Programms als auch verschiedene Einzelheiten, wie die Bearbeitung des 3D-Modells genauer beschrieben. Selbstverständlich kann nicht auf alle Besonderheiten eingegangen werden, jedoch wird versucht, die wesentlichen Punkte der Programm-Struktur und des Programm-Ablaufs darzustellen und verständlich zu machen.

5.2.1 3D-Modell

Die Grundlage der implementierten 3D-Visualisierung mit mechanischem Umgebungsmodell bildet ein passendes 3D-Modell. Um Zeit zu sparen wurde nach bereits vorhandenen Modellen gesucht, die zur Verwendung einer 3D-Visualisierung geeignet schienen. Als Kriterien wurden dabei sowohl der Detailgrad (Interieur) als auch die Zerlegbarkeit in Einzelteile berücksichtigt. Erste Anlaufstelle hierbei war die Design-Abteilung von BMW, in der diverse 3D-Fahrzeugmodelle verwaltet und bearbeitet werden. Zwar konnten hier passende Modell gefunden werden, jedoch waren diese äußerst detailliert ausgearbeitet und bestanden aus einer Unmenge an Polygonen. Auf Anraten der dortigen Experten wurde davon Abstand genommen, ein solches Modell automatisch zu vereinfachen und herunterrechnen zu lassen, da die Ergebnisse so einer Aktion nicht vielversprechend seien. Es musste daher eine Alternative gesucht werden, die ein erstelltes Modell bietet, das einen hohen Detailgrad aufweist, d.h. dass z. B. das Interieur vorhanden ist.
An dieser Stelle wurde auf Google SketchUp [41] zurückgegriffen. Dies ist eine sehr einfach zu handhabende Software mit der man jedoch durchaus ansprechende 3D-Modelle und Szenen ge-

stalten kann. Die Software ist in ihrer Grundausstattung frei verfügbar, so dass es mittlerweile eine große Anzahl an Nutzern gibt, die mit dem Programm aufwändige 3D-Modelle erstellen. Über eine interne Programmfunktion können diese Modelle sehr einfach auf Google-Server hochgeladen werden und dort von jedem anderen betrachtet und weiterverarbeitet werden. In der sog. 3D-Galerie, über die auf die online verfügbaren Modelle zugegriffen werden kann, ist auch eine Vielzahl an Fahrzeug-Modellen verfügbar. Dort wurde schließlich ein modellierter BMW mit den passenden Spezifikationen gefunden - das Modell war sehr detailliert, aber die Datei war nur wenige Megabyte groß. Abbildung 5.6 zeigt das Modell auf der linken Seite in seiner Ausgangsform. Im Folgenden wurde dieses Modell in einzelne Bauteile zerlegt und diese jeweils ins 3ds-Format exportiert, um sie anschließend in 3ds Max wieder zusammenzufügen.

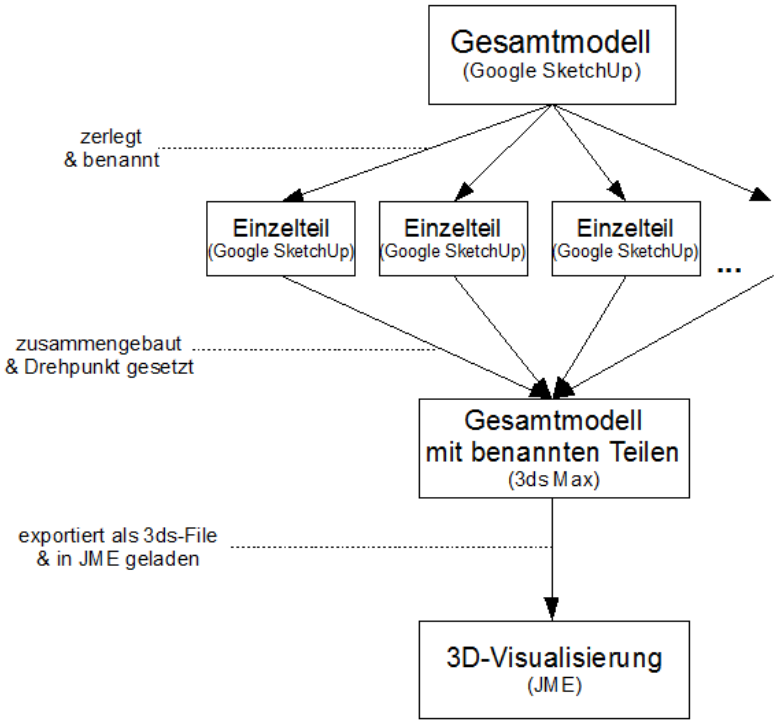


Abbildung 5.5: Vorgehensweise zur Erstellung des Modells für die Visualisierung.

Dieser Zwischenschritt war leider notwendig, da das Modell selbst in SketchUp nicht korrekt in Teile zerlegt werden konnte, da die jeweiligen Teilenamen bei der Konvertierung verloren gingen. Nachdem sämtliche Teile einzeln exportiert und wieder zusammengefügt wurden, bedurfte es einer passenden Nomenklatur, um jedes Teil individuell ansprechbar zu machen, also wurden die Teile gleichzeitig noch korrekt benannt und deren Pivot Punkte, also die Mittelpunkte einer Rotation des Körpers, wurden gesetzt. Ein Screenshot, der während der Arbeit am Modell entstanden ist, ist in Abbildung 5.6 auf der rechten Seite zu sehen.
Nach diesen Arbeitsschritten war es noch nötig, das gesamte Modell, das wieder zusammengesetzt wurde, als eine einzige 3ds-Datei zu speichern, für die die JME wiederum eine Loader-Klasse bereitstellt. Diese Schritte waren insgesamt eigentlich unproblematisch, allerdings sind beim Export von SketchUp zu 3ds Max einige unschöne Verzerrungen im Modell entstanden, die sich durch Dreiecke auf einigen Seitenflächen äußern. Trotz diverser Optimierungsversuche konnte dieses Phänomen nicht beseitigt werden.
Nach der Fertigstellung des Fahrzeugmodells wurden noch vier Bus-Systeme in das Modell integriert. Dabei handelt es sich um verschiedene CAN-Busse, zu denen jeweils auch Trace-Daten

einer Testfahrt vorliegen. Somit ist sichergestellt, dass die dargestellten Bus-Systeme später auch durch hinterlegte Daten korrekt zur Visualisierung genutzt werden können. Wie im Abschlusskonzept beschrieben, wurde das Bus-Netz teils realistisch, teils abstrakt eingebaut (siehe Abbildung 5.11). Die Steuergeräte haben zwar eine nahezu einheitliche Form, die einem Standard-Steuergerät ähnelt, aber sowohl Größe als auch Position sind zu Gunsten der Übersicht nicht völlig realistisch. Derselbe Weg wurde beim Kabelbaum gegangen: hier dienen einfache Formen als Kabel und sind möglichst realistisch, aber dennnoch übersichtlich im Fahrzeug verteilt und bilden so das virtuelle Bordnetz.

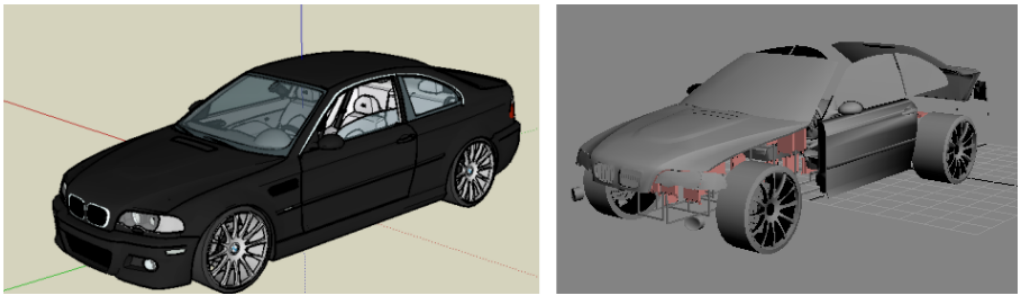


Abbildung 5.6: (a) Ursprungsmodell in Google SketchUp[41], (b) Screenshot während der Modellbearbeitung in 3ds Max[31].

Im Endeffekt steht ein vollständiges Modell zur Verfügung, dessen Einzelteile über eine klare Benennung separat ansprechbar sind. Diese Modell kann nun im weiteren Verlauf der Arbeit zu Visualisierungszwecken genutzt werden. Dabei sind sowohl die mechanischen Komponenten als auch die Bordnetz-Elemente über Namen ansprechbar, korrekt positioniert und bei den entsprechenden mechanischen Teilen sind die Drehpunkte zur Rotation festgelegt. Als weiterer Vorteil dieser Vorgehensweise steht die bereits angesprochene Austauschbarkeit des Modells. Solange die Namen der Bauteile übereinstimmen kann jedes beliebige andere Fahrzeug in die 3D-View eingebunden werden.

5.2.2 Programmstruktur

Das Programm wurde basierend auf einem simplen Schichtenprinzip entworfen, um die einzelnen Schichten austauschbar bzw. sehr einfach wartbar zu machen, ohne das gesamte Projekt überarbeiten zu müssen. In diesem Abschnitt werden die einzelnen Schichten vorgestellt, ihre Bedeutung erklärt und die einzelnen Methoden, die bereitgestellt werden, näher umschrieben.
Aus den festgelegten Anforderungen an das Programm ergibt sich die erste und oberste Ebene von allein - das Interface. Das Interface bietet nach außen die Methoden an, die andere Programme benötigen, um die 3D-Visualisierung zu steuern. Um dies zu erreichen, kommuniziert die Schnittstelle mit der nächsttieferen Schicht und übersetzt dabei die Befehle, wie „Öffne die Tür vorne links“, in ein Format, das die darunter liegende Schicht umsetzen kann. Das Interface bietet also alle Funktionen der Visualisierung auf einer hohen Abstraktionsstufe an. Wenn Programme darauf zugreifen, müssen sie sich nicht um Details wie Rotation etc. kümmern, sondern können direkt die entsprechenden Methoden, z. B. (`openDoorFL()`) aufrufen. Durch die einfache Implementierung der Methoden in der obersten Schicht ist es problemlos möglich, die Anzahl der Methoden nachträglich zu erhöhen, ohne die interne Funktionsweisen der darunterliegenden Methoden verstehen zu müssen. Stattdessen werden in der nächsten Schicht, dem ViewManager, Methoden, wie z. B. `rotate()` mit verschiedenen Parametern, aufgerufen.

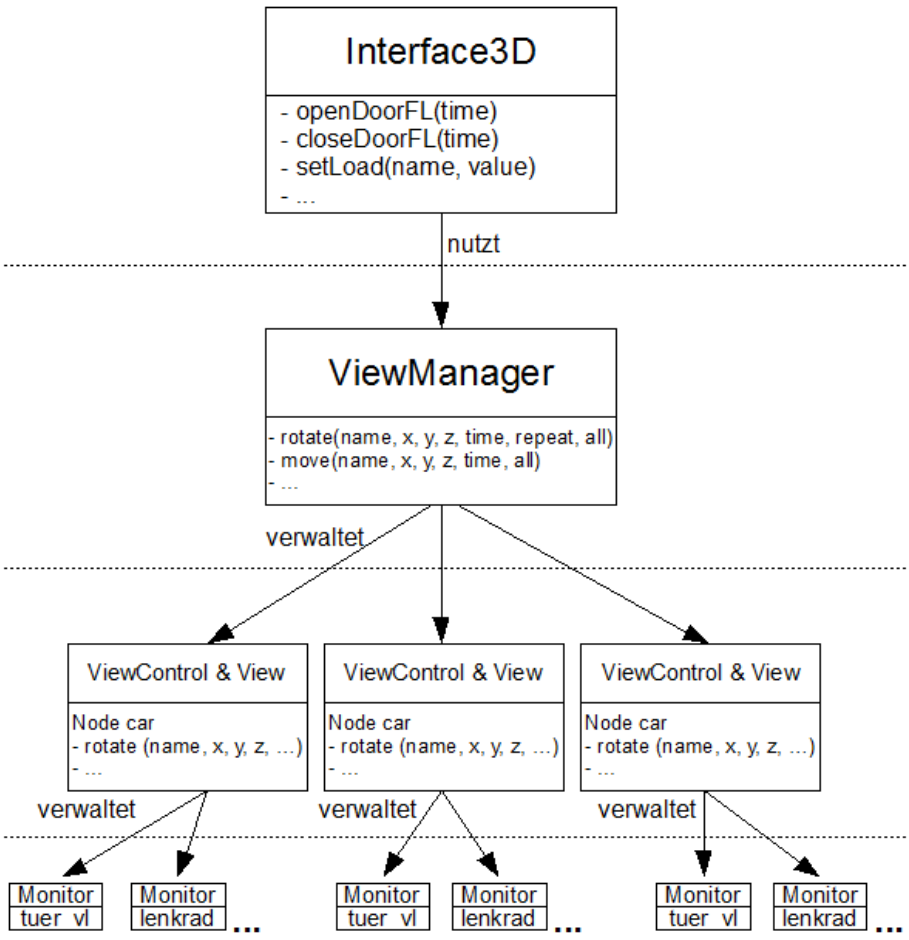


Abbildung 5.7: Programmstruktur des 3D-Visualisierungstools.

Der ViewManager, die nächste Schicht unterhalb des Interfaces, stellt die Verwaltung der 3D-Ansichten dar. Er bildet außerdem die zentrale Komponente der Anwendung, verwaltet zusätzlich zu den Views auch Grundeinstellungen der 3D-Engine (Framerate, Auflösung, Farbtiefe) und generiert die Beleuchtung der 3D-Szene. Der ViewManager ist dafür zuständig, einzelne Anweisungen, die vom Interface an ihn weitergegeben werden, auf die entsprechenden Ansichten zu verteilen bzw. die Anweisungen weiterzuleiten. Dabei kann unterschieden werden, ob sich die gemachte Anweisung (z. B. Einfärben einer Komponenten) auf alle existierenden 3D-Ansichten bezieht oder lediglich auf die Ansicht, die aktuell im Fokus des Anwenders ist. An dieser Stelle wird auch eine Art State-Machine verwaltet, um stets den aktuellen Fahrzeugzustand verfügbar zu haben. Sollte im Programmablauf eine neue Sicht benötigt werden, so kann durch die State-Machine garantiert werden, dass sich das Fahrzeugmodell in der neuen Ansicht im korrekten Status befindet und von da an alle Veränderungen am Status korrekt abbildet. Unabhängig davon, ob es eine oder mehrere Ansichten gibt, werden Anweisungen, die an den Manager übergeben werden, in die untere Schicht der Programmarchitektur, also zur View-Ebene, weitergeleitet.
Auf View-Ebene existieren zwei wichtige Klassen - die View-Klasse und der dazugehörige View-Controller. Die Anweisungen vom ViewManager treffen beim ViewController ein und dieser setzt sie letztendlich für die entsprechenden Objekte um. Die Objekte bzw. das Gesamtmodell werden in der View-Klasse, in der sich auch andere zentrale Variablen (Kameraposition, Blickrichtung, 2D-Anzeigen) jeder einzelnen 3D-View befinden, verwaltet.

Die letzte Schicht in der Programmstruktur stellen die Monitore der einzelnen Bauteile des Modells dar. Jedes Element hat einen eigenen Monitor, der alle relevanten Parameter zu diesem Objekt speichert. So werden hier sowohl die bereits vollzogenenen Rotationswinkel, als auch die Strecke, die sich ein Objekt bewegt hat, sowie diverse Parameter zur Farbgebung und Transparenz gehalten und abgespeichert. Zusätzlich werden z. B. auch die Standard-Parameter wie Farbe, Transparenz und andere Faktoren gespeichert, um im weiteren Verlauf darauf zurückgreifen zu können.
Die gesamte Programmstruktur lässt sich am ehestem mit einer Baumstruktur vergleichen (siehe Abbildung 5.7), deren Wurzel den Zugriff auf die darunterliegenden Schichten ermöglicht. Die Wurzel ist in diesem Fall das Interface, über das Befehle an die 3D-Ansichten gegeben werden können. Diese Befehle werden im ViewManager analysiert und an die unterschiedlichen Views verbreitet. Innerhalb der Views gibt es wiederum eine Vielzahl von `Monitor`-Objekten, die die Eigenschaften jedes einzelnen Elements speichern. Diese `Monitor`-Objekte bilden die Grundlage des Programms, da sie sämtliche verfügbaren Aktionen unterstützen und überhaupt erst ermöglichen.

5.2.3 Programmablauf

Dieses Kapitel beschreibt den allgemeinen Programmablauf, wobei die Startphase des Programms lediglich kurz angeschnitten wird - der größte Teil des Kapitels beschreibt die Umsetzung von Anweisungen innerhalb des Programms. In den folgenden Kapiteln 5.2.4 bis 5.2.6 werden dann noch einige Spezialfälle genauer beschrieben, wie die Erstellung und Verwaltung von zusätzlichen 3D-Ansichten oder die Anbindung an ein vorhandenes Programm, welches dann die Steuerung der 3D-Sicht übernimmt.
Wie schon in Kapitel 5.2.2 deutlich wurde, wird ein einzelner Methodenaufruf durch unterschiedliche Schichten geführt, bis er dann lettztendlich entsprechend umgesetzt wird. Der Start jeder Aktion ist der Aufruf einer Methode in der Interface-Klasse des 3D-Clients. Jede dieser Methode wurde zuvor entsprechend der gewünschten Aktion implementiert - so ruft die Methode `openDoorFL()` in der nächsten Schicht eine Methode auf, die die Tür vorne links (`tuer_vl_gr`) um einen festdefinierten Winkel in einer angegebenen Zeit rotiert. Für einen einfachen Zugriff auf die einzelnen Modellteile und weil bei der Benutzung der Engine nur 10 Zeichen zur Benennung von Teilen verfügbar waren, wurden diese nach einem festen Schema benannt. Dieses Schema kennzeichnet durch die Benennung `name_position` mit dem Zusatz `_gr`, wenn es sich um eine Gruppe von mehreren einzelnen Elementen handelt, jedes Teil des Fahrzeugs eindeutig. Abgesehen von dieser einfach Beispiel-Methode sind auch komplexere Vorgänge, wie Lenken oder das Herunterfahren eines Fensters, ohne große Probleme zu implementieren - sie müssen lediglich in einzelne Teilvorgänge zerlegt werden. In der obersten Schicht lassen sich also nach dem Baukasten-Prinzip beliebig komplexe Aktionen durch eine Ansammlung von mehreren Einzelaktionen beschreiben. So besteht die Aktion „lenken“, sowohl aus Rotationsaufrufen für die beiden Achsobjekte an der Fahrzeugfront als auch einem Rotationsaufruf für das Lenkrad.
Zur genaueren Beschreibung des weiteren Programmablaufs wird jedoch nun die Methode `openDoorFL` ausgewählt, da diese auch im späteren Verlauf direkten Nutzen aus der Verwaltung mit einem Szenegraph zieht und so die Vorteile der gewählten Vorgehensweise besser veranschaulichen kann. In diesem Beispiel soll davon ausgegangen werden, dass eine externe Anwendung einen 3D-Client erstellt hat. Durch die Erstellung des Clients erhält die Anwendung das Interface, auf welchem wiederum die unterschiedlichen Aktionen, wie bereits beschrieben, implementiert sind. Zur besseren Veranschaulichung soll angenommen werden, die externe Anwendung habe `openDoorFL` im Interface aufgerufen. Dieser Methodenaufruf benötigt verschiedene Parameter, wie den Winkel um den die Tür geöffnet werde soll und die Zeit, die diese Animation dauern soll. Um dies zu erreichen sähe der Aufruf auf dem Interface bespielsweise so aus: `openDoorFL(2000, 60)`. Dabei gibt der erste Parameter die Dauer der Aktion in Millisekunden an - hier also 2000ms - und der zweite Parameter den Winkel, um den die Tür geöffnet werden soll - hier also 60°. Da

von einer Methode durch Überladen auch verschiedene Versionen angeboten werden können, sind auch Aufrufe wie `openDoorFL(2000)` denkbar. Bei diesem Aufruf wird versucht die Tür in zwei Sekunden um einen festdefinierten Winkel geöffnet. Daraufhin wird vom Interface anhand der State-Machine im ViewManager geprüft, ob die Tür bereits geöffnet ist oder nicht. Wenn sie noch nicht geöffnet wurde, wird eine Rotation von `tuer_vl_gr`, der Tür vorne links, veranlasst. Zu diesem Zweck wird folgende Methode auf dem ViewManager aufgerufen, wobei `anlge=60` und `time=2000` ist:

```
viewManager.rotate("tuer_vl_gr", 0, 0, -angle, time, false, true);
```

Hier sieht man sehr gut, dass das Öffnen der Tür durch eine einfache Rotation dargestellt wird. Der erste Parameter gibt dabei den Namen des Bauteils an. Die nächsten drei Werte sind die jeweiligen Grad-Angaben pro Achse, d.h. der erste Wert ist der Rotationswinkel der X-Achse, der zweite Wert für die Y-Achse und der letzte Wert (hier `-angle`) für die Z-Achse. Im Anschluss wird die Aktionszeit übergeben, gefolgt von zwei Boolschen Werten. Dabei gibt der erste Wert an, ob es sich bei der Rotation um eine wiederholende Rotation handelt (z. B. drehende Räder) und der zweite Wert bestimmt, ob die Aktion bei allen aktiven Views angezeigt wird. Nach dem erfolgten Aufruf der Methode im ViewManager wird noch der State-Machine mitgeteilt, dass die Tür nun geöffnet ist. Dadurch wird verhindert, dass eine weitere Öffnung der Tür stattfinden kann. Außerdem ist nun für den Fall, dass eine weitere Sicht erstellt wird (siehe Kapitel 5.2.5) gewährleistet, dass auch dort die Tür vorne links um den entsprechenden Winkel geöffnet dargestellt wird.
Sobald im ViewManager die entsprechende Methode aufgerufen wird, wird zuerst geprüft, ob die anstehende Aktion für alle oder nur für die aktive View, die im Fokus ist, ausgeführt werden soll. Der weitere Verlauf unterscheidet sich dann darin, dass die Weiterleitung des Aufrufs entweder an alle vorhandenen ViewController oder eben nur an den ViewController der aktiven Sicht weitergeleitet wird.

```
for (int i = 0; i < viewControlVec.size(); i++)
{
   viewControlVec.elementAt(i).rotate(name, angleX, angleY,angleZ, time, repeat);
}
```

Es wird also entweder der entsprechende Vektor mit allen ViewControllern durchlaufen und der Aufruf weitergegeben oder nur der aktive ViewController, dessen Position im Contoller-Vector gespeichert wurde, wird per Methodenaufruf über die Aktion informiert. Die übergebenen Parameter sind in beiden Fällen nahezu identisch mit denen aus dem ViewManager-Methodenaufruf, lediglich der letzte boolean-Wert bzgl. der beteiligten Views fällt weg.
Im jeweiligen ViewController wird der rotate-Aufruf nun letztendlich in eine Rotation umgesetzt bzw. vielmehr werden die Rotationsdaten im entsprechenden Monitor-Objekt des Bauteils eingetragen. Dazu wird zuerst geprüft, ob das entsprechende Bauteil überhaupt vorhanden ist, um eine NullPointerException zu verhindern, falls ein nicht-vorhandenes Teil rotiert werden soll. Im Anschluss daran wird geprüft, ob das Teil bereits ein Monitor-Objekt besitzt, wenn nicht wird ein Monitor erstellt und dem Objekt zugewiesen. Unabhängig davon, ob bereits ein Monitor vorhanden war oder nicht, wird nocheinmal geprüft, ob das Objekt aktuell schon eine Rotation ausführt bzw. rotiert wird oder nicht. Falls noch keine Rotation stattfindet, wird zuerst ein Rotations-Flag auf `true` gesetzt und im Anschluss daran werden die verschiedenen Parameter im Monitor gespeichert. Dazu gehören bei der Rotation die Winkel, die Dauer der Rotation, der Startzeitpunkt der Aktion und ob es sich um eine wiederholende Rotation handelt.

```
monitor.setRotation(true);

monitor.setTargetRotX(angleX);
monitor.setTargetRotY(angleY);
monitor.setTargetRotZ(angleZ);

monitor.setRotTime(time);
monitor.setRotStart(System.currentTimeMillis());

monitor.setRepeatRot(repeat);
```

Natürlich werden allein durch das Eintragen der Werte noch keine Rotationen ausgeführt. Hierfür ist die `update()`-Methode verantwortlich, die von der 3D-Engine bei jeder Neuberechnung eines Bildes aufgerufen wird. Beim vorliegenden Programm wurde die Framerate auf 15 Bilder pro Sekunde festgelegt, um nicht zu viel Rechenleistung zu beanspruchen. Innerhalb dieser Methode werden alle Elemente des Modells überprüft, ob sie bereits ein Monitor-Objekt besitzen. Ist dies der Fall, so werden die Monitor-Objekte auf alle verfügbaren Aktionen überprüft - dazu zählen Rotation, Verschiebungen, Einfärbungen und Veränderungen der Transparenz. Wird dabei festgestellt, dass ein Objekt - in unserem Beispiel die Tür vorne links - rotiert werden soll, wird aufgrund des hinterlegten Startzeitpunkts, der aktuellen Systemzeit und der Rotationsdauer der Winkel bestimmt mit dem das Objekt zu diesem Zeitpunkt rotiert sein müsste, um passend zum berechneten Rotations- bzw. Animationsende den Zielort bzw. den Zielwinkel zu erreichen. Anschließend wird das Objekt um den entsprechenden Winkel rotiert und die Veränderung im Monitor-Objekt eingetragen, so dass beim nächsten Durchlauf der aktuelle Zustand des Elements korrekt ausgelesen werden kann.
Vor der jeweiligen Teilanimation wird geprüft, ob die Animationszeit bereits abgelaufen ist. Ist dies der Fall wird keine Teilanimation durchgeführt, sondern das Objekt anhand im Monitor hinterlegter Daten zum entsprechenden Zielpunkt bewegt. Im Anschluss daran wird die zugehörige boolean-Flag auf `false` zurückgesetzt, um erneute Änderungen in diesem Bereich zuzulassen.
Denn sollte während einer laufenden Veränderung eine Aktion desselben Typs auf dem Objekt aufgerufen werden, wird diese zunächst nicht ausgeführt. Stattdessen wird ein Objekt vom Typ Event erstellt, welches alle Einzelheiten der Aktion speichert. In jedem weiteren Durchlauf von `update()` wird also nun geprüft, ob das Event bzw. alle gespeicherten Events abgearbeitet werden können oder nicht. Im konkreten Beispiel würde etwa die Methode `closeDoorFL()` aufgerufen, während die Tür sich noch öffnet.

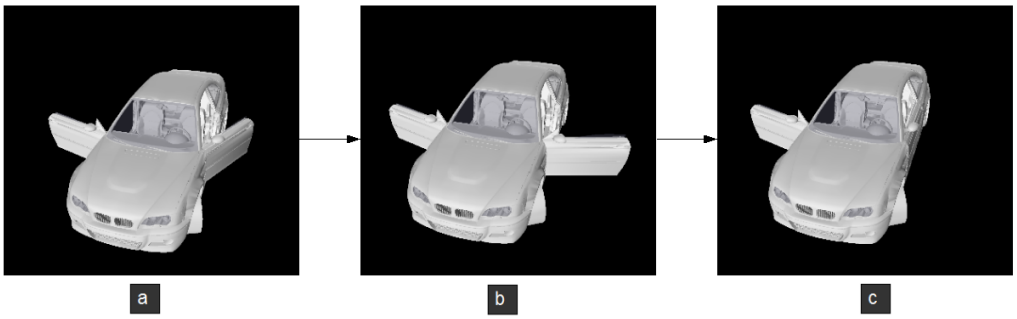


Abbildung 5.8: (a) Rotation zur Öffnung der tür läuft, `closeDoofFL()` wird aufgerufen, (b) Öffnen der Tür abgeschlossen, (c) Schließen der Tür eigentlich auch abgeschlossen -> Tür springt sofort in Endzustand

Der Aufruf wird dann entsprechend durch die Schichten weitergeleitet, bis auffällt, dass noch eine Rotation läuft (Abbildung 5.8 (a)). Es wird dann ein entsprechendes Event erstellt und abgespeichert. Sobald das Öffnen der Tür abgeschlossen und die Rotations-Flag auf `false` gesetzt wird (Abbildung 5.8 (b)), kann die neue Rotation begonnen werden. Allerdings mit der Besonderheit, dass der Startzeitpunkt der Rotation im Event gespeichert ist. Die Tür wird aufgrund des gespeicherten Startzeitpunkts zu der Position bewegt, in der sie sich befinden würde, wenn keine andere Aktion beim Methoden-Aufruf aktiv gewesen wäre. Im Anschluss daran läuft die Rotation normal zu Ende, falls noch Restlaufzeit vorhanden ist, ansonsten springt das Element direkt zur Zielposition (5.8 (c)) und die Rotations-Flag wird wieder auf `false` gesetzt.
Anhand des aufgeführten Beispiels wurde der Programmablauf bei einer Rotation vorgestellt. Der Ablauf bei einer Verschiebung, Einfärbung oder einer Transparenzänderung gleicht dem hier vorgestellten Ablauf in hohem Grad. Selbstverständlich müssen zum Teil andere Parameter übergeben werden, wie z. B. Entferungen, Farben oder Transparenzwerte, jedoch ist die Umsetzung der einzelnen Aufrufe und die Abarbeitung anstehender Änderungen über die `update()`-Methode immer dieselbe. Die Pufferung von Ereignissen, die nicht sofort ausgeführt werden können, werden auch bei allen anderen Änderungstypen mit einem Event-Objekt realisiert.

5.2.4 Erstellung und Verwaltung von 2D-Anzeigen

Bei der Festlegung des endgültigen Konzepts wurde entschieden, dass es dem Nutzer möglich sein soll, zweidimensionale Anzeigeelemente mit Objekten im 3D-Raum zu verknüpfen und sich auf den 2D-Flächen verschiedene Daten der Objekte anzeigen lassen zu können. Zu diesem möglichen Daten gehören bei mechanischen Bauteilen z. B. die unterschiedlichen Rotationswinkel und bei den Bordnetz-Komponenten, wie Steuergeräten, sind es unter anderem die Auslastung oder die empfangenen Nachrichten pro Sekunde. Diese Werte müssen von externen Programmen an die 3D-Visualisierung übergeben werden. Dazu stehen im Interface Methoden wie `setLoad(name, value)` zur Verfügung, die den Auslastungswert des benannten Bauteils aktualisieren. Wie schon im Konzept in Abschnitt 4.4 erläutert, wurden zwei Arten von 2D-Elementen ins Konzept aufgenommen und entsprechend umgesetzt. Zum einen die Anzeige von konkreten Werte und die Visualisierung eines Werteverlaufs in einem Wertegraph. Beide Visualisierungsarten werden auf den 2D-Anzeigeflächen, die in diesem Abschnitt beschrieben werden, dargestellt.

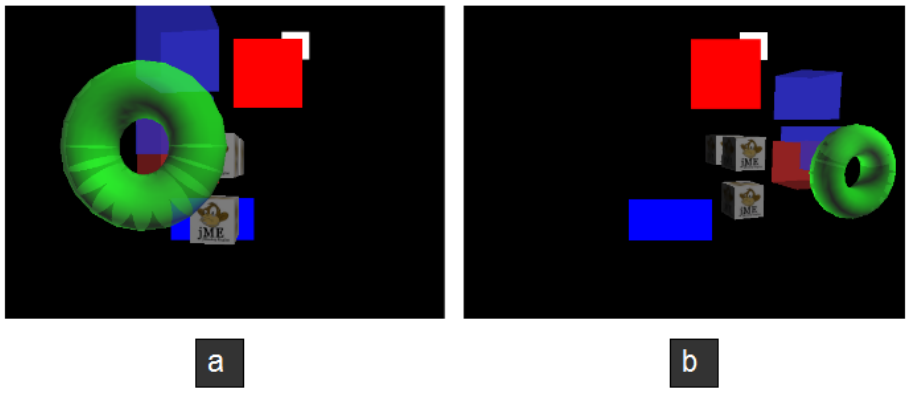


Abbildung 5.9: Das rote, das blaue und das weiße Rechteck werden orthogonal gerendert, die restlichen 3D-Objekte nicht [45]; (a) Ausgangsposition, (b) veränderte Kameraposition

Die Umsetzung der 2D-Anzeigen baut in beiden Fällen auf Rechtecken im 3D-Raum auf. Diese Rechtecke werden hierfür in der sogenannten „orthogonalen Render-Queue“ platziert. Dadurch ist gewährleistet, dass sie stets rechtwinklig zur Kamera dargestellt werden und somit die Seiten-

fläche immer in vollem Umfang vom Benutzer gesehen werden kann. Damit diese 2D-Anzeigen auch alle Kamerabewegungen nachvollziehen und nicht unbeweglich im Raum stehen, werden sie im Szenegraph unterhalb des sich verändernden Kameraknotens platziert. Dadurch ist es möglich, dass sie ohne weiteren Aufwand immer dem Kamerabild folgen und orthogonal zur Kamera gerendert werden. Abbildung 5.9 zeigt dieses Verfahren anhand von drei orthogonal gerenderten Rechtecken, während die verbleibenden 3D-Objekte fest im Raum stehen. Bei den Bildern handelt es sich um dieselbe Szene aus unterschiedlichen Blickwinkeln betrachtet - dabei fällt auf, dass die drei Rechtecke immer an derselben Position auf dem Bildschirm und rechtwinklig zur Kamera angezeigt werden.
Doch die Rechtecke allein zeigen natürlich noch keine Werte oder Wertegraphen an. Hierzu wird eine Möglichkeit der 3D-Engine ausgenutzt. Zuerst müssen den Rechtecken Texturen zugewiesen werden und im Anschluss daran ist es möglich innerhalb dieser Texturen einen Java2D-Kontext zu erstellen. Mit Hilfe einer eigenen Klasse kann dann innerhalb der Textur mit Java2D-Befehlen gearbeitet werden, während die Textur bei jeder Bildberechnung der 3D-Fensters ebenfalls neu gezeichnet wird. Dabei ist es noch interessant festzuhalten, dass die Auflösung des 2D-Zeichenbereichs unabhängig von der Größe des Rechtecks ist, da sie separat festgelegt wird. Es ist also auch denkbar, eine sehr niedrige Auflösung zu wählen und dieses Bild anschließend auf ein großes Rechteck zu texturieren - der Effekt wäre eine entsprechende Vergrößerung und ggf. auch eine Verzerrung, wenn die Seitenverhältnisse von Auflösung und Rechtecklänge bzw. -breite nicht übereinstimmen.

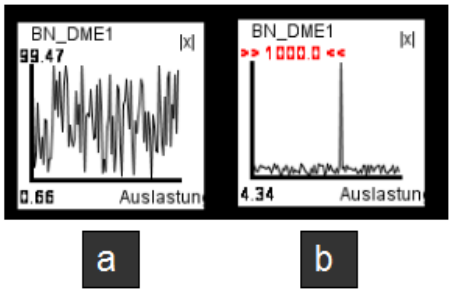


Abbildung 5.10: (a) Detailansicht ohne starke Veränderung des Wertebereichs, (b) Hervorhebung eines stark abweichenden Werts mit Hervorhebung der Wertebereichsänderung.

Wenn die Werte in einem Koordinatensystem eingetragen und auf einem 2D-Element visualisiert werden sollen, ist es nötig, eine gewisse Menge an alten Daten im Speicher zu behalten, um den Graph korrekt anzeigen zu können. Zu diesem Zweck werden bis zu 100 Werte, die über das Interface an die 3D-Ansicht übergeben wurden, in einem Vektor gespeichert, sobald eine Wertegraph-Darstellung für ein Objekt gestartet wurde. Die neuen Werte werden am Ende des Vektors eingefügt bis dieser eine Länge von 100 Objekten erreicht hat. Dann werden immer so viele Werte am Anfang des Vektors herausgestrichen, wie neue Werte am Ende eingefügt werden, so dass die Länge immer unverändert auf 100 bleibt, sobald diese Länge erreicht wurde. Um nun ein entsprechend skaliertes Koordinatensystem zu bieten, wird der Vektor vor jedem Zeichnen des Graphen nach dem minimalen und dem maximalen Wert durchsucht. Diese werden an vorgegebenen Punkten der 2D-Textur geschrieben, um dem Nutzer den aktuellen Wertebereich darzustellen. Sollten dabei Abweichungen zu vorhergehenden Minimal- oder Maximalwerten festgestellt werden, d.h. zum Beispiel der Maximalwert hat sich um mehr als 10% verändert, wird die Schriftfarbe geändert und es werden weitere Schriftzeichen um die Zahlen herum eingefügt, um eine präattentive Hervorhebung zu erreichen. So soll der Nutzer darauf aufmerksam gemacht werden, dass sich eine - möglicherweise zu große - Änderung im Wertebereich vollzogen hat. Anschließend wird

der Abstand der beiden Extrem-Werte zueinander berechnet und danach wird die zur Verfügung stehende Anzeigefläche für den Wertebereich, also die Anzahl der Pixel in y-Richtung des Koordinatensystems, durch diese Differenz geteilt. Mit Hilfe des so errechneten Verhältnisses ist es möglich, stets ein Koordinatensystem zu zeichnen, dass die zur Verfügung stehende Fläche komplett ausnutzt und so zum einen stark abweichende Werte deutlich hervorhebt und ansonsten eine detaillierte Ansicht des Werteverlaufs bietet. Diese beiden Fälle werden in Abbildung 5.10 einander gegenüber gestellt, um den Unterschied zu verdeutlichen.
Durch die Integration der 2D-Anzeigefläche in die 3D-Umgebung - das texturierte Rechteck befindet sich trotz orthogonalen Renderings in der 3D-Umgebung - ist es möglich, eine Verbindung zwischen 2D-Anzeige und zugehörigem 3D-Objekt im Fahrzeugmodell mittels eines semitransparenten Körpers zu erstellen.
Zu diesem Zweck werden über eine Methode des DisplaySystems der JME die 3D-Koordinaten des Mittelpunkts der 2D-Anzeigefläche ermittelt. Sobald diese Koordinaten bekannt sind, ist es in jedem neuen Frame möglich auf Basis von Polygonen einen 3D-Körper zu definieren, der einer verzerrten Pyramide ähnelt. Deren Basis befindet sich auf der Rückseite der 2D-Fläche, wobei der Mittelpunkt der 2D-Fläche gleichzeitig der Mittelpunkt des dynamischen 3D-Körpers ist.
Die Spitze des Körpers befindet sich im Mittelpunkt des zugehörigen Objekts im Fahrzeugmodell.

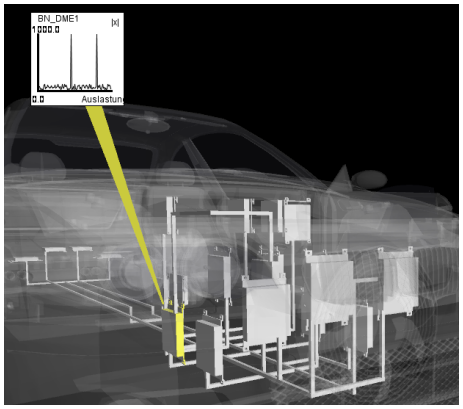


Abbildung 5.11: Dynamisch erzeugtes Verbindungsobjekt zwischen 2D-Anzeige und 3D-Element in dessen Farbe.

Wird nun die Kamera bewegt, ändert sich zwar die Position der 2D-Anzeige im 3D-Raum, doch da das Verbindungsobjekt zwischen Anzeige und Fahrzeug-Element in jedem Frame neu berechnet wird, passt sich die Darstellung entsprechend an, sobald die neue Kameraposition erreicht wird. Für eine bessere Übersicht bei gleichzeitiger Anzeige mehrerer solcher Verbindungen wird das verbindende Objekt jeweils in der Farbe des verbundenen 3D-Objekts dargestellt - in Abbildung 5.11 also z. B. in gelb.

5.2.5 Erstellung und Verwaltung mehrerer 3D-Views

In diesem Kapitel soll die Erstellung und Verwaltung mehrerer 3D-Ansichten beschrieben werden. Dabei spielt sowohl der *technische Hintergrund* der Umsetzung eine Rolle als auch die *logische Verwaltung* durch den ViewManager. Es wird außerdem auf die *Unterschiede zwischen der aktiven Sicht und den verkleinerten Ansichten* eingegangen.
Zur Erstellung einer neuen 3D-View innerhalb des 3D-Clients gibt es zwei Möglichkeiten: bei der ersten findet von außen ein entsprechender Aufruf auf der Interface-Klasse statt, die wiederum auf Methoden im ViewManager zurückgreift. Die zweite Möglichkeit ist die Erstellung und gleichzeitige Benennung einer neuen View im Konfigurationsmenü des Programms. Abgesehen von den unterschiedlichen Aufruf-Orten verlaufen beide Arten der Erstellung anschließend identisch ab

und werden daher in der folgenden Beschreibung nicht weiter unterschieden.
Da der ViewManager für die Verwaltung mehrerer Views verantwortlich ist, liegt auch die Erstellung einer zusätzlichen View auf dieser Schicht-Ebene. Dort existieren die Methoden `createView(String name)` und `createNew3DHud()`, die nacheinander aufgerufen werden. Dabei wird durch die erste der beiden Methoden ein neues View-Objekt erstellt, während dessen Erstellung das Fahrzeug-Modell erneut eingeladen wird. Anschließend wird dieses Modell mit Hilfe der State-Machine im ViewManager auf den aktuellen Fahrzeugzustand gebracht und die Kamera, die in jeder View-Klasse definiert wird, wird auf die Ausgangsposition gesetzt.
Durch die zweite Methode werden zwei einfache, zweidimensionale Rechtecke erstellt, die, wie die 2D-Anzeigen in Kapitel 5.2.4, orthogonal zur Kamera gerendert werden. Diese dienen als Anzeigeflächen für die verkleinerten 3D-Ansichten, die aktuell nicht im Fokus des Nutzers sind, und als Umrahmung der Ansichten, um einen bestimmten Kontrast zum Rest des 3D-Fensters garantieren zu können. Das größere der beiden Rechtecke dient sozusagen als Fenster bzw. Rahmen für die jeweilige 3D-Ansicht. Wie schon bei den 2D-Anzeigen wird auch hier in einer Java2D-Umgebung auf die Textur des Rechtsecks gezeichnet. So können in der Umrahmung der kleinen 3D-Ansichten sowohl Beschriftung als auch der Button zum Schließen der Miniatur-Ansicht dargestellt werden. Durch diese Umrandung entsteht außerdem ein hoher Kontrast zum großen 3D-View, so dass die einzelnen kleinen Ansichten gut zu erkennen sind (5.12).
Das kleinere der beiden Rechtecke wird genutzt, um die 3D-Szene in verkleinerter Form darzu-

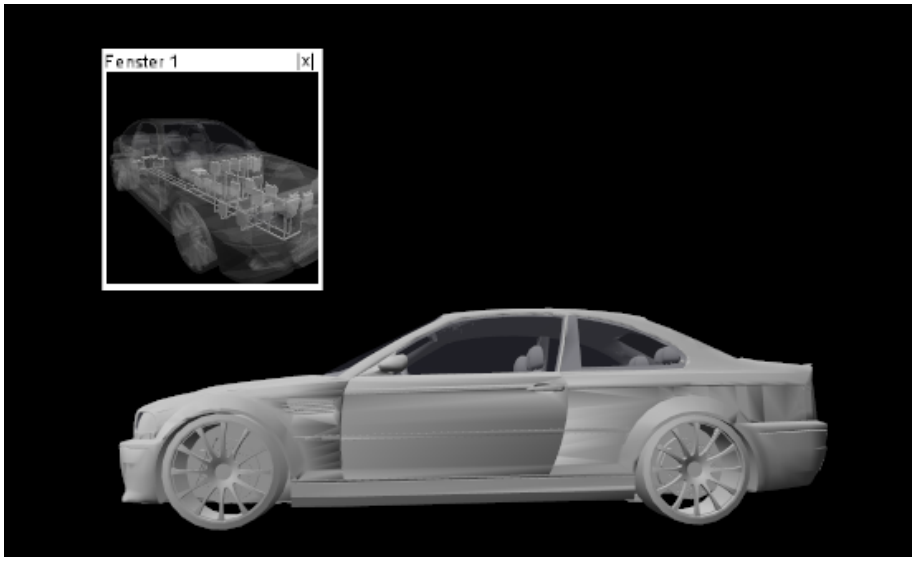


Abbildung 5.12: Hauptansicht mit kleiner 3D-Ansicht.

stellen - in Abbildung 5.12 kann man sowohl das kleine 3D-Fenster als auch den umgebenden weißen Rahmen erkennen. Bei der Erstellung einer neuen View muss jedoch zunächst das aktuelle Modell aus dem Szenegraph der Hauptszene entfernt werden. Dazu sucht man von der Wurzel ausgehend den entsprechenden Knoten und löst ihn von seinem Elternknoten. Das daran hängende Modell ist jedoch nicht verloren, denn es wird ja im entsprechenden View-Objekt weiterhin verwaltet. Im Anschluss an das Entfernen (engl.: detachen) des Knotens wird das neu eingeladene Modell an der entsprechenden Stelle im Szenegraph eingefügt, so dass fortan das neue Modell im Hauptfenster erscheint.
Für das Rendering der kleinen Ansichten ist zunächst eine Zuweisung zwischen vorhandenen Rechtecken - den sog. ViewQuads, die zur Anzeige der kleinen 3D-Views genutzt werden - und den noch vorhandenen Views erforderlich. Diese Zuordnung ist nötig, um die Views immer konstant im gleichen kleinen Fenster anzeigen zu können. Die Anzeige der 3D-Szene in dem Fenster,

wurd durch eine JME-Methode realisiert. Diese ermöglicht es, eine gegebene 3D-Szene auf eine flache Textur rendern zu lassen - die zu rendernde 3D-Szene ist in diesem Fall das Fahrzeugmodell, welches im entsprechenden View-Objekt hinterlegt ist. Anschließend kann die Textur dem View-Quad zugewiesen werden und das 3D-Modell wird somit in verkleinerter Form auf dem Rechteck angezeigt. Dadurch, dass dieser Rendering-Vorgang in jedem Frame wiederholt wird, sind auch die Animationen von Bewegungen und Farbveränderungen in den kleinen 3D-Ansichten stets auf demselben Stand wie im großen Hauptfenster.
Diese Form der Realisierung eines MCV-Systems in einem anderen MCV-System hat den Vorteil, dass die Grenzen des 3D-MCV-Systems klar definiert sind und somit nicht die Gefahr besteht, dass der Nutzer Fenster verschiedener Ebenen durcheinander wirft. Die zusätzlichen 3D-Fenster sind immer innerhalb der 3D-Ansicht zu finden und können, da sie Teil der 3D-Szene sind, auch nicht aus dem Fenster herausgezogen werden.

5.2.6 Prototypische Anbindung an ein Programm

Da der 3D-Client keine selbstständige Daten-Analyse vornimmt, ist er auf die Kommunikation mit anderen Anwendungen angewiesen, die entsprechend aufbereitete Daten zur Verfügung stellen. Diese aufbereiteten Daten können dann im weiteren Verlauf vom 3D-Client interpretiert bzw. umgesetzt werden. Da es Ziel des Konzepts ist, eine möglichst einfache Anbindung zu ermöglichen wurde bei der prototypischen Anbindung eine Client-Server-Architektur verwendet, die es ermöglicht, auf der Basis von String-Nachrichten, über Sockets zu kommunizieren. Dies hat den enormen Vorteil, dass auch Anwendungen, die nicht in Java geschrieben sind mit der 3D-Ansicht zusammenarbeiten können.
Als Server diente hierbei die AutobahnView aus Kapitel 2.2, da sie eine mögliche Datenquelle darstellt und einen Überblick über die gesamten Daten bietet. Durch diese Daten kann der Nutzer scrollen und es wurde eine Methodik eingebaut, die beim Scrollen über den Zeitstrahl in der AutobahnView überprüft, welche Nachrichten und Ereignisse zwischen dem letzten betrachteten Zeitpunkt und dem nun aktiven Zeitpunkt liegen. Wird in diesem Zeitrahmen eine Nachricht gefunden, die eine Auswirkung auf den mechanischen Fahrzeugstatus hat, wird anhand der beiden Zeitstempel (alte Zeit, neue Zeit) geprüft, ob in der Zeit vor- oder zurücknavigiert wurde.
So kann das korrekte Ereignis ausgelöst und an die 3D-Ansicht übertragen werden. Dies wäre beispielsweise eine Aktion der Fahrertür - wird vorwärts durch die Zeit navigiert, öffnet sich die Tür zu dem gegebenen Zeitpunkt, wird jedoch rückwärts navigiert, schließt sie sich zu diesem Zeitpunkt. In Abbildung 5.13 wird der Zusammehang zwischen dem Scrollen über die Zeit und dem Öffnen und Schließen der Tür dargestellt.
Durch die Bestimmung der eingetretenen Ereignisse anhand der vorliegenden Kommunikationsdaten und der Analyse der Navigationsrichtung, können entsprechend kodierte String-Nachrichten in den Socket-Puffer der Verbindung zwischen AutobahnView und 3D-Ansicht geschrieben werden. Diese Nachrichten werden durch ein eindeutig definiertes Trennsignal voneinander abgeschnitten, so dass es später möglich ist, die Nachrichten wieder auseinander zu schneiden. Der Puffer wird dann in festgelegten Intervallen abgeschickt und anschließend geleert.
Auf der Seite des Clients werden die über die Socket-Verbindung eintreffenden Nachrichten wiederum in einen Puffer geschrieben. Dieser Puffer wird periodisch nach den eingefügten Trennzeichen durchsucht. Sobald die erste Nachricht vollständig gelesen werden konnte, d.h. beim Auffinden des ersten Trennzeichens, beginnt der Client damit, den restlichen Puffer nach entsprechend entgegengesetzten Ereignissen auf dieselbe Weise zu durchzusuchen. Ist das erste Ereignis zum Beispiel ein `openDoorFL` wird nach einem `closeDoorFL` gesucht. Wird eine solche entgegengesetzte Anweisung gefunden, werden beide aus dem Puffer gelöscht und die Suche beginnt wieder von vorn, d.h. es wird das erste Trennsymbol und anschließend die zugehörige entgegensetzte Aktion gesucht. Sobald keine solche Aktion gefunden wird, wandelt der Client die Nachricht in den passenden Befehl um und ruft auf dem Interface, das auch zur direkten Anbindung genutzt werden

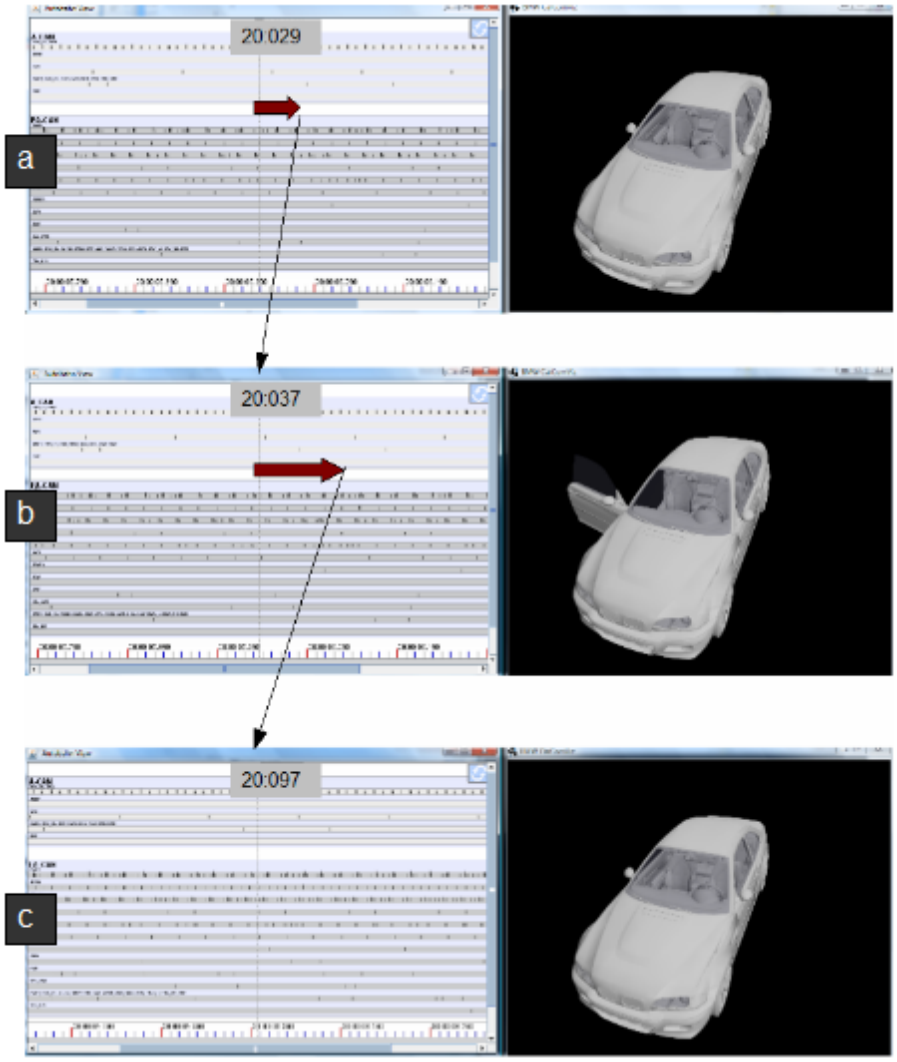


Abbildung 5.13: Zusammenspiel von AutobahnView (links) und 3D-Client(rechts); (a) Zeit: 20:029, (b) Zeit: 20:037, (c) Zeit: 20:097

kann, die entsprechenden Methoden auf. Sobald der Aufruf erfolgt ist, wird die Nachricht aus dem Puffer entfernt und die Pufferanalyse startet erneut.
Dabei stellt die Pufferung auf der Client-Seite sicher, dass nur vollständige Nachrichten analysiert werden. Das Entfernen von sich gegenseitig aufhebenden Aktionen aus dem Puffer trägt dazu bei, dass beim schnellen Scrollen durch die Daten in der AutobahnView und der damit verbundenen Masse an eintreffenden Befehlen keine extremen Animationssprünge im Modell auftreten. Darüber hinaus wird dadurch die Abarbeitungsgeschwindigkeit bei vielen gepufferten Nachrichten deutlich erhöht, da weniger Befehle ausgeführt werden müssen, wenn einige zuvor schon durch die Filtermethode entfernt wurden.

6 Abschluss-Evaluation

Nach Abschluss der Implementierungsarbeiten am Prototyp wurde dieser - genau wie die bereits vorhandenen Prototypen in Kapitel 3 - mit der Hilfe von Experten bei BMW evaluiert. Bis auf zwei Ausnahmen konnten erneut die gleichen Testpersonen und zusätzlich eine neue Person für die Evaluation gewonnen werden, d.h. an dieser Abschlussevaluation nahmen fünf Analyse- und Diagnose-Experten von BMW teil. Die jeweiligen Testsitzungen dauerten pro Person etwa eine Stunde und wurden im Rahmen eines Leitfadeninterviews durchgeführt. Um möglichst viele individuelle Rückmeldungen zu erhalten, wurde also erneut der Ansatz der qualitativen Evaluation gewählt. So sollten auch nach Abschluss der Implementierung noch weitere Verbesserungsvorschläge und Visualisierungsideen durch die Gespräche gefunden werden, um eine spätere Weiterentwicklung in diesem Bereich einschätzen zu können und durch die Ergebnisse zu unterstützen.

6.1 Ablauf

Der Ablauf der Evaluation ähnelt stark der Befragung aus der Anforderungsanalyse, auch hier wird zunächst die erste Meinung zum Prototyp abgefragt nachdem dieser den Experten kurz präsentiert wurde. Dabei wurde die Steuerung erklärt und es gab eine kurze Einführung in die Benutzung des Konfigurationsmenüs. Im Anschluss daran wurden dann die ersten positiven und negativen Eindrücke abgefragt. Dabei ist es wichtig festzuhalten, dass keine genaue Erklärung des späteren Einsatzzwecks erfolgte. Nach diesen allgemeinen Eindrücken gingen die Fragen konkret auf einzelne Details des Visualisierungskonzepts ein. Dabei lagen die Schwerpunkte auf dem mechanische Umgebungsmodell und dessen Möglichkeiten zur Visualisierung und der Positionierung und Verwendung der 2D-Anzeigen. Am Ende dieses Teils der Befragung wurde wieder Raum gelassen für allgemeine Anmerkungen der Teilnehmer, die bis dahin nicht von spezifischen Fragen abgedeckt wurden, und es wurde die Frage nach der möglichen Verwendung in Verbindung mit anderen Programmen gestellt.
Nach der Evaluation des alleinstehenden 3D-Clients wurde zusätzlich die Anbindung an die AutobahnView (vgl. Kapitel 2.1.2) aktiviert, so dass die Nutzer nun ein einfaches MCV-System (siehe Abbildung 5.13) vor sich hatten. Auch hier wurde kurz die Funktionsweise erklärt, bevor die allgemeine Frage nach Vor- und Nachteilen einer solchen Kombination gestellt wurde. Um bereits in dieser frühen Phase der Entwicklung eines MCV-Systems mit 3D-View einen ersten Eindruck von der Benutzerfreundlichkeit zu erhalten, wurden den Test-Teilnehmern verschiedene Aufgaben gegeben, die mit Hilfe des MCV-Systems gelöst werden sollten. Sie sollten bestimmte Zeitpunkte finden, an denen vorgegebene Ereignisse und Statusänderungen eintreten. Hierzu zählten Zeitpunkt wie das erste Öffnen einer Tür, das erste Blinken eines Blinkers und das Erreichen eines Auslastungswerts bei einem Steuergerät. Beim Bearbeiten der Aufgaben wurden sie beobachtet und es wurden Notizen zu den jeweiligen Vorgehensweisen gemacht.
Bevor eine allgemeine Bewertung des Programms durch die Teilnehmer erfolgen konnte, wurde ihnen zuerst das Programm vorgestellt. In diesem Rahmen wurden ihnen kurz sämtliche Funktionen gezeigt und die Grundprinzipien der Steuerung erklärt. Hierzu gehörten sowohl die verkleinerten 3D-Ansicht, als auch die 2D-Anzeigen und die Einstellung von Transparenzen für Objekte innerhalb der 3D-Szene. Bei der Steuerung wurde die unterschiedliche Belegung der Maustasten erklärt und wie die Navigation der Kamera durch die 3D-Welt erfolgen kann.
Insgesamt sollten mit dieser Evaluation mehrere Hypothesen überprüft werden. Die *Fahrzeugsreaktions-Hypothese* sagt aus, dass eine 3D-Ansicht gut geeignet ist, um mechanische Fahrzeugreaktionen zu visualisieren. Die *Bordnetz-Hypothese* unterstellt, dass die 3D-Ansicht in Verbindung mit 2D-Anzeigen gut geeignet ist, um Daten zur Bordnetzkommunikation zu visualiseren. Die dritte zu prüfende Hypothese, die *MCV-Hypothese*, vermutete, dass eine einzelne 3D-Ansicht nicht ausreicht, um die Bordnetzkommunikation und den Fahrzeugstatus angemessen zu visualisieren. Daher ist es nötig die 3D-Ansicht mit vorhandenen 2D-Visualisierungen zu komibi-

nieren, um einen Mehrwert zu bekommen.

6.2 Allgemeine Bemerkungen zum Prototyp

Bei der Frage nach den *positiven Eigenschaften einer solchen Visualisierungsform* wurde von den meisten Probanden sofort der Detailgrad des Modells angesprochen. Sie navigierten meist direkt ins Fahrzeug und waren überrascht, dass selbst im Innenraum eine Vielzahl an Schaltern und weiteren Details sichtbar und gut zu erkennen ist. Dabei wurde zum Teil auch die Vielzahl der einzelnen Elemente positiv hervorgehoben, die in der Liste der Bauelemente und Steuergeräte selektierbar sind. Ein weiterer positiver Aspekt, der von den Befragten gleich zu Beginn genannt wurde, war die Einstellmöglichkeit der Transparenz. Es wurde hierbei zwar nicht näher darauf eingegangen warum und weshalb dies positiv sei, jedoch wurde bereits hier davon ausgegangen, dass es bei der späteren Arbeit mit einem 3D-Modell nützlich sein könnte. Eine der Testpersonen war sich bereits zu diesem Zeitpunkt sicher, dass man mit einem 3D-Modell den Fahrzeugstatus deutlich besser erkennen kann als durch mühsames Lesen von Trace-Daten.
Es wurde danach auch nach ersten *negativen Eindrücken* gefragt. Zentraler Kritikpunkt der Aussagen hierbei war vor allem der fehlende Mehrwert einer allein genutzten 3D-Visualisierung im Bereich der Analyse/Diagnose. Konkret wurde an dieser Stelle bemängelt, dass eine 3D-Visualisierung kaum in der Lage sei, sämtliche Daten, die von vorhandenen 2D-Darstellung präsentiert werden, sinnvoll und übersichtlich in einem solchen Fahrzeugmodell unterzubringen. Auch die 2D-Anzeigen seien kein möglicher Vorteil einer reinen 3D-Visualisierung, da sie im Prinzip lediglich eine Nachbildung von 2D-Visualisierungen darstellen. Auf Nachfrage stellte sich hier aber heraus, dass nicht die grundsätzliche Verwendung von 2D-Anzeigen kritisiert wurde, sondern lediglich betont werden sollte, dass diese als einzige Visualisierungsfläche für konkrete und komplexe Werte zu wenig seien. In diesem Zusammenhang wurde auch vermutet, dass eine reine Bordnetzdarstellung in 3D zu unübersichtlich sei - es wurde an dieser Stelle auf vorhandene 2D-Visualisierungen verwiesen und dass 3D hier eben keinen großen Vorteil bei der Darstellung biete.

6.3 Bewertung der Visualisierungskonzepte

Nach der anfänglichen allgemeinen Bewertung sollten die Teilnehmer konkrete Eigenschaften des Prototyps bewerten. Hierzu zählen die Visualisierung des mechanischen Umgebungsmodells, die Darstellung des Bordnetzes und die 2D-Anzeigen. Diese Punkte wurde nacheinander explizit abgefragt, um die Meinungen der Experten zu den jeweiligen Features zu erhalten. Nach Abschluss des Frageteils gab es die Möglichkeit für die Teilnehmer, noch weitere Bemerkungen zu machen.

6.3.1 Mechanisches Umgebungsmodell und Detailgrad

Die Befragung wurde mit dem offensichtlichsten Punkt des Programms begonnen - dem mechanischen Umgebungsmodell. Da den Teilnehmern zuvor die Funktionsweise des Programms erläutert wurde, konnten sie ab diesem Zeitpunkt selbstständig mit dem Prototyp arbeiten. Zur besseren Veranschaulichung der Funktionsweise während der Tests, lief im Hintergrund ein Script, das durch einige Aktionen permanent den Fahrzeugstatus veränderte (Türen auf / zu, Lenkbewegungen, ...). So sollte auf einfache Weise verdeutlicht werden, wozu das Modell genutzt werden kann, jedoch ohne auf den späteren Einsatzort in einem MCV-System hinzuweisen.
Die Frage nach dem Fahrzeugmodell und dem *Detailgrad des Fahrzeugs* wurde nahezu einheitlich beantwortet. Das häufigste Attribut, mit dem der Prototyp zuerst bedacht wurde war „cool“. Anschließend wurde das Modell von allen Teilnehmern näher betrachtet, die Kamera wurden zum Fahrzeug oder darum herum bewegt. Schon nach kurzer Zeit wurde gesagt, dass es gut sei, wie man den Fahrzeugzustand erkennen könne. Gerade bei Trace-Daten, in denen viele mechanische Aktionen gleichzeitig codiert seien, sei so eine Ansicht sehr nützlich, da man nicht manuell den Trace

durchsuchen müsse, sondern den letztendlichen Fahrzeugzustand direkt präsentiert bekommt. Es bestand also unabhängig voneinander eine Übereinstimmung darin, dass man den Gesamtstatus des Fahrzuegs sehr gut erkennen könne.
Nach dem ersten genaueren Betrachten des Modells wurden jedoch auch kritische Kommentare geäußert, was die *Sichtbarkeit von kleinen Veränderungen* im Modell angeht. Als Beispiel wurde in einem Gespräch das Schließen der Zentralverriegelung angeführt, welches visuell am Modell eher schlecht zu erkennen ist. In diesem Zusammenhang wurden auch Aktionen angesprochen, die verdeckt vom Modell stattfinden können und die möglicherweise dadurch vom Benutzer übersehen werden könnten. Mehrfach wurde dabei vorgeschlagen, laufende Aktionen auf eine beliebige Weise (meist wurde hier als Beispiel Einfärben aufgeführt) hervorzuheben, um den Nutzer darauf aufmerksam zu machen. Aber auch gerade bei der oben angesprochenen Situation, in der viele Aktionen gleichzeitig ablaufen, müssten Filteroptionen zur Hervorhebung angeboten werden. Zur Abschwächung dieses Problems kam der Vorschlag auf, eine Kameraposition anzubieten in der sowohl die Fahrer-Instrumente (Tacho, Drehzahl, ...) als auch weitere wichtige Anzeigeelemente für den Fahrer auf einen Blick sichtbar seien. Dadurch sei es auch akzeptabel, wenn im 3D-Modell einige Aktionen nicht direkt sichtbar seien, aber in der dargestellten Cockpit-View angezeigt würden. Ein mögliches Beispiel für einen solchen Fall sei das Blinken der Blinker auf der dem Nutzer abgewandten Seite des Modells.
Auf die Frage nach dem Detailgrad des genutzten Modells waren bis auf einen Nutzer alle begeistert, „[...]wie genau das Modell ist“. Alle Befragten waren sich an dieser Stelle einig, dass es keinesfalls zu viele Details seien, die am Modell dargestellt werden. Einige gingen sogar soweit noch mehr Details einbauen zu wollen. Als Grund hierfür nannten sie die Einstellmöglichkeit der Transparenz von Objekten, so dass man sich das Modell selbst so gestalten könne, dass man die für sich wichtigen Details im Blick hat.

6.3.2 Darstellung des Bordnetzes

Der nächste Punkt, der mit Hilfe der Abschlussevaluation geklärt werden sollte, war die Darstellung der Steuergeräte und des gesamten Bordnetzes. Hier stand in Frage, wie detailliert die Positionierung der Steuergeräte und Kabelverläufe sein muss bzw. wie detailliert diese sein dürfen, ohne dass die Übersicht im Modell leidet. Zusätzlich zur Positionierung ging es außerdem um Faktoren wie das Aussehen von Steuergeräten und die jeweilige Größe von dargestellen Bordnetz-Komponenten.
Nachdem die Testpersonen zum *Bordnetz* befragt wurden, suchten alle Tester umgehend im Konfigurationsmenü die Steuerung für die Transparenz der Fahrzeugbauteile, um einen besseren Blick auf das Bordnetz zu erhalten. Hier zeigt sich bereits, dass die Steuerung der Transparenz scheinbar ein wesentlicher Faktor im Umgang mit einer 3D-Visualisierung ist, da sich der Nutzer das Modell auf diesem Wege nach seinen Wünschen und Use-Case-Anforderungen selbst gestalten kann.
Abgesehen von dieser Beobachtung gingen die Meinungen der Benutzer zu dieser Frage ziemlich auseinander. Dies war allerdings auch schon fast zu erwarten, denn bei der Evaluierung der Konzeptentwürfe wurden sehr unterschiedliche Vorschläge gemacht und jeweils auf die Use-Cases verwiesen. Sobald sich der Use-Case ändert, ändert sich, so alle Befragten bei der Bewertung der Konzeptentwürfe, auch die benötigte Bordnetzdarstellung. Dementsprechend gingen auch die Meinungen zum gewählten Mittelweg bzw. Kompromiss zwischen „sehr detailliert“ und „völlig abstrakt“ auseinander.
Entgegen der Erwartung, dass die *Darstellung von Steuergeräte-Namen* im 3D-Raum nicht sinnvoll sei (siehe Kapitel 2.22), wurde sie mehrfach von den Testpersonen angesprochen. Allerdings nicht ausschließlich in Form von 3D-Text, sondern zum Beispiel als Tooltip, wenn der Benutzer mit dem Mauszeiger über einem Objekt der 3D-Szene verweilt. Sicherlich ist diese Aussage jedoch in Verbindung mit der oben genannten Use-Case-Abhängigkeit zu sehen. Denn, wenn die Position und die anderen Parameter der Steuergeräte nicht wichtig sind, dann sind es sicherlich

auch nicht die dargestellten Namen im 3D-Fenster. Jedoch könnten die Namen in der 3D-Szene eine sinnvolle Information sein, wenn die Position und weitere Informationen zum Bauelement eine entscheidende Rolle spielen.
Unabhängig von der statischen Darstellung gab es auch ein paar Äußerungen zur *Visualisierung von Nachrichtenflüssen*. Interessant dabei ist, dass die Anmerkungen lediglich von Teilnehmern der ersten Evaluation mit dem Prototyp der Purdue University [20] (siehe Kapitel 2.2.2) gemacht wurden. Es ist also sehr wahrscheinlich, dass hier eine Beeinflussung stattgefunden hat. Dennoch sollte die Idee der Visualisierung von Nachrichtenflüssen nicht völlig übergangen werden, da es ja scheinbar ein ernsthaftes Interesse daran gibt, sobald diese Idee einmal bekannt ist. Dabei ist allerdings noch zu beachten, dass eine Visualisierung in Echtzeit in diesem Bereich nicht mit bisher versuchten Mitteln adäquat umzusetzen ist.

6.3.3 2D-Anzeigen

Die Idee, die 3D-Visualisierung eines Fahrzeugs mit zweidimensionalen Anzeigeflächen zu erweitern, wurde allgemein gut aufgenommen. Dabei wurde besonders die Möglichkeit gelobt, sich einige wichtige Werte gezielt anzeigen lassen zu können, um die genauen Daten zu erhalten und sich nicht ausschließlich auf den dargestellen 3D-Fahrzeugstatus verlassen zu müssen. Auch die möglichen Darstellungsmodi wurden insgesamt positiv bewertet. Die *konkrete Anzeige von Werten* wurde direkt mit den entsprechenden 2D-Visualisierungen existierender Tools verglichen (siehe Kapitel 2.1). Die Idee, sich diese Informationen auch im 3D-Bereich anzeigen lassen zu können, schnitt bei den Beurteilungen gut ab.

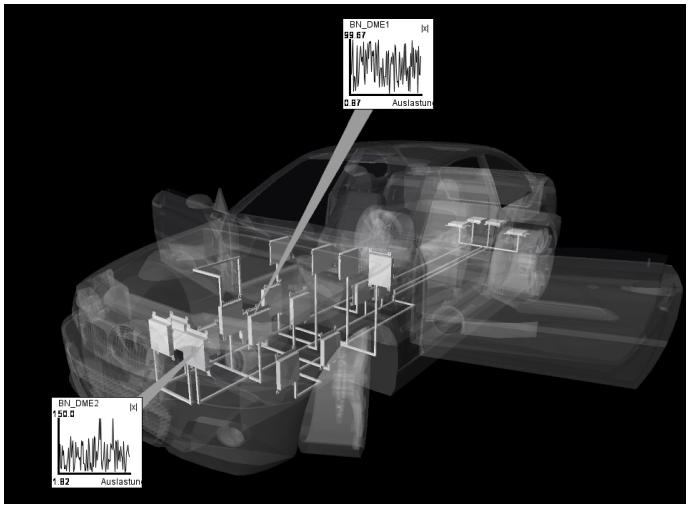


Abbildung 6.1: Freie Positionierung von 2D-Anzeigen mit Wertegraph-Visualisierung.

Ebenso positiv wurde die *Wertegraph-Visualisierung* aus Abbildung 6.1 bewertet, da hier vor allem die History-Funktion, d.h. die Darstellung von Werten, über die Zeit, nützlich sei. Dies war in Anbetracht der Tatsache, dass die Idee des radialen Farbverlaufs während der Konzeptentwicklung und -bewertung sehr gut abgeschnitten hatte, durchaus zu erwarten gewesen. Von dieser Idee ausgehend, wurden aber auch Überlegungen angestellt, ob es nicht sinnvoll sei, noch weitere 2D-Anzeigemöglichkeiten zu bieten und sich nicht lediglich auf diese beiden Typen (siehe Kapitel 5.2.4) festzulegen.
Diese Aussagen der Teilnehmer zeigen, dass die Erweiterung des 3D-Raums um zweidimensionale Anzeigen konkreter Werte eine gute Möglichkeit bietet, das Fahrzeugmodell mit weiteren Informationen anzureichern, ohne jedoch direkt Nachrichten oder Werte im 3D-Modell darstel-

len zu müssen. Dieser Ansatz wurde bei der Evaluation in Kapitel 2.2.1 eher abgelehnt, da die Textinformationen in dreidimensionaler Form schwer zu lesen und bei mehreren, möglicherweise bewegten, Objekten auch nicht übersichtlich waren.
Eine Kritik, die allgemein an der 2D-Darstellung geäußert wurde, war das *fehlende Look&Feel* des jeweiligen Betriebsystems, in diesem Fall Microsoft Windows XP. Mehrfach wurde gesagt, dass das Fehlen des Look&Feels, also die Optik des Fensters, das Aussehen der Titelleiste, eine Beeinträchtigung bei der Benutzbarkeit der 2D-Anzeigen erschwere, da das Konzept der Benutzung sich nicht von allein erschließe.
Abgesehen von der Anzeigefläche selbst, war auch die *Positionierungstechnik der 2D-Anzeigen* ein Thema dieses Fragebereichs. Zur Verfügung standen den Testpersonen zwei unterschiedliche Techniken. Die erste dabei war eine In-Place-Positionierung, d.h. die Anzeigen werden direkt an den zugehörigen Elementen im 3D-Modell bzw. besser gesagt an deren Position auf dem Bildschirm angezeigt (siehe Abbildung 6.2).
Die andere Möglichkeit war die freie Positionierung auf dem Bildschirm, wobei die 2D-Anzeige dann durch ein 3D-Objekt mit dem zugehörigen Bauteil verbunden ist (siehe Abbildung 6.1). Bei der Bewertung der beiden Techniken wurde jedes Mal die freie Positionierung bevorzugt. Einen Grund dafür fasste einer der Teilnehmer sehr passend zusammen mit dem Kommentar zur In-Place-Positionierung: „Ich sehe, dass ich nichts sehe“.

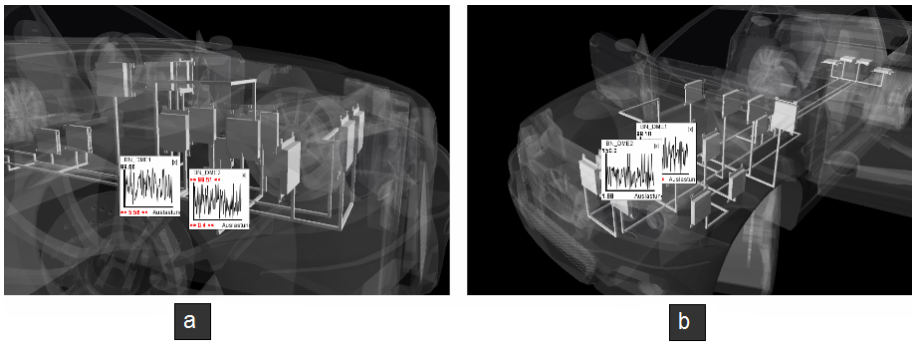


Abbildung 6.2: In-Place-Positionierung von 2D-Anzeigen aus unterschiedlichen Sichtwinkeln: (a) ohne Verdeckungsproblem, (b) mit Verdeckungsproblem.

Wie schon vermutet wurde, sind die entstehenden Verdeckungen bei der In-Place-Positionierung zu groß, so dass die freie Positionierung an dieser Stelle einfach die bessere Wahl zu sein scheint. Es wurde auch positiv bewertet, dass es zwischen 2D-Anzeige und 3D-Bauteil eine Verbindung gibt, so dass hier der Benutzer unterstützt wird, indem die jeweiligen Positionen der beobachteten Bauteile angezeigt werden.
Auch in diesem Bereichen wurden von den Befragten weitere Vorschläge zur *Verbesserung der gezeigten Ansätze* gemacht. Zum einen wurde von einem Befragten vorgeschlagen, eine gewisse Transparenz-Stufe bei der In-Place-Positionierung zu verwenden, so dass die Verdeckung anderer Objekte abgemildert wird. Doch selbst dann wäre die freie Positionierung immer noch zu bevorzugen, so der Teilnehmer. Eine weiterführende Idee in diese Richtung ist die Möglichkeit, die Positionierungstechnik pro 2D-Fenster selbst bestimmen zu können. Als Vergleich oder Metapher diente eine Pin-Nadel mit der man die Anzeigeflächen an die entsprechenden Bauteile pinnen könnte. So könnte der Nutzer völlig frei entscheiden, in welcher Situation er welche Technik bevorzugt. Auch eine Mischung von beiden Techniken wäre so problemlos möglich.

6.3.4 Konfigurationsmenü

Auch wenn das Konfigurationsmenü nicht direkt Teil der Visualisierung ist, wurde auch dieser Teil des Prototyps durch eine entsprechende Frage evaluiert. Dabei wurde besonders die Filtermöglichkeit gelobt, die durch die Verbindung der Auflistung aller Bauteile und die Einstellmöglichkeit der Transparenz eben jener Teile entsteht. Gerade dieses Feature des Menüs wurde auch von allen Teilnehmern problemlos genutzt und als äußerst positiv empfunden.
Ein Vorschlag, der die Darstellung der Teilenamen betrifft (Abbildung 6.3(a)), war die *Organisation der Teile in einer Baumstruktur*, ähnlich der Filtermöglichkeiten des getesteten Prototyps in Kapitel 2.2.1 (siehe Abbildung 3.2 (a)). Es wäre von Vorteil, so die Aussage, wenn man auf diesem Wege gleich ganze Baugruppen des Fahrzeugs ausblenden könne, ohne erst mühsam die Einzelteile aus der Gesamtliste zu wählen. Dieselbe Aussage wurde auch für die Filterliste der Steuergeräte gemacht. Gerade hier ließe sich ja das Konzept aus dem bereits getesteten Prototyp von Mozdari - also die Umsetzung der Struktur der Bordnetzdatenbank - gut übernehmen. Abgesehen davon erfüllte das Menü seinen Zweck, den Nutzern die gegebenen Konfigurationsmöglichkeiten übersichtlich zu präsentieren. Gerade die Erstellung von 2D-Anzeigen (Abbildung 6.3(c)) stellte für keinen der Benutzer Probleme dar, ebenso wie die Erstellung einer zusätzlichen 3D-View.

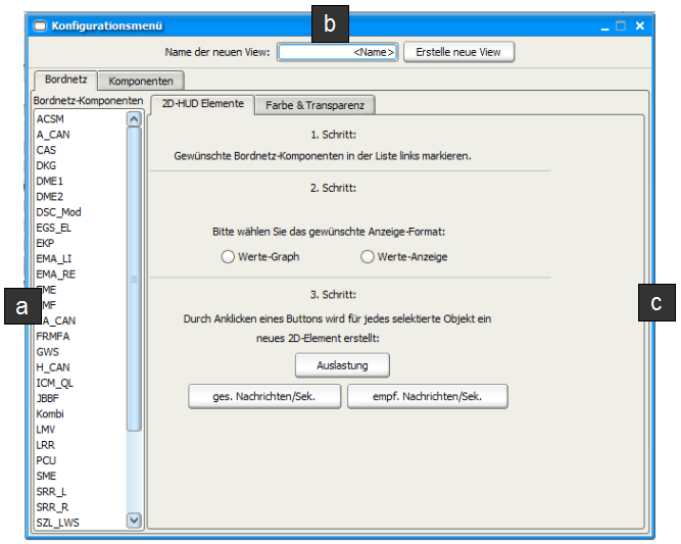


Abbildung 6.3: Konfigurationsmenü der 3D-Ansicht: (a) Liste der auswählbaren Bordnetz-/Fahrzeugkomponenten, (b) Dialog zur Erstellung einer neuen 3D-Ansicht, (c) Dialog zur Erstellung einer 2D-Werteanzeige.

Ein Punkt, der hier jedoch auch krititsiert wurde, war das *Fehlen jeglicher Vorkonfigurationen*, die im Menü abrufbar sein sollten. Es wurden unter anderem vordefinierte Kamerapositionen (Cockpit-View, ...), Transparenzeinstellungen oder eine bestimmte Zusammensetzung an 2D-Anzeigen vorgeschlagen, für die in der späteren Benutzung einer solchen 3D-Ansicht Bedarf bestehen könnte. Gerade die Möglichkeit, eigene Kamera- und Transparenz-Konfigurationen abzuspeichern, wurde als zentrales Element an dieser Stelle hervorgehoben. Da es sich jedoch um einen Prototyp zur Ermittlung der Möglichkeiten einer 3D-Visualisierung handelt und nicht um ein fertig entwickeltes Programm, kann diese Kritik zur weiteren Entwicklung genutzt werden, darf aber nicht als Negativpunkt in die Bewertung des Prototyps einfließen.
Insgesamt lassen die erhaltenen Antworten darauf schließen, dass eine entsprechende 3D-Visualisierung kein umfangreiches Konfigurationsmenü benötigt. Vielmehr sind der schnelle und einfache Zugriff auf wichtige Steuerungselemente, wie die Erstellung von 3D-Anzeigen (Abbil-

dung 6.3(b)) und die Erstellung und Verwaltung von abgespeicherten Konfigurationen wichtig. In Verbindung mit einer gut strukturierten Auflistung aller verbauten Modell-Elemente würde also ein einfach gehaltenes Menü die Benutzbarkeit der Visualisierung fördern.

6.3.5 Abschließende Bemerkungen

Am Ende dieses Teils der Evaluation wurde den Teilnehmern die Möglichkeit gegeben, weitere Bemerkungen zu machen und sonstiges Feedback zum bisher gesehenen Prototyp zu geben. Dies sollte dazu dienen, noch weitere Bewertungen zu erhalten, die bis jetzt nicht von einem der Fragepunkte abgedeckt wurden, aber dennoch wichtig sein könnten.
Als zentraler Punkt, der von allen Befragten angesprochen wurde, stellte sich schnell die *Steuerung der 3D-Ansicht* heraus, also die Bewegung der Kamera im 3D-Raum. Diese wurde von durchweg alles als schwierig beurteilt, jedoch wurde ihr auch durchaus eine gewisse Erlernbarkeit bescheinigt. Es war also zu Beginn jedesmal schwierig für die Nutzer, die Kamera auf die gewünschte Position zu bringen und den gewünschten Blickwinkel zu erreichen. Dabei machte vor allem die gleichzeitige seitliche Bewegung in Verbindung mit einer Rotation der Sicht größere Probleme. Jedoch wurden diese Probleme weniger, je länger die Nutzer die Möglichkeit hatten, mit der Kamera zu experimentieren. Ein interessanter Aspekt, der in diesem Zusammehang auftrat, ist die Abhängigkeit der Probleme bei der Kamerasteuerung, mit der Erfahrung der einzelnen Personen im Bereich der Videospiele, speziell im Bereich der Ego-Shooter. Dies lässt sich dadurch erklären, dass die gewählte Kameraführung in wesentlichen Punkten mit denen eines Ego-Shooters übereinstimmt, wie sie in Kapitel 2.5.3 als „First person camera“ beschrieben wurde. Je mehr Erfahrung man mit dieser Steuerung also bereits hat, desto einfacher fällt am Anfang auch die Steuerung der Kamera des 3D-Prototyps.
Ein anderer Punkt, der angesprochen wurde, ist die *Orientierung im 3D-Raum*. Es wurde vorgeschlagen dem Benutzer, egal in welcher Position sich die Kamera aktuell befindet, einen 3D-Würfel anzuzeigen, der visualisiert, von welcher Seite der Nutzer gerade auf das Modell schaut und der auch eine Rotation der Kamera um den Modell-Mittelpunkt ermöglicht. Als Beispiel kann hier gut ein vergleichbarer Ansatz von 3ds Max [31] dienen, da hier exakt dieser Vorschlag umgesetzt ist und sowohl die Orientierung in der 3D-Szene als auch die Steuerung der Kamera erleichtert. Eine weitere Idee in diese Richtung ist das Anbieten von Rotationspfeilen auf dem Bildschirm, die mit der Maus ausgewählt und per Klick aktiviert werden können. Sie stellen dann eine deutlich vereinfachte Art der Kamerasteuerung dar, die aber gerade den Einstieg in die Benutzung eines 3D-Programms vereinfachen kann, wenn der Nutzer ansonsten noch keine Erfahrungen in diesem Bereich hat.
Interessant ist es an dieser Stelle festzuhalten, dass die meisten der Befragten bis jetzt noch keinen genauen *Use-Case für eine solche 3D-Visualisierung* sehen konnten. Zwar wurden vereinzelt Präsentationen oder Schulungen als möglicher Einsatzort für ein solches Programm genannt, aber einen konkreten Einsatz im Bereich der Analyse oder Diagnose konnte sich keiner so recht vorstellen. Auch eine Verbindung der 3D-Darstellung mit vorhandenen 2D-Visualisierungen wurde bis zu diesem Zeitpunkt nur ein einziges Mal angesprochen, und selbst dabei eher ohne eine konkrete Vorstellung eines genauen Einsatz-Szenarios. Als dann jedoch die Frage nach der 3D-Ansicht als Ergänzung zu vorhandenen Tools gestellt wurde, nannten alle die Möglichkeiten zur Vereinfachung der Einarbeitung ins Stoffgebiet „Bordnetz-Kommunikation“ und zur Besprechung von Lösungen und Änderungen am Bordnetz. Es wurde allerdings auch klar, dass alle Beteiligten die Nützlichkeit einer 3D-Visualisierung an den zu bearbeitenden Daten - also wiederum dem jeweiligen Use-Case - festmachen. Dabei wurde festgestellt, dass vor allem Daten, die einen direkten Bezug zum Fahrzeugstatus haben, von einer 3D-View profitieren würden. Bei anderen Daten ist eine solche Ansicht, nach Aussage der Befragten, nicht nötig, da für diesen Use-Case dann einfach zu viele unwichtige Informationen präsentiert würden.

6.4 Test des prototypischen MCV-Systems

Nach Abschluss der Befragung zur allein dargestellten 3D-Ansicht, wurde die 3D-Ansicht in einem MCV-System zusammen mit der AutobahnView [13] gestartet. Den Teilnehmern wurde kurz die Funktionsweise der AutobahnView und der Verbindung zwischen beiden Ansichten erklärt und anhand eines einfachen Beispiels demonstriert. Bereits jetzt waren einige User sehr positiv von der Zusammenstellung der unterschiedlichen Visualisierungen überrascht und fanden das Zusammenspiel der beiden Visualisierungen „sehr geil“. Es wurde auch von einem Nutzer ein Vergleich mit einem Programm von BMW gemacht, das eine ähnliche Funktionsweise besitzt, allerdings den Fahrzeugstatus mit Hilfe zweidimensionaler Grafiken darstellt.
Nach der ersten Erklärung der Funktionsweise, wurden die Teilnehmer gebeten, selbst ein wenig mit dem Programm zu experimentieren und wurden danach gefragt, *wozu sich die 3D-Ansicht gut eigne* bzw. was diese Art der Darstellung besonders gut präsentieren kann. Wie schon beim Test der 3D-View ohne MCV-System, wurde auch hier die gute Erkennbarkeit von mechanischen Veränderungen am Fahrzeug gelobt. Die Art des Zusammenspiels würde das Verständnis für die vorliegenden Daten steigern und gerade bei komplexen mechanischen Geometrien sei eine solche Kombination von Visualisierungen extrem praktisch. Dies lässt zum einen darauf schließen, dass eine einzelne 3D-Visualisierung eher schlecht geeignet ist um einen vollständigen Funktionsumfang zur Kommunikaitons-Visualisierung zu bieten, aber im Verbund mit anderen Visualisierungen ihre Stärken ausspielen kann, während die Schwächen von anderen Visualisierungen abgedeckt werden.
Auf die Frage, ob die *3D-Ansicht in einem MCV-System verwirrend* wirken könnte, gaben viele Teilnehmer an, dass dies nur der Fall sei, wenn die 3D-Ansicht allein genutzt würde - gerade durch das MCV-System würde diese Art der Darstellung einen Sinn bekommen. Es gab die allgemeine Äußerung, dass die Ansicht eigentlich in keinem denkbaren Fall verwirrend wirken würde, da ja in einem MCV-System auch stets noch andere Visualisierungen vorhanden seien und man somit nicht allein auf die 3D-Darstellung beschränkt sei. In eine etwas andere Richtung ging der Kommentar eines Nutzers, der an dieser Stelle gemacht wurde, dass Analyse-/Diagnose-Experten auch eine „kryptischere“ Ansicht nützlich finden könnten. Da es sich bei diesen Nutzern um Experten ihrer jeweiligen Fachgebiete handelt, sei es nicht von der Hand zu weisen, dass eine spezielle 3D-Visualisierung, die nicht zwangsläufig ein Fahrzeugmodell enthalten muss, genauso nützlich oder vielleicht sogar noch nützlicher sei.
Die Vorgehensweise beim anschließenden *Lösen der gestellten Aufgaben* war eigentlich bei allen Nutzern identisch - sie navigierten die Kamera vor oder über das virtuelle Fahrzeug und richteten den Blickwinkel auf das Fahrzeug aus. Anschließend navigierten sie mit der Maus über die Zeitleiste und beobachteten das Modell und mögliche Statusveränderungen. Auf diese Weise konnten alle Teilnehmer die erste Aufgaben ohne größere Probleme und Hilfestellungen lösen und fanden den Zeitpunkt der ersten Türöffnung sehr schnell.
Etwas schwieriger war es für die Probanden im Anschluss daran herauszufinden, ob in den gegebenen Daten zu irgendeinem Zeitpunkt Blinker aktiv sind und wenn ja, welche Blinker und zu welchem Zeitpunkt. Hier waren die Nutzer im Vorteil, die bereits eine Kameraposition vor dem Fahrzeugs gewählt hatten, da sie beide Frontblinker beobachten konnten. An diesen versuchten herauszufinden, durch Navigieren über die Zeit, wann die Blinker aktiv sind. Die anderen Teilnehmern mussten die Kamera noch neu positionieren, wählten aber auch hier einen Blickpunkt, der die Übersicht über beide Blinkerseiten gewährleistete. Nach kurzer Zeit hatten alle Teilnehmer diese Aufgabe gelöst, allerdings konnte beobachtet werden, dass der Kontrast zwischen aktivem Blinker und dem Rest des Fahrzeugs - je nach Blickwinkel und Beleuchtungssituation - nicht sonderlich hoch war, weswegen das Finden des richtigen Zeitpunkts manchmal schwierig war.
Die letzte zu lösende Aufgabe bestand darin den Zeitpunkt zu bestimmen, wann das Steuergerät DME1 einen Auslastungswert von 800 erreicht. Ohne großes Zögern wurde von den Testpersonen das Konfigurationsmenü geöffnet, um eine 2D-Anzeige des Auslastungswerts von DME1 zu

erstellen. Bei dieser Aufgabe erstellten alle Teilnehmer umgehend eine 2D-Anzeige und scrollten anschließend über die Zeitleiste in den Daten bis sie den Zeitpunkt gefunden hatten. Erstaunlicherweise stellten zwei Nutzer vorher die Transparenz der mechanischen Fahrzeugteile ein, um einen besseren Blick auf das Bordnetz zu erhalten. Dies kann als Zeichen interpretiert werden, dass die Idee der 2D-Anzeigeflächen zu diesem Zeitpunkt noch nicht vollständig klar gewesen ist.

6.5 Überprüfung der Hypothesen

Die durchgeführte Evaluation sollte die zu Beginn des Kapitels 6 beschriebenen Hypothese überprüfen und dadurch eine genaue Bewertung des Prototyps ermöglichen. Im folgendenen Abschnitt werden nun die Hypothesen einzeln betrachtet und die Ergebnisse der Evaluation dazu genutzt, um die Hypothesen entweder zu bestätigen oder zu widerlegen.

6.5.1 Überprüfung der Fahrzeugreaktions-Hypothese

Die erste Hypothese sagte aus, dass eine 3D-Visualisierung gut geeignet sei, um *mechanische Fahrzeugreaktionen* zu visualisieren und sie dem Benutzer gut erkennbar darzustellen. Diese Hypothese kann, bis auf eine kleine Einschränkung, als bestätigt angesehen werden. Die Nutzer haben durchweg ausgesagt, dass der Fahrzeugzustand gut zu erkennen sei und man selbst komplexe mechanische Veränderungen gut anhand des Modells nachvollziehen kann. Die angesprochene Einschränkung bezieht darauf, dass kleine Veränderungen am Modell, wie eine schließende Zentralverriegelung, zum Teil nur schwer zu erkennen seien, so die Befragten. Dasselbe Problem wurde bei farblichen Änderungen, wie zum Beispiel beim Blinken, gesehen, wenn der farbliche Kontrast zwischen Umgebung und Verfärbung gering ist. Beide Einschränkungen haben ihren Ursprung in der Implementierung des vorliegenden Prototyps, da Aktionen im Modell nicht automatisch hervorgehoben werden und bei farblichen Veränderungen keinerlei andere Hervorhebungstechniken zum Einsatz kommen. Es wäre also noch zu untersuchen, inwieweit sich diese Probleme beseitigen oder minimieren lassen. Ansonsten wurde die Hypothese vollständig bestätigt, da die befragten Experten der Visualisierung durchweg eine gute Erkennbarkeit von mechanischen Zuständen und Veränderungen bescheinigten.

6.5.2 Überprüfung der Bordnetz-Hypothese

In der zweiten Hypothese wurde die Behauptung aufgestellt, die 3D-Visualisierung eigne sich, ergänzt durch 2D-Anzeigen, außerdem auch gut für die Visualisierung von *Bordnetzkommunikation*. Die Überprüfung hat hierbei ergeben, dass diese Hypothese zwar als bestätigt gesehen werden kann, jedoch wurde während der Evaluation klar, dass die 2D-Visualisierungen in diesem Bereich noch viele Entwicklungsmöglichkeiten bieten. Die angebotenen Datenvisualisierungen in Form konkreter Werte bzw. eines selbstskalierenden Wertegraphen wurden gut aufgenommen und auch die zugehörige Aufgabe im letzten Teil der Evaluation wurde problemlos gelöst. Jedoch wurde von einem Experten angemerkt, dass diese Form der Visualisierung noch mehr Potential bereithalte und sich die 2D-Visualisierung in den Anzeigeflächen nicht auf die gezeigten Formen beschränken müsse. Es bleibt also festzuhalten, dass sich die 2D-Anzeige zwar scheinbar gut zur Darstellung von Bordnetzkommunikation bzw. zugehöriger Daten (Auslastung, Bus-Last, ...) eignen, aber es auch noch weitere Möglichkeiten in diesem Bereich gibt.

6.5.3 Überprüfung der MCV-Hypothese

Die letzte zu prüfende Hypothese ging auf die *Integration einer 3D-Visualisierung in vorhandene 2D-Visualisierungstools* aus der Analyse bzw. Diagnose ein. Diese Hypothese wurde durch die qualitative Evaluation vollständig bestätigt. Alle Nutzer waren von der Test-Konfiguration, mit

der AutobahnView[13] als 2D-Visualisierung und der 3D-Visualisierung aus Ergänzung, begeistert. Vor allem in Bereichen, in denen mechanische Reaktionen am Fahrzeug eine entscheidende Rolle bei der Analyse und Diagnose spielen, scheint eine 3D-Visualisierung ein sehr guter Ansatz zu sein, um eine große Menge an Daten sinnvoll aufzubereiten. Die Anmerkung einer Testperson, durch die 3D-Darstellung entfalle das mühsame Herauslesen des Status aus den Trace-Daten, bestätigt diese Vermutung.

7 Future Work

Diese Diplomarbeit beschäftigt sich mit einer 3D-Visualisierung im Bereich der Analyse und Diagnose von Bordnetzkommunikation. Dabei konnten viele Bereiche des komplexen Themengebiets der 3D-Visualisierung von Bordnetzkommunikation nicht vollständig evaluiert und entwickelt werden. Es gibt daher noch eine Vielzahl von Möglichkeiten, die von dieser Arbeit nicht abgedeckt sind. Diese Möglichkeiten und die weitere Entwicklung des implementierten Prototyps werden in diesem Kapitel umschrieben, um eine Idee davon zu vermitteln in welche Richtung sich die 3D-Visualisierung in diesem Themengebiet noch entwickeln kann.
Im Rahmen der Diplomarbeit wurde ein Konzeptkatalog entwickelt, der eine Vielzahl von Konzepten beinhaltet, die auf unterschiedliche Weise den Nutzer bei der Arbeit mit Bordnetzdaten unterstützen sollen. Da jedoch lediglich ein Teil dieser Konzepte im letztendlichen Prototyp berücksichtigt wurden, gibt es an dieser Stelle noch viele Möglichkeiten, die vorgestellten *Konzepte oder weiterführende Ideen zu entwickeln und zu evaluieren*. Besonders interessant sind dabei die unterschiedlichen Arten der Kodierung von Auslastung direkt im Modell. Gerade der Ansatz des radialen Farbverlaufs zur Visualisierung der Auslastungswerte der letzten Sekunden scheint eine relativ neue Idee zu sein und daher wäre eine Evaluation dieses Konzepts ein möglicher nächster Schritt bei der Weiterentwicklung einer 3D-Visualisierung. Ein anderes Konzept, das die zum Teil komplizierte Steuerung einer 3D-Kamera vereinfachen kann, ist die Nutzung eines realen Fahrzeugmodells zur Positionierung der Kamera mit Hilfe von AR-Markern. Dieses Konzept wurde zwar von allen Beteiligten eher als Spielerei abgehandelt, jedoch zeigen die enormen Steuerungsprobleme mit der Kamera, dass es auf diesem Gebiet weiteren Forschungsbedarf zu geben scheint.
Auch wenn einige der Konzepte bereits in Richtung *Usability* gehen, so ist auch hier noch einige Arbeit erforderlich, um die gezeigten Visualisierungskonzepte in aktuelle Analyse-/Diagnose-Umgebungen zu integrieren und über das prototypische Stadium hinaus benutzbar zu machen. Dabei sind vor allem die Menüführung und die Filterungsoptionen innerhalb des Menüs zu optimieren. Gerade im Bereich der Auflistung der Bauteile und Bordnetzkomponenten kann eine durchdachte Struktur viel in Richutng Usability erreichen. Ein weiterer Punkt wäre die Bereitstellung diverser vordefinierter Konfigurationen für die 3D-Visualisierung, so dass zum Beispiel auf Anhieb bestimmte Sichtpositionen geöffnet werden. Eine Erweiterung dessen wäre dann die Möglichkeit, eigene Sichtkonfigurationen abspeichern und später erneut laden zu können.
Ein anderer Schritt kann auch die *Weiterentwicklung* des jetzigen Prototyps und das Ergänzen neuer Funktionen sein. Ein mögliches Beispiel hierfür ist die Integration von Audio-Rückmeldungen für den Benutzer, da dieses Konzept während der Konzeptevaluation sehr gut angenommen wurde. Hier wäre dann zu untersuchen, ob eine realisitische Soundkulisse oder einfache Warntöne den größeren Mehrwert für den Benutzer bringen. Bei der realistischen Vertonung wäre auch besonderer Wert darauf zu legen, dass die Bilder und Töne synchron ablaufen, um zu verhindern, dass es durch zeitlich unpassende Geräusche zur Verwirrung des Nutzer führt. Ein anderer Punkt bei der Weiterentwicklung ist die Erhöhung der Performance des Prototyps. Zwar ist es auf aktuellen Rechnern problemlos möglich, mehrere 3D-Ansichten parallel zu betreiben, jedoch benötigen diese einen verhältnismäßig hohen Teil der Rechenleistung. Gerade im Hinblick auf den Einsatz in einem MCV-System sollte hier, zum Beispiel durch Instanziierungen des Fahrzeugmodells, versucht werden eine bessere Performance zu erreichen.

Inhalt der beigelegten CD

Literatur

[1] M. Baldonado, A. Woodruff, A. Kuchinsky: Guidelines for using multiple views in information visualization.. Proceedings of the working conference on Advanceed visual interfaces, 2000

[2] D. A. Bowman, J. Chen, C A. Wingrave, J. Lucas, A. Ray, N. F. Polys, Q. Li, Y. Haciahmetoglu, J. Kim, S. Kim, R. Boehringer, T. Ni: New Directions in 3D User Interfaces.. Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality

[3] M. Christie, R. Machap, J.-M. Normand, P. Olivier, J. Pickering: Virtual Camera Planning: A Survey. Proceedings of the 5th International Symposium on SmartGraphics, 2005

[4] A. Davison: Pro Java 6 3D Game Development.. Apress Verlag, 2007

[5] N. Elmqvist: BalloonProbe: reducing occlusion in 3D using interactive space distortion.. Proceedings of the ACM symposium on Virtual reality software and technology, SESSION: Interaction and design – II, pages: 134 - 137, 2005

[6] N. Elmqvist, P. Tsigas: A Taxonomy of 3D Occlusion Management for Visualization. IEEE Educational Activities Department, 2008

[7] C. G. Healey, K. S. Booth, and J. T. Enns: High-Speed Visual Estimation Using Preattentive Processing ACM Transactions on Computer-Human Interaction, Vol. 3, No. 2, June 1996

[8] K. Husoy, C. Skourup: 3d visualization of integrated process information.. NordiHCI, 2006

[9] Ji-Young Oh, W. Stuerzlinger: Moving Objects with 2D Input Devices in CAD Systems and Desktop Virtual Environments.. Proceedings of Graphics Interface, 2005

[10] M. Kallmann, D. Thalmann: Direct 3D interaction with smart objects.. VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology

[11] D. A. Keim: Information Visualization and Visual Data Mining.. IEEETransactions On Visualization And Computer Graphics, Vol. 7, No. 1, 2002

[12] U. Kuckartz, T. Dresing, S. Rädiker, C. Stefer: Qualitative Evaluation. Der Einstieg in die Praxis.. VS Verlag für Sozialwissenschaften, 2008

[13] B. Kunze: Diplomarbeit: Time-Based Visualization of Car Communication Processes.. LMU München, LFE Medieninformatik

[14] K. Larson, M. van Dantzich, M. Czerwinski, G. Robertson: Text in 3D: Some Legibility Results. Conference on Human Factors in Computing Systems, 2000

[15] M. Limniou, D. Roberts, N. Papadopoulos: Full immersive virtual environment CAVETM in chemistry education Computers & Education, Volume 51 , Issue 2, September 2008

[16] A. Mack, I. Rock: Inattentional Blindness: An Overview.. Psyche 5(3), 1999

[17] M. Mozdari: Diplomarbeit: Konzept zur 3D-Visualisierung von Fahrzeugbordnetzarchitekturen TU München, Lehrstuhl für Mensch-Maschine-Kommunikation

[18] M. Nnadi, U. Fischer, M. Boyce, M. Nitsche: Effect of Dynamic Camera Control on Spatial Reasoning in 3D Spaces. Sandbox '08: Proceedings of the 2008 ACM SIGGRAPH symposium on Video games

[19] C. North, B. Shneiderman: Snap together visualization: a user interface for coordinating visualizations via relational schemata.. Proceedings of the working conference on Advanceed visual interfaces, 2000

[20] M. Pastko, A. Mackie, M. Simon: Virtual Environments for the use of Data Visualization.. Envision Center for Data Perceptualization, CGT 411, Purdue University

[21] B. Preim, A. Raab, T. Strothotte: Coherent Zooming of Illustrations with 3D-Graphics and Text.. Proceedings of Graphics Interface '97, pages 105-113, Canadian Information Processing Society, 1997

[22] B. Preim, F. Ritter: Techniken zur interaktiven Hervorhebung von Objekten in medizinischen 3d-Visualisierungen Magdeburg, 2002

[23] Robert Bosch GmbH: Autoelektrik / Autoelektronik.. Vieweg Friedr. + Sohn Verlag, 2002

[24] Robert Bosch GmbH: Kraftfahrtechnisches Taschenbuch.. Vieweg+Teubner, 2007

[25] K. Ruhland, S. Bioletti, C. O'Sullivan, H. Hußmann: LibViz: A visualisation toolkit to support the Preservation of the Old Library Eurographics Ireland, pp60 - 66, 2007

[26] M. C. Stone: Color in Information Display Principles, Perception, and Models. Course 20, SIGGRAPH 2004

[27] M. Tory, T. Möller: Human Factors in Visualization Research.. IEEETransactions On Visualization And Computer Graphics, Vol. 10, No. 1, 2004

[28] E. R. Tufte: Envisioning Information. Graphics Press, 2001

[29] J. Viewga, M. J. Conway, G. Williams, R. Pausch: 3D Magic Lenses.. Proceedings of the 9th annual ACM symposium on User interface software and technology, 1996

[30] S. Zhai, W. Buxton, P. Milgram: The „Silk Cursor": Investigating Transparency for 3D Target Acquisition Human Factors in Computing Systems, Boston, 1994

Web-Referenzen

[31] 3ds Max, Autodesk [http://www.autodesk.de/adsk/servlet/index?siteID=403786&id=10612077], accessed December 30, 2008.

[32] About JME http://www.jmonkeyengine.com/wiki/doku.php?id=about_jme, accessed January 11, 2009.

[33] Adobe Flash http://www.adobe.com/de/products/flash/, accessed January 11, 2009.

[34] ARToolKit [http://www.hitl.washington.edu/artoolkit/], accessed December 28, 2008.

[35] CANoe, Vector [http://www.vector.com/vi_canoe_de.html], accessed December 25, 2008.

[36] Colin McRae Dirt, Codemasters 2008 [http://www.codemasters.de/dirt], accessed December 23, 2008.

[37] CyberVRML'97 - Virtual Reality Modeling Language Development Library, CyberGarage http://www.cybergarage.org/vrml/cv97/, accessed January 9, 2009.

[38] Far Cry 2, Ubisoft 2008 [http://farcry.de.ubi.com/], accessed December 23, 2008.

[39] Global-i from Infomagnet - 3Dglobe full of world data [http://infoview.infomagnet.com/view.php], accessed December 19, 2008.

[40] Google Earth [http://earth.google.de/, accessed January 9, 2009.

[41] Google SketchUp [http://sketchup.google.com/, accessed January 1, 2009.

[42] „JAVA3D am Beispiel eines Getriebemodells", TU Chemnitz [http://archiv.tu-chemnitz.de/pub/2001/0069/data/java3d.html, accessed January 4, 2009.

[43] Java 3D File Loaders http://java3d.j3d.org/utilities/loaders.html, accessed January 11, 2009.

[44] JAXB Reference Implementation Project https://jaxb.dev.java.net/, accessed January 12, 2009.

[45] JMonkeyEngine [http://www.jmonkeyengine.com, accessed January 1, 2009.

[46] JOGL - Java Binding for the OpenGL API https://jogl.dev.java.net/, accessed January 11, 2009.

[47] LWJGL - Lightweight Java Game Library http://lwjgl.org/, accessed January 11, 2009.

[48] OpenGL - The Industry's Foundation for High Performance Graphics http://www.opengl.org/, accessed January 12, 2009.

[49] OpenGL ARB - „Architecture Review Board" http://www.opengl.org/about/arb/, accessed January 12, 2009.

[50] Review of High Speed Visual Estimation Using Preattantive Processing [http://www.cs.umd.edu/class/spring2002/cmsc838f/preattentive.ppt#267], accessed December 22, 2008.

[51] School of Computer Science: Palaeoceanography and Climate Visualisation Models: EarthSeaTrilogy [http://www.hpv.cs.bangor.ac.uk/palaeo.php], accessed December 19, 2008.

[52] The CAVE Virtual Reality System [http://www.evl.uic.edu/pape/CAVE/], accessed December 25, 2008.

[53] Tomb Raider, Eidos Interactive 1996 [http://www.rawgamer.com/games/de/pc/tomb-raider.html], accessed December 23, 2008.

[54] VirTools, A Dassault Systèmes Technology [http://www.virtools.com/], accessed December 30, 2008.

[55] VECTOR http://www.vector.com, accessed January 12, 2009.

[56] Virtual Earth, Microsoft [http://local.live.com], accessed December 24, 2008.

[57] World of Warcraft, Blizzard Entertainment http://www.wow-europe.com, accessed January 12, 2009.

Anhang

Anhang A: Ideensammlung

Interaktion / Steuerung
Maus ↔ Tastatur ↔ 3D-Input

- 2D Eingabegerät schneller & genauer als 3D-Eingabegerät (vgl. „Guidelines für 3D Positioning Techniques“, Teather et al, 2007) bzgl. Auswahl, Verschieben und Platzieren von Objekten im 3D-Raum
-

Art	Beschreibung	Anmerkung
Maus	„Smart Objects“ zeigen dem User Interaktionspunkte an (Bsp.: bei MouseOver erscheint bei Fensterheber eine virt. Hand, die entsprechende Aktion anzeigt), bei Klick wird Aktion ausgeführt; mehrere Aktionen mit Mausrad durchscrollen → „intelligente Objekte“	siehe Paper!
Maus	Maus reagiert auf Interaktionsflächen; bei MouseOver verändert sich der Zeiger; bei anschließendem Klick wird Menü mit möglichen Aktionen eingeblendet (Bsp.: MouseOver Fensterheber → **klick** → Menüeinblendung an der Position & Auswahl durch User → **klick** löst Aktion aus	Menü 2D (Kreisdiagramm) oder 3D (vgl. Tomb Raider Inventar; 3D-Kreis/Flip) möglich
Tastatur	Auf Knopfdruck werden die Interaktionspunkte (Schalter, Hebel...) kurzzeitig markiert (blinken, farblich kennzeichnen,...)	Vgl. Interaktionspunkte in Point'n'Click Adventures hervorheben
Maus	SG und Busse können auf Auswahlflächen innerhalb des 3D-Fensters gezogen werden, um anschließend im MCV weitere Details über die Bauteile/Busse zu erhalten	Drag'n Drop
Tastatur	CommandLine-Steuerung, z.B. zum schnellen Einfärben eines Busses, Ausblenden von Bauteilen etc...	Zur besseren Integration könnten Maus-Anweiseungen „live“ in CommandLine-Anweisungen umgewandelt & angezeigt werden
3D-Input	User kann mit virtueller Hand im Fahrzeug navigieren; alle 6 Freiheitsgrade durch einen Inputdevice abgedeckt	

Kamera

statisch ↔ dynamisch; 2D ↔ 3D (Display)

- ein Sichtfeld auf das 3D-Modell
- Kameraparameter
  - Zoom
  - Ausrichtung
  -

Art	Beschreibung	Anmerkung
Dynamisch, 2D/3D	Frei steuerbare Kamera, Steuerung ähnlich wie bei einem Ego-Shooter; Pfeiltasten = vor, zurück, Drehung links, rechts; Maus = Drehung, Neigung; Benutzer „fliegt“ durch die 3D-Szene und kann sich frei positionieren	
statisch, 2D/3D	Kameraposition ist fix (z.B. Cockpit, Außen vorne, Außen links, ...), die per Button oder Shortcut durchgeschaltet werden können (inkl. Anzeige welche Kamera aktuell gewählt ist); dabei optional: weiterhin Drehung der Kamera mit Maus möglich	
statisch, 2D/3D	Baugruppen können nach Kategorien gefiltert und komplett ausgeblendet werden → Verringerung der optischen Dichte von Bauteilen und somit weniger Verdeckung	siehe Occlusion!
dynamisch, 2D/3D	Passive bzw. nicht-aktive Elemente werden (nahezu) durchsichtig gerendert; erst bei Aktivität (z.B. Nachricht pro Sekunde) wird die Transparenz stückweise zurückgesetzt; ggf. pulsierende Veränderung oder in Verbindung mit geringer Größenfluktuation	
dynamisch, 2D/3D	User erhält um die Mausposition herum die Möglichkeit ein kugelförmiges Kraftfeld aufzubauen, dass Objekte von ihrem ursprünglichen Platz verdrängt und an der Feldoberseite platziert (auf direkter Linie vom Feldmittelpunk durch die original Position zum Feldrand); ursprüngliche Lage wird durch Gitternetz-Objekte symbolisiert; kann anschließend um Sphäre herum navigieren und so verdeckte Bauteile erkennen/auswählen	Siehe BalloonProbe
Dynamisch, 2D/3D	User steuert Lupe/Linse, die eine genaue Sicht auf SG-Topologie etc. erlaubt; außerhalb der Linse werden die SG etc. nur rudimentär (wenn überhaupt bzw nur bei Aktivität oder Selektion...) angezeigt	Siehe FishEyeView

dynamisch, 2D/3D	Fahrzeug und Bauteile sind semi-transparent, um eine bessere Navigation & Orientierung zu ermöglichen	Verstärkung des Effekts durch 3D-Brille
dynamisch	Optionale automatische Kameraführung/-einrichtung (vgl CamPlan Paper)	Benötigt griffige Optionen zur Definition des gewünschten Views („communicative goals“)

Informationsvisualisierung

statisch ↔ dynamisch

- Möglichkeiten zur InfoVis:
  - Farbe, Farbverlauf
    nur begrenzte Farbanzahl
  - Textur
    versch. Texturen für versch. Zustände
  - Größe, Größenveränderung, Größenfluktuation
    Größe je nach Auslastung
  - Ausrichtung, Rotation (nur sehr bedingt möglich!)
  - Form, Formveränderung (Pulsieren,...)
  - Transparenz
    Ausblenden von Informationen; Verdeckung!
  - Umgebung
    Abstrahlen von Bauteilen
  - Audio (Tonaufnahmen [Tür auf/zu; Blinker])

- darzustellende Informationen:
  - 3D-Animationen in Echteit (Tür auf/zu,...)
  - 3D-Positionierung von SG und Bussen/Kabeln
  - Nachrichtenfluss bzw. Nachrichtenmenge
  - Auslastungen
    - Netz
    - SG
  - Welche FBs in welchem SG
  - Vorgänger/Nachfolger von FB/SG (MCV)

- vorhandene Symbole / Metaphern in bisherigen Programmen?

Art	Beschreibung	Anmerkung
gemischt	2D-Leinwand in 3D-Umgebung integriert; dort werden Nachrichten ausgewählter Steuergeräte o.ä. Angezeigt; inkl. Visualisierung des Nachrichtenflusses im 3D-Modell	Leinwand stellt Vermischung des MCV dar
dynamisch	3D-Lupe/-Linse wird genutzt, um Fahrzeug partiell durchsichtig zu machen; somit gezielter Einblick in SG-Topologie in bestimmtem Bereich; alternativ: Fahrzeug und SGs semi-transparent, wird durch Lupe aufgehoben → Objekte innerhalb der 3D-	Vgl. „3D-Magic-Lense“, „The Partial-Occlusion Effect“

	Lupe vollkommen sichtbar	
statisch	FB innerhalb einer ECU werden per Textur dargestellt, auf jeder Seite dieselbe Textur / Ansicht	
statisch	ECU setzt sich aus FB-Blöcken zusammen (Blockgröße abhängig von Codegröße/Auslastung), die den ECU Block ergeben); Größe der FB-Blöcke kann sich dynamisch ändern (wenn z.B. Auslastung visualisiert wird)	
dynamisch	Darstellung des Nachrichtenflusses/Datenmenge durch längliches Einfärben des Kabels; Farbkodierung bzgl. Systemzugehörigkeit, Echtzeitanforderung etc.	
dynamisch	Verdickung des Kabels entsprechend der Auslastung; Je mehr SG sendet, desto weiter wächst Verbreitung auf den Bus → komplett „dicker“ Bus = Überlastung	Zusätzlich einzelne Nachrichten als Symbole ober-/unterhalb der Leitungen möglich (vgl. BT-Client)
dynamisch	Sensoraktivitäten werden durch animierte Wellenlinien in Sensornähe dargestellt oder Sensor pulsiert zwischen 2 Farben	Sound?
dynamisch	Bei nicht-sichtbaren Ereignissen (wegen Kameraposition) kleine Infosymbole einblenden; bei Klick kommt man zum „Ort des Geschehens“ (vgl. Strategiespiele)	
Statisch	Bezeichnungen & Statusinformationen über Tooltips („Sprechblase“ eines SG) an den SG darstellen oder als Textur auf den SG	Erweiterbar zu: Tooltips ausgewählter SG dauerhaft anzeigen mit Statusinformationen
dynamisch	2D-Messgeräte bzw. entsprechende Visualisierungen lassen sich via Drag'n'Drop auf SG oder Busse ziehen, um in Echtzeit Nachrichten, Auslastung etc. einsehen zu können	

Mechanische Visualisierung:

- Ideensammlung zur mechanischen Visualisierung:
  - !! Mechanik-Trace: einfache Trace Version mit schlichten Statusänderungen des Fzg-Zustands („Fahrertür geöffnet.“, „Blinker links betätigt.“)
  - Hervorheben von Elementen bei Aktion:
    - Tür bewegt sich und pulsiert kurze Zeit (farbig / transparent); direkt nach Beginn der Aktion schnelles Pulsieren, wird danach langsamer... → welche Aktion war zuerst da?
    - Elemente beginnt Aktion stark eingefärbt und verliert Färbung wieder gegen Ende der Aktion (Color Fading); ggf. über Transparenz? (Transparency Fading)
    - Bauteile lassen sich kategorisieren: „wichtig → blinkt lang/kräftig“, „weniger wichtig → blinkt kurz/schwach“, ...); dabei default-Einstellungen vorhanden; Bauteile lassen sich den verschiedenen Gruppen zuweisen (Baumstruktur auf der linken Seite vgl. Meriem gekoppelt mit der direkten Auswahl durch Anklicken)
    - Tooltips an Elementen bei Aktionen/Eintreten definierter Ereignisse
  - Elemente ziehen bei Bewegung einen Schatten hinter sich her (vgl. Mausssschatten bei Windows)
  - Sensorabfragen/-events sichtbar machen
    - Kreise verbreiten sich ausgehend vom Sensor und verblassen
    - Sensor blinkt/pulsiert
  - Auslagerung von Aktionen weg vom 3D-Modell:
    - Teile können auf seitliche Slots gezogen werden und werden dort einzeln angezeigt (Bsp.: Tür) → tritt das Bauteil in Aktion, wird sowohl im 3D-Modell als auch das einzelne Teil in der Slot Ansicht animiert
      → Benutzer kann Fokus auf einzelne – ihm wichtige – Bauteile setzen, die dann in keinem gewählten Blickwinkel verdeckt werden
  - Kamera schwebt halb-automatisch um Fzg herum und zeigt selbstständig die eingetretenen Ereignisse (Ereignisse sind selbst definierbar → Bsp.: Fenster fährt herunter)
  - kleines Aktivitätsfenster in dem Aktivitäten mit eigener Kameraperspektive dargestellt werden (vgl. Rückspiegel im Auto); pro (zuvor definierter) Aktion ein Fenster?
  - Darstellung von Nutzerinteraktion mit Fahrzeug durch virtuelle Hand:
    - Bsp.: Nutzer aktiviert per Knopfdruck das Radio → am Radioknopf erscheint kurzzeitig eine virtuelle Hand/Pfeil und symbolisiert den Knopfdruck
      → kleine unscheinbare Aktionen werden deutlich dargestellt

Anhang B: Evaluation vorhandener Tools

A) Was ist Ihre Meinung bzgl. einer 3D-Visualisierung (des Fahrzeugs / der BN-Kommunikation)?

Purdue University

1. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp?**

2. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp nicht?**

3. **Wie bewerten Sie die gewählten Visualisierungstechniken zur Darstellung der Auslastung (SG, Busse) und des Fahrzeugzustands? Haben Sie sonstige Anmerkungen zum gezeigten Prototyp?**

   - (Kontinuierliche) Darstellung des Fahrzeugstatus
   - Auslastung der Steuergeräte
   - Darstellung der BN-Topologie (u.a. Bus-Aktivität)
   - Detailgrad des Fahrzeugmodells
   - sonstiges:

DA-Tool von M. Mozdari

4. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp?**

5. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp nicht?**

6. **Wie bewerten Sie die gewählten Visualisierungstechniken zur Darstellung der Auslastung (SG, Busse) und des Fahrzeugzustands? Haben Sie sonstige Anmerkungen zum gezeigten Prototyp?**

   - Darstellung der einzelnen Bussysteme
   - Anzeige & Animation einzelner Nachrichten bzw. Funktionen
   - Darstellung der BN-Topologie
   - Verständnis der Zusammenhänge
   - Detailgrad des Fahrzeugmodells
   - Filteroptionen
   - sonstiges:

A) Können Sie sich eine Ansicht, die einem dieser Prototypen ähnelt, als Ergänzung zu vorhandenen Tools vorstellen?

Anhang C: Ergebnisse der Evaluation vorhandener Tools

Auswertung der User Study

Allgemeine Äußerungen zu 3D-Visualisierung vor der Tool-Nutzung:

- allgemein:
  - Erwartung, dass es positive Effekte hat (man kennt die Möglichkeiten von CAD-Modellen)
  - 3D eher für Navigation benutzen und nicht für InfoVis
  - selbst 2D zum Teil komplexer InfoVis nicht gewachsen
  - Gefahr: Design rutscht in den Vordergrund
  - Zielgruppe fraglich
  - eventuell mit Zeitbogen kombinieren
  - gut für Ausstellungen geeignet
  - Joy of View
  - gute Eingängigkeit (vgl Google Sketchup) nötig
  - „aha-Effekt“

- positiv:
  - Ort der SG wird dargestellt
  - Vernetzung gut zu erkennen
  - mechanisches Umgebungsmodell
  - Auslastung von Bussen direkt sichtbar/darstellbar
  - in Verifikationsphase gut sichtbar, ob Nachrichten rechtzeitig ankommen

- negativ
  - unübersichtlich bei Darstellung von Kommunikation
  - Unsinn für Entwicklung, da zu komplex
  - wichtige Informationen fehlen / können nicht dargestellt werden

Prototyp der Purdue University:

Was gefällt Ihnen?

- prinzipielle Kommunikationsdarstellung
- Region of Interest (durch Einfärbung)
- Mechanik und Elektronik kombiniert
- Messgrößen-Visualisierung durch Farben
- Ort der SG erkennbar und Aufbau des BN
- Verkabelungsaufwand ist halbwegs abschätzbar
- dynamische Darstellung → kaskadierende Nachrichten
- InfoVis gut gelöst
- SG werden bei Überlast/Fehlern rot eingefärbt → erkennt man sofort
- gute Steuerung / Bedienung
- gefällige Optik
- realitätsnah / hoher Detailgrad
- Transparenz gut eingesetzt
- involvierte SG, Busse etc. gut zu erkennen
- involvierte Komponenten gut zu erkennen

Was gefällt Ihnen nicht?

- mehr Details anzeigen (mehr Mechanik)
- wird bei gesamtem Bordnetz sicherlich unübersichtlich („selbsttragender Kabelbaum“) → fehlende Filtermöglichkeiten
- Problem der Skalierung
- (keine realen BN-Daten)
- Art der Interaktion mit dem Fahrzeug
- Aussehen der SG / Gateways
- andauernde Rotation stört
- eigentliche Nachrichten werden nicht angezeigt

(Kontinuierliche) Darstellung des Fahrzeugstatus

- gut
- Fenster schlecht zu erkennen
- ggf. 2D-Modell übersichtlicher? (→ Explosionszeichnung)
- Fehler/Fehlfunktionen schnell erkennbar
- eventuell Richtung Simulation verfolgen?
- Zustand des Fahrzeugs muss deutlich werden

Auslastung der SG

- gut
- intuitiv
- Metrik muss definiert werden (zum Teil 100% Auslastung gewünscht, dann nicht rot darstellen etc...)
- auch sinnvoll bei Bus-Systemen (z.B. CAN hat definierte maximale Auslastung)
- Icons außerhalb des Point of View besser
- Einfärbung ist gut und leicht zu erkennen
- Buslast zum Teil wichtiger als SG-Last
- Timing ist bei Echtzeit kritisch → ist zu schnell

Darstellung der BN-Topologie

- gut
- verständlich

- bei guter Pflege der Datensätze sehr sinnvoll: physikalische ↔ logische Integration
- muss übersichtlich bleiben
- SG-Position sollte stimmen
- für InfoVis weniger realistische Darstellung besser, ansonsten abhängig vom UseCase

Detailgrad des Fahrzeugmodells
- reicht aus
- kommt auf Funktionen an → räumliche Orientierung leidet nicht an weniger Details
- Detailgrad sollte konfigurierbar sein (→ bei Zoom mehr Details?)
- Darstellung des Kombi?
- Mehr Details, aber Transparenz sehr wichtig
- zu viele Details erschweren Sicht auf die Daten

Sonstiges:
- [zuerst die Buttons angeklickt statt Zahlen zu drücken]
- Konturlinien des Modells zum Teil für Verkabelung gehalten
- sollte im Verbund mit anderen Tools genutzt werden
- zuerst direkte Interaktion mit dem Modell vermutet
- flexiblere Anordnung wäre nicht schlecht
- Auswahl von Bauelement mit Anzeige aller betroffenen BN-Teile
- mehr Informationen über Nachrichten

Prototyp M. Mozdari:

Was gefällt Ihnen?

- SG-Ort sichtbar
- Nachrichtenfluss sichtbar
- für spezielle UseCases sehr sinnvoll (Lernprogramm, Schulung, ...)
- Struktur gut zu erkennen
- für Trace-Analyse geeignet
- Darstellung und Transparenz okay
- SG-Ort-Darstellung
- Fahrzeugdarstellung erlaubt guten Blick auf BN
- Filtermöglichkeiten

Was gefällt Ihnen nicht?

- Größerer Kontext bei Nachrichtenfluss nötig
- unübersichtlich
- dargestellte Inhalte besser in 2D visualisierbar
- Nachrichten vermutlich nur schwer erfassbar
- für Fehlersuche ungeeignet → Nachrichten zu schnell, man erkennt nichts
- reine Anzeige von Spezifikationen uninteressant
- Metapher: Nachricht geht aus SG rein/raus ist fraglich
- Filter ist umständlich zu bedienen
- Bedienung nicht intuitiv → mehr an Kundenfunktionen anlehnen
- mechanisches Modell fehlt
- SG-Optik
- Auslastung nicht dargestellt

Darstellung der einzelnen Bus-Systeme

- Abstraktion und Ort der SG wichtig – Verkabelung halb realistisch
- besser als Purdue: deutlich erkennbar
- keine Unterschiede zwischen den SG
- noch Infos zu einzelnen Teilen gewünscht
- wegen Namen unübersichtlich
- gut: Busse haben unterschiedliche Farben
- durch Winkel im Busverlauf gute Übersicht
- wichtig: Beschriftung

Anzeige & Animation einzelner Nachrichten & Funktionen

- unübersichtlich
- Nachrichten nur schwer zu erfassen
- Identifier bei CAN anzeigen
- Filter für spezielle Nachrichten und diese dann einfärben
- bei Fokus auf Nachricht gut, aber eher zu viele Informationen auf einmal
- wäre in Tabelle einfacher
- gut, aber mehr zusätzliche Informationen

Darstellung der BN-Topologie

- Verkabelung so okay, wird sonst zu kompliziert
- Unterschiede der Verkabelung (Aufbau der Busse) wichtig z.B. bei Kabelbruch
- Anordnung zu sehr an BN
- grobe SG-Position gut
- Kabelverlauf sollte realisitischer sein

- Verständnis der Zusammenhänge
- nicht gegeben (man sieht nicht was im SG passiert)
- grundsätzlich verständlich, jedoch sollte Aktivität im SG angezeigt werden
- „man macht mehr kaputt als gut“

Detailgrad des Fahrzeugmodells
- gewisser Trade-Off → kann ruhig detaillierter sein
- ist übersichtlich
- Leitungen abstrahiert
- bei abstrahiertem Leitungsverlauf auch abstraktes Modell okay
- Detailgrad kann wichtig sein (je nach Use Case)
- eher zu wenig, da kein mechanisches Modell
- ruhig etwas mehr Details

Filteroptionen
- gut
- definitiv nötig (eher noch mehr Filtermöglichkeiten)
- mehr farbliche Kodierung im Menü
- vielleicht Hover-Info
- nicht intuitiv
- Baum und Topologie gut, aber Symbolik nicht ganz klar
- man sollte alles zu einer Funktion einblenden können

Sonstiges:
- Filter verständlich & Steuerung problemlos
- muss intuitiver sein
- zu viele Nachrichten gleichzeitig
- Kamerapositionen definieren?
- Druckfunktion der aktuellen Anzeige
- BN-Darstellung in MCV integrieren mit 2D-Tool zur Topologiedarstellung

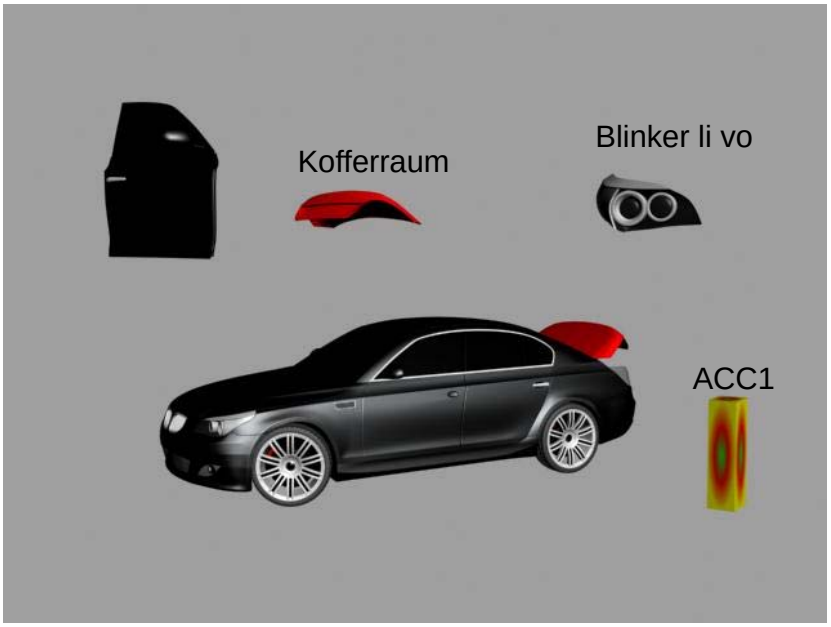
Können Sie sich eine Ansicht, die einem dieser Prototypen ähnelt, als Ergänzung zu vorhandenen Tools vorstellen?

- In einigen Bereichen: Änderungsmanagement, Werkstatt, Schulung („der Bringer“)
- nur für einfache Fälle in denen das mechanische Modell zur InfoVis genügt
- Fahrzeug-Architektur: Abhängigkeiten gut zu erkennen, SG-Änderungen ggf. gut zu erkennen
- Detailgrad sollte definierbar sein
- Zusammenhänge sollten durch 3D klar werden
- unter Vorbehalt, da „Last-Situation“ erst die eigentliche Eignung zeigt (Anmerkung: Zustandsvektor)
- 3D zur Netzplandarstellung geeignet, aber ohne Fahrzeug übersichtlicher (Anmerkung: Autobahnviews als Ergänzung!)

Anhang D: View-Konzepte

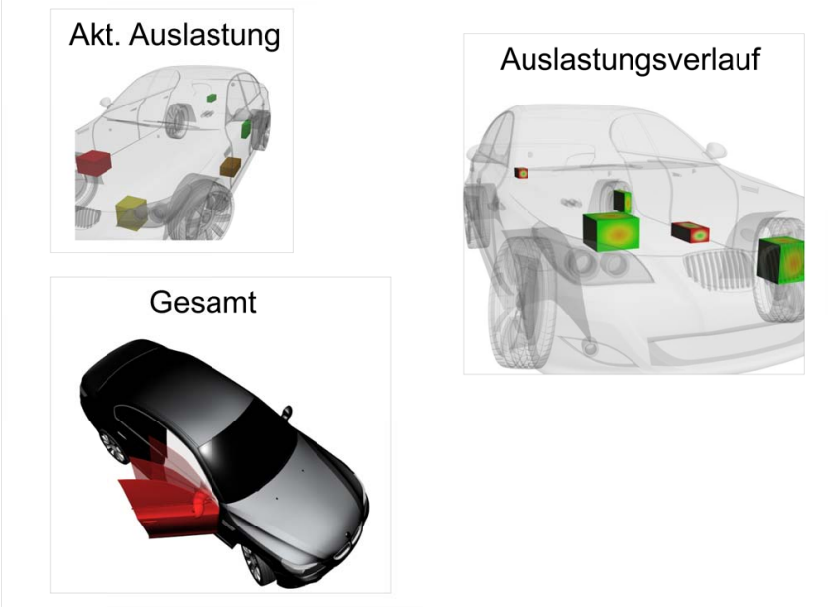
View-Konzept 1:

Es exisitiert eine View auf ein 3D-Modell. Es werden die mechanischen Reaktionen angezeigt und sobald Teile des Bordnetzes auf Wunsch des Users eingeblendet werden, wird das Fahrzeug transparent.



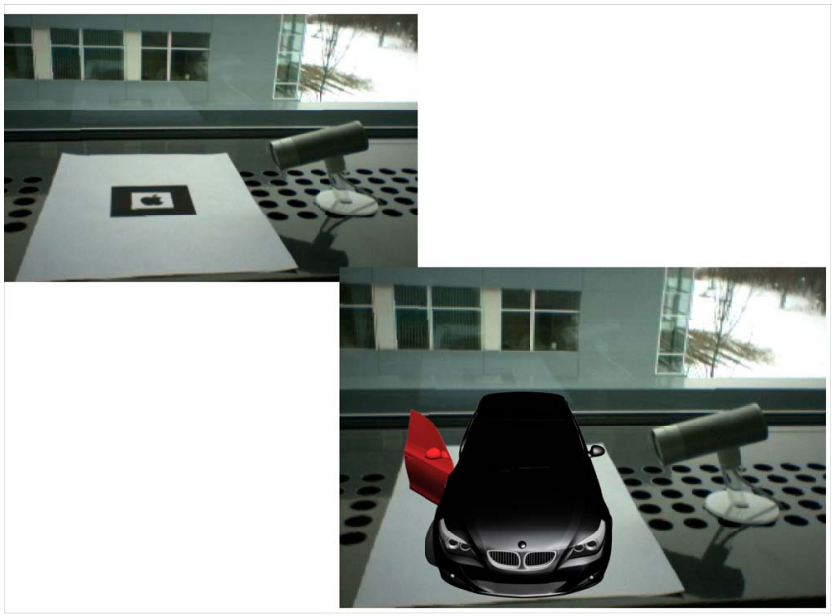
View-Konzept 2:

Es exisitiert eine View auf ein 3D-Modell. Zusätzlich können beliebige Fahrzeugkomponenten auf dem Bildschirm / um das Fahrzeug angeordnet werden. Diese können einzeln konfiguriert werden (Zoom, Ausrichtung, Bezeichnung, ...). Außerdem lassen sich 2D-Messgeräte ans Fahrzeug anschließen, die in abstrakter Form Auskunft über messbare Faktoren (Auslastung, Nachrichtenanzahl etc. geben)



View-Konzept 3:

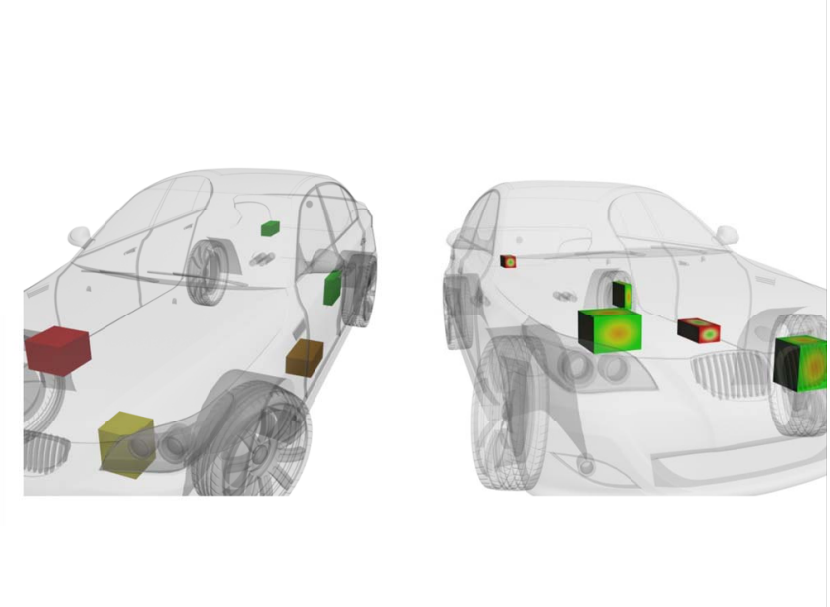
Es existieren beliebig viele Views auf das 3D-Modell. Jede View ist frei-konfigurierbar und bietet eine spezielle Ansicht auf den aktuellen Datensatz. Es lassen sich jeweils Einstellungen für die Bordnetzdarstellung und die Interaktionsmöglichkeiten vornehmen. Bei vordefinierten Events kann eine View automatisch vergrößert werden und in den Vordergrund rutschen. Die Organisation der verschiedenen Views wird wahlweise über Reiter, verschiebbare Fenster oder ähnliches realisiert.



View-Konzept Ergänzung:

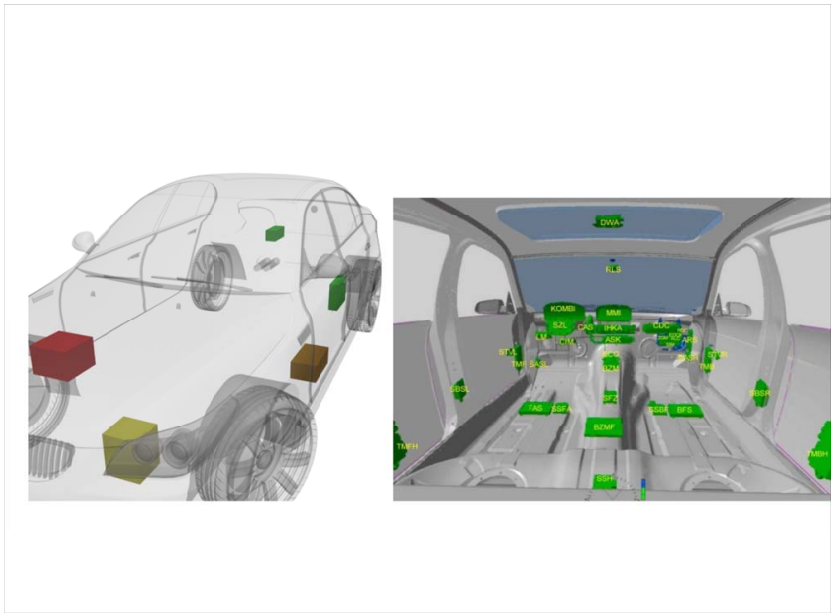
Es exisitieren verschiedene Views auf ein 3D-Modell, jedoch wird das Modell mit Hilfe eines Tracking-Systems („auf dem Schreibtisch“ des Benutzers) dargestellt: Eine Webcam zeichnet ein Bild des Schreibtischs auf, erkennt einen speziellen Marker und das 3D-Modell wird an dieser Markerposition ins Kamerabild eingefügt.
Je nach erkanntem Marker werden individuell konfigurierte Modelle gezeigt. (Bsp.: Marker A = volltransparentes Modell mit A-CAN Bus; Marker B = nicht-transparentes Modell)

Anhang E: InfoVis-Konzepte



InfoVis-Konzepte Auslastung:

- SG oder Busse werden singulär eingefärbt
- radialer Farbverlauf (von außen zum Zentrum) auf jeder SG-Seite
- inaktive SG / Busse werden im Laufe der Zeit mehr und mehr transparent
- Busse haben Standard-Durchmesser und werden proportional zur Auslastung „dicker“ bis zu einer bestimmten Grenze



InfoVis-Konzepte SG-Darstellung:

- SG werden möglichst realistisch dargestellt
- SG werden abstrakt und vereinfacht dargestellt
- semantische 3D-Lupe erlaubt Überblick welche Funktionen in einem SG untergebracht sind

InfoVis-Konzepte Fahrzeugverhalten:

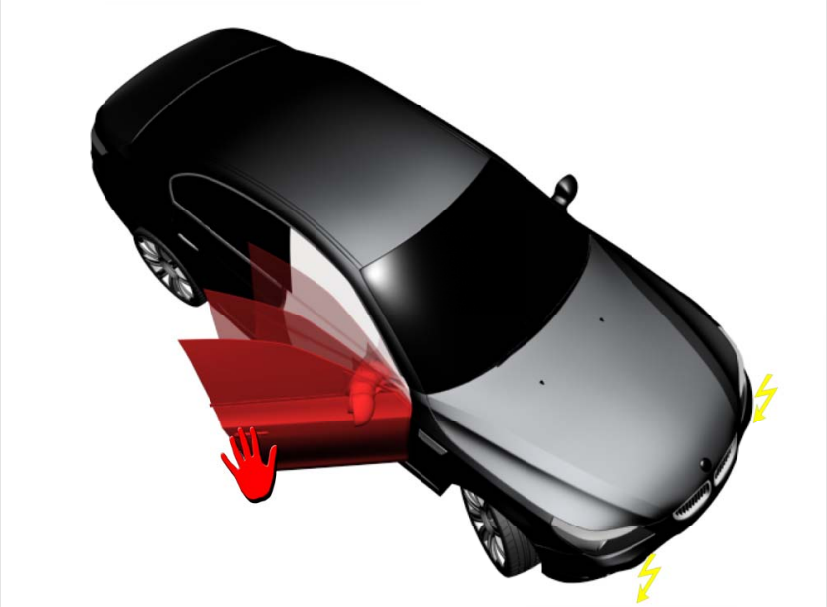
- Fahrzeug verhält sich entsprechend der Trace-Daten, es findet keinerlei Hervorhebung statt

- aktive Elemente werden eingefärbt

- aktive / sich bewegende Elemente ziehen einen Schatten hinter sich her (ggf. zusätzlich eingefärbt)

- Elemente pulsieren/leuchten mit abnehmender Intensität (starkes Leuchten zu Beginn)

- zusätzliche Fenster definierbar, die voreingestellte Aktionen vergrößert darstellen und nur bei der entsprechenden Aktion erscheinen

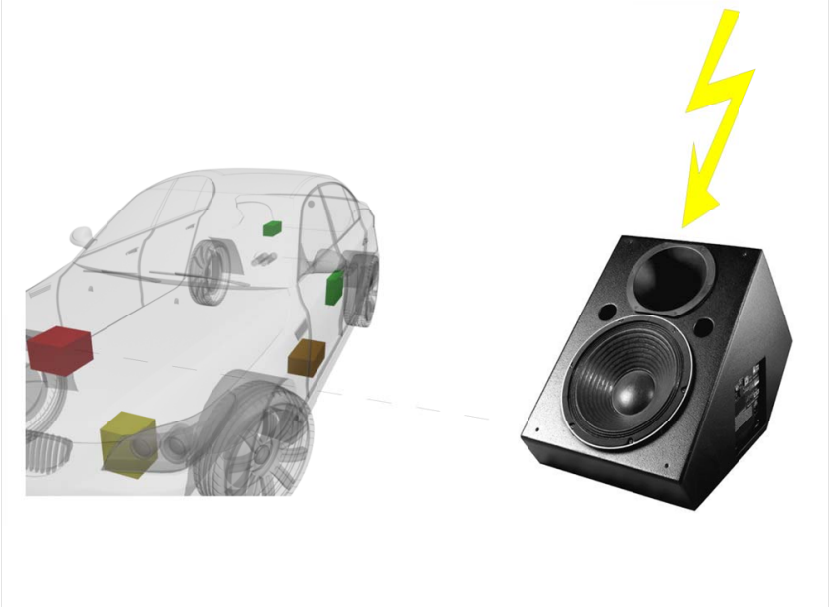


InfoVis-Konzepte Fahreraktion:

- bediente Elemente werden gesondert eingefärbt (anders als Fahrzeug(re)aktionen
- virtuelle Hand deutet Interaktion an

InfoVis-Konzepte Sensorereignisse:

- Sensoren werden eingefärbt
- vom Sensor ausgehende Wellen/Symbole deuten Aktivität an



Konzepte Sound:

- realistische Sounds zur Unterstützung der Immersion

- als Codierung / Signal für festgelegte Informationen oder Ereignisse (z.B. Auslastung überschreitet definierten Grenzwert)

Konzepte Nutzerinteraktionen:

- SmartObjects geben Hilfestellung (was kann ich mit welchem Element machen?)
- Rechtsklick auf Objekt öffnet Auswahlmenü mit möglichen Aktionen
- Fahrzeugrotation mit Hilfe eines Joysticks/Gamepads
- View-Steuerung mit Kombination von AR-Markern und konventioneller Steuerung

Anhang F: Konzeptbewertungen

Konzeptbewertungen

Views:
- View 1
- View 2
  - als Grundlage
  - Zustände wie gebraucht anzuordnen
  - 3D Modell bietet weiterhin komplette Übersicht
  - bietet gute Übersicht („clever“)
  - vgl. „Zustandsvektor“
- View 3
  - bietet mehr Informationen als die anderen Tools
  - gute Verbindungsmöglichkeit von Elektronik und Mechanik
  - bietet Möglichkeit für mehrere Abstraktionsschichten
  - ggf. vordefinierte Kamerapositionen
  - Darstellung von Elektronik und Mechanik gut trennbar
- sonstiges:
  - AR ist „lustige“ Idee, aber 3D-Steuerung simpel genug
  - Kombination von 2 & 3 optimal (!!!) → fast einstimmig: Grundkonzept = View 2 + 3

InfoVis:
- Auslastung:
  - Farbverlauf gut (nicht ganz intuitiv; von innen nach außen?) → nur in Extremsituationen deutliche Farbveränderung
  - Transparenz gut/okay (gut für Übersicht)
  - Bewertung vom Farbverlauf schwierig, da neue Idee
  - Bus-Dicke gute Idee (Achtung: Lie-Factor!)
  - Bus-Dicke passt nicht → eventuell zur Darstellung der Bus-Bandbreite?
  - Nutzer muss selbst definieren was gut und was schlecht ist (zur Farbkodierung)
  - Farben suggerieren eventuell Temperatur?
  - Bus-Last abhängig von Bus-System
  - singuläres Einfärben kann auch Baugruppen kennzeichnen
  - Transparenz + Bus-Dicke eher optional

- SG-Darstellung:
  - realistische Darstellung hat Wiedererkennungseffekt
  - ist nutzerabhängig
  - für InfoVis keine realen Positionen nötig → Abstraktion gut
  - Lupe/Hoverinfo praktisch (Zusatzinfo auf Wunsch immer gut!)
  - eigentliche Kommunikation läuft ohnehin abstrakt, aber mit möglicher Detailansicht
  - verschiedene Detailstufen und Details-on-Demand
  - Örtlichkeit der SG nicht so wichtig
  - Darstellung zwischen Purdue und Meriem, aber SG sollte erkennbar bleiben
  - Lupe schwierig, weil Funktionen über Fahrzeug verteilt
  - → UseCase-abhängig

- Fahrzeug-Verhalten:
  - Mechanik mit einbeziehen
  - färben gut → sollte komplett konfigurierbar sein
  - Schatten („muss nicht sein“) und Zusatzfenster („sinnvoll“) interessant → optional
  - Gesamtmodell ggf. nicht übersichtlich gneug, daher generelles Zerteilen vllt. Sinnvoll!
  - „Bei Purdue sieht man nichts“
  - komplette Fenster sind zuviel zu konfigurieren → sollte möglichst intuitiv bleiben
  - abstrakt wäre übersichtlicher

- Fahreraktion:
  - Wie kann ich im MCV interagieren?
  - Darstellung/Hervorhebung ist wichtig
  - definitiv sinnvoll für Trace-Daten

- Sensoren:
  - gut, kommt aber auf Detailgrad an
  - zentrale Frage: Welche Sensoren visualisieren?
  - Äußere Einflüsse darstellen; eingehende, nicht ausgehende Wellen
  - wegen komplexer und stark unterschiedlicher Systeme kaum umsetzbar
  - Wirkketten bei Sensorereignissen?
  - Sinnvoll für Trace-Daten

- Sound:
  - Problem: Großraumbüro
  - eventuell **gong** bei definierten Ereignissen abspielen statt realem Sound
  - gute Möglichkeit zusätzlichen Info-Kanal zu öffnen, unauffällige Geräusche (menschliche Sinne nutzen) [Bsp.: Zentralverriegelung]
  - muss abschaltbar sein
  - synchron mit Bildern bei realen Sounds
  - Signal-Sound wichtiger als Real-Sound

- Interaktion:
  - bei reiner Visualisierung überflüssig/sinnlos → eher konventionelle 3D-Steuerung mit vorhandenen Input Devices
  - bei Simulation stressig und Timing nicht wiederholbar
  - Point&Click Steuerung (vgl LucasArts-Adventure)
  - Problem: Innen Knopf drücken → außen Reaktion... → nicht sichtbar!
  - AR: Spielerei

Anhang G: Zielsetzungsdokument

Zielsetzungsdokument

Auf Grundlage der während der User-Tests gewonnenen Ergebnisse werden in diesem Dokument Design-Entscheidungen für den Prototyp meiner Diplomarbeit getroffen. Dabei dienen die Prototypen der Purdue University und die Diplomarbeit von Meriem Mozdari also erste Anhaltspunkte. In Verbindung mit den Bewertungen der „getesteten" Benutzer lassen sich hieraus bereits erste wesentliche Punkte für eine 3D-Visualisierung von Kommunikation im Fahrzeug herleiten. Jedoch bereits die Einstiegsfrage zur allgemeinen Meinung bzgl. 3D-Visualisierung brachte interessante Erkenntnisse für die spätere Entwicklung.

Erkenntnisse der Einstiegsfrage:
- 3D bietet bei guter Umsetzung Joy-of-View,
- Position der Steuergeräte im Modell sollte sofort erkennbar sein.
- Mechanisches Umgebungsmodell wird gewünscht.
- Vernetzung der Steuergeräte sollte gut zu erkennen sein.

Erkenntnisse zum Purdue-Tool:
- Einfärben der Steuergeräte je nach Last (Region of Interest) ist gut, da rote SG direkt ins Auge fallen.
- Kombination von Elektronik und Mechanik ist gut gelungen und bietet gute Ansätze für die Kommunikationsvisualisierung.
- Es gibt ein Skalierungsproblem – werden zu viele SG und Busse dargestellt, wird das 3D Modell schnell unübersichtlich. („selbsttragender Kabelbaum")
- Durch eingesetzte Transparenz sind die Positionen der SG im Fahrzeug gut zu erkennen und der gewählte Detailgrad bietet einen guten Kompromiss aus wenigen Details und möglichst vielen.
- Mechanische Reaktion des Fahrzeugs je nach Perspektive nur schwer erkennbar.
- Auslastungsanzeige von Steuergeräten ist gut, wenn auch in Echtzeit zum Teil problematisch. Ein ebenfalls wichtiger Punkt – die Buslast – könnt auf ähnliche Weise visualisiert werden.
- Position der SG und halbwegs realistischer Kabelverlauf wären bei gepflegten Datensätzen ein praktisches Feature.
- 3D-Visualisierung sollte besonders im Verbund mit anderen Tools betrachtet werden.

Erkenntnisse zur DA von Meriem Mozdari:
- Für spezielle UseCases ist die nachrichtendarstellung sinnvoll (Schulung, Lernprogramm, ...), aber bei Echtzeit in Analyse/Diagnose zu komplex.
- Darstellung der Vernetzung sorgt für eine gute Übersicht und durch unterschiedliche Einfärbung der Bus-Systeme behält man auch bei mehreren Systemen den Überblick.
- Anzeige der einzelnen Nachrichten vermutlich besser in 2D zu visualisieren → zu viele Informationen auf einmal.
- 3D Modell darf ruhig detaillierter sei, solang die Übersicht nicht darunter leidet.
- Filtermöglichkeiten sind absolut notwendig, allerdings ist die starke Anlehnung ans Bordnetz fraglich bzw. nicht von Beginn an intuitiv.
- Statt fliegender Nachrichten könnten Hover-Informationen über Steuergeräten nützlicher sein.

Erkenntnisse der Abschlussfrage:
- In einigen Bereichen sicherlich sehr sinnvoll. (Werkstatt, Schulung, ...)
- 3D-Modell kann keine hochkomplexen Informationen darstellen, ist jedoch gut geeignet um einfache Informationen mit Hilfe des mechanischen Modells darzustellen und bei der Unterstützung der räumlichen Vorstellungskraft zu helfen.

Zielsetzungsdokument

Auf Grundlage der während der User-Tests gewonnenen Ergebnisse werden in diesem Dokument Design-Entscheidungen für den Prototyp meiner Diplomarbeit getroffen. Dabei dienen die Prototypen der Purdue University und die Diplomarbeit von Meriem Mozdari also erste Anhaltspunkte. In Verbindung mit den Bewertungen der „getesteten“ Benutzer lassen sich hieraus bereits erste wesentliche Punkte für eine 3D-Visualisierung von Kommunikation im Fahrzeug herleiten. Jedoch bereits die Einstiegsfrage zur allgemeinen Meinung bzgl. 3D-Visualisierung brachte interessante Erkenntnisse für die spätere Entwicklung.

Erkenntnisse der Einstiegsfrage:
- 3D bietet bei guter Umsetzung Joy-of-View,
- Position der Steuergeräte im Modell sollte sofort erkennbar sein.
- Mechanisches Umgebungsmodell wird gewünscht.
- Vernetzung der Steuergeräte sollte gut zu erkennen sein.

Erkenntnisse zum Purdue-Tool:
- Einfärben der Steuergeräte je nach Last (Region of Interest) ist gut, da rote SG direkt ins Auge fallen.
- Kombination von Elektronik und Mechanik ist gut gelungen und bietet gute Ansätze für die Kommunikationsvisualisierung.
- Es gibt ein Skalierungsproblem – werden zu viele SG und Busse dargestellt, wird das 3D Modell schnell unübersichtlich. („selbsttragender Kabelbaum“)
- Durch eingesetzte Transparenz sind die Positionen der SG im Fahrzeug gut zu erkennen und der gewählte Detailgrad bietet einen guten Kompromiss aus wenigen Details und möglichst vielen.
- Mechanische Reaktion des Fahrzeugs je nach Perspektive nur schwer erkennbar.
- Auslastungsanzeige von Steuergeräten ist gut, wenn auch in Echtzeit zum Teil problematisch. Ein ebenfalls wichtiger Punkt – die Buslast – könnt auf ähnliche Weise visualisiert werden.
- Position der SG und halbwegs realistischer Kabelverlauf wären bei gepflegten Datensätzen ein praktisches Feature.
- 3D-Visualisierung sollte besonders im Verbund mit anderen Tools betrachtet werden.

Erkenntnisse zur DA von Meriem Mozdari:
- Für spezielle UseCases ist die nachrichtendarstellung sinnvoll (Schulung, Lernprogramm, ...), aber bei Echtzeit in Analyse/Diagnose zu komplex.
- Darstellung der Vernetzung sorgt für eine gute Übersicht und durch unterschiedliche Einfärbung der Bus-Systeme behält man auch bei mehreren Systemen den Überblick.
- Anzeige der einzelnen Nachrichten vermutlich besser in 2D zu visualisieren → zu viele Informationen auf einmal.
- 3D Modell darf ruhig detaillierter sei, solang die Übersicht nicht darunter leidet.
- Filtermöglichkeiten sind absolut notwendig, allerdings ist die starke Anlehnung ans Bordnetz fraglich bzw. nicht von Beginn an intuitiv.
- Statt fliegender Nachrichten könnten Hover-Informationen über Steuergeräten nützlicher sein.

Erkenntnisse der Abschlussfrage:
- In einigen Bereichen sicherlich sehr sinnvoll. (Werkstatt, Schulung, ...)
- 3D-Modell kann keine hochkomplexen Informationen darstellen, ist jedoch gut geeignet um einfache Informationen mit Hilfe des mechanischen Modells darzustellen und bei der Unterstützung der räumlichen Vorstellungskraft zu helfen.

Kennzeichnung (virtuelle Hand & Einfärben) hervorgehoben.

Sensoren:
Bei diesem Thema wurde mehrfach die Komplexität von Sensorabfragen, Verarbeitung der Daten und Auslösen von Ereignissen angesprochen. Dabei kamen jedoch Wirkketten bei Sensorereignissen als interessanter Visualisierungspunkt ins Gespräch, die sich aber auhc als recht komplex darstellten.
Daher wird es keine gesonderte Hervorhebung von Sensorereignissen im 3D-Modell geben, da die Komplexität dieser Aufgabe in keinem Verhältnis zum nutzen des resultierenden Features steht.

Sound:
Die Idee einen Audiokanal als zusätzlichen Infomationskanal zu nutzen wurde von allen Testpersonen gut bzw. interessant bewertet. Zwar wurde stets das Problem des Großraumbüros angesprochen (Funktion sollte deaktivierbar sein), aber ebenso wurden viele positiven Eigenschaften von allen Testern genannt.
Daher wird die 3D-Visualisierung um eine Soundkomponente ergänzt werden, die – frei konfigurierbar – die Möglichkeit bietet realen Sound (Tür auf / Tür zu, ...) oder verschiedene Warnsignale bei bestimmten Ereignissen abzuspielen („gong“ bei Überschreitung eines Grenzwerts).

Interaktion:
Die Interaktion mit dem Fahrzeug wurde als weniger wichtig für die Visualisierung bewertet und zum Teil sogar als „eventuell störend“ dargestellt. Im Rahmen einer 3D-Simulation sei eine Interaktion jedoch sehr praktisch und vor allem sinnvoll, um eigene Test-Traces aufnehmen zu können.
Daher wird sich die Interaktion mit dem Modell auf die Kamerasteuerung und Konfiguration der unterschiedlichen Grenzwerte, Metriken usw. zur korrekten/gewünschten Einfärbung von Elementen beschränken.

Anhang H: Namen der mechanischen Bauteile im 3D-Modell

Teilenamen & -zuordnungen:

Abblendlicht vorne links	-	abblend_vl
Abblendlicht vorne rechts	-	abblend_vr
Achse vorne links	-	achse_vl
Achse vorne rechts	-	achse_vr
Auspuff	-	auspuff
Blinker hinten links	-	blinker_hl
Blinker hinten rechts	-	blinker_hr
Blinker mittig links	-	blinker_ml
Blinker mittig rechts	-	blinker_mr
Blinker vorne links	-	blinker_vl
Blinker vorne rechts	-	blinker_vr
Blinkerhebel	-	blinkerheb
Bremse hinten links	-	bremse_hl
Bremse hinten rechts	-	bremse_hr
Bremse vorne links	-	bremse_vl
Bremse vorne rechts	-	bremse_vr
Bremslicht hinten links	-	bremsli_hl
Bremslicht hinten mittig	-	bremsli_hm
Bremslicht hinten rechts	-	bremsli_hr
Display vorne mittig	-	Display_vm
Fenster hinten links	-	fenster_hl
Fenster hinten rechts	-	fenster_hr
Fernlicht vorne links	-	fernli_vl
Fernlicht vorne rechts	-	fernli_vr

Frontscheibe	-	scheibe_v
Gaspedal	-	gaspedal
Handbremse	-	handbremse
Handschuhfach	-	handschuhf
Heckscheibe	-	scheibe_h
Klima-Steuerung	-	klima_kont
Kofferraum	-	kofferraum
Bremslicht links	-	brems_kr_l
Bremslicht rechts	-	brems_kr_r
Klappe	-	koffer_kl
Kombi-Instrument	-	kombi_inst
Lenkrad	-	lenkrad
Lichtschalter	-	lichtschal
Lüftung vorne links	-	lueft_vl
Lüftung vorne mittig	-	lueft_vm
Lüftung vorne rechts	-	lueft_vr
Motorhaube	-	motorhaube
Nebelscheinwerfer vorne links	-	nebelli_vl
Nebelscheinwerfer vorne rechts	-	nebelli_vr
Rad hinten links	-	rad_hl
Rad hinten rechts	-	rad_hr
Rad vorne links	-	rad_vl
Rad vorne rechts	-	rad_vr
Rückfahrlicht hinten links	-	rueckfahrl
Rückfahrlicht hinten rechts	-	rueckfahrr
Schaltknauf	-	schaltknau
Scheibenwischer vorne links	-	wischer_vl

Scheibenwischer vorne recht	-	wischer_vr
Scheibenwischerhebel	-	wischerheb
Scheinwerferglas vorne links	-	scheibe_fl
Scheinwerferglas vorne rechts	-	scheibe_fr
Schlüssel	-	schluessel
Sensor hinten links	-	sensor_hl
Sensor hinten mittig links	-	sensor_hml
Sensor hinten mittig rechts	-	sensor_hmr
Sensor hinten rechts	-	sensor_hr
Sitz vorne links	-	sitz_vl
Lehne	-	sitRueck_vl
Kopfstütze	-	kopf_vl
Rückenlehne	-	rueckLe_vl
Sitzfläche	-	sitzfla_vl
Sitz vorne rechts	-	sitz_vr
Lehne	-	sitRueck_vr
Kopfstütze	-	kopf_vr
Rückenlehne	-	rueckLe_vr
Sitzfläche	-	sitzfla_vr
Standlicht vorne links	-	standli_vl
Standlicht vorne rechts	-	standli_vr
Tankdeckel	-	tankdeckel
Tür vorne links (gesamt)	-	tuer_vl_gr
Tür vorne links	-	tuer_vl
Fenster vorne links	-	fenster_vl
Tür vorne rechts (gesamt)	-	tuer_vr_gr
Tür vorne rechts	-	tuer_vr
Fenster vorne rechts	-	fenster_vr

Anhang I: Abschlussevaluation

1. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp?**

2. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp nicht?**

3. **Wie bewerten Sie die gewählten Visualisierungstechniken zur Darstellung des Fahrzeugzustands? Haben Sie sonstige Anmerkungen zum gezeigten Prototyp?**
   - (Kontinuierliche) Darstellung des Fahrzeugstatus
   - Darstellung der Steuergeräte und Busse
   - Detailgrad des Fahrzeugmodells
   - Menü-Struktur
   - Positionierung der 2D-Anzeige [Positionierung variieren!]
   - sonstiges:

A) Können Sie sich eine Ansicht, die einem dieser Prototypen ähnelt, als Ergänzung zu vorhandenen Tools vorstellen?

B) [Autobahn-View starten & kurz erklären]

1. **Was lässt sich aufgrund der 3D-Ansicht besonders gut erkennen bzw. wofür eignet sich diese Darstellungsform?**
2. **In welchen Situationen verwirrt die 3D-Ansicht im MCV?**

C) Tasks:

1. **Wann sind beide Türen geöffnet?**

   - Welche Tür öffnet sich zuerst?
2. **Wann blinken die Blinker zum ersten Mal?**
3. **Wann erreicht DME1 eine Auslastung von 800?**

Anhang J: Auswertung der Abschlussevaluation

1. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp?**
   - „coole Spielerei“
   - Modell sehr detailliert
   - [insgesamt sehr positive Grundeinstellung zum 3D-Thema]
   - [im Verlauf des Tests: man erkennt die mechanischen Veränderungen]
   - Transparenz-Einstellung gut für Übersicht
   - [wenn mech. Status von Interesse: dieser ist gut zu erkennen]
   - [Status deutlich einfacher auslesbar als aus manueller Trace-Analyse]
   - Recht detailliert
   - (später im MCV: Fzg.-Zustand sofort sichtbar)

2. **Was gefällt Ihnen an der (Informations-)Visualisierung des gezeigten Prototyp nicht?**
   - 3D allein bringt nicht viel ohne weitere 2D-Anzeigen, da es zu viele Informationen sind, um sie schlüssig in 3D darstellen zu können
   - Steuerung
   - [Feinheiten (Zentralverriegelung schwer zu erkennen)]
   - Steuerung gewöhnungsbedürftig [kein Gamer]
   - Fahrzeug aus Bild rausdrehen sollte verhindert werden [[ist gelöst!]]
   - für sich allein genommen bzw. die reine Bordnetzdarstellung weniger geeignet als 2D bzw. 3D bietet dann kein herausragendes Merkmal

3. **Wie bewerten Sie die gewählten Visualisierungstechniken zur Darstellung des Fahrzeugzustands? Haben Sie sonstige Anmerkungen zum gezeigten Prototyp?**

   - (Kontinuierliche) Darstellung des Fahrzeugstatus
     - „ist cool“
     - Busdaten ggf. schlecht visualisierbar... „bringt vllt nicht viel in 3D“
     - mech. Aktionen gut zu erkenne
     - ggf. Cockpit View?
     - Region of Interest (einfärben/hervorheben bei Aktionen)
     - bei viel „Action“ gut, da viel Action = unübersichtlicher Trace bzw. Trace ist aufwendig zu lesen
     - am Modell gut zu erkennen
     - Kleinigkeiten schwer zu sehen (Bsp.: ZV)
     - „ziemlich cool“
     - gut erkennbar

   - Darstellung der Steuergeräte und Busse
     - gut
     - Aussehen ist Entwicklern egal
     - Namen sollten in 3D dargestellt werden → Tooltip?
     - realistischere SG/Gateway-Darstellungen wären „cool“
     - durch Ausblenden/Transparenzen gute Filterungsmöglichkeit
     - SG und Busse sind gut erkennbar
     - Größe der Bauteile gut gewählt
     - sagt nichts aus → Topologie fehlt/ist nicht zu erkennen
     - Positionsbestimmung gut für Werkstatt
     - SG-Namen darstellen wäre sinnvoll
     - Verbindung mit Mechanik sinnvolle Darstellung für Mechaniker
     - „leider“ keine Darstellung wie bei Purdue mit Nachrichtenfluss etc.

  - bzgl. Realismus: bei zu vielen Informationen gleichzeitig wird es unübersichtlic
  - [Anm.: Eventuelle Beeinflussung durch Erinnerung an Purdue-Prototyp?]

- Detailgrad des Fahrzeugmodells
  - sehr hoch
  - „Wahnsinn wie genau das Modell ist“
  - keinesfalls zu wenig Details
  - durch einstellbare Transparenz kein Problem
  - guter Detailgrad
  - [ggf. Motor ergänzen]
  - keinesfalls zu wenig Details
  - im Menü: eher Baugruppen bilden statt einzelne Elemente anzubieten!

- Menü-Struktur
  - „ist okay“
  - einarbeiten notwendig, danach okay
  - Resize wäre gut, um Liste zu vergrößeren
  - „2D-HUD“-Tab umbenennen?
  - Eingegebener Name sollte für „alte“ View gelten und nicht für die neue
  - Transparenz ist sehr wichtig → ins erste Tab zusammen mit 2D-Elementen
  - Farbe eher sekundär, kann „versteckt“ werden
  - Vorkonfigurationen!
  - Erklärung der Schritte gut
  - vllt. Baum-Struktur in Liste anwenden?
  - A-CAN / [A-CAN gesamt] anders benennen?

- Positionierung der 2D-Anzeige [Positionierung variieren!]
  - bei In-Place: vllt Halb-Transparenz, damit umliegende Elemente nicht verdeckt werden?
  - Umschalten ermöglichen via „Pin-Nadel“
  - Manuelle Positionierung vorgezogen
    - mehr Freiheitsgerade
    - weniger Verdeckungen
  - „Wie soll das aussehen wenn da 3 oder 4 2D-Elemente angezeigt werden?“
  - Position am SG „totaler Käse“... → „ich sehe, dass ich nichts sehe“
  - variable Lösung bevorzugt
  - Anm. zu 2D-HUD: Windows/Linux etc. Look&Feel fehlt
  - flexible Position ist eindeutiger Favorit!
  - Bei Am-Platz-Darstellung stört die Verdeckung anderer Teile – außerdem: „Wo bin ich?“

- sonstiges:
  - Steuerung macht am Anfang Probleme
  - Navigationspfeile zur Unterstützung der Orientierung wären gut

  - Kamerasteuerung gewöhnungsbedürftig [kein Gamer]
  - ggf. Simulationsaspekt mit reinbringen bzw. 3D-Modell würde sich dafür eignen („Wass passiert, wenn ich die Tür öffne“)

- gewisse Gamer-Erfahrung
- Steuerung: nicht intuitiv... lieber „oben/unten“ bewegen mit W und S statt „vorn/hinten“
  - wenn Steuerung einmal klar, kein Problem
- Färbung: keine Anzeige der alten Werte
- Skalierung des 3D-Fensters wäre gut
- zur Erkennung „kleiner“ Veränderungen (Bsp.: ZV) eventuell Front- und Seitenansicht per Standard bieten oder kleine Statuslampen
- zusätzlich noch Wärme darstellbar?
- [bis zu diesem Zeitpunkt ist Use Case nicht klar]
- Steuerung: Buttons zum Rotieren (Idee: Wie Rotationswürfel bei 3DS)
- Steuerung zuerst problematisch
- BN sichtbar machen ist nicht intuitiv
- 2D-Elemente sollten eine Titelleiste wie bei Windows haben, zwecks Verschieben usw.
- 2D bietet noch mehr Darstellungsmöglichkeiten
- bewegliches Modell suggerierte zuerst die Möglichkeit der Interaktion mit dem Modell → Simulationsaspekt
- Anregung zur Kamerasteuerung: MultiTouch-Steuerung, z.B. bei TableTop

A) Können Sie sich eine Ansicht, die einem dieser Prototypen ähnelt, als Ergänzung zu vorhandenen Tools vorstellen?

- Vereinfacht die Einarbeitung ins Stoffgebiet
- nützlich zur Besprechung von Lösungen/Änderungen
- Mehrwert durch 3D ist möglich
- entscheidend:
  - welche Daten?
  - Echtzeit-Darstellung ist kritisch
  - „break points“, wenn definiertes Ereignis eintritt
  - sequentielle Folge „durchscrollbar“ (vgl. Autobahnview)
- Wenn untersuchter Bus etwas mit Fahrzeugstatus zu tun hat ist die 3D-Ansicht nützlich, ansonsten sind es zuviele Informationen

B) [Autobahn-View starten & kurz erklären]

erster Eindruck ohne gefragt zu haben: „sehr geil“

Aussage: „Das ist ja wie DIERK in 3D“

1. Was lässt sich aufgrund der 3D-Ansicht besonders gut erkennen bzw. wofür eignet sich diese Darstellungsform?

- fördert das Verständnis
  - [im Nachhinein: Nachrichten im BN in 3D zeigen und bei Selektion Details zur Nachricht darstellen]
- Mechanik ist gut
- Statusveränderungen über die Zeit lassen sich so sehr gut herausfinden
- Gute Möglichkeit der Darstellung
- sehr gut zur Visualisierung von komplexen Geometrien
- bei erster Erklärung der Verbindung zur AutobahnView in Bezug auf 3D-View: „Ah, dazu ist das da!“

2. **In welchen Situationen verwirrt die 3D-Ansicht im MCV?**
- Experten könnten „kryptischere“ Ansicht vllt eher gebrauchen, d.h. gezielte Informationsdarstellung, auch wenn kein realistisches 3D-Modell mehr dargestellt werden sollte
- Verwirrt im MCV nicht, sondern nur, wenn allein genutzt

C) Tasks:

1. **Wann sind beide Türen geöffnet?**

   - Welche Tür öffnet sich zuerst?

2. **Wann blinken die Blinker zum ersten Mal?**
   - Anmerkung: andere Farbgebung für höheren Kontrast
   - problemlos, aber mit Vermerk, dass es leicht übersehen werden könnte

3. **Wann erreicht DME1 eine Auslastung von 800?**
   zuerst Transparenz eingestellt und dann 2D erstellt... [Warum Transparenz einstellen?! O.o]

   sofort 2D zur Hilfe genommen und Stelle gefunden

INSGESAMT: alles problemlos gelöst
INSGESAMT: alles problemlos gelöst, nach leichten Problemen mit der Steuerung

ABSCHLUSS:
- Gesamtfazit: „Passt so.“

- generell nicht schlecht
- gut, weil voll konfigurierbar
- automatischer Transparenz-Modus wäre „perfekt“ (Teile faden automatisch aus, wenn „hinter“ ihnen eine (per Definition) wichtige Aktion passiert)
- Standard-Transparenz und Undurchsichtigkeit zur Hervorhebung?
- Anwendungsbeispiel: SG-Analyse... An welcher Stelle geht welche Nachricht verloren?

- Kombination aus 2D/3D (MCV) wurde als wichtig bewertet
- MCV wurde wie im Paper von selbst „erstellt“
- UseCase: Suchte gleich „echte“ Nachricht für Tür-öffnen....
- Problem: bei langen Traces ist es mühselig/nicht möglich über Scrolling die „richtige“ bzw. die gesuchte Stelle zu finden
- Position der SG ist in vielen Fällen nicht wichtig