Interaktive Visualisierungssysteme zur beschleunigten Analyse von Simulationsergebnissen im Fahrzeugentwicklungsprozess

Von der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Martin Schulz

aus Uelzen

Hauptberichter:	Prof. Dr. T. Ertl
Mitberichter:	Prof. Dr. G. Greiner
Tag der mündlichen Prüfung:	13.3.2002

Institut für Informatik der Universität Stuttgart
2002

Danksagung

Diese Arbeit wurde am Lehrstuhl für graphische Datenverarbeitung der Universität Erlangen-Nürnberg begonnen und nach meinem Wechsel in der Abteilung für Visualisierung und interaktive Systeme der Universität Stuttgart fortgeführt. Das hervorragende Arbeitsklima der beiden Arbeitsgruppen erleichterte mir die Arbeit, ließ mich die ganze Zeit über wohlfühlen und meine Kochkünste beim gemeinsamen Kochen verbessern.

Mein spezieller Dank gilt meinem Doktorvater Prof. Dr. Thomas Ertl, der mir zu jeder Zeit mit Rat und Tat zur Seite stand. Durch seine Visionen, seine Ideen und letztendlich seinem Verhandlungsgeschick entstand aus so manchem Gespräch eine fruchtbare Zusammenarbeit mit einem Industriepartner. Meinem Zweitgutachter Prof. Dr. Günther Greiner möchte ich für zahlreiche Diskussionen und die Hilfe bei mathematischen Problemen danken.

Die BMW Group München förderte meine Arbeit mit einem Promotionsstipendium, wobei Thomas Reuding mich von industrieller Seite betreute. Er unterstützte mich beim Knüpfen von Kontakten, bei zahlreichen Veröffentlichungen, die zu Präsentationen in Übersee führten, und auch beim Zugang zur BMW-CAVE zu später Stunde. Für sein Engagement und seinen kameradschaftlichen Umgang bedanke ich mich.

Auch möchte ich allen Mitarbeitern bei BMW danken, die mir durch ihren persönlichen Einsatz halfen, mir die Tiefen des Fahrzeugentwicklungsprozesses nahebrachten und die Prototypen meiner Arbeit in der praktischen Anwendung untersuchten. Dies sind neben vielen anderen Dr. Wolf Bartelheimer, Daniel Heiserer und Ulf Tipkemper. Besonders herausheben möchte ich Dr. Michael Holzner für seinen Weitblick und Ideenreichtum, der meine Arbeit mitbegründete und ihr viele Impulse gab.

Meinen Kollegen an den Universitäten möchte ich für den kameradschaftlichen Umgang, viele interessante Diskussionen (teils bei mexikanischem Essen) und unterhaltsame Stunden bei schweißtreibender Arbeit in Simulationsprogrammen danken. Besonders erwähnen möchte ich Dr. Roberto Grosso, Dr. Christian Teitzel, Dr. Peter Hastreiter, Prof. Dr. Rüdiger Westermann, Ove Sommer, Klaus Engel, Matthias Hopf, Dr. Sven Kuschfeldt, Martin Kraus, Maria Baroti.

Den Studien- und Diplomarbeitern Manfred Weiler, Frank Reck, Jürgen Beckmann und Edith Zimmermann danke ich für ihre fruchtbare und erfolgreiche Zusammenarbeit.

Für ihre starke Unterstützung bei der Korrektur dieser Arbeit möchte ich mich bei Martin Kraus, Sabine Iserhardt-Bauer, Elke Hösch, Frank Reck, Matthias Hopf und Dr. Christian Teitzel bedanken. Es war für sie sicherlich nicht immer einfach.

Schließlich möchte ich meinen Eltern und meiner Familie für ihre volle Unterstützung in jeder Situation danken. Ohne Petras Unnachgiebigkeit beim Zusammenschrieb wären unsere beiden Kinder Maike und Henrik in der Zeit meiner Dissertation nicht nur geboren sondern auch eingeschult.

Martin Schulz

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Der kosten- und zeitintensive Prozess der Produktentwicklung ist für die Industrie von zentraler Bedeutung. Ohne sichere Gewinnaussichten finanzieren Firmen die Entwicklungskosten vor und koppeln so ihre finanzielle Lage an den Erfolg des Produktes beim Kunden. Die Kaufentscheidung des Kunden wird von der Qualität und dem Preis des Produktes geleitet, der neben den Herstellungskosten zu einem großen Teil aus den Entwicklungskosten resultiert. Ein kostspielig entwickeltes Produkt von hoher Qualität kann sich daher ähnlich schlecht auf die Firmenbilanz auswirken, wie ein qualitativ minderwertiges Produkt. Folglich zielen die Methoden der Produktentwicklung auf einen hohen und effizienten Wirkungsgrad ab und befinden sich immer auf dem schmalen Grad zwischen Qualitätssicherung und Kostenersparnis.

Ein großer Teil der Entwicklungskosten resultiert aus den Gehältern der Spezialisten. Die Volksweisheit „ Zeit ist Geld “ trifft auch hier den Kern des Problems und eine Senkung der Kosten kann über eine Reduktion der Entwicklungszeit erreicht werden. Eine optimale Ausrichtung der Arbeitsumgebung des Ingenieurs steigert die Effizienz seiner Arbeitskraft und wird durch den Einsatz von neuen Techniken erzielt. Die Arbeitsabläufe im Entwicklungsprozess werden durch die Einführung von neuen Methoden zur Kosten- und Zeitersparnis kontinuierlich verändert.

Ein großes Potenzial zur Kostenreduktion bietet die digitale Technik. Digital gespeicherte Informationen lassen sich schnell und einfach manipulieren, transportieren, verwalten und sind gerade durch die Globalisierung der Märkte ein erfolgsentscheidender Faktor. In vielen Bereichen der Produktentwicklung hat der Einsatz von digitaler Technik bereits zu einem großen Wandel geführt und die alten Arbeitsmethoden ersetzt bzw. revolutioniert. Angepasst an die einzelnen Teilschritte des Entwicklungsprozesses, führt die Computerunterstützung zu einer deutlichen Effizienzsteigerung.

Das Anforderungsprofil der unterschiedlichen Entwicklungsschritte an die Hard- und Softwarelösungen hängt von den Aufgaben im Arbeitsumfeld ab. Die einzelnen Aufgabenbereiche des Produktentwicklungsprozesses lassen sich grob in Design, Konstruktion, Evaluation und Werkzeugkonstruktion untergliedern und mit den entsprechenden digitalen Verfahren unterstützen.

Der Einzug digitaler Modellierungs- und Visualisierungstechniken im Designschritt hat dazu beigetragen, dass zu einem sehr frühen Zeitpunkt bereits ein digitales Modell vorliegt. Teilweise werden die realen Modelle durch Mustererkennung erfasst, oder virtuelle dreidimensionale Modelle durch den Einsatz von modernen Eingabegeräten erstellt. Die Visualisierung des digitalen

Modells auf großen Projektionsanlagen oder Workbenches trägt zu einem schnelleren Entscheidungsprozess über das Design bei.

Im darauf folgenden Teilschritt werden die Designentwürfe durch Konstruktion in ein exaktes Modell umgesetzt. Das so genannte Nullmodell entsteht, an dem sich die Folgeschritte des Entwicklungsprozesses orientieren werden. Ursprünglich wurde die Konstruktion von Produkten an Zeichentischen durchgeführt. Die Einführung des computerunterstützten Designs, dem „Computer Aided Design“ (CAD), führte in diesem Entwicklungsschritt zu einer Beschleunigung und qualitativen Verbesserung der Konstruktion. Mittlerweile wird kaum noch ein Produkt ohne CAD entwickelt.

Die Qualität der Konstruktion in Hinblick auf Fugenmaße, Stabilität und Herstellbarkeit wird traditionell durch den Bau von Prototypen evaluiert. In iterativen Schritten werden Schwachstellen des Produktes untersucht und in der Konstruktion durch Verbesserungen behoben, bis die vorgegebenen Anforderungen erfüllt sind. Je später Probleme und Fehler im Entwicklungsprozess erkannt werden, umso teurer sind deren Behebung. Das „Computer Aided Engineering“ (CAE) ermöglicht durch die numerische Simulation digitaler Modelle das so genannte „information front loading“ und hat zu einem ähnlichen Wandel geführt, wie der Einsatz von CAD-Arbeitsplätzen in der Konstruktion. Die frühzeitige numerische Analyse der Konstruktion beschleunigt nicht nur den Entwicklungsprozess, sondern spart durch Reduktion der kostspieligen realen Prototypen auch Kosten ein. Die digitalen Modelle können schon auf unvollständigen Datensätzen Schwachstellen aufzeigen und somit die Konstruktion im frühen Stadium beeinflussen. Zur Analyse der Berechnungsergebnisse gibt es noch keine automatisierten Methoden, wodurch die Erfahrung der Ingenieure beim Analyseschritt gefordert ist. Die Arbeit der Ingenieure muss daher durch die aktuellste Technik zur Effizienzsteigerung im höchsten Maße unterstützt werden.

Die Konstruktion der Herstellungswerkzeuge eines Produktes ist im eigentlichen Sinne ein eigenständiger Produktentwicklungsprozess. Das Werkzeug wird anhand der Produktvorgaben konstruiert und die gewünschte Funktionsweise bzw. die Einhaltung von Toleranzen evaluiert. Falls sich mit einem Werkzeug ein Produkt in der geforderten Form bzw. Qualität nicht herstellen lässt, muss eventuell die Konstruktion des Produktes den Herstellungsmöglichkeiten angepasst werden. Die Sicherstellung der Herstellbarkeit sollte natürlich möglichst früh erfolgen, um kostspielige Änderungen am fertigen Produkt zu verhindern. Die Werkzeugkonstruktion verläuft also idealerweise parallel zur Produktkonstruktion und auch hier bieten numerische Simulationen die Möglichkeit zur frühzeitigen Fehlererkennung und Behebung.

Optimierungen von Entwicklungsprozessen haben die Parallelisierung der Teilschritte zur Zeit- und Kostenersparnis als Ziel. Im optimalen Fall verlaufen die Teilschritte des Entwicklungsprozesses nicht sequentiell, sondern möglichst parallel, wie am Beispiel der Werkzeugkonstruktion erkennbar. Die numerischen Simulationen erlauben eine frühzeitige Evaluation der Konstruktion bereits an Teilmodellen und verbessern somit nicht nur den Prozess sondern auch das Produkt. Die Grenzen der Automatisierung im Produktentwicklungsprozess liegen klar in der Beurteilung von Berechnungsergebnissen. Hier ist auf absehbare Zeit der qualifizierte Ingenieur die einzige Alternative. Sein Fähigkeiten müssen durch eine intuitive und aussagekräftige Visualisierung unterstützt werden.

1.1 Computerunterstützung der Fahrzeugentwicklung

Das Aufgabengebiet dieser Arbeit ist speziell der Entwicklungsprozess von Personenkraftwagen. Die Entwicklung von Fahrzeugen ist im Vergleich zu vielen anderen Produkten ein sehr komplexer Prozess. Nicht nur die Qualität der Verarbeitung und der Preis spielen beim Kauf eines Fahrzeuges eine wichtige Rolle sondern auch das Design und das Prestige. Letztendlich sind die Preise enorm und ein Fahrzeug soll über mehrere Jahre gefahren werden, ohne dass die Kaufentscheidung bereut wird. Somit muss ein Käufer von den Vorzügen des Produktes im höheren Maße überzeugt werden.

Die Käuferansprüche sind im Laufe der Zeit gestiegen. Eine ruhige Straßenlage und bequeme Sitze sind seit langem selbstverständlich. Arbeitsgebiete für die heutigen Entwicklungen sind beispielsweise passive Sicherheit und die Berücksichtigung von subjektiven Empfindungen.

Im Bereich der passiven Sicherheit sind durch die Entwicklung von Airbags und neuen Rückhaltesystemen wichtige Komponenten entstanden. Die Sicherheitszelle des Fahrzeuges muss jedoch für jeden Fahrzeugtypen neu konstruiert und evaluiert werden. Numerische Simulationen kommen in diesem Bereich mittlerweile massiv zum Einsatz und die exponentielle Steigerung der Komplexität der digitalen Modelle stellt immer höhere Anforderung an die Auswertung der Berechnung, das so genannte Postprocessing.

Das Design eines Fahrzeuges muss heutzutage nicht nur ansprechend sein sondern zur Einsparung von Treibstoff auch möglichst strömungsoptimiert. Auf diesem Gebiet werden ebenfalls numerische Simulationen mit wachsender Modellkomplexität eingesetzt.

Von immer größerer Bedeutung wird der subjektive Eindruck des Fahrzeuges. Das Schließgeräusch der Fahrzeugtüren zählt beispielsweise zu dieser Kategorie oder auch die Geräuschentwicklung im Fahrzeuginneren in Bezug auf Lautstärke und Frequenz im Leerlauf oder Fahrbetrieb. Numerische Kinematik-, Schwingungs- und Akustiksimulationen werden zur objektiven Bewertung dieser Kriterien erprobt.

Ohne den Fortschritt in der Hard- und Softwaretechnik wäre der Einsatz von numerischen Simulationen nicht denkbar. Der produktive Einsatz wurde erst durch die Einführung von Supercomputern mit niedrigen Betriebs- und Anschaffungskosten, wie der *SGI Origin*, ermöglicht. Parallel wurden die Softwaresysteme mit zusätzlichen Funktionalitäten, wie der Unterstützung von Mehrprozessorsystemen, optimiert. Die Dauer der numerischen Berechnungen wurden so trotz steigender Modellkomplexität gesenkt.

Die Analyse der Berechnungsergebnisse lässt sich nicht allein durch schnellere Rechner und optimierte Algorithmen beschleunigen, da hier noch immer die Auffassungsgabe und Beurteilung der Ingenieure entscheidend ist. Die Entwicklung der Analysetechniken trat durch die großen Anstrengungen im Bereich der Simulation in den Hintergrund und konnte mit der Entwicklung nicht standhalten. Im produktiven Einsatz zeigt sich jetzt der Bedarf an schnellen und intuitiv zu bedienenden Postprocessing-Systemen.

Ein Postprocessing-System muss in der Lage sein, die Berechnungsergebnisse detailgetreu als dreidimensionale Abbildung darzustellen und zusätzlich physikalische Simulationsgrößen zu visualisieren. Die physikalischen Parameter können beispielsweise Eigenschaften der Geometrie oder des angrenzenden Volumens beschreiben.

Im Fahrzeugentwicklungsprozess verdoppelt sich die Komplexität der Modelle und somit die

Größe der darzustellenden Geometrie alle zwei Jahre. Beispielsweise besteht ein durchschnittlicher Crashtest-Datensatz momentan aus der enormen Datenmenge von ca. 300.000 Finiten-Elementen in jedem der 60 Zeitschritte und kann nur mit moderner Graphikhardware dargestellt werden.

Die Grenzen moderner Graphiksysteme wurden durch die Entwicklung von Graphikhard- und software immer weiter nach oben verschoben. Die obere Grenze von Seiten der Graphikhardware wird momentan durch das *SGI ONYX Infinite Reality* Graphiksystem mit 10^7 Polygonen pro Sekunde definiert. Immer neue Entwicklungen gerade im PC- und Workstationsegment dringen jedoch in diese Leistungsbereiche vor. Auch im Bereich der Ein- und Ausgabegeräte eröffnen Neu- und Weiterentwicklungen, wie beispielsweise die *CAVE* oder elektromagnetisch bzw. optische Tracking-Systeme, neue Einsatzprofile. Softwareseitig hat die Entwicklung von Szenengraphen die effiziente Speicherung und leichte Handhabbarkeit von komplexen Geometrien ermöglicht.

Von den Entwicklungen der Graphiksysteme und -software haben die Techniken der virtuellen Realität (*VR*) stark profitiert. Die VR wandelte sich von reinen Unterhaltungs- und Demonstrationsanwendungen in ein leistungsstarkes Instrument als Mensch-Maschine-Schnittstelle. In den letzten Jahren wurde die VR weltweit zum Gegenstand der Forschung und wird von vielen Universitäten und Firmen auf Leistungspotenzial und Einsatzmöglichkeiten untersucht. Die Eröffnungen von vielen VR-Labors und VR-Demozentren sind ein deutliches Indiz für die produktive Reife der neuen Technik. Große Firmen setzen bereits auf den Nutzen der VR zur Einsparung von Kosten im Entwicklungsprozess.

Dieses bereitgestellte neue Potenzial kann von den meisten etablierten Postprocessing-Systemen nur zu einem Teil genutzt werden. Die Basis dieser Systeme besitzt eine proprietäre Schnittstelle zur Graphik, die sich häufig nur schwer erweitern lässt. Die Vorteile durch die Nutzung einer Szenengraphen-API und den darin enthaltenen Optimierungsalgorithmen, wie beispielsweise das *Occlusion Culling*, bleibt ihnen deshalb verwehrt. Die typischen VR-Ein- und Ausgabegeräte können von diesen Postprocessing-Systemen ebenfalls nicht angesteuert werden. Das Manko aller VR-Kompetenz- und Demozentren ist der Mangel an angepassten leistungsfähigen Anwendungen, die das Potenzial der bereitgestellten Hardware ausnutzen und sich zusätzlich in den produktiven Einsatz einfügen.

Die Einsatzmöglichkeiten von optimierten Visualisierungsalgorithmen gekoppelt mit den Techniken der virtuellen Realität zur Steigerung der Produktivität bei der Analyse von Berechnungsergebnissen im Fahrzeugentwicklungsprozess wird in dieser Arbeit untersucht. Es wird gezeigt, welche VR-unterstützten Arbeitsumgebungen für die einzelnen Schritte des Postprocessings geeignet sind und beispielhaft für den Ingenieursarbeitsplatz, das Visualisierungslabor und Besprechungsräume vorgestellt.

Diese Arbeit demonstriert den Einsatz von VR-Techniken im produktiven Umfeld anhand zweier prototypischen Anwendungen. Neue VR-gestützte Visualisierungsmethoden werden in den beiden Prototypen für verschiedene Simulationsergebnisse erprobt. So bieten beide Programme Unterstützung für eine Reihe an Ein- und Ausgabegeräten, wie beispielsweise die *CAVE*, den *BOOM*, die *Spacemouse* oder verschiedene Tracking-Systeme. Die Applikationen sollen durch die Unterstützung dieser Gerätschaften und die direkte Manipulation von virtuellen Objekten leicht und intuitiv bedienbar sein. Ziel ist die Bereitstellung eines leistungsstarken Instrumentes

für den spezialisierten Ingenieur, das auch dem unbedarften Anwender ohne großen Lernaufwand einen Einblick in den dargestellten Sachverhalt gewährt.

Die erste Applikation *VtCrash* ist spezialisiert auf die Visualisierung von geometriebezogenen Simulationen, wie beispielsweise Crashtest-, Schwingungs-, Kinematik und Umformberechnungen. Der Schwerpunkt hierbei liegt in einer effizienten Verwaltung der enormen Datenmengen und der Bereitstellung von Interaktionsmechanismen für zeitabhängige Oberflächen. In diesem Umfeld werden auch die Einsatzmöglichkeiten von Intra- und Internet für kooperatives Arbeiten untersucht.

In einer Erweiterung von *VtCrash* werden Ergebnisse von Akustikuntersuchungen parallel mit den zu Grunde liegenden Schwingungsanalysen visualisiert. Die gleichzeitige Visualisierung von Geometrie- und Volumendaten stellt hierbei neue Anforderungen an die Datenverwaltung und es werden zusätzlich Echtzeitinteraktionsmechanismen für zeitabhängige Volumendatensätze implementiert und bewertet.

Die Umströmung der Fahrzeugaußenhaut ist das Gebiet der zweiten Applikation. Für den speziellen Volumendatentyp der lokal verfeinerten kartesischen Gitter von Berechnungen mit dem numerischen Solver *PowerFLOW* von *Exa* werden angepasste Visualisierungsalgorithmen und Interaktionstechniken im Visualisierungssystem *PowerVIZ* demonstriert.

Die Steuerung einer Applikation in einer virtuellen Umgebung erfordert Interaktionsmechanismen, die ohne eine 2D-Maus und ein 2D-Menü auskommen. Zur Lösung dieses Problems werden in dieser Arbeit Alternativen untersucht und ein 3D-Interaktionsmenü implementiert.

Die implementierten Programme werden beispielhaft im produktiven Entwicklungsprozess der *BMW Group* eingegliedert und ihre Einsatz- und Leistungseigenschaften beurteilt.

1.2 Beiträge der Arbeit

Das Betätigungsfeld der vorliegenden Arbeit breitet sich über eine Vielzahl von Teilgebieten des Postprocessings im Fahrzeugentwicklungsprozess aus. Für die Analyse von Finite-Element-Berechnungen und Strömungssimulationen wurde jeweils ein interaktives Visualisierungssystem entworfen und implementiert. Neben den spezialisierten Applikationen für bestimmte Datenformate wurden mit der Untersuchung und Implementierung einer dreidimensionalen graphischen Benutzerschnittstelle auch Ergebnisse für den Querschnittsbereich virtuelle Realität im Fahrzeugentwicklungsprozess erzielt. Die gesammelten Erfahrungen im Umgang mit *VRML*-Dateien und deren Darstellung in kooperativen Visualisierungstreffen über das Intra-/Internet lassen sich in verschiedenen Visualisierungsbereichen einsetzen.

Die Arbeit hat in den folgenden Gebieten Beiträge durch die Methodenentwicklung, -untersuchung und die Implementierung von Prototypen geleistet. Teilergebnisse wurden auf internationalen Konferenzen veröffentlicht.

- **Entwicklung eines dreidimensionalen graphischen Interaktionsmenüs:** In immersiven virtuellen Umgebungen verfügt der Benutzer nicht über die traditionellen Eingabegeräte 2D-Maus und Tastatur, sondern ist auf die Benutzung spezieller VR-Geräte, wie beispielsweise elektromagnetische Tracker, angewiesen. Üblicherweise steht dem Anwender ein zweidimensionales Menü basierend auf *Motif*, *MFC* o.ä. zur Verfügung. In

virtuellen Umgebungen muss das Interaktionsmenü in die virtuelle Welt und somit in den Szenengraphen integriert werden. Für diese Aufgabe gibt es noch keine Standard-Programmierschnittstelle.

Das in dieser Arbeit realisierte dreidimensionale Interaktionsmenü *G3Menu* wurde so konzipiert, dass es in die Applikationen der verschiedenen Szenengraph-APIs *OpenInventor*, *WorldToolKit* und *OpenGL Optimizer* integriert werden kann. Zum Einsatz kommt das *G3Menu* in den Visualisierungssystemen *VtCrash* und *PowerVIZ*.

- **Entwicklung eines VR-basierten Tools zur Visualisierung von Strukturs imulationen:** Etablierte Postprocessing-Systeme bieten dem Anwender nur einen geringen Grad an direkter Interaktion mit den dargestellten Ergebnisdaten. Der Anwender ist auf die Funktionalität des zweidimensionalen Menüs angewiesen. Beispielsweise ist die Selektion einer Fahrzeugkomponente nicht über einen interaktiven Auswahlmechanismus durch die Maus sondern nur über die Materialnummer der Komponente möglich.

  In dieser Arbeit wird das entstandene Postprocessing-System für Finite-Element-Simulationen *VtCrash* vorgestellt. Die Unterstützung einer Reihe von VR-Geräten und die direkten Benutzerinteraktionen sind die Besonderheiten dieses Systems. Die einfache, intuitive Bedienung erlaubt so auch „Nicht-Spezialisten“ den Umgang mit diesem Werkzeug.

- **VRML-Dateien als entwicklungsbegleitendes Medium:** Das *VRML*-Dateiformat ermöglicht als Internet-Standard für dreidimensionale Geometrien die Nutzung von plattformunabhängigen Applikationen für die Visualisierung von Berechnungsergebnissen. PCs und leistungsschwächere Workstations lassen sich somit auch zur Visualisierung einsetzen und der Internetzugang der Applikationen erlaubt zusätzlich kooperatives Arbeiten mehrerer Ingenieure in einer Visualisierungssitzung.

  Einzelne Fahrzeugkomponenten können über eine Schnittstelle von *VtCrash* als *VRML*-Dateien gespeichert werden. In dieser Arbeit entstandene *Java*-Applets und HTML-Seiten bieten dem Anwender auch in Standardbrowsern die Möglichkeit zum interaktiven Arbeiten und die synchronisierte Darstellung von zwei Berechnungsvarianten in einem Browser. Über eine Socket-Verbindung können zusätzlich Standardbrowser zu kooperativen Sitzungen zusammengeschlossen werden.

- **Visualisierung von zeitabhängigen Akustikdatensätzen:** Die Simulation der Luftdruckverteilung im Innenraum der Fahrzeugkabine basiert auf den Ergebnissen von Simulationsberechnungen der Karosserieschwingungen. Ziel der Untersuchungen ist die Reduktion der Geräuschentwicklung für die Passagiere. Die Lokalisierung des Ursprungs der Luftdruckwellen an den Übergängen Karosserie-Volumen ist ein wichtiger Schritt der Optimierung.

  Die kombinierte Visualisierung von zeitabhängigen Geometrie- und Volumen-Informationen stellt neue Anforderungen an die graphische Darstellung. Zur Bewältigung der Aufgabe wurde *VtCrash* um eine Struktur zum Laden und Visualisieren von

zeitabhängigen, unstrukturierten Volumengitter erweitert. Die entwickelten Volumenproben erlauben dem Anwender eine interaktive Analyse der simulierten Daten.

- **Interaktive Visualisierung von Strömungen in lokal verfeinerten, kartesischen Gittern:** Die Simulation der Außenhautumströmung hilft in einer frühen Phase der Entwicklung bei der Beurteilung des Karosseriedesigns. Prototypische Applikationen anderer Forschungsvorhaben demonstrierten die Vorteile der Interaktivität für die Visualisierung von Strömungen, fanden allerdings keine Anwendung im produktiven Prozess.

  Ziel von *PowerVIZ* ist die interaktive Visualisierung von Ergebnissen des Solvers *PowerFLOW* sowohl am Arbeitsplatz als auch im VR-Labor im produktiven Einsatz. Die lokal verfeinerten, kartesischen Gitter werden in dem System mit neu entwickelten und angepassten Algorithmen visualisiert. Die graphische Ausgabe erfolgt über die Programmierbibliothek *OpenGL Optimizer*.

- **Postprocessing-Systeme in der CAVE:** Die CAVE besitzt als immersive VR-Hardware-Umgebung ein hohes Potenzial für die Analyse von Berechnungssystemen. In ihr können auch unerfahrene Anwender Visualisierungssysteme steuern und so Berechnungsergebnisse analysieren. Die Verwendung einer CAVE erfordert nicht nur die Ansteuerung von VR-Geräten und mehrerer Graphiksysteme durch verschiedene Prozesse sondern auch neue Interaktionskonzepte.

  Für die beiden Visualisierungssysteme dieser Arbeit wurde eine Unterstützung der CAVE integriert. Hierzu wurde für die Szenengraph-APIs die Ansteuerung für CAVEs implementiert und die Einsatzmöglichkeiten des *G3Menus* untersucht.

- **Einbringung der Ergebnisse in den produktiven Prozess bei BMW:** Ziel dieser Arbeit ist nicht nur die Untersuchung und Implementierung neuer Visualisierungsansätze, sondern auch deren Einsatz im produktiven Prozess der Fahrzeugentwicklung. Beide Visualisierungssysteme der Arbeit wurden im produktiven Einsatz von BMW erfolgreich erprobt und demonstrierten so das Potenzial des interaktiven Arbeitens. Die Umwandlung der prototypischen Systeme in kommerzielle Produkte ist die konsequente Weiterentwicklung.

Zu den wesentlichen Beiträgen der Arbeit wurden Teilergebnisse in Tagungsbänden und Artikeln publiziert:

Gegenstand der Publikationen [74, 75, 106, 108] ist die Visualisierung von geometriebezogenen Finite-Element-Berechnungen mit *VtCrash*. Die Publikation der Aktivitäten im Bereich der kombinierten Darstellung von Schwingungs- und Akustikergebnissen erfolgte in [111]. Der Übersichtsartikel [107] präsentiert einen Querschnitt des kompletten Aufgabengebiets von *VtCrash*.

Die Ergebnisse der Internet-basierten Visualisierung sind Gegenstand von [109, 110]. In ihnen wird sowohl auf die entwickelten Interaktionsmechanismen in Standard-Internet-Browsern als auch auf die Techniken zur Synchronisation von *VRML*-Viewern und Internet-Browsern eingegangen.

Das Visualisierungssystem *PowerVIZ* wurde in [105] vorgestellt. Der Schwerpunkt dieses Artikels liegt auf den neuen und angepassten Visualisierungsmethoden für lokal verfeinerte Gitter.

Ein kurzer Überblick der gesamten Arbeit wurde in [104] gegeben. Er beschäftigt sich mit den wichtigsten Aspekten der interaktiven Visualisierung im Fahrzeugentwicklungsprozess.

Im Rahmen dieser Arbeit wurden Studien- und Diplomarbeiten betreut, die nachfolgend im Überblick zusammengestellt sind:

- Studienarbeiten
  - Frank Reck, *Strömungsvisualisierung in lokal verfeinerten kartesischen Gittern*, 1998, [94]
  - Manfred Weiler, *Visualisierung von Schwingungs- und Akustiksimulationen in einer Virtuellen-Realitäts-Umgebung*, 1998, [133]
- Diplomarbeiten
  - Edith Zimmermann, *Virtual Reality Modeling Language (VRML) als entwicklungsbegleitendes Medium zur Kommunikation von Konstruktionsinformationen*, 1998, [140]
  - Jürgen Beckmann, *3D Interaktionselemente als Benutzerschnittstelle in virtuellen Umgebungen*, 1999, [6]

1.3 Gliederung der Arbeit

Die ersten beiden Kapitel führen in das Umfeld der Arbeit ein. In den folgenden fünf Kapiteln werden Untersuchungen und Ergebnisse auf den Gebieten der interaktiven Visualisierung präsentiert und in den letzten beiden Kapiteln zusammengefasst.

Das **Kapitel 2** gibt einen Überblick der Visualisierungstechniken für numerische Simulationen. Es werden die Simulationstypen mit den entsprechenden Datenformaten und den Grundlagen der Visualisierungstechniken vorgestellt. Die angeführten Techniken bilden die Basis für die Entwicklungen dieser Arbeit.

Der aktuelle Stand der Entwicklung von VR-Techniken ist Gegenstand des **Kapitels 3**. Die Einführung in die VR-Techniken beinhaltet eine kurze Historie und gibt eine Übersicht der zur Zeit aktuellen Soft- und Hardwarelösungen. In dieser Arbeit kommen zwei der vorgestellten Szenengraph-Implementierungen und verschiedene Ein- und Ausgabegeräte zum Einsatz.

Auf den Fahrzeugentwicklungsprozess wird in **Kapitel 4** näher eingegangen. Es werden die einzelnen Teilschritte mit ihren Problemstellungen vorgestellt und Lösungsansätze präsentiert. Die Überlegungen dieses Kapitels fließen in die Struktur und Ausrichtung der beiden prototypischen Anwendungen ein.

Gegenstand von **Kapitel 5** ist die Mensch-Maschine-Schnittstelle. In diesem Kapitel wird die Diskussion über die verschiedenen Formen, mit denen der Mensch in einer virtuellen Umgebung intuitiv Eingaben tätigen kann, geführt. Das realisierte dreidimensionale Menü für verschiedene Szenengraphen wird als ein Lösungsansatz vorgestellt.

Die virtuelle Umgebung *VtCrash* zur Visualisierung von geometriebezogenen Simulationsergebnissen wird in **Kapitel 6** beschrieben. Neben einer Übersicht der Systemstruktur wird auf die realisierten Interaktionsmechanismen eingegangen. *VtCrash* spaltet sich in einen High-End-VR und einen *VRML*-basierten Bereich auf. Die Einsatzgebiete und der Funktionsumfang der beiden Bereiche wird vorgestellt und deren Vor- und Nachteile diskutiert. Insbesondere werden die Einsatzmöglichkeiten von Internet-Browsern im produktiven Einsatz und für kooperatives Arbeiten beleuchtet.

Eine Erweiterung von *VtCrash* erlaubt die synchrone Visualisierung von geometriebezogenen Vibrationsanalysen der Fahrzeugkarosserie und Akustiksimulationen in der Fahrzeugzelle. Die hierfür entwickelten Interaktionsmechanismen werden in **Kapitel 7** präsentiert. In dieser Arbeit werden Techniken zur interaktiven Berechnung von Schnittebenen in zeitabhängigen Volumen und die Berechnung von propagativ wachsenden Isoflächen vorgestellt.

Das speziell auf die Berechnungsergebnisse in lokal verfeinerten Gittern abgestimmte Visualisierungssystem *PowerVIZ* wird in **Kapitel 8** präsentiert. Mit dieser prototypischen Anwendung werden Lösungsansätze für eine effiziente und interaktive Darstellung von Berechnungsergebnissen der Strömungsberechnung aufgezeigt. Diese Applikation zeigt, wie auch *VtCrash*, interaktive Visualisierungstechniken für einen Teilbereich des Entwicklungsprozesses auf. *PowerVIZ* ist in der Lage Schnittebenen und Partikelbahnen für interaktive Bildwiederholungsraten mit mehr als 10 Hz zu berechnen. Wie auch *VtCrash* unterstützt *PowerVIZ* eine Reihe von Ein- und Ausgabegeräten.

Die Ergebnisse der einzelnen Forschungsbeiträge werden in **Kapitel 9** zusammengefasst. In diesem Kapitel werden zusätzlich die Erfahrungen aus dem produktiven Einsatz der Prototypen bei der *BMW Group* diskutiert und weiterführende Arbeiten vorgestellt, die aus Ergebnissen dieser Arbeit resultieren. Ein Gesamtresümee mit Ausblick findet sich in **Kapitel 10**.

Kapitel 2

Visualisierung numerischer Simulationen

Die Aufgabengebiete der numerischen Simulation und der wissenschaftlichen Visualisierung sind eng miteinander verflochten. Ohne die Simulation gibt es keine zu visualisierenden Berechnungsergebnisse und ohne die Visualisierung ist die Auswertung von großen Ergebnisdaten nicht möglich. Um das gemeinsame Ziel der effizienten Modellanalyse zu erreichen, müssen die Visualisierungstechniken an die Datenstrukturen der Simulationergebnisse angepasst und diese im Gegenzug einen leichten Zugang zu den Daten bereitstellen. Im folgenden Kapitel werden Datenstrukturen und Visualisierungstechniken als Ausgangspunkt dieser Arbeit vorgestellt.

2.1 Numerische Simulationen

Zur Beschleunigung des Fahrzeugentwicklungsprozesses werden numerische Simulationen in vielen Bereichen der digitalen Produktentwicklung eingesetzt. In dieser Arbeit werden Berechnungsergebnisse mit unterschiedlichen Datenstrukturen von Struktur- und Strömungssimulationen visualisiert.

2.1.1 Struktursimulationen

Finite-Element-Simulationen bestimmen auf Basis eines räumlich diskretisierten Modells (Finite-Element-Modell) das Verhalten der gegenseitigen Beeinflussung der einzelnen Geometrieelemente und ermöglichen die Analyse der Struktureigenschaften. Die diskrete Modellstruktur setzt sich aus eindimensionalen Balken-, zweidimensionalen Schalen- (dreieckige und viereckige Polygone) und den Volumenelementen (Tetraedern, Pyramiden, Prismen und Hexaedern) zusammen. Das Finite-Element-Modell enthält zusätzlich Informationen über die Materialeigenschaften der Elemente und die Randbedingungen.

Die inneren Spannungen und Dehnungen der Elemente sowie die verschiedenen Kräfte (Kontakt, Reibung, Trägheit und Dämpfung) werden zusammen mit den äußeren Kräften und Randbedingungen durch ein Differentialgleichungssystem beschrieben. Das Prinzip von d'Alembert in Lagrangescher Fassung besagt, dass die Arbeit der äußeren Kräfte gleich der Arbeit der inneren Kräfte ist, die an bzw. in einem Körper verrichtet werden. Zum Zweck von Finite-Element-

Analysen kann das Gleichgewicht für Körper, die in Kontakt miteinander stehen, in schwacher bzw. gerichteter Form ausgedrückt werden:

$$\underbrace{\int_B S\,\delta\varepsilon\,dv}_{a} + \underbrace{\int_B \rho_0 \ddot{x}\,\delta u\,dv}_{b} - \underbrace{\int_B \rho_0 g\,\delta u\,dv}_{c} - \underbrace{\int_{A_\sigma} F_0\,\delta u\,dA}_{d} + K + R = 0 \tag{2.1}$$

In die Gleichung gehen ein die Volumen der Körper (B), die von den äußeren Kräften beaufschlagte Oberfläche (A_σ), die Arbeit der Spannungen an den virtuellen Verzerrungen (innere Arbeit) (a), die Arbeit der Trägheitskräfte an den virtuellen Verschiebungen (b), die Arbeit der Schwerkraft an den virtuellen Verschiebungen (c), die äußeren Kräfte und Randbedingungen (d), die Kontaktarbeit (K) und die Reibungsarbeit (R).

Verschiedene numerische Verfahren kommen zur Lösung der Gleichung zum Einsatz und bestimmen für den zeitlichen Verlauf die Veränderungen der Kräfte, deren Ausbreitung und Auswirkung auf das Modell. Die numerischen Simulationen lassen sich in lineare und nichtlineare zeitabhängige Probleme unterteilen. Nicht umkehrbares Verhalten der Finiten-Element-Modelle, wie beispielsweise Crashtest- und Tiefzieh-Untersuchungen, wird von nichtlinearen Vorgängen simuliert. Periodisches Verhalten bei Schwingungsanalysen hingegen wird von linearen Verfahren behandelt.

Nichtlineare zeitabhängige Probleme können durch direkte Integration für diskrete Zeitbereiche gelöst werden. Dabei wird für einzelne Zeitpunkte das statische Gleichgewicht gesucht. Dies kann sowohl mit Hilfe impliziter als auch expliziter Verfahren erfolgen [60]. Mit impliziten Algorithmen (beispielsweise dem Newmark-Verfahren) können zwar bei beliebiger Zeitschrittgröße numerisch stabile Ergebnisse erzielt werden, jedoch erfordern die bei der notwendigen Linearisierung entstehenden Systemmatrizen eine rechenzeitaufwendige Invertierung innerhalb einer iterativen Lösung.

Die Industrie setzt oft das auf einem expliziten Algorithmus basierende Zentrale Differenzen Verfahren (ZDV) ein, bei dem die Zeitschrittgröße durch Massendichte, Steifigkeit und Elementgröße bestimmt wird und meist sehr klein ist. Im Gegensatz zum Newmark-Verfahren kann das Gleichungssystem beim ZDV unter Voraussetzung einer diagonal besetzten Massenmatrix mit einer einfachen Division nach den Beschleunigungen aufgelöst werden. Der Speicherbedarf ist verhältnismäßig gering, da die Berechnung elementweise erfolgt.

Die Nichtlinearität wird beim expliziten Verfahren ausschließlich über die inneren Kräfte, Kontaktkräfte und Reibungskräfte eingebracht, beim impliziten Ansatz zusätzlich noch über die Steifigkeitsmatrix. Das ZDV wird in Crash-Simulationen bevorzugt, auch weil die Komplexität der Modelle und ihre starken Änderungen infolge von Deformationen einen sehr kleinen Zeitschritt ohnehin notwendig machen.

Die Entwicklung der numerischen Verfahren hat zu einer Reihe am Markt verfügbarer Produkte geführt. Die einzelnen Simulationscodes sind meistens auf einen Simulationstyp optimiert und besitzen ihr eigenes Ergebnisdatenformat. Die folgende kurze Übersicht fasst die Aufgabengebiete, die Form der Datenformate und Vertreter kommerzieller Simulationscodes zusammen.

Crashtest-Simulationen werden mittlerweile intensiv im produktiven Fahrzeugentwicklungsprozess zur optimalen Auslegung von Karosserien für die Aufnahme maximaler Energie bei

Deformationen eingesetzt. Mit dem Ziel des Insassenschutzes werden komplette Deformationsvorgänge für Front-, Heck-, Seiten- und Pfahl-Crashs an digitalen Modellen numerisch simuliert. Von den heute verfügbaren Systemen für Crashtests wird die Oberflächenstruktur im Simulationsverlauf nicht verfeinert und es reicht somit aus, die Knotenkoordinaten und Zustandsparameter des Fahrzeugmodells in regelmäßigen Abständen abzuspeichern. Der Visualisierung wird daher eine statische, nicht adaptive Gitterstruktur mit zeitabhängigen Knoten- und Parameterlisten bereitgestellt. Weit verbreitete Crashtest-Simulationscodes sind *PAM-CRASH* der *ESI Group* und *DYNA3D* von *LS-DYNA*.

Kinematik-Simulationen analysieren das Bauteilverhalten in bestimmten mechanischen Bewegungsabläufen und geben so Aufschluss über das Deformationsverhalten und die Belastbarkeit zur Auslegung der Komponenten. Die zu speichernden Datenstrukturen entsprechen denen von Crashtest-Simulationen. Auf dem Markt befindet sich beispielsweise der Simulationscode *MARC* von *MSC Software*.

Die ebenfalls nichtlinearen Tiefzieh-Simulationen berechnen die Pressvorgänge zur Produktion von Fahrzeugkomponenten und haben die optimierte Auslegung der Bauteile und Presswerkzeuge zur Absicherung des Herstellungsprozesses als Ziel. Bei Pressvorgängen treten sehr starke Deformationen und Dehnungen in den Modellen auf, die eine adaptive Anpassung der Gitterstruktur erfordern. Ohne die Adaption verlieren die Tiefzieh-Simulationen stark an Genauigkeit und werden für die Analyse unbrauchbar. Somit muss die adaptive Gitterstruktur für jeden Zeitschritt zusätzlich zu den Koordinaten- und Parameterlisten abgespeichert werden und stellt neue Anforderungen an das Visualisierungssystem. Ein bekanntes System zur Tiefziehsimulation ist *PAM-STAMP* der *ESI Group*.

Die Analyse von Schwingungen ist ein Schwerpunkt von linearen numerischen Simulationen. Untersucht wird beispielsweise das Schwingungsverhalten einer Karosserie bei vorgegebener Motordrehzahl. Anhand der Ergebnisse werden Fahrzeugkomponenten zur Minimierung der Schwingungen verändert, um auftretenden Lärm zu vermeiden. Ein weiteres Anwendungsgebiet ist die Analyse des Schwingungsverhaltens von Motoren zur Erzielung hoher Laufruhe durch eine bessere Auslegung des Motorblocks. Die Ergebnisse der Schwingungsanalyse bilden häufig den Ausgangszustand für lineare Akustikuntersuchungen im Volumen des Fahrzeuginnenraumes bzw. des Motorraums und geben Aufschluss über die Form und Ausdehnung von Luftdruckwellen. Im Gegensatz zu den nichtlinearen Simulationen können die Bewegungen von vibrierenden Fahrzeugkomponenten oder Luftdruckänderungen in unstrukturierten Volumengittern durch reell- oder komplexwertige Schwingungsgleichungen abgebildet werden. Zur Speicherung der Ergebnisdaten reicht daher die statische Gitterstruktur inklusive der Funktionsparameter aus. Ein häufig verwendeter Simulationscode hierfür ist *NASTRAN* von *MSC-Software*.

Die Ergebnisse der verschiedenen Simulationscodes sind mittlerweile so genau und realitätsnah, dass sie sich in der Streubreite der realen Versuche befinden.

2.1.2 Strömungssimulationen

Die Strömungsmechanik entwickelt Verfahren für eine möglichst genaue Abbildung der Realität in numerische Modelle zur Berechnung des Strömungsverhaltens beliebiger Flüssigkeiten und Gase. Das Finite-Volumen- und das Lattice-Boltzmann-Verfahren sind zwei bekannte Ansätze.

Die Basis des Finite-Volumen-Verfahrens bildet die *Navier-Stokes-Gleichung* zur allgemeinen Beschreibung des Strömungsverhaltens inkompressibler Medien. Die Eigenschaften der Flüssigkeit oder des Gases werden durch die Dichte ρ und den Viskositätskoeffizienten μ des Mediums definiert. Die Differentialgleichung beschreibt die Bewegung von Flüssigkeiten und Gasen unter dem Einfluss von Druck p, kinematischer Viskosität $\nu = \frac{\mu}{\rho}$ und den Körperkräften $\vec{F}$ (beispielsweise der Gravitation):

$$\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \vec{\nabla} \vec{v} = -\frac{1}{\rho} \vec{\nabla} p + \nu \vec{\nabla}^2 \vec{v} + \frac{1}{\rho} \vec{F} \tag{2.2}$$

In vielen Fällen kann durch die Vernachlässigung der Körperkräfte $\vec{F} = 0$ eine Vereinfachung der Gleichung erzielt werden.

Der Finite-Volumen-Ansatz bietet durch die Unterteilung des Raums in kleine diskrete Volumina eine numerische Lösung der Navier-Stokes-Gleichung an. Unter Wahrung des Gleichgewichts von Masse, Impuls und Energie treten die diskreten Volumina in Interaktion. Die Geschwindigkeit des Zustroms, der statische Druck des Abstroms und die Wandgeschwindigkeit $\vec{v} = 0$ an Oberflächen definieren als wichtigste Randbedingungen das Strömungsfeld. Das Ergebnis einer Simulation wird nach der Initialisierung durch Bilanzierung der Volumina in einem iterativen Prozess berechnet, der zu einer stationären Strömung konvergieren kann. Praktische Anwendung findet der Finite-Volumen-Ansatz beispielsweise im Simulationscode *STAR-CD* von *Computational Dynamics Limited* und *Fluent* von *Fluent Inc.*.

Das Lattice-Boltzmann-Verfahren hingegen benutzt als grundlegendes Primitiv keine Finiten-Volumen, sondern die Partikel des Mediums. Die Bewegung eines Partikels wird hierzu an der Position $\vec{x}$ mit der Geschwindigkeit $\vec{v}$ zum Zeitpunkt t mit der Funktion $f(\vec{x}, \vec{v}, t)$ beschrieben. Durch die Vernachlässigung von Kollisionen würde sich ein Partikel in der Zeitdifferenz Δt von $\vec{x}$ nach $\vec{x} + \vec{v}\Delta t$ bewegen und die Geschwindigkeit von $\vec{v}$ nach $\vec{v} + \vec{a}\Delta t$ ändern, wobei F die Körperkräfte des Partikels sind. Die Betrachtung aller Partikel des Mediums unter dem Einfluss der Kollisionsfunktion Ω und $\Delta t \to 0$ führen schließlich zu der Boltzmann-Gleichung:

$$\frac{\partial f}{\partial t} + v_i \frac{\partial f}{\partial x_i} + F_i \frac{\partial f}{\partial v_i} = \Omega f \tag{2.3}$$

Die Lattice-Gas-Verfahren starten mit einer Verteilung der Partikeln in einem Strömungsfeld, in dem Raum, Zeit und Geschwindigkeit diskret sind. Die Partikel bewegen sich in Echtzeitperioden und können mit anderen Partikeln oder Objekt-Oberflächen kollidieren. Die diskrete Geschwindigkeit erlaubt die Berechnung von Kollisionseffekten ohne Rundungsfehler und numerische Artefakten. Bei Kollisionen im Strömungsfeld werden Masse, Impuls und Energie exakt erhalten, wohingegen bei Kollisionen mit Oberflächen eventuell auch Oberflächenmodelle und Randbedingungen beachtet werden. Die Berechnungsergebnisse sind immer zeitabhängig, sogar beim Auftreten einer statischen Strömung, bei der die Ergebnisse sich zwischen den verschiedenen Zeitschritten nicht unterscheiden.

Die Simulationsergebnisse dieser mikroskopischen Beschreibung von Partikelbewegungen und -kollisionen werden durch statistische Mittelung berechnet. Die Masse, Impuls und Energie

von einzelnen Partikeln wird addiert, um makroskopische Messungen von Dichte, Geschwindigkeit, Druck usw. zu erhalten. Der Simulationscode *PowerFLOW* [20] von *Exa Corporation* und der sich in der Entwicklung befindliche Code *BEST* vom *Lehrstuhl für Strömungsmechanik der Universität Erlangen* sind Lattice-Gas-Verfahren.

Der Finite-Volumen-Ansatz ist bereits seit längerer Zeit sowohl für Innen- als auch Außenumströmungen im Gebrauch. Die Gitterstruktur der Finiten-Volumen muss sich jedoch der Außenhülle von Objekten anpassen und kann somit nur in einem aufwendigen und kostenintensiven Schritt generiert werden. Der Lattice-Gas-Ansatz hingegen ist ein vergleichsweise junges Verfahren, das in jüngster Zeit einen größeren Kundenkreis in der Außenumströmung von Fahrzeugen besitzt. Sein entscheidender Vorteil ist die einfache und kostengünstige Generierung einer Gitterstruktur, die sich nicht den Objekthüllen anpassen muss. Daher kommen für Lattice-Boltzmann-Simulationen häufig lokal, verfeinerte kartesische Gitter zum Einsatz, die sich halbautomatisch erstellen lassen.

Im Fahrzeugentwicklungsprozess werden die beiden numerischen Lösungsansätze zur Strömungssimulation produktiv eingesetzt.

2.2 Visualisierung

Das Ziel der wissenschaftlichen Visualisierung ist die Generierung aussagekräftiger Darstellungen zur Analyse großer Datenmengen. Die Ergebnisdatensätze werden hierzu in einem ersten Schritt aufbereitet und anschließend auf geometrische Objekte abgebildet. Die beiden Schritte erfordern effiziente Algorithmen, die auf die Problemstellung angepasst wurden. Vorgestellt werden einige grundlegende Algorithmen, die für die vorliegende Arbeit benötigt wurden.

2.2.1 Statische und zeitabhängige Geometrien

Mittelfristig werden durchschnittliche Finite-Element-Modelle von Strukturs imulationen aus mehr als einer Million Elementen bestehen, womit die Modellkomplexität stärker steigt als die Leistung der Graphikhardware. Ein zeitabhängiger Datensatz dieser Größe stößt durch die Auslastung von Graphikhardware, Hauptspeicher und Bus-System-Bandbreite an die Grenzen der interaktiven Darstellbarkeit. Zur Handhabung der steigenden Datenmenge existieren Aufbereitungsalgorithmen und -techniken, um die geometrischen Repräsentationen im Vorfeld zu optimieren.

Die Reduktion der Polygonanzahl durch Ausdünnung und Optimierung der Polygonstruktur beschleunigt nicht nur die graphische Darstellung sondern spart auch Speicher und reduziert den Datentransport über das Bus-System. Mit geeigneten Algorithmen [103, 58, 88, 96] lässt sich die Polygonanzahl auf bis zu 10% reduzieren, jedoch müssen die simulierten Parameter der Materialeigenschaften bei der Polygonreduktion mit berücksichtigt werden, damit lokale Effekte nicht verloren gehen. Problematisch sind ebenfalls zeitabhängige Geometrien, da die Reduktion die Deformationen aller Zeitschritte beachten muss, schließlich darf die Ausdünnung nicht die Aussagekraft von Simulationsergebnissen beeinflussen.

Für die Darstellung werden die graphischen Objekte in einem Szenengraphen angeordnet, der die Knoten- und Indexlisten der Geometrien verwaltet. Eine Vielzahl von Szenengraph-APIs, die sich in vielen Merkmalen unterscheiden, befindet sich auf dem Markt. Durch die Wahl einer geeigneten Szenengraph-API und der Ausnutzung seiner Strukturen und Listen kann für zeitabhängige Geometrien Speicher eingespart und die Bildwiederholungsrate erhöht werden. Beispielsweise kann durch die Mehrfachreferenzierung von Indexlisten Speicher eingespart werden, da sich im zeitlichen Verlauf lediglich die Knotenkoordinaten ändern.

Die verschiedenen Knoten- und Gruppenobjekte der Szenengraphen stellen weitere Funktionalitäten für eine interaktive Visualisierung zur Verfügung. So können mit Hilfe von *Switch*-Knoten zeitlich veränderbare Geometrien dargestellt und durch den Einsatz von *Transform*-Knoten bewegt werden (siehe Kapitel 6.3). *Level of Detail*-Knoten können durch den automatischen Wechsel zwischen verschiedenen Auflösungsstufen einer Geometrie, komplexe Gitterstrukturen mit interaktiven Bildwiederholungsraten abbilden. Szenengraphen bieten durch diese Möglichkeiten der interaktiven Veränderung von Geometrien die direkte Manipulation der Visualisierung und eröffnen dem Benutzer eine neue Form im Umgang mit Berechnungsergebnissen. Die meisten Szenengraphen besitzen zusätzlich Optimierungsfunktionen um die Darstellungsgeschwindigkeit zu steigern, wie beispielsweise Sortiermechanismen zur Beschleunigung des Renderings oder zum Aussortieren nicht sichtbarer Polygone (engl. *culling*).

Gegenstand der Forschung sind momentan neue Visualisierungstechniken zur Generierung von Ersatzgeometrien aus den Finite-Element-Modellen. Kraftflussröhren [73, 95] bieten beispielsweise durch die Integration von Kräften in mehreren Geometrie-Schnitten entlang einer Oberflächenlinie Aufschluss über die Kräfteverteilung innerhalb von Fahrzeugkomponenten. Auch die direkte Ausnutzung von Graphikhardware für die Visualisierungsalgorithmen spielt eine immer wichtigere Rolle [71].

2.2.2 3D Skalar- und Vektorfelder

Die skalaren und vektoriellen Parameter werden in kartesischen, perimetrischen, curvilinearen und unstrukturierten Gittern angeordnet. Die Visualisierung der diskreten Datenmengen erfordert einige grundlegende Algorithmen auf dem Weg zur Generierung von Visualisierungsgeometrie.

2.2.2.1 Zellfindung

In den Gitterstrukturen liegen die Datenpunkte an den Zelleckpunkten oder den Zellmittelpunkten vor. Zur Bestimmung der Parameter zu einem vorgegebenen Punkt, muss zunächst die umschließende Zelle gefunden werden. Für diesen Schritt gibt es zu den einzelnen Gittertypen verschiedene Zellfindungsalgorithmen.

Das Auffinden der Zelle in einem kartesischen Gittern ist durch die parallelen Achsen und gleichgroßen Zellen vergleichsweise einfach. Mit der Gleichung 2.4 wird zur einer Position im physikalischen Raum $\vec{x}_p$ in einem Gitter mit der minimalen Koordinatengrenze $\vec{minBound}$ und

der konstanten Zellgröße *cellSize* die entsprechende Position $\vec{x}_c$ im diskreten Raum (engl. *computational space*) berechnet.

$$\vec{x}_c = \frac{\vec{x}_p - \vec{minBound}}{cellSize} = \begin{pmatrix} i \\ j \\ k \end{pmatrix} + \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{2.4}$$

Der ganzzahlige Anteil (i, j, k) von $\vec{x}_c$ ist der Achsenindex und die Nachkommastellen sind der Offset (α, β, γ) innerhalb der Zelle. Ähnlich lassen sich in perimetrischen Gittern mit variabler Zellgröße Index und Offset bestimmen.

Curvilineare und unstrukturierte Gitter benötigen kompliziertere Algorithmen, da ihnen die parallelen, geradlinigen Achsen fehlen. Häufig kommen iterative Algorithmen zum Einsatz, die sich ausgehend von einer geratenen Position schrittweise der gesuchten Position nähern. Beginnend am geratenen Punkt wird in Richtung der gesuchten Zellen die Nachbarzelle bestimmt, die der Ausgangspunkt für den nächsten Iterationsschritt ist.

Die Bestimmung der Nachbarzelle ist in diesen Algorithmen der aufwendigste Teilschritt. In curvilinearen Gittern nutzen Algorithmen wie beispielsweise die *Stencil Walk Method* [16] die in der Struktur vorhandene implizite Nachbarschaftsbeziehung durch Abbildung auf ein kartesisches Gitter aus. In unstrukturierten Gittern hingegen werden zu jeder Zelle die Nachbarn explizit gespeichert und der Algorithmus muss in jedem Schritt überprüfen, welche dieser Zellen sich in Richtung der gesuchten Position befindet. Hierzu wird der Richtungsvektor zur gesuchten Position mit den Seitenwänden geschnitten und anschließend mit der Nachbarzelle der geschnittenen Seitenwand fortgefahren. Die iterativen Algorithmen stoppen, sobald der Richtungsvektor die vorgegebenen Mindestlänge unterschreitet (siehe Kapitel 7.1.2).

Die Zellfindungsalgorithmen sind Teil vieler Visualisierungstechniken. Ihr Anteil am Berechnungsaufwand ist im Allgemeinen sehr hoch und daher sollten sie für kurze Rechenzeiten optimal an die Gitterstruktur angepasst sein.

2.2.2.2 Interpolation

An die erfolgreiche Zellsuche schließt die Bestimmung der gesuchten Parameter zu der gegebenen Position an. Die Interpolationsalgorithmen liefern zu jeder Raumposition im Gitter die Parameter und wandeln so die diskreten Datenrepräsentationen in eine kontinuierlich definierte Funktion um. Nahezu alle Visualisierungsalgorithmen basieren daher auf einem Interpolationsalgorithmus, der an die zugrunde liegende Gitterstruktur angepasst ist.

Der einfachste Algorithmus zur Parameterinterpolation ist das Nächste-Nachbar-Verfahren (engl. *nearest neighbour*) 0-ter Ordnung, das der gesuchten Position die Parameter des nach dem Euklidischen Abstand nächsten Nachbarn zuweist. In Zellen mit zentral definierten Parametern ist die Verteilung konstant und somit nach dem Auffinden der Zelle bestimmt. Die sich hieraus ergebene stückweise konstante Verteilung wird anhand von deutlichen Kanten in der Visualisierung sichtbar.

Bessere Interpolationsergebnisse werden mit Verfahren höherer Ordnung durch die Gewichtung aller Eckparameter einer Zelle erzielt. Für Hexaederzellen mit paarweise parallelen Seitenflächen bietet sich, durch die Abbildung auf einen Einheitswürfel, die trilineare Interpolation an.

Die Parameter an der gesuchten Position werden durch Aufsummieren der Eckpunktparameter ermittelt, die entsprechend ihrer Distanz zur gesuchten Position gewichtet werden. Die Gewichtung ist das Produkt der Differenzen von 1 und dem Abstand (α,β,γ) der einzelnen Koordinatenkomponenten des gesuchten Punktes innerhalb des Einheitswürfels. Nach der Zellfindung in einem kartesischen Gitter kann so innerhalb der Zelle (i,j,k) der Parameter p an der Position mit dem Offset (α,β,γ) bestimmt werden:

$$\begin{aligned} p \;=\; & p_{(i,j,k)} \cdot (1-\alpha)(1-\beta)(1-\gamma) + p_{(i+1,j,k)} \cdot \alpha(1-\beta)(1-\gamma) + \\ & p_{(i,j+1,k)} \cdot (1-\alpha)\beta(1-\gamma) + p_{(i,j,k+1)} \cdot (1-\alpha)(1-\beta)\gamma + \\ & p_{(i+1,j+1,k)} \cdot \alpha\beta(1-\gamma) + p_{(i+1,j,k+1)} \cdot \alpha(1-\beta)\gamma + \\ & p_{(i,j+1,k+1)} \cdot (1-\alpha)\beta\gamma + p_{(i+1,j+1,k+1)} \cdot \alpha\beta\gamma \end{aligned} \tag{2.5}$$

Aufwendigere Algorithmen sind für beliebig komplexe und deformierte Zellen erforderlich. Beispielsweise eignet sich das *Inverse Distance Weighting* für etliche Zelltypen. In diesem Verfahren bestehen die Gewichte w_i aus den invertierten Euklidischen Abständen d_i der n Zelleckpunkte $\vec{x}_i$ zur gesuchten Position $\vec{x}$:

$$d_i = \|\vec{x}_i - \vec{x}\| \tag{2.6}$$

$$w_i = \frac{\frac{1}{(d_i^k)}}{\sum_{j=1}^{n} \frac{1}{(d_j^k)}} \tag{2.7}$$

Eine schnelle Interpolation lässt sich mit $k = 2$ durchführen, da sich in diesem Fall die Wurzel des Euklidischen Abstands mit der Zweierpotenz aufhebt. Die gesuchten Parameter p werden anschließend als gewichtete Summe der Eckparameter p_i berechnet:

$$p = \sum_{i=1}^{n} w_i \cdot p_i \tag{2.8}$$

Falls ein Abstand d_i sehr klein wird, ist es ratsam, dem entsprechenden Gewicht w_i den Wert eins zu geben, und die verbleibenden Gewichte mit null zu bewerten, da durch Rundungsfehler bei der Berechnung des Bruchs $\frac{1}{d_i^k}$ eine Division durch Null auftreten kann.

Denkbar sind ebenfalls Interpolationsverfahren höherer Ordnung, jedoch sinkt mit steigender Komplexität der Algorithmen die Geschwindigkeit der Berechnungen. Für Echtzeitvisualisierungen bieten sich daher die vorgestellten einfachen Interpolationsverfahren an.

2.2.2.3 Isoflächen

Die Visualisierung von dreidimensionalen Skalarfeldern durch Isoflächen ist eine weit verbreitete und etablierte Visualisierungstechnik. Sie bietet dem Anwender die Gewinnung eines globalen Überblicks der inneren Struktur des Datensatzes und ist auch bei der Orientierung im Datenfeld behilflich. Zusätzlich lassen sich durch die Begrenzung auf Teilräume auch lokale Effekte analysieren. Die Berechnungsverfahren von Isoflächen können in indirekte und direkte Algorithmen unterteilt werden.

Indirekte Algorithmen generieren aus dem skalaren Datenfeld Polygone, die als Geometrie mit Standardverfahren dargestellt werden. Der *Marching Cubes* Algorithmus [82] ist das bekannteste indirekte Verfahren und extrahiert die Isoflächen zellenweise aus einem uniformen Gitter. Eine Isofläche schneidet eine Zelle nicht, wenn sich alle Parameter an den Hexaederecken unter- oder oberhalb des Isowertes befinden. In den restlichen Fällen wird die Isofläche innerhalb der Zelle mit Polygonen angenähert. Durch die Klassifikation der Zelleckpunkte $(0 : Parameter < Isowert; 1 : Parameter >= Isowert)$ in einem 8 Bit-Muster wird die Triangulierung der Isoflächen-Polygone innerhalb einer Zelle bestimmt. Die $2^8 = 256$ möglichen Triangulierungen können durch Ausnutzung von Symmetrien auf 14 Fälle reduziert und in einer Lookup-Tabelle zusammengefasst werden. Die Koordinaten der Polygonknoten werden durch lineare Interpolation zwischen den Zelleckpunkten auf den Zellkanten berechnet.

Die separate Betrachtung einzelner Zellen ist die Basis des einfachen *Marching Cubes* Verfahrens und lässt sich auch auf unstrukturierte Gitter übertragen, wie in Kapitel 7.1.3 gezeigt wird. Die ebenfalls in Kapitel 7.1.3 vorgestellte propagative Erweiterung ermöglicht den interaktiven Einsatz des *Marching Cubes* Algorithmus.

Das grundlegende Primitiv der direkten Ansätze ist die eigentliche Volumenzelle. Die Bilder von direkten Verfahren werden beispielsweise durch die Berechnung des Strahlungstransports (engl. *Ray-Casting*) oder durch die Verwendung von Texturspeicher generiert. Beim Ray-Casting werden Lichtstrahlen durch das semi-transparente Volumen geschickt und ergeben so die Projektion des Volumens in der Bildebenen. Die Semi-Transparenz wird durch eine Abbildungsfunktion aus den skalaren Parametern gewonnen. Die Ray-Casting Verfahren sind mit der momentan verfügbaren Rechnerleistung nicht interaktiv möglich, daher wurde in letzter Zeit massiv an den Einsatzmöglichkeiten von Textur-Hardware für die direkte Volumenvisualisierung gearbeitet [18, 134]. Diese Verfahren bilden uniforme Gitter auf den Textur-Speicher ab und nutzen die Interpolationsleistung der Graphik-Hardware für die Visualisierung des Volumens. Die Textur-basierten Verfahren erreichen zwar interaktive Raten, sind aber durch die Größe des Textur-Speichers begrenzt, der für große Datensätze noch nicht ausreicht.

Aus Performanz- und Speichergründen kommen in dieser Arbeit nur indirekte Verfahren zum Einsatz.

2.2.2.4 Berechnung von Partikelbahnen

Partikelbahnen geben sowohl Aufschluss über das globale Flussverhalten innerhalb eines räumlichen Vektorfeldes, als auch über lokale Effekte in Wirbeln. Ausgehend von einer Startposition werden masselose oder massebehaftete Teilchen durch das Simulationsvolumen verfolgt und können beispielsweise als Linien, Rotationsbänder oder Glyphen visualisiert werden. Das reale Pendant zu den virtuellen Partikelbahnen ist die Rauchsonde im realen Windkanal.

Die Berechnung einer masselosen, zeitabhängigen Partikelbahn (*Path Line*) basiert auf der numerischen Integration der gewöhnlichen Differentialgleichung:

$$\frac{d\vec{x}(t)}{dt} = \vec{v}(\vec{x}(t), t) \tag{2.9}$$

Das Partikel befindet sich zum Zeitpunkt t an der Position $\vec{x}$ und bewegt sich mit der Geschwin-

digkeit $\vec{v}(\vec{x}(t),t)$. Der Startpunkt $\vec{x}_0$ liefert die Anfangswertbedingung $\vec{x}(t_0) = \vec{x}_0$ für die Differentialgleichung. Das Ergebnis dieser Berechnung entspricht der Photographie eines bewegten Partikels mit einer langen Belichtungszeit.

Stream Lines hingegen entsprechen der Lösung der Differentialgleichung:

$$\frac{d\vec{x}(s)}{ds} = \vec{v}(\vec{x}(s),t) \tag{2.10}$$

Die Zeit t wird in diesem Fall als konstant angenommen und die resultierende Kurve hängt vom Parameter s ab. Stream Lines repräsentieren somit Partikelbahnen im stationären Vektorfeld zum Zeitpunkt t. In vielen Anwendungen werden stationäre Datensätze als Mittelung eines zeitabhängigen Vorgangs gespeichert und mit Stream Lines visualisiert.

Numerisch berechnet werden Partikelbahnen in drei Schritten. Im ersten wird zu der Position $\vec{x}$ die umschließende Zelle gesucht, in der anschließend die Geschwindigkeit $\vec{v}$ an der Position $\vec{x}$ interpoliert wird. Durch Integration im Vektorfeld werden die folgenden Teilchenposition iterativ berechnet, bis das Teilchen das Volumen verlässt oder eine maximale Schrittanzahl erreicht wurde.

Die numerische Integration erfolgt durch expliziten Algorithmen, die schrittweise den Linienzug der Partikelbahn bestimmen. Das Euler-Schema und die Runge-Kutta-Verfahren verschiedener Ordnung werden im Allgemeinen mit fester Schrittweite eingesetzt:
Euler-Schema:

$$\vec{x}_{n+1} = \vec{x}_n + \Delta t \cdot \vec{v}(\vec{x}_n) \tag{2.11}$$

(2.12)

Runge-Kutta 2. Ordnung:

$$\begin{aligned} \vec{x^*}_{n+1} &= \vec{x}_n + \Delta t \frac{1}{2} \cdot \vec{v}(\vec{x}_n) \\ \vec{x}_{n+1} &= \vec{x}_n + \Delta t \frac{1}{2} \cdot (\vec{v}(\vec{x}_n) + \vec{v}(\vec{x^*}_n)) \end{aligned} \tag{2.13}$$

Runge-Kutta 3. Ordnung:

$$\begin{aligned} \vec{k}_1 &= \Delta t \cdot \vec{v}(\vec{x}_n) \\ \vec{k}_2 &= \Delta t \cdot \vec{v}(\vec{x}_n + 0.5 \cdot \vec{k}_1) \\ \vec{k}_3 &= \Delta t \cdot \vec{v}(\vec{x}_n + 0.25 \cdot (\vec{k}_1 + \vec{k}_2)) \\ \vec{x}_{n+1} &= \vec{x}_n + \frac{1}{6}(\vec{k}_1 + \vec{k}_2 + 4 \cdot \vec{k}_3) \end{aligned} \tag{2.14}$$

Die Genauigkeit eines Integrationsschrittes steigt mit der Anzahl an Abtastpunkten und somit mit der Höhe der Ordnung. Der größere Rechenaufwand pro Integrationsschritt wird durch eine mögliche längere Schrittweite wieder ausgeglichen.

Die Integrationsschemata mit fester Schrittweite bilden die Basis für die adaptive Schrittweiten-Steuerung. Mit zwei kombinierten Schemata mit fester Schrittweite wird ein Integrationsschritt berechnet und die Differenz als Fehlerabschätzung zur Anpassung der Schrittweite des adaptiven Verfahrens herangezogen. Die neue Schrittweite τ_j^* wird mit der Schätzformel aus der alten Länge τ_j bestimmt:

$$\tau_j^* = \sqrt[p+1]{\frac{\rho \cdot TOL}{|\varepsilon_{j+1}|}} \quad , \rho \leq 1 \tag{2.15}$$

Falls für ein Verfahren der Ordnung p der lokale Fehler $|\varepsilon|$ über der Toleranzschwelle TOL liegt, wird mit einer kürzeren Schrittweite versucht das Fehlermaß einzuhalten. Ansonsten verlängert der Algorithmus den Schritt, solange $|\varepsilon| \leq TOL$ eingehalten wird. Der lokale Fehler $|\varepsilon|$ wird aus der Differenz von zwei Verfahren mit fester Schrittweite, wie beispielsweise Runge-Kutta 2. und 3. Ordnung, gewonnen.

Die Anpassung und Optimierung der Partikelbahnberechnung an verschiedenen Gittertypen und deren Integration in Visualisierungssysteme wird von einer Vielzahl an Arbeiten untersucht [99, 126, 76, 90, 127, 123, 47, 46]. Häufig bilden die Algorithmen der Teilchenbahnberechnung auch die Basis für weiterführende Visualisierungstechniken. Beispielsweise benutzt LIC (*Line Integral Convolution*) Partikelbahnen zur Faltung einer verrauschten Textur [129, 114, 62, 63] und Westermann [135] visualisiert mit direktem Volumenrendering Verwirbelungen durch die Abbildung der Krümmung von Teilchenbahnen in ein dreidimensionales Skalarfeld.

2.3 Erweiterungen der existierenden Methoden

Die existierenden Methoden wurden an vielen Stellen den Anforderungen der vorliegenden Arbeit nicht gerecht, so dass Erweiterungen und Anpassungen der Algorithmen nötig wurden. Die speziellen Datenstrukturen der Simulationsergebnisse erforderten die Anpassung der Visualisierungstechniken und die Voraussetzung der Echtzeitfähigkeit führte zu Geschwindigkeitsoptimierungen. Ein Augenmerk wurde zusätzlich auf die Einsatzfähigkeit der Algorithmen im produktiven Einsatz geworfen.

Für die Visualisierung großer Finite-Element Datensätze aus Struktursimulationen wurde ein Algorithmus zur Reduktion der Polygonanzahl entwickelt, der unter Berücksichtigung der Zeitabhängigkeit des Datensatzes die Anzahl der Polygone in der Ladephase des Visualisierungssystems *VtCrash* auf 50% reduziert. Extrem wichtig für den produktiven Einsatz ist die schnelle Reduktion in der Ladephase, da Anwender im Allgemeinen eine Konvertierung der Daten oder eine lange Ladephase nicht akzeptieren. Die Szenengraph-Struktur von *VtCrash* wurde auf zeitabhängige Geometrien angepasst und bietet eine optimale Schnittstelle für eine Reihe von Interaktionsmechanismen, wie beispielsweise das Bewegen von animierter Geometrie oder die Berechnung von zeitabhängigen Schnitten.

Die Netzwerk-Tauglichkeit der Visualisierungsalgorithmen wurde in der VRML-basierten Erweiterung von *VtCrash* erprobt. Die Interaktionsmechanismen wurden an Standard-*VRML*-Browser angepasst und bieten die Möglichkeit von kooperativem Arbeiten über das Inter-/Intranetz.

Die Visualisierung von kombinierten Schwingungs- und Akustiksimulationen erfordert die parallele Visualisierung von Geometrie- und Volumendatensätzen inklusive einer effiziente Datenverwaltung. Interaktive Bildwiederholungsraten wurden durch die Anpassung des Zellfindungsalgorithmus an unstrukturierte Gitter erreicht. Die ebenfalls entstandenen Volumenproben bieten dem Anwender einen intuitiven Interaktionsmechanismus, mit dem im Alltagsgeschäft einfach und gezielt gearbeitet werden kann. Der Standard *Marching Cubes* Algorithmus wurde für das interaktive Arbeiten um eine propagative Variante erweitert, deren Saatpunkt interaktiv durch eine Volumenprobe definiert wird und dessen Isofläche sich anschließend im Volumen ausbreitet. Die Parallelisierung der Visualisierungsalgorithmen erhöhte zusätzlich die Reaktionszeiten des Systems.

Im Bereich der Strömungsvisualisierung wurde eine effiziente Datenstruktur für lokal verfeinerte kartesische Gitter speziell für die Belange der Visualisierung entwickelt. Die Algorithmen für Zellfindung, Schnitt- und Partikelbahnberechnung wurden zur Optimierung der Geschwindigkeit auf diesen Gittertyp angepasst. Die Separierung von Geometrie- und Volumendaten erforderte eine Kollisionsberechnung zwischen den Partikelbahnen und der Fahrzeugoberfläche, die durch die Verwendung eines Octrees Aufschlagpunkt-Berechnungen in Echtzeit ermöglicht. Für die Visualisierung von skalaren Parametern auf der Fahrzeugoberfläche und den Schnittebenen kommen Hardware-Texturen zum Einsatz.

Kapitel 3

Virtuelle Realität

3.1 Definition von virtueller Realität

Der Begriff *Virtual Reality* (VR) entstand Ende der 80er Jahre. Er bezeichnet im weitesten Sinne eine neue Art von Benutzungsschnittstelle zwischen Mensch und Rechner. Die Kernidee von Virtual Reality ist es, den Menschen unmittelbar in eine dreidimensionale, computergenerierte Welt zu integrieren. Ziel ist es, die „Immersion“ zu erreichen, das heißt, dem Benutzer das Gefühl zu vermitteln, in der virtuellen Welt anwesend zu sein. Die Interaktion mit dem Rechner erfolgt dabei nicht über eine graphische Benutzungsoberfläche, Monitor, Tastatur oder Maus. Vielmehr erfolgt die Kommunikation mit dem Rechner in Echtzeit im dreidimensionalen Raum in einer intuitiven Art und Weise, wobei die natürlichen Fähigkeiten des Menschen genutzt werden sollen. Eine perfekte Virtual-Reality-Anwendung sollte mehrere Sinne des Menschen gleichzeitig ansprechen, beispielsweise das Sehen, das Hören und das Tasten.

Die perfekte Virtual-Reality-Anwendung im oben beschriebenen Sinn ist aufgrund technologischer Grenzen heute noch in weiten Teilen eine Vision. Das immersive Arbeiten mit großen und komplexen 3D-Modellen, wie beispielweise dem 3D-Modell eines Automobils, eines Flugzeugs oder einer Industrieanlage, ist jedoch bereits heute möglich. Zunehmende Verbreitung finden hier Mehrseiten-Stereoprojektionsräume (CAVE). Ein Hochleistungs-Graphikrechner erfasst dabei die Kopf- und Handbewegungen des Benutzers (Tracking) und generiert für den aktuellen Betrachterstandort perspektivisch korrekte stereoskopische Bilder des 3D-Modells. Wichtig ist, dass dies in Echtzeit erfolgt. Eine Kopfdrehung des Benutzers muss eine sofortige entsprechende Änderung der Ansicht des 3D-Modells zur Folge haben, andernfalls ist das VR-System nicht benutzbar. Echtzeitfähige 3D-Graphik ist daher eine der wichtigsten technologischen Grundlagen für ein solches High-End-VR-System.

Während für voll-immersive VR-Anwendungen, auch als *Immersive Virtual Reality* bezeichnet [128], Bildwiederholungsraten größer als 10 Bilder pro Sekunde [11] wünschenswert sind, kann bei einer reinen Visualisierung großer 3D-Modelle in Form eines „Walkthrough“ bzw. „Flythrough“ eine Bildrate von 5 bis 10 Bildern pro Sekunde noch ausreichend sein. Ein Tracking ist dann jedoch nicht mehr sinnvoll und in solchen Fällen erfolgt die Navigation optimalerweise über das dreidimensionale Eingabegerät *Spacemouse* mit der Ausgabe über eine einfache Projek-

tionswand oder einen Monitor in Kombination mit einer graphischen 2D-Benutzungsoberfläche. Die Definition von virtueller Realität wird in diesem Fall jedoch sehr weit gefasst, denn im strengen Sinne handelt es sich bei einer solchen interaktiven Darstellung nicht um virtuelle Realität und wird folglich häufig als *Fish Tank VR* bezeichnet. In der Praxis wird der Begriff VR jedoch weiter gefasst, so dass Visualisierungen großer 3D-Modelle durchaus in den Kontext der Technologie Virtual Reality gestellt werden können. Diese weiter gefasste Definition von VR schließt auch die Visualisierung von 3D-Modellen mittels der Internet-Technologie auf Basis von *VRML* („Virtual Reality Modelling Language“) ein.

Im Umfeld der virtuellen Realität haben sich im Laufe der Zeit weitere Echtzeitdarstellungen von dreidimensionaler Graphik entwickelt, die sich in entscheidenden Punkten absetzen.

Die Projektion von generierter Geometrie in das reale Sichtfeld des Benutzers wird als *Augmented Reality (AR)* bezeichnet. Im Gegensatz zu der virtuellen Realität wird der Benutzer nicht von der realen Welt abgeschottet, sondern die reale Welt wird durch synthetische Bilder erweitert. Neben dem Bau von geeigneten *see-through*-Brillen ist die präzise Ausrichtung der virtuellen an der realen Welt ein wichtiges Forschungsgebiet [86]. Zum Einsatz kommt diese Technik beispielsweise im Flugzeugbau zur Unterstützung der Arbeiter beim Einbau von Kabelsträngen und elektronischen Elementen [85].

Unter dem Begriff *Mixed Reality* wird der gleichzeitige Einsatz von Ein- und Ausgabegeräten der virtuellen und der Augmented Realität zusammengefasst. Der erforderliche Hardware-Aufbau kann beispielsweise aus der Kombination von Touchscreens und durchsichtigen LCD-Bildschirmen in einer Workbench-ähnlichen Anordnung für kollaboratives Arbeiten, wie in [33] vorgestellt, bestehen.

3.2 Historie

Die virtuelle Realität hat eine lange und ereignisreiche Historie, die sehr weit gefasst mit dem Zeichnen von Bildern auf Felswände und im engeren Sinne mit den ersten graphikfähigen Monitoren beginnt. In diesem Kapitel werden die Meilensteine in der Hard- und Software-Entwicklung aufgeführt, die schließlich zum momentanen Stand der Techniken der virtuellen Realität geführt haben. Wichtig für die Entwicklung waren nicht nur die bahnbrechenden Entdeckungen im Bereich der Ein- und Ausgabegeräte, sondern auch deren Kommerzialisierung, ohne die die virtuelle Realität keine Beachtung und Verbreitung gefunden hätte.

1963 Ivan Sutherland veröffentlicht mit seiner Doktorarbeit *Sketchpad – A Man-Machine Graphical Communication System* [118] die erste interaktive, graphische Mensch-Maschine-Schnittstelle (siehe Abbildung 3.1 links).

1965 Ivan Sutherland veröffentlicht das *Ultimate Display* [119]

1967 Fred Brooks et al. entwickeln das Force Feedback System *GROPE* [5, 9] an der Universität von North Carolina.

1968 Ivan Sutherland und Bob Sproull entwickeln aus einem Helikopter Infrarot-Display das erste Virtual Reality Head-Mounted-Display [120] (siehe Abbildung 3.1 rechts). Die dargestellte Graphik bestand nur aus dem einfachen Drahtgittermodell eines Raumes.

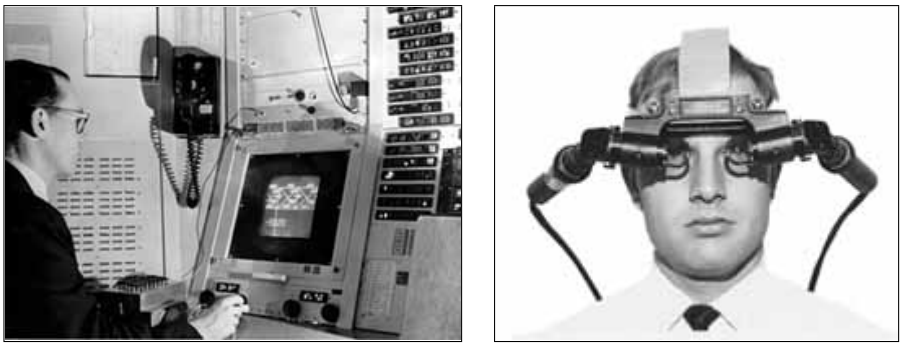

Abbildung 3.1: Historische Bilder aus den Anfängen der virtuellen Realität. Links: Das Sketchpad von Ivan Sutherland, 1965. Rechts: Erstes Head-Mounted-Display entwickelt von Sutherland, 1968.

1979 Raab et al. beschreiben die theoretischen Grundlagen für das magnetische Tracking-System *Polhemus*. In [92] werden auch Überlegungen zur Applikationsanpassung, die Sender/Empfänger-Problematik und die Veränderungen des Magnetfeldes durch Metallgegenstände angesprochen.

1981 Thomas Furness entwickelt das virtuelle Cockpit [41], Projektname „Super Cockpit“.

1983 Mark Callahan baut das erste „see-through“ Head-Mounted-Display am MIT. Dieses HMD erlaubt die Projektion von virtueller Graphik in das Betrachterblickfeld ohne die Verdeckung der realen Welt.

1984 Mike McGreevy und Jim Humpries entwickeln VIVED (VIrtual Visual Environment Display) für zukünftige NASA Astronauten [36]. Dieses System war seinerzeit mit $20.000 das preiswerteste HMD weltweit.

1984 Gründung von VPL Research Inc. (Visual Programming Language) durch Jaron Lanier. VPL entwickelt und verkauft VR-Hard- und Software. Bekannte Produkte sind:

- *DataGlove*: Datenhandschuhe
- *EyePhone*: farbiges Head-Mounted-Display
- *AudioSphere*: System zur Echtzeitgenerierung von 3D Sound
- *Isaac*: graphisches Echtzeit-Renderingsystem

1987 Thomas Zimmermann entwickelt den ersten Datenhandschuh.

1989 Jaron Lanier, CEO von VPL prägt den Begriff Virtual Reality.

1989 Mark Bolas gründet *Fakespace Labs* und vermarktet den *BOOM*.

1990 *Sense8 Corporation* wird von Pat Gelband gegründet. Das Produkt *WorldToolKit* ist eine Szenengraph Bibliothek mit einer Vielzahl an Gerätetreibern.

1993 Präsentation der CAVE auf der SIGGRAPH [24, 23].

1993 SGI kündigt das Graphiksystem *RealtiyEngine* an.

1994 Die „Virtual Reality Society“ wird gegründet

Ausgehend von der ersten interaktiven Graphikschnittstelle von Ivan Sutherland hat sich die Technik der virtuellen Realität zu einer großen Vielfalt an Ein- und Ausgabegeräten entwickelt. Softwareseitig haben problemorientierte Algorithmen die Leistungsfähigkeit der VR-Systeme gesteigert.

3.3 Begriffsdefinitionen

Einige Begriffe der Virtuellen Realität besitzen Mehrdeutigkeiten, die zum besseren Verständnis dieser Arbeit durch eindeutige Definition ausgeräumt werden.

- Als **virtuelle Welt** oder **virtuelle Umgebung** wird eine Applikation bezeichnet, wenn sie Interaktionsmechanismen mittels VR-Geräten anbietet und der Anwender zumindest semi-immersiv unterstützt wird.

- Die Schnittstelle zwischen Benutzer und virtueller Welt sind die **VR-Geräte**. Auf der einen Seite erfassen sie reale Positionen und bilden diese in den virtuellen Raum ab und zum anderen stellen sie die generierten Bilder dar.

- Ein **Projektionsraum** bietet für mehr als 10 Teilnehmer Platz und besitzt eine fest-installierte, stereofähige und mindestens 2x2 Meter große Leinwand. Die Projektion erfolgt entweder von vorne (Aufprojektion) oder durch die Leinwand (Durch- oder Rückprojektion). Teilweise sind Projektionsräume auch mit VR-Eingabegeräten ausgestattet.

- Als **Immersion** wird der Eindruck des Eintauchens in eine virtuelle Welt bezeichnet, wobei dieser Effekt durch die VR-Ein- und Ausgabegeräte erzeugt wird. In dieser Arbeit wird der Begriff Immersion enger gefasst und das subjektive Eintauchen, wie beispielsweise durch die Handlung eines Spiels, ausgeschlossen.

- In **immersiven** oder **vollimmersiven Umgebungen** ist das Blickfeld des Betrachters vollständig mit einer Stereodarstellung abgedeckt und der Anwender kann nur über VR-Geräte, wie Datenhandschuh und Trackingsystem, mit der virtuellen Welt interagieren. Solche Hardware-Aufbauten, wie beispielsweise die *CAVE* oder der *BOOM*, befinden sich hauptsächlich in VR-Labors und sind sehr kostenintensiv.

- In einer **semi-immersiven Umgebung** ist das Blickfeld der Betrachters nicht komplett durch eine Stereodarstellung abgedeckt und es entsteht somit keine vollständige Immersion. Ebenfalls findet kein massiver Einsatz von VR-Geräten statt. Arbeitsplatz- und Powerwall-Umgebungen sind beispielsweise semi-immersiv.

- **Interaktives Arbeiten** in einer virtuellen Umgebung erfordert Bildwiederholungsraten der dreidimensionalen Graphik mit mindestens 10 Hz und Antwortzeiten der Eingabegeräte von höchstens 0,1 Sekunden, da andernfalls der Eindruck der Immersion verloren geht.

- Mit **High-End-VR** werden virtuelle Umgebungen mit hoher Immersion und massivem VR-Geräteeinsatz bezeichnet. Im Gegensatz hierzu besitzt die **Low-End-VR** nur eine sehr geringe Immersion des Benutzers. Diese auch als **Fish-Tank VR** bezeichnete Umgebung beschränkt sich auf einen stereofähigen Monitor kombiniert mit einem 3D/6D-Eingabegerät, wie beispielsweise der Spacemouse.

3.4 VR-Hardware

Die Schnittstelle zwischen dem Menschen und der virtuellen Realität bilden die speziell entwickelten Ein- und Ausgabegeräte, deren Ziel die vollständige Immersion des Anwenders ist. Seit dem ersten *Head-Mounted-Display* von Sutherland werden immer neue Geräte entwickelt und auf den Markt gebracht, die so entstandene Vielfalt bietet bereits für viele Arbeitsgebiete umfangreiche Lösungen an. Die Gerätepalette kann nach ihren Aufgaben in Display-, Tracking- und Interaktionssysteme unterteilt werden. Mit Augenmerk auf die Visualisierung von Berechnungsergebnissen werden im Folgenden die wichtigsten Geräte aus Forschung und Industrie vorgestellt.

3.4.1 Display-Systeme

Kernstück der Hardware-Umgebung ist die Abbildung der virtuellen Welt auf die Netzhaut des Anwenders. Für die Immersion entscheidend ist die vollständige Abdeckung des 200 Grad umfassenden menschlichen Blickfeldes mit einer möglichst hochaufgelösten, stereoskopischen Darstellung [2, 98]. Die gegenwärtig erhältlichen Geräte lassen sich aufgrund ihrer Bauart in bewegliche und fest installierte Display-Systeme aufteilen.

Bewegliche Display-Systeme

Bei den beweglichen Display-Systemen wird das generierte Bild direkt vor die Augen des Anwenders projiziert. Hierzu trägt der Anwender eine am Kopf befestigte Brillenkonstruktion, engl. *Head-Mounted-Displays (HMD)*, bestehend aus zwei LCD- oder Röhrenbildschirmen kombiniert mit einem Trackingsystem zur Bestimmung der momentanen Position und Orientierung des Kopfes. Die Ergonomie dieser Systeme hängt stark vom Gewicht der Konstruktion ab, da bei einem zu hohen Gewicht dem Anwender die Immersion verloren geht. Um das Blickfeld des Menschen vollständig abzudecken, werden also leichte und hochauflösende Displays benötigt.

Die auf dem Markt befindlichen LCD-Bildschirme sind zwar leicht und besitzen mit bis zu 1280x1024 Pixeln eine ausreichende Auflösung, decken aber maximal nur 80 Grad des Blickfeldes ab (siehe Abbildung 3.2). Bedingt durch das geringe Blickfeld ist die Immersion in solchen Geräten nicht ausreichend. Kleine Röhrenmonitore hingegen können durch den Einsatz von Linsen größere Bereiche des Blickfeldes abdecken. Mit einem Gewicht von mehr als einem Kilogramm ist das Arbeiten mit solchen Röhrenhelmen sehr anstrengend und nur in einem sehr begrenzten Zeitrahmen möglich. Der hohe Preis von *HMDs* ist ein weiterer Grund, warum sie sich bisher nicht durchgesetzt haben.

Abbildung 3.2: Eine Auswahl an Head-Mounted-Displays, die momentan auf dem Markt erhältlich sind. Vlnr: n-vision Datavisor (78 Grad FOV, 1280x1024), Virtual Research V8 (60 Grad FOV, 1920x480), Kaiser ProView 50 (augmented, 50 Grad FOV, 1024x768), Kaiser Proview 100 (augmented, 100 Grad FOV, 1024x768).

Mit dem *BOOM (Binocular Omni-Oriented Monitor)* brachte die Firma *Fakespace* ein Display-System auf den Markt, das sich der Anwender ohne Gewichtsbelastung vor das Gesicht halten kann (siehe Abbildung 3.3). Die Aufhängung zweier Röhrenbildschirme an einem Galgen sollen die Probleme der *HMDs* beseitigen. Die Gelenke des Galgens erlauben die Bewegung des *BOOM*-Displays in allen sechs Freiheitsgraden. Mit einer Auflösung von 1280x960 Pixeln und einer Abdeckung des Blickfeldes mit 140 Grad besitzt der *BOOM* die qualitativ besten Eigenschaften. Ein weiterer Vorteil ist die hohe Genauigkeit der Mechanik von 2 mm gegenüber den elektromagnetischen Trackern der *HMDs*. Einschränkungen besitzt der *BOOM* durch den geringen Aktionsradius und das störende Standbein, um das der Anwender herumlaufen muss. Auch reduziert das Nachschwingen des Displays bei Kopfbewegungen bedingt durch die Mas-

Abbildung 3.3: Die verschiedenen Varianten des *Fakespace BOOM*. Vlnr.: Fakespace FS2, FS2+PinchGlove, BOOM 3C, PUSH BOOM.

senträgheit den Immersionseffekt. In verschiedenen Derivaten, wie dem arbeitsplatzorientierten *PUSH-BOOM* oder einem am Kopf fixierten Display *BOOM*, wurde versucht, die Probleme zu beseitigen (siehe Abbildung 3.3). Der ebenfalls sehr hohe Preis verhinderte eine weite Verbreitung des *BOOMs*, der so hauptsächlich in VR-Labors Anwendung findet.

Die Idee der Galgenkonstruktion wurde auch in anderen Display-Systemen erprobt. Das in 3.4 abgebildete *Window VR* von *Virtual Research Systems* besteht aus einem herkömmlichen LCD-Display, das zur Positionsbestimmung an einem Galgen aufgehängt ist. Durch dieses „Fenster“ kann der Betrachter nun die virtuelle Welt durch Positionierung des Bildschirms erforschen. Aufgrund des Fehlens der stereoskopischen Darstellung ist die Immersion des Gerätes sehr gering, es eignet sich aber gut für Produktpräsentationen in Verkaufsräumen und auf Messeständen.

Abbildung 3.4: Das *Window VR* von *Virtual Research Systems*

Fest installierte Display-Systeme

Ein großer Nachteil der beweglichen Systeme ist die vollständige Abschottung des Anwenders von der realen Welt. In den festinstallierten Display-Systemen führt die Sichtbarkeit der realen Hände oder auch anderer Teilnehmer zu einem höheren Immersionsgrad, der durch die vollständige Abdeckung des menschlichen Blickfeldes mit Hilfe großer Leinwände noch verstärkt wird. Ebenfalls wirkt sich das Fehlen einer schweren Brille positiv aus.

Der Stereoeffekt wird in fest installierten Systemen nicht wie beim *HMD* durch zwei separate Displays erzeugt, sondern durch die aktive oder passive Stereoprojektion. In beiden Fällen wird das Stereobild auf ein Display abgebildet und durch Brillen wieder getrennt. Für ein aktives Stereodisplay werden die Bilder für das linke und rechte Auge mit einer Frequenz > 100 Hz auf die Leinwand projiziert. Die Synchronisation einer Shutter-Brille, wie sie beispielsweise von *CrystalEyes* (siehe Abbildung 3.5) angeboten wird, mit der Frequenz des Bildschirmes erfolgt über einen Infrarot-Emitter und erzeugt durch abwechselndes Abdecken eines Auges den Ste-

reoeffekt. Die passive Stereoprojektion hingegen benutzt Polarisationsfilter an den Projektoren und in der Brille, wodurch das Bild eines Projektors nur auf jeweils ein Auge abgebildet wird. Die Polarisationsbrillen sind zwar deutlich günstiger und auch für große Projektionsräume geeignet, jedoch sind dann für jedes Display zwei teure Beamer nötig, wohingegen das aktive Stereo mit einem Beamer auskommt. In kleinen Projektionsaufbauten wird deshalb meistens das aktive Stereo verwendet, während in großen Räumen passives Stereo eingesetzt wird.

Abbildung 3.5: Die *CrystalEyes* Shutter-Brille mit Emitter.

Ein stereofähiger Monitor mit einer Bildwiederholungsrate von $>$100 Hz in Kombination mit einer Shutter-Brille für aktives Stereo ist die einfachste und günstigste Variante eines VR-Display-Systems. Die geringe Abdeckung des Blickfeldes brachte diesem Aufbau auch den Namen *Fish-Tank-VR* ein. Gut geeignet ist diese Variante für den Ingenieursarbeitsplatz, da sie keine größeren Erweiterungen zum Standardbildschirmarbeitsplatz bedeutet. Um den Immersionseffekt zu steigern, kann auch am Arbeitsplatz ein Trackingsystem zur Erfassung der Kopfposition eingesetzt werden.

Projektionsräume mit einer großen Rückprojektion (siehe Abbildung 3.6) wurden in den letz-

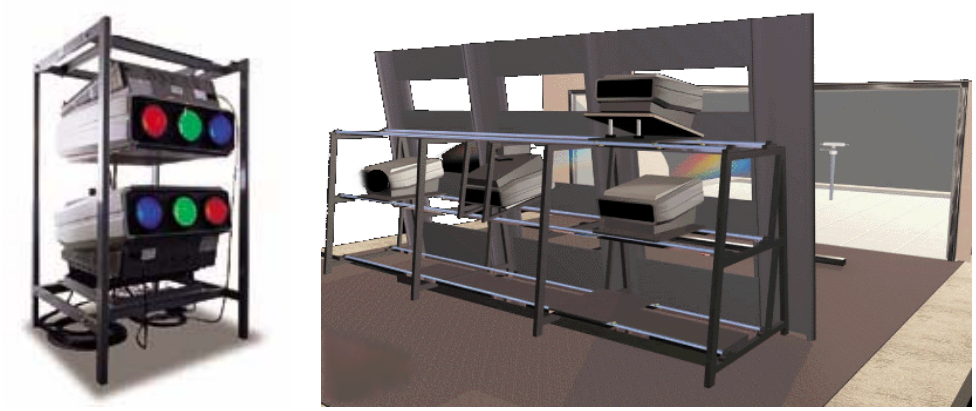

Abbildung 3.6: Die großflächige Leinwand einer *Powerwall* (rechts) wird von mehreren separaten Projektionseinheiten (links) beleuchtet.

ten 3 Jahren verstärkt in der Industrie und an Universitäten installiert. Von mehreren LCD- oder Röhrenbeamern wird eine milchige Plexiglasscheibe mit einer Mindestgröße von 2x2 Metern hochauflösend angestrahlt [102], wobei der optimale Aufbau einer solchen Konstruktion noch von vielen Seiten untersucht wird [54, 79]. An den Übergängen der Projektionsflächen verhindert das *Edgeblending* das Auftreten von harten Kanten und erzeugt so im Idealfall eine gleichmäßig beleuchtete Leinwand. Je nach Projektionsaufbau und Größe des Raumes kommt passives oder aktives Stereo zum Einsatz. Die Erfahrungen haben gezeigt, dass aktives Stereo nur in Räumen mit maximal 10 Personen sinnvoll genutzt wird. Entscheidend für die Größe des Raumes ist die Wahl der Projektoren, während LCD-Beamer durch eine hohe Leuchtkraft glänzen, bestechen Röhrenbeamer durch flexiblere Einstellungsmöglichkeiten der Projektion. Zu beachten ist, dass LCD-Beamer nur in passiven Stereoumgebungen eingesetzt werden können, aber aufgrund des niedrigen Preises immer attraktiver werden. In einigen Aufbauten wird auch der Einsatz von elektromagnetischen- und optischen Trackern erprobt, um beispielsweise das realitätsnahe Abschreiten eines virtuellen Fahrzeuges in Originalgröße zu ermöglichen. Klassische Einsatzgebiete für Projektionsräume sind die wissenschaftliche Visualisierung und das Automobildesign [132, 17].

Naiv betrachtet ist eine *Workbench* [38] ein großer Monitor bestehend aus einer horizontalen, vertikalen oder im Winkel variablen Rückprojektion (siehe Abbildung 3.7). Allerdings bietet sie durch ihre Größe von ca. 1.7 x 1.2 Metern und der räumlichen Nähe zum Betrachter eine engere Kopplung der dreidimensionalen Darstellung und einer zweidimensionalen Oberfläche. Die Erweiterung durch eine *Touchscreen*-Oberfläche eröffnet der Benutzerführung sowohl den immersiven Eindruck, als auch die Vorzüge eines zweidimensionalen GUIs und somit mehr Intuitivität als am Monitor oder an einer *Powerwall*. Interaktionsgeräte wie Datenhandschuhe, Tracking-System mit Pointer oder das *Phantom* verstärken zusätzlich den Grad der Immersion, da auch hier die reale Hand des Benutzers in der virtuellen Welt sichtbar ist. Dreidimensionales Sehen wird an einer *Workbench* im Allgemeinen mit einem aktiven Stereosystem in Kombination mit *Head-Tracking* erzeugt. Eine stärkere Immersion bietet eine zweiseitige Variante der *Workbench*,

Abbildung 3.7: Die Projektionsfläche einer Workbench kann horizontal, vertikal oder variabel sein. Die Holobench (rechts) besteht aus zwei Projektionsflächen.

da durch eine horizontale und vertikale Bildfläche dem Anwender das Gefühl vermittelt wird in die virtuellen Objekte hineinzugreifen. Die Anzahl der Teilnehmer in einem effektiven Arbeitstreffen an einer *Workbench* ist durch die Breite des Bildschirmes auf maximal 3 Personen beschränkt.

Der mehrseitige Projektionsraum *CAVE (CAVE Automatic Virtual Environment)*, siehe Abbildung 3.8, wie er von Cruz Neira [22, 24, 23] vorgestellt wurde, bietet zur Zeit den höchsten Grad an Immersion. Eine CAVE ist ein rechteckiger Raum mit einer aktiv-stereoskopischen Rückprojektion auf vier bis sechs Seiten und einem Tracking-System für Kopf und Hände des Benutzers. Auf den Leinwänden wird die virtuelle Welt in alle Blickrichtungen entsprechend dem Benutzerstandpunkt abgebildet, womit die Abbildungsfehler, bedingt durch die Latenzzeit des Tracking-Systems, vom Anwender nur vermindert wahrgenommen werden. Die Ausrichtung der perspektivischen Abbildung zum Standpunkt des Benutzers beschränkt die optimale Nutzung einer *CAVE* auf einen getrackten Teilnehmer. Jedoch können mit leichten Einschränkungen bis zu 4 Personen in einer *CAVE* arbeiten. Die Anschaffungskosten einer *CAVE* liegen im siebenstelligen DM-Bereich und auch die Unterhaltskosten sind enorm. Sie entstehen durch die großen Baumaßnahmen, die Projektionstechnik und den leistungsstarken *SGI*-Großrechner mit mindestens zwei Graphiksystemen. Der Raumverbrauch einer normalen vierseiten *CAVE* ist mit 6x4x3 Metern bereits zu hoch für den Bürobereich und für sechsseitige Aufbauten werden mit 6^3 Metern praktisch eigene Gebäude benötigt. Die VR-Software für eine *CAVE* muss Multi-Pipe und somit Multi-Thread fähig sein, damit mehrere Displays unterstützt werden. Nur speziell angepasste Software bietet diese Möglichkeiten, während Standardsoftware nicht in einer *CAVE* betrieben werden kann. Viele Institutionen schaffen sich trotz der deutlichen Nachteile eine *CAVE* an, da sie eine unerreichte Immersion bietet.

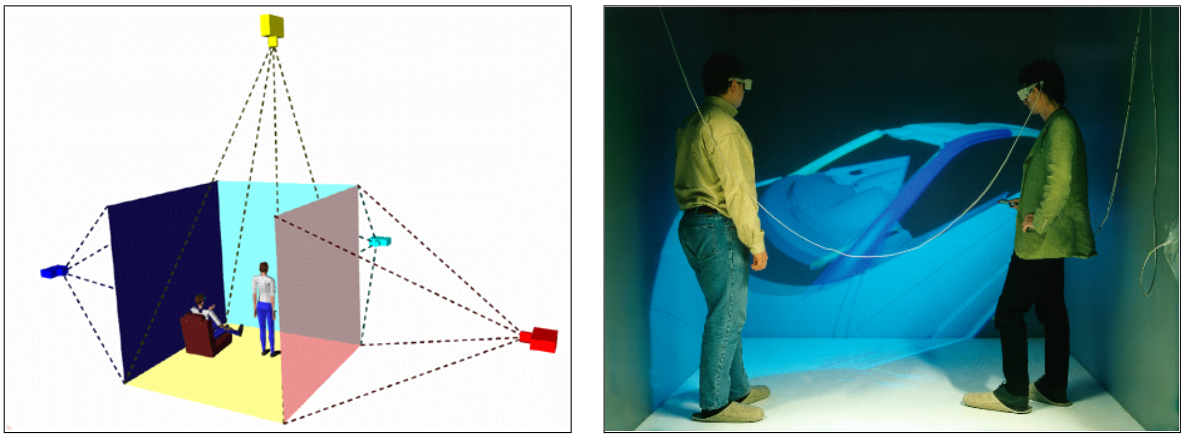

Abbildung 3.8: Der mehrseitige Projektionsraum in einer schematischen Darstellung und in Realität.

3.4.2 Tracking-Systeme

Die Übertragung einer Raumposition von der realen in die virtuelle Welt ist eine weitere Schlüsselfunktionalität der virtuellen Realität. Das so genannte *Tracking* besteht aus der Bestimmung von Position und Richtung realer Objekte an dezidierten Punkten durch den Computer. Komplexe Objekte, wie beispielsweise der komplette Körper eines Menschen, können mit dem heutigen Stand der Technik nur sehr aufwendig bestimmt werden. Für die Interaktion des Anwenders reicht daher das Tracking von Kopf und Händen aus. Anhand der Kopfposition wird die stereoskopische Projektion berechnet und die Manipulation von virtuellen Objekten erfolgt über die Hände.

Die momentan käuflichen Tracking-Systeme basieren im Wesentlichen auf fünf verschiedenen Ansätzen: den elektromagnetischen Feldern, dem Ultraschall, der Mechanik, der Beschleunigung und den Infrarot-Wellen.

Am weitesten verbreitet sind die auf elektromagnetischen Feldern basierenden Tracking-Systeme *Flock of Birds* von *Ascension* und das *FASTRAK*-System von *Polhemus*, siehe Abbildung 3.9. Der *Flock of Birds* kann mit Hilfe eines gepulsten Gleichstrom-Magnetfeldes gleichzeitig die Position von bis zu 30 Empfängern bei einer Abtastrate von 144 Hz und einer Auflösung von 0,8 mm für die Translation bzw. 0,1° für die Rotation bestimmen. Mit einer Genauigkeit von 2,5 mm und 0,5° besitzt ein einzelner *Extended Emitter* einen Operationsradius von 2,4 m, der sich optional durch das Kaskadieren von mehreren Sendern erweitern lässt. Der *FASTRAK* arbeitet mit einem magnetischen Wechselfeld und ist in der Auflösung(0,6 mm, 0,025°), der Genauigkeit (0,8 mm, 0,15°) und dem Aktionsradius (3 m) besser als der *Flock of Birds*. Die maximale Abtastrate des *FASTRAK* von 120 Hz muss jedoch durch die Anzahl der Empfänger geteilt werden. Der *Flock of Birds* ist daher besser für Applikationen mit vielen Abtastpunkten geeignet, während der *FASTRAK* genauere Ergebnisse liefert. Beide Geräte sind gegenüber magnetischen Strukturen (z.B. Stahlträger) und elektromagnetischen Störstrahlen (z.B. Monitore) anfällig, was zu einem Zittern der Positionsangabe führen kann, wobei das magnetischen Wechselfeld des *FASTRAK* empfindlicher ist. Entwickelte Eichungsmethoden, Filter- und Glättungsalgorithmen [32] führen zu einer Verbesserung der Messdaten.

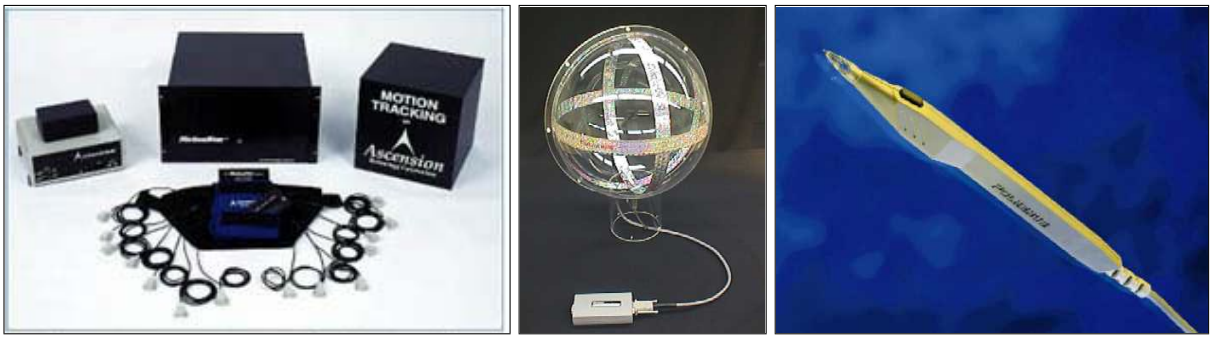

Abbildung 3.9: Die bekanntesten Tracking-Systeme: *Ascension Flock of Birds* (links), *Polhemus Long Range* Sender (Mitte), *Polhemus Stylus* Empfänger (rechts).

Ultraschall-Tracking-Systeme bestimmen die Lage und Orientierung eines Objektes durch die Messung der Laufzeitunterschiede kurzer Ultraschall-Impulse zwischen drei im Dreieck angeordneten Sendern zum Empfängerdreieck. Durch die geringe Triangulierungsbasis kann die Auflösung von 0,1 mm bzw. 0,1° nur im Abstand von höchstens 1,5 m innerhalb eines Kegels mit einem Öffnungswinkel von 100° bei 50 Hz Abtastrate erzielt werden. Neben der beschränkten Reichweite grenzt vor allem die Notwendigkeit einer direkten „Sichtverbindung“ zwischen Sender und Empfänger die Einsatzmöglichkeiten stark ein. So werden Ultraschall-Systeme hauptsächlich für Arbeitsplatzsysteme in Verbindung mit Shutter-Brillen und 3D-Mäusen oder zur Ergänzung anderer Messmethoden (z.B. elektromagnetischer Felder) eingesetzt.

Das mechanische Tracking findet seine Anwendung in kleineren Desktop-Interaktionsgeräten, wie beispielsweise im *Phantom*, und hauptsächlich im Zusammenhang mit dem *BOOM*. Mittels optischer Winkelaufnehmer lässt sich die Lage der Monitoreinheit auf bis zu 4 mm bzw. 0,1° in einem Radius von 1,8 m mit mindestens 70Hz bestimmen. Interessant für die virtuelle Realität ist das mechanische Tracking durch die Unempfindlichkeit gegenüber elektromagnetischen Einflüssen, wodurch eine präzise Positionsbestimmung im ganzen Aktionsradius sichergestellt ist. Ein weiterer Vorteil ist die mit 200 ns sehr geringe Latenzzeit gegenüber den 4000 ns der elektromagnetischen *Tracker*.

Wenig Verbreitung fanden bisher die gyrotechnischen *Tracker* der Firma *Intersense*, die erst seit kurzem auf dem Markt erhältlich sind. In diesen Systemen erkennt ein kleiner Kreisel die Beschleunigungs- und Scherbewegungen, aus denen in Kombination mit einem Ultraschall-System die Position und Orientierung bestimmt werden. Die Auflösung der Systeme ist mit 1,5 mm bzw. 0,05° und einer Genauigkeit von 4 mm bzw. 0,1° bei 180Hz Abtastrate ähnlich gut wie bei den elektromagnetischen *Trackern*. Ihr großer Vorteil ist die Unempfindlichkeit gegen elektromagnetische Einflüsse, jedoch sind die Systeme bisher vergleichsweise teuer und die Empfänger verhältnismäßig groß.

Der Empfänger der genannten Tracking-Systeme ist teilweise sehr groß und erfordert eine Kabelverbindung zum Rechner, wodurch der Immersionseffekt reduziert wird. Neue Tracking-Systeme setzen auf die Infrarot-Technik, bestehend aus Infrarotlicht-Sendern, Kameras und einer kleinen Reflektionsmarkierung am Benutzer. Die Infrarotstrahlen werden mit unterschiedlichem Winkel von mindestens zwei Kameras erfasst und erlauben so die Positionsbestimmung des Anwenders. Die Erfassung der Orientierung ist mit diesem System nicht möglich und auch die Verdeckung der Reflektionsmarke behindert das Tracking. Trotz dieser Einschränkungen kann das Infrarot-Tracking sehr gut für das Head-Tracking an *Powerwalls* eingesetzt werden, da hier die Blickrichtung des Anwenders nicht stark variiert. Die ungehinderte Bewegungsfreiheit ist gerade für *Powerwall*-Anwendungen von großem Vorteil.

Zielsetzung der Forschung ist daher momentan die Realisierung eines Kabel-unabhängigen Tracking-Systems. Untersucht werden unter anderem Ansätze mit bildbasierter Objekterkennung [78] oder optischen Systemen in der CAVE [113]. Die Marktreife solcher Systeme ist allerdings erst in einigen Jahren zu erwarten.

3.4.3 Interaktionssysteme

Mit Hilfe der Interaktionsgeräte kann der Anwender die Applikation steuern und Manipulationen auslösen. Das Spektrum der verfügbaren Geräte reicht von Desktop-Systemen über den klassischen Datenhandschuh bis hin zu haptischen *Force-Feedback*-Systemen. Anhand einiger Beispiele soll die große Vielfalt der am Markt käuflichen Geräte aufgezeigt werden.

Das klassische VR-Interaktionsgerät für den Arbeitsplatz ist die *DLR-Spacemouse*, siehe Abbildung 3.10. Der Benutzer kann zur Eingabe einen etwa handtellergroßen Knopf um ca. 1,5 mm in jede Richtung verschieben bzw. um ca. 4° verdrehen. Die Eingaben werden dann zu einer sechsdimensionalen relativen Transformation in der virtuellen Umgebung umgerechnet. Durch variable Empfindlichkeit und eine gute Dosierbarkeit der Aktionen ist ein genaues Arbeiten möglich.

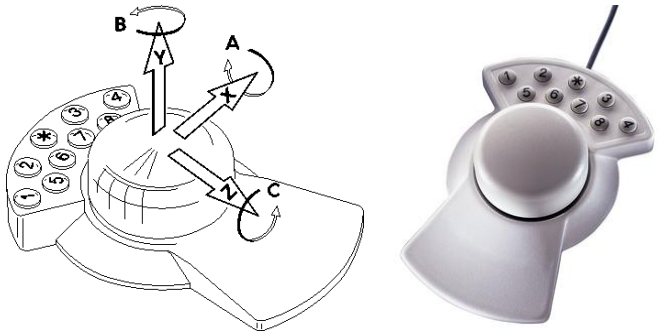

Abbildung 3.10: Die *DLR-Spacemouse* erlaubt Interaktionseingaben in allen sechs Freiheitsgraden.

Mit dem *Phantom*, siehe Abbildung 3.11, der Firma *SensAble* kann intuitiver gearbeitet werden als mit der *Spacemouse*. Der an einem mechanischen Galgen aufgehängte Zeiger kann vom Anwender beliebig positioniert werden und zusätzlich durch eine Kraftrückkopplung, dem so

Abbildung 3.11: Das *SensAble Phantom* gibt es in drei verschiedenen Größen.

genannten *Force Feedback*, einen haptischen Eindruck eines virtuellen Objekts vermitteln. Viele aktuelle Arbeiten beschäftigen sich mit Einsatzgebieten für dieses Gerät [84, 100, 21, 19], da es sich hervorragend für das Entwerfen und Bewerten von Oberflächen eignet.

In immersiven Umgebungen werden die Interaktionsgeräte zur Positionserkennung an ein Tracking-System gekoppelt. Das einfachste immersive Eingabegerät ist der virtuelle Zeigestab, bestehend aus einem Tracking-Empfänger und einigen Eingabeknöpfen, die als Auswahl-Auslöser fungieren. Die Positionierung des Zeigestabes kann durch die Probleme des elektromagnetischen Trackings sehr ungenau sein. Die *Cubic-Mouse* der *GMD* (siehe Abbildung 3.12) bietet die präzise Positionierung von Schnittebenen durch virtuelle Objekte in interaktiven, virtuellen Umgebungen [39], eignet sich allerdings nur für Anwendungen mit der Beschränkung auf Schnittebenen.

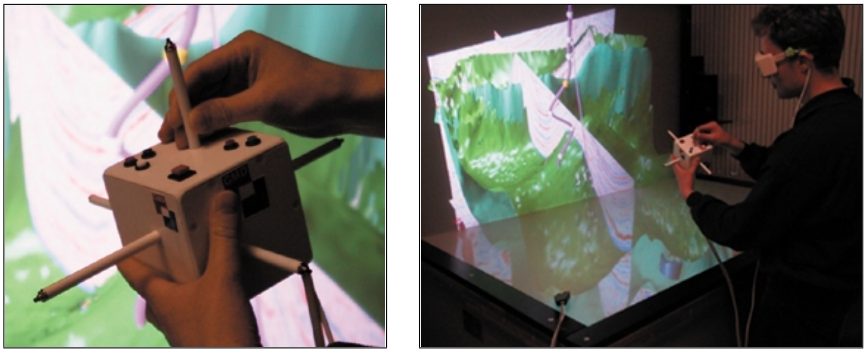

Abbildung 3.12: Die *Cubic Mouse* wurde entwickelt, um in immersiven Umgebungen Schnittebenen achsenparallel zu verschieben.

Das bekannteste immersive Interaktionssystem ist der Datenhandschuh, wie beispielsweise der *PinchGlove* von *Fakespace* oder der *CyberGlove* von *Virtual Technologies VTi*. Ein Datenhandschuh bestimmt mit Hilfe von Dehnungsstreifen die Stellung der Finger, anhand derer die Treibersoftware Gesten erkennt und somit Ereignisse auslöst. Die neuentwickelten Varianten von *VTi* besitzen durch Vibrationsspulen (*CyberTouch*), durch das Zurückziehen der Finger (*CyberGrasp*) oder durch die Kombination mit einem mechanischen Galgen (*CyberForce*) die Eigenschaften des *Force Feedbacks* (siehe Abbildung 3.13). Jedoch ist der haptische Effekt dieser Geräte noch nicht ausgereift und bedarf noch einiger Entwicklung.

So ist gegenwärtig die Entwicklung von neuartigen haptischen [121, 45] und taktilen [3, 61] Systemen Gegenstand der Forschung. Ein weiterer Entwicklungsschwerpunkt ist die Untersuchung von *Tretmühlen* die dem Anwender das Navigieren in der virtuellen Welt durch Laufen auf einem Laufband [64] ermöglichen. Erst wenn der Anwender vollständig in die virtuelle Welt eintauchen kann, ist das Ziel der Entwicklungen erreicht.

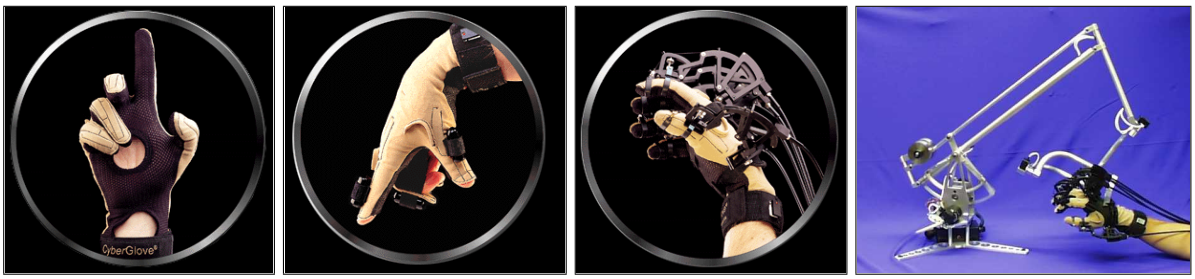

Abbildung 3.13: *Virtual Technologies* vertreibt verschiedene Varianten von Datenhandschuhen. Bis auf den CyberGlove weisen sie haptisches Feedback auf. Vlnr.: CyberGlove, CyberTouch, CyberGrasp, CyberForce.

3.5 VR-Software

Das Systemdesign einer VR-Applikation muss zum Erreichen einer interaktiven Umgebung in allen Teilbereichen auf die Anwendung abgestimmt werden [11]. Die Teilbereiche umfassen die effiziente Organisation der zu visualisierenden Daten, die Schnittstelle zur Graphik-Hardware, die Treiber-Software für die VR-Geräte und eine graphische 2D/3D Benutzerschnittstelle. Eine VR-Applikation kann entweder komplett neu entwickelt oder auf bereits existierende Teillösungen aufgesetzt werden. Im ersten Fall ist viel Entwicklungsarbeit für bereits gelöste Probleme zu leisten, die im zweiten Ansatz durch das Ausnutzen von vorhandenem Wissen entfällt. Die Verwaltung der zu visualisierenden Daten ist der Kernbereich der Applikation, für den es in den wenigsten Fällen standardisierte Lösungen gibt, wohingegen für die graphische Schnittstelle, die Treibersoftware und die GUI-Problematik vielversprechende Ansätze existieren.

In der industriellen und in der universitären Entwicklung befinden sich eine Reihe von VR-Software-Bibliotheken (*VR-API*) mit großen Funktionsumfängen. Sie verfügen über einen Szenengraphen als Schnittstelle zur Graphik-Hardware, einen häufig sehr umfangreichen Satz an Gerätetreibern und teilweise über spezielle 2D- und 3D-GUIs. Die Verwendung einer VR-API sichert meistens die Unterstützung verschiedener Plattformen und die zügige Einbindung von neuen *Graphik-Extensions* und neuer Hardware. Die Wahl der richtigen API hängt stark von den Anforderungen der Anwendung ab. Die wichtigsten Entscheidungskriterien sind:

- Flexibilität des Szenengraphen
- Datenmenge der Applikationen
- Rechnerplattform
- VR-Gerätetreiber

Die Flexibilität des Szenengraphen kann entscheidend sein, wenn die Anwendung den direkten Aufruf von OpenGL-Funktionsaufrufen, oder einen sehr schnellen und guten Zugriff auf

die Datenstrukturen erfordert. Auch muss die Datenverwaltung die Erfordernisse für die Datenmengen der Applikation erfüllen, da ansonsten die Ressourcen schnell aufgebraucht sind. Selbstverständlich muss der Szenengraph auf den vorgesehenen Plattformen verfügbar sein, wohingegen VR-Gerätetreiber der Hersteller notfalls auch nachträglich in die Szenengraphen eingefügt werden können.

Die wichtigste Entscheidung muss also im Bereich des Szenengraph-API getroffen werden und sie ist trotz der großen Anzahl an Produkten nicht einfach zu fällen.

3.5.1 Szenengraph-APIs

Eine Szenengraph-API ist kein vollständiges VR-System, sondern bildet die Rendering-Basis für VR-Applikationen. Die graphischen Objekte werden innerhalb von Szenengraphen in einer Baumstruktur verwaltet. Der Ursprung der virtuellen Welt entspricht darin dem Wurzelknoten, gefolgt von den Verzweigungen zur Zusammenfassung der graphischen Objekte und den Blattknoten mit den eigentlichen geometrischen Beschreibungen. Sowohl von den Verzweigungs- als auch den Blattknoten besitzen die Szenengraphen verschiedene Typen für unterschiedliche Aufgaben. So gibt es beispielsweise *Transformations-*, *Switch-* und *Level of Detail*-Knoten zur Gruppierung von Knoten. Die Geometrie-Knoten werden häufig noch in Teilknoten wie Koordinatenlisten, Indexfelder und Materialeigenschaften zur Beschreibung von Dreiecks- und Viereckesnetzen unterteilt. Die Betrachtung der Szenengraphen erfolgt durch Kameras, die schließlich die Bilder liefern.

Die Effizienz, mit der die Blattknoten die Geometrie speichern und auf Funktionen der Graphik-Hardware abbilden, entscheidet über den Speicherverbrauch und die Geschwindigkeit der Darstellung. Die Erfahrung zeigt, dass flexible und speicheroptimierte Szenengraphen eine oft schlechte Graphik-Performance bieten, wohingegen die auf Geschwindigkeit getrimmten meist Unmengen an Speicher benötigen. Ausnahmen gibt es natürlich im negativen wie auch positiven Sinne.

Die einzelnen Szenengraphen unterscheiden sich zusätzlich im Umfang der Ausstattung. Hierzu gehören die bereits angesprochenen Gerätetreiber, aber auch Optimierungsfunktionen, wie beispielsweise für beschleunigtes Rendering (z.B. *Occlusion-Culling*) oder zur Polygonreduktion. Zu einigen Szenengraphen existieren bereits graphische Benutzerschnittstellen zur interaktiven Erstellung von virtuellen Welten. Ein Beispiel hierfür ist das Werkzeug *dVISE* zum Aufbau eines *dVS*-Szenengraphen.

1990 brachte *Sense8* mit *WorldToolKit* eine der ersten kommerziellen Szenengraph-APIs auf den Markt. Eine Besonderheit ist die umfangreiche Treiber-Bibliothek, mit der fast jede VR-Komponente angesprochen werden kann und die mittlerweile auch über eine *CAVE*-Unterstützung verfügt. Leider ist der Szenengraph sehr stark gekapselt und verhindert so die schnelle Manipulation von graphischen Objekten, auch lassen sich erst seit dem Release 7 OpenGL-Funktionen über spezielle Knoten direkt aufrufen. Seit dem Release 5 verfügt *WorldToolKit* mit dem Werkzeug *WorldUp* über eine graphische Benutzerschnittstelle zur interaktiven Erstellung von virtuellen Welten. Der einfache Aufbau von *WorldToolKit* ist sehr gut für das *Rapid Prototyping* geeignet und führt schnell zu ersten Ergebnissen, jedoch stößt man aufgrund des hohen Speicherverbrauches schnell an die Grenzen der Datenstrukturen.

Direkter Konkurrent von *WorldToolKit* ist der von *dVision* entwickelte Szenengraph *dVS* mit dem dazugehörigen Szenen-Editor *dVISE*. *dVS/dVISE* verfügen über einen ähnlichen Funktionsumfang inklusive einem dreidimensionalen graphischen Menü, jedoch ist die Gerätetreiber-Bibliothek nicht ganz so umfangreich wie bei *WorldToolKit*.

Neben den auf mehreren Plattformen verfügbaren rein kommerziellen Szenengraphen gibt es auch spezielle Lösungen, die von Hardwareherstellern für ihre Systeme entwickelt wurden und kostenlos verfügbar sind. *SGIs OpenInventor* ist der bekannteste und in vielen Bereichen erfolgreichste Szenengraph dieser Art. Anfangs war der *OpenInventor*, alte Bezeichnung *Inventor*, nur auf *IRIX*-Plattformen verfügbar, wurde aber mittlerweile zu einem OpenSource-Projekt und anschließend auf die wichtigsten Plattformen *Linux, HP-UX* und *Windows* portiert. Der *OpenInventor*-Szenengraph erlaubt einen umfangreichen Zugriff auf die gespeicherten Geometrie-Objekte und ermöglicht somit sehr leicht Manipulationen. Der Preis für die Flexibilität ist eine vergleichsweise schlechte Performanz in der Darstellung und die horizontale Vererbungsreihenfolge von Knoteneigenschaften verhindert eine parallele Abarbeitung. Das Fehlen von umfangreichen Gerätetreibern ist für VR-Applikationen zusätzlich ein Problem. Viele in der darauf folgenden Zeit entwickelten Szenengraphen orientierten sich am Konzept und der Struktur des *OpenInventor*.

Speziell für hochperformante Applikationen entwickelte *SGI* den *IRIS Performer*. Die Spezialität des *Performer* ist die Integration von Multi-Processing in die Szenengraph-Struktur. Multi-Pipe-Systeme können so mit einer hohen Geschwindigkeit bedient werden. Nachteilig wirkt sich der komplizierte Zugriff und die schlechte Handhabung der Geometrieobjekte aus. Auch der *Performer* war anfangs nur auf *IRIX* verfügbar und wurde erst durch die neue Strategieausrichtung von *SGI* auf *Linux* portiert. Viele VR-Applikationen und VR-Systeme basieren auf dem *IRIS Performer*.

Im Laufe der Zeit erkannte *SGI*, dass eine Szenengraph-API nur als Standard anerkannt werden kann, wenn sie über die Flexibilität von *OpenInventor* und die Performanz von *Performer* verfügt. Aus den Entwicklungen in diese Richtung entstand der *Cosmo3D* Szenengraph und die darauf basierende Funktionsbibliothek *Inspector*, später in *OpenGL Optimizer* umbenannt. *Cosmo3D/Optimizer* besitzt einen flexiblen Szenengraphen mit der Möglichkeit von direkten OpenGL-Funktionsaufrufen, unterstützt Multi-Threading und bietet eine hohe Performanz. *SGI* verfolgte zu dieser Zeit die Politik den neuentwickelten Szenengraphen, ähnlich wie *OpenGL*, als Standard auf allen Plattformen durchzusetzen. Um dieses Ziel zu erreichen, wurde mit anderen Soft- und Hardwareherstellern zusammengearbeitet.

Ein erster Versuch wurde durch die Zusammenarbeit mit *SUN* gestartet. Das Projekt scheiterte an der Anforderung, sowohl *Java* als auch *C++* zu unterstützen. *SUN* führt diese Arbeiten als *Java3D* fort. Das folgende Projekt mit *HP* sollte als *OpenGL++* die Vorteile der *HP*-Szenengraph-API *DirectModel* und *Cosmo3D/Optimizer* vereinigen. Allerdings ließen Unstimmigkeiten zwischen den Partnern auch diese Zusammenarbeit scheitern. Als letzter Versuch wurde von *SGI* eine Zusammenarbeit mit *Microsoft* unter dem Codenamen *Fahrenheit* angestrebt. *Fahrenheit* sollte zu großen Teilen aus *Cosmo3D/Optimizer* bestehen und mit einer neuen *Low Level API* auf *Windows* verfügbar sein. Die strikte Weigerung von *Microsoft Fahrenheit* auch für *Linux* anzubieten, beendete das Projekt. *Microsoft* entwickelt an *Fahrenheit* weiter und *SGI* besaß zu diesem Zeitpunkt nicht mehr die Kapazität zur Weiterentwicklung von *Cosmo3D/Optimizer*.

Keine der genannte Szenengraph-APIs hat bisher den Durchbruch als Standard auf allen Plattformen geschafft. Vielmehr wird auf die bewährten Bibliotheken *OpenInventor*, *Performer* und *Cosmo3D/Optimizer* zurückgegriffen.

Das Fehlen eines Standards lässt viele OpenSource Projekte aus dem Boden sprießen. Bekannte Vertreter dieser Gattung sind *OpenScenegraph*, *OpenRM* und *OpenSG*. Die ersten beiden Projekte sind teilweise kommerziell, während *OpenSG* in einem öffentlichen Forum unter Leitung des *Fraunhofer Institut für Graphische Datenverarbeitung* entwickelt wird. Leider sind auch diese Projekte nicht erfolgreicher als die industriellen. Eine leistungsstarke, freiverfügbare und plattformunabhängige Szenengraph-API wäre eine große Bereicherung für die Computergraphik und speziell die virtuelle Realität.

3.5.2 VR-Visualisierungssysteme

Aufbauend auf den Szenengraph-APIs entstanden eine Vielzahl an universitären und industriellen VR-Visualisierungssystemen. Einige Systeme sind auf eingegrenzte Problemstellungen beschränkt, während andere größtmögliche Flexibilität bieten möchten. Die folgende Liste stellt einen kleinen Teil der weltweit verfügbaren Software zur Visualisierung von Berechnungsergebnissen vor.

Vom *Fraunhofer Institut für Arbeitswirtschaft und Organisation* wird das modulare Entwicklungssystem *Lightning* [7, 8] zur Erstellung von interaktiven High-End-Virtual-Reality-Anwendungen entwickelt und kommerziell vertrieben. Die Module von Lightning beschreiben die virtuelle Umgebung und deren Funktionen für 3D-Modelle, Ein- und Ausgabegeräte, Funktionsblöcke und Kommunikationsmechanismen. Der Anwendungsprogrammierer kann über eine Tcl/Tk Schnittstelle Tcl oder C/C++ Module sogar zur Laufzeit hinzufügen. Dies erlaubt die Implementierung von anwendungsspezifischen Erweiterungen, wie sie beispielsweise für die Sparte Presswerk der *BMW Group* entwickelt wurden. In dem Beispiel können im *Inventor*-Format gespeicherte Simulationsergebnisse in eine virtuelle Umgebung geladen und analysiert werden. *Lightning* basiert auf der Szenengraph-API *Performer* und ist auf *IRIX*-Systemen verfügbar.

Das objektorientierte Framework zur Entwicklung von verteilten, interaktiven VE Applikationen *Avango* entstand bei der *GMD/IMK* [125]. *Avango* bietet dem Anwendungsprogrammierer das Konzept eines Szenengraphen für verteiltes Arbeiten mit mehreren Prozessen innerhalb einer Anwendung. Jeder Prozess besitzt eine lokale, synchronisierte Kopie des Szenengraphen inklusive Zustandsinformationen. Basierend auf dem *IRIS Performer* zielt *Avango* auf das Anwendungsgebiet von SGI High-End- und Echtzeitumgebungen, wie beispielsweise die *CAVE* und *Workbenches* ab.

Ziel des *SIM-VR*-Projektes [124] war die Untersuchung des Einsatzes von virtueller Realität für die Visualisierung von Crashtest-Berechnungsergebnissen. Partner des Konsortiums waren die *BMW Group* auf der Anwenderseite, *ESI* als Entwickler von numerischen Simulationen, *TAN* als Hersteller von Projektionstechniken und die *GMD* als Forschungsinstitut. Für die Tests wurde eine spezielle Version des Crash-Solvers *PAM-CRASH* auf einem leistungsstarken Rechner installiert, der ein moderates Modell innerhalb von 30 Minuten berechnen konnte. Die Berechnungsergebnisse wurden über eine *CORBA*-Anbindung direkt auf einer *Responsive Workbench* in einer virtuellen Umgebung dargestellt. Dieses Projekt hat die Machbarkeit von *Interactive*

Steering für Finite-Element-Simulationen nachgewiesen.

Ein weiteres Projekt der *GMD* in Zusammenarbeit mit der *DaimlerChrysler AG* war *FLUVIS*, dessen Zielsetzung die dreidimensionale, interaktive Visualisierung von Strömungssimulationsergebnissen in einer virtuellen Umgebung war. Die Visualisierung fand auf einer *Responsive Workbench* oder dem *CyberStage* in Verbindung mit 3D-Interaktionsgeräten statt. Als Graphik-Hardware kam eine *SGI ONYX* zum Einsatz mit einer direkten Kopplung an eine IBM SP2 via HIPPI zur Berechnung von Schnittebenen, Partikelbahnen und Isoflächen. Die Leistung der Graphikrechner ist mittlerweile stark angestiegen, so dass der Ansatz der getrennten Berechnung von Geometrie-Elementen auf einem separaten Compute-Server nicht mehr zeitgemäß ist.

Der Demonstrator *VRGeo* [37] entstand in einem Projekt zur Erforschung des Einsatzes von virtueller Realität für die Unterstützung bei der Suche nach neuen Rohstoffvorkommen. Entwickelt wurde *VRGeo* in einem internationalen Konsortium, bestehend aus den beiden Software-Entwicklern *VETL* der Universität Houston und der *GMD* und den Partnern aus der Öl- und Gasindustrie Arco, Amoco, BHP, EXXON, Landmark, Mobil, Saga, Schlumberger, Shell, Smedvig und Statoil. Die virtuelle Umgebung wurde speziell für die Belange einer High Performance Analyse von geophysikalischen Daten ausgelegt. Das Projektziel war die Generierung eines einheitlichen Blickes über verschiedene Datenquellen zur Unterstützung von multidisziplinären Teams in verteilten Anwendungen.

MuSE Software Development Environment 2000 ist ein kommerzielles Visualisierungssystem von *Muse Technologies* zur Erstellung von Multi-User-, Multi-Plattform-, Multi-Sensor-Applikationen. Eine Sammlung von brauchbaren C und C++ Funktionen in Kombination mit verfügbaren Werkzeugen ermöglichen die Erstellung von kundenorientierten Applikationen. Es werden sowohl *High-End*- als auch *Low-Level*- VR-Umgebungen mit einer Vielzahl an Gerätetreibern unterstützt. Der Kunde kann so den Grad an Immersion den Anforderungen entsprechend wählen. Für reine Endbenutzer verfügt *MuSE* über ein umfangreiches Funktionsangebot für schnelle Standardvisualisierungen. *MuSE* unterstützt sowohl UNIX-, als auch WindowsNT-Systeme in kollaborativen Applikationen.

Das modulare Visualisierungssystem *COVISE* [137, 136] entstand am Rechenzentrum der Universität Stuttgart und wird von *Vircinity IT-Consulting* als kommerzielles Produkt vertrieben. Der modulare Aufbau von *COVISE* erlaubt durch die Entwicklung und das Hinzufügen von speziellen Modulen die Anpassung an beliebige Problemstellungen. Die einzelnen Module besitzen die unterschiedlichsten Funktionen, wie beispielsweise das Laden und Bearbeiten von Daten, oder auch das Betrachten von Geometrie in einer VR-Umgebung. Jedes Modul ist ein eigenständiger Prozess und wird über seine Ein- und Ausgabeschnittstellen mit anderen Modulen zu einer *Map* verbunden. Eine *Map* beschreibt die Aufgabenverteilung und den Datenfluss zwischen den verschieden Modulen, die auf verschiedene Rechner verteilt sein können. Schwerpunktgebiete sind die Visualisierung von Crashtest- und Strömungssimulationen. In verschiedenen Projekten wurde in *COVISE* das *Interactive Steering* und das collaborative Arbeiten aus VR-Umgebungen heraus integriert. *COVISE* besitzt sowohl auf *Inventor* als auch auf *Performer* basierende Betrachtungsmodule. Der modulare Ansatz hat allerdings die Nachteile eines hohen Kommunikationsaufkommens zwischen den Modulen und das Fehlen einer einheitlichen graphischen Benutzerschnittstelle.

3.6 Einsatzgebiete

Virtuelle Realität wird breitgefächert in vielen Disziplinen untersucht und eingesetzt [10]. Beschränkten sich die ersten Anwendungen auf die Betrachtung von dreidimensionaler statischer Geometrie, so gibt es mittlerweile vielfältige Interaktionsmechanismen mit denen in verschiedensten Anwendungen gearbeitet wird. Es entstanden sowohl wissenschaftliche als auch kommerzielle Applikationen, die auf den Techniken der virtuellen Realität aufbauen. Im Folgenden werden Beispiele von VR-Applikationen verschiedenster Einsatzgebiete vorgestellt.

Die ersten Anwendungen fand die virtuelle Realität in so genannten *Walk*- und *Flythrough* Umgebungen, wie sie in Architekturvisualisierungen, Fahr- und Flugsimulatoren eingesetzt werden. Zu Beginn wurden lediglich einfache Polygonmodelle dargestellt, um möglichst hohe Bildwiederholungsraten zu erzielen. Neue Techniken der Computergraphik, wie beispielsweise *image based rendering* [93] für Modelldetails und voxel-basierte Terrain-Modelle [130], ermöglichen mittlerweile die Darstellung von detaillierteren Modellen bei hoher Interaktivität.

In der Lehre wird die virtuelle Realität zur Unterstützung des klassischen Schulunterrichts sowohl für die Unterrichtsform [66] als auch für Unterrichtsinhalte [1] erprobt. Ebenfalls können extrem gefährliche Situationen von Feuerwehreinsätzen [122] und medizinischen Notfällen [115] in virtuellen Umgebungen kostengünstig für das Training simuliert werden.

Die interaktive und realitätsnahe Darstellung von dreidimensionalen Objekten ist auch hervorragend für das Produktmarketing geeignet. In vollimmersiven Umgebungen oder in Kombination mit dem Internet wird die virtuelle Realität für den Verkauf von beispielsweise Möbeln [31] oder Autos [27] untersucht. Ziel dieser Applikationen ist es, dem Kunden jede erdenkbare Variante des Produktes virtuell zu präsentieren.

Auch für militärische Zwecke wird die virtuelle Realität eingesetzt. Die Unterstützung der Ausbildung von Soldaten durch die kostengünstige Simulation von gefährlichsten Situationen in Fahr- und Flugsimulationen ist nur ein Teilbereich. Mittlerweile wird der Einsatz von VR ebenfalls für die Visualisierung von Gefechtsschauplätzen [67, 56] und das Training von Krankenpersonal nach atomaren Einsätzen [115] untersucht.

Ein friedlicheres Gebiet für die virtuelle Realität ist die wissenschaftliche Visualisierung, in der viele namhafte Forscher [49, 35] das größte Einsatzpotenzial sehen. Gerade immersive Hardware-Umgebungen sind hierfür geeignet und auch erwünscht [12, 128]. Als ein besonders erfolgreiches Projekt ist in diesem Zusammenhang der virtuelle Windtunnel [13] zu nennen. Mit ihm wurden zu einem sehr frühen Zeitpunkt die Vorteile der VR für die Visualisierung von Strömungssimulationen demonstriert.

Zum wissenschaftlichen Bereich der VR werden auch die Aktivitäten in der Medizin gezählt. Tumore werden beispielsweise aus CT- oder MR-Aufnahmen in virtuellen Umgebungen visualisiert [30, 53] und die anschließende Operation vorbereitet [48]. Ein anderes medizinisches Einsatzgebiet ist die Bekämpfung von psychischen Störungen. Durch die virtuelle Simulation der Problemsituationen [57] soll der Patient seine Ängste in der realen Welt bekämpfen.

Hinter den VR-Aktivitäten der großen Ölkonzerne stehen sowohl wissenschaftliche als auch kommerzielle Interessen. Die Auswertung von Bohrungsergebnissen zur Lokalisierung von neuen Erdölvorkommen ist problematisch, da die großen Ergebnisdaten eine möglichst verständliche Darstellung für die Ingenieure erfordern. Ein Weg ist die interaktive Visualisierung von

Volumendaten in virtuellen Umgebungen [37, 80].

Im Automobilbau werden seit geraumer Zeit Einsatzszenarien für die virtuelle Realität im produktiven Umfeld [97, 101] erprobt. In vielen Bereichen des Fahrzeugentwicklungsprozesses setzt man bereits auf die Vorteile dieser Technik. In der Phase der virtuellen Prototypen wird die VR zur Visualisierung und Begutachtung der digitalen Modelle eingesetzt [15, 26, 25]. Neben der reinen Modellbetrachtung werden auch Einbauvorgänge [65] und Ergonomieuntersuchungen [29] an den virtuellen Daten vorgenommen, um so in einem frühen Stadium Fehler zu vermeiden. Dieselbe Zielsetzung hat auch die Visualisierung von Finite-Element-Berechnungen. Dem Problem der großen Anzahl an Geometrieelementen wurde anfangs mit der Erstellung von Video-Sequenzen [72] begegnet. Die fehlende Interaktivität schränkt dieses Verfahren jedoch stark ein und so wurde für kleinere Modelle auch die interaktive Visualisierung von Finiten-Elementen untersucht [42, 138, 139]. Fernziel in diesem Bereich ist die Kopplung von Simulation und interaktiver Visualisierung, das *Interactive Steering*, so dass die numerische Berechnung über eine graphische Schnittstelle vom Benutzer gesteuert und beeinflusst werden kann [68].

Die Nutzung des Internet spielt für die virtuelle Realität eine stärker werdende Rolle. Ziel ist es, Anwender über das Internet in einer virtuellen Umgebung zusammenzubringen, wie es bereits in vielen Multi-User-VR-Applikationen untersucht wurde [4, 69, 131, 116, 40, 89]. Eine interessante Vorgehensweise ist die Verwendung der *Virtual Reality Modelling Language* [77, 52], da sie als standardisiertes Dateiformat in Kombination mit Internet-Browser-PlugIns die schnelle Verbreitung von virtuellen Welten über das Internet erlaubt. Die Kopplung von *VRML*-Viewern ermöglicht kooperatives Arbeiten [112, 81] und besitzt zusätzlich den Vorteil der Plattformunabhängigkeit.

3.7 Einordnung der Arbeit in das VR-Umfeld

Im Rahmen dieser Arbeit sollen die Möglichkeiten zum Einsatz der virtuellen Realität im Fahrzeugentwicklungsprozess untersucht werden. Die entwickelten prototypischen Applikationen setzen auf Szenengraphen auf und unterstützen verschiedene VR-Hardware-Umgebungen, die anschließend auf ihre Tauglichkeit im produktiven Einsatz erprobt werden.

Die verwendete Arbeitsplatzumgebung setzt sich aus dem als *Fish-Tank VR* bezeichneten Aufbau bestehend aus Stereo-Shutter-Brille und *DLR-Spacemouse* zusammen. Ebenfalls untersucht wurden Projektionsräume mit bis zu drei Projektionseinheiten und entsprechend vielen Graphiksystemen, jedoch kommt hier aufgrund der fehlenden Verfügbarkeit kein Tracking-System zum Einsatz. Der *BOOM* und die *CAVE* wurden als Vertreter der immersiven Umgebungen inklusive Tracking-System, Button-Eingabegerät und Datenhandschuh betrachtet.

Erfahrungen in Bezug auf die Leistungsfähigkeit und die Flexibilität von Szenengraphen wurden in dieser Arbeit durch deren praktischen Einsatz gesammelt. Der *WorldToolKit*-Szenengraph wurde aufgrund der großen Auswahl an VR-Gerätetreibern für das erste Visualisierungssystem *VtCrash* der geometriebezogenen numerischen Simulationen ausgewählt. Das zweite Projekt *PowerVIZ* benötigt eine höhere Flexibilität beim Zugriff auf die Geometrieobjekte und so fiel hier die Wahl auf *Cosmo3D/OpenGL Optimizer*.

Vergleichbare Projekte werden häufig auf selbstentwickelte Szenengraphen aufgesetzt, dieser

Weg wurde hier bewusst gemieden, um den hohen Entwicklungsaufwand zu sparen. Es konnte so auf die implizit vorhandene Erfahrung der Szenengraph-APIs zurückgegriffen und schneller die angestrebten Ziele erreicht werden.

Einige der in 3.6 vorgestellten Arbeiten, wie beispielsweise *SIM-VR*, sind in demselben Arbeitsgebiet angesiedelt. Im Gegensatz zu der vorliegenden Arbeit liegen hier die Schwerpunkte mehr auf der Erprobung und Entwicklung neuer VR-Geräte und der Ankopplung an laufende Simulationsprozesse. Die Echtzeitdarstellung dieser Projekte beschränkt sich fast immer auf das reine Betrachten der virtuellen Szene mit minimalen Interaktionsmechanismen. Zielsetzung dieser Arbeit ist die Bereitstellung vielfältiger Interaktionsmechanismen zur verbesserten Analyse von Berechungsergebnissen und der Umsetzung der Techniken in den produktiven Ablauf.

Kapitel 4

Visualisierungsszenarien im Fahrzeugentwicklungsprozess

Der traditionelle Fahrzeugentwicklungsprozess setzt auf den Einsatz von realen Prototypen, deren zeitaufwendige und kostspielige Erstellung die Effizienz in der Entwicklung senkt. Die Neu- und Weiterentwicklungen im Bereich des Hochleistungsrechnens und die Beschleunigung von numerischen Algorithmen ermöglichen die digitale Produktentwicklung für die Automobilindustrie. Die numerische Simulation kann durch die schnelle Modellerstellung bereits in der frühen Phase und in vielen Bereichen des Entwicklungsprozesses angewandt werden.

4.1 Der Fahrzeugentwicklungsprozess

Die Optimierungszyklen des iterativen Prozesses der Fahrzeugentwicklung haben die Einhaltung der Vorgaben an Qualität und Kosten zum Ziel. Im traditionellen Entwicklungszyklus (siehe Abbildung 4.1) wird auf den Einsatz von digitaler Technik zur numerischen Simulation vollständig verzichtet.

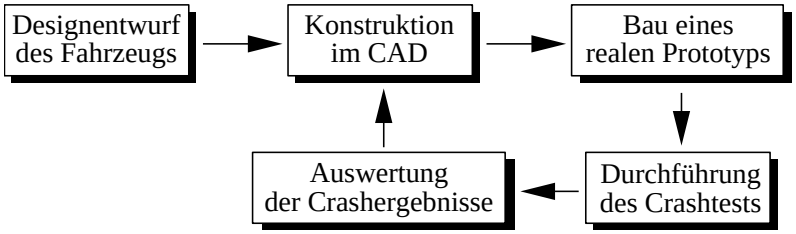

Abbildung 4.1: Skizze des traditionellen Entwicklungsprozesses.

Das Fahrzeug wird ausgehend vom Designentwurf mit CAD-Systemen konstruiert und in den anschließenden Untersuchungen anhand von real erstellten Teilmodellen und vollständigen Prototypen auf Problem- und Schwachstellen untersucht. Die Verarbeitungsqualität und Passgenauigkeit sind ebenso Gegenstand der Untersuchungen, wie Funktionalität und Handhabbarkeit des

Fahrzeuges im Fahrbetrieb. Schließlich wird mit einigen Prototypen in Crashtestuntersuchungen die Karosseriestabilität und die passiven Sicherheitssysteme überprüft. Erkannte Schwachstellen werden durch Konstruktionsänderungen behoben und der neue Konstruktionsstand bildet den Ausgangspunkt für den nächsten Zyklus, bis die Vorgaben schließlich erfüllt werden.

Kernstück des traditionellen Entwicklungszyklus ist das Evaluieren der Konstruktion an realen Prototypen, deren Erstellung auch heute noch überwiegend Handarbeit und somit zeit- und kostenintensiv ist. Ein Zyklus kann daher mehrere Monate in Anspruch nehmen. Parallel zur Konstruktionsevaluierung wird an dem Fahrzeugprojekt weitergearbeitet und die erhaltenen Versuchserkenntnisse können daher nicht mehr mit dem aktuellen Konstruktionsstand übereinstimmen. Die Verlagerung der Konstruktionsänderungen in eine späte Phase des Entwicklungsprozesses lässt die Kosten zusätzlich steigen, da im Verlauf der Entwicklung Konstruktionsänderungen mehr Seiteneffekte und somit auch höhere Kosten erzeugen.

Ein Ansatz zur Problemlösung ist die frühzeitige und parallel zur Entwicklung verlaufende Evaluierung der Konstruktion, das *information Front Loading*. Diese Vorgehensweise kann jedoch nicht durch den Bau von realen Prototypen erzielt werden, sondern erfordert neue Methoden zur Konstruktionsanalyse und Bewertung. Die Entwicklungen von neuer Hard- und Software im Bereich des Hochleistungsrechnens haben den Einsatz von Finite-Element-Simulationen für den produktiven Einsatz ermöglicht und sollen zur Einsparung von realen Prototypen führen.

Im digitalen Entwicklungszyklus wird der reale Prototyp durch ein digitales Fahrzeugmodell abgelöst und der reale Versuch durch die numerische Simulation ersetzt (siehe Abbildung 4.2).

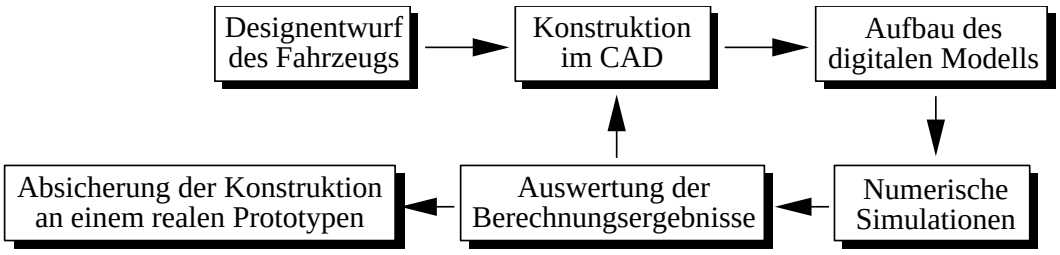

Abbildung 4.2: Skizze des digital-unterstützten Entwicklungsprozesses.

Ein digitaler Entwicklungszyklus ist in wenigen Wochen vollständig durchführbar. Die Berechnung von Varianten mit nur kleinen Änderungen kann in noch kürzeren Zeiträumen erfolgen, da das bestehenden digitale Modell lediglich geändert wird und bereits einige Simulationssysteme in der Lage sind, Teilergebnisse vorheriger Berechnungen wiederzuverwenden. Auch für kleine Konstruktionsänderungen können numerische Simulationen zur Absicherung eingesetzt werden und führen so zu einem besseren Produkt.

Beide Varianten des Entwicklungszyklus erfordern grundsätzliche Investitionen. Für den traditionellen Zyklus werden Versuchsstände und Werkstätten benötigt und der digitale Zyklus erfordert die Beschaffung von Hochleistungsrechnern und Softwarelizenzen. Wie aus Abbildung 4.2 hervorgeht, muss das Ergebnis des digitalen Prozesses durch einen realen Versuch bestätigt werden. Durch diese Abhängigkeit kann der digitale Versuch den realen Test niemals vollständig

ersetzen, jedoch erlauben die numerischen Simulationen eine Reduktion der zu erstellenden Prototypen bzw. der Kosten auf ein Minimum und rechtfertigen so die doppelte Investition.

Die vergleichsweise geringen Kosten einer numerischen Simulation gegenüber einem realen Versuchslauf sind deren großer Vorteil. Während beispielsweise die Kosten für einen realen Crashtestversuch mit einem Prototypen durchaus siebenstellig sein können, liegen die Kosten für eine numerische Berechnung maximal im fünfstelligen DM-Bereich.

Der größte Anteil der Kosten eines Simulationslaufs resultiert aus dem Arbeitsaufwand der Ingenieure für die drei Arbeitsschritte Preprocessing, Postprocessing und der Simulationsdurchführung. Beobachtungen bei *BMW* haben gezeigt, dass durch die steigende Komplexität der Finite-Element-Modelle eine Aufwandsverschiebung vom Pre- zum Postprocessing stattgefunden hat (siehe Tabelle 4.1). Die Verbesserung der Simulationsalgorithmen und die schnellere Hardware haben zu einer prozentualen Reduktion der eigentlichen Berechnungszeit geführt. Im Bereich des Preprocessings führten bessere Diskretisierungsalgorithmen und verbesserte Modellbeschreibungen ebenfalls zu einer Beschleunigung. Die zeitaufwendigste Aufgabe ist somit das Auffinden von Schwachstellen des Fahrzeugmodells im Postprocessing-Schritt. Dieser Schritt lässt sich praktisch nicht automatisieren und kann nur durch eine verbesserte Visualisierung der Berechnungsergebnisse zu Zeiteinsparungen führen.

Jahr	Preprocessing	Simulationsberechnung	Postprocessing
1991	50%	30%	20%
1996	30%	10%	60%

Tabelle 4.1: Prozentuale Verteilung der Ingenieursarbeitszeit auf die einzelnen Schritte eines Simulationslaufs.

Der Postprocessing-Vorgang ist in mehrere Teilschritte untergliedert (siehe Abbildung 4.3). Im ersten Schritt nach der Simulation verschafft sich der Ingenieur einen ersten Eindruck der Berechnungsergebnisse am Gesamtfahrzeug und versucht in den folgenden Detailbetrachtungen an Komponenten die Schwachstellen der Konstruktion zu finden. Im Erfahrungsaustausch zwischen dem Berechnungsingenieur und den verantwortlichen Konstrukteuren werden Änderungsvorschläge zur Behebung von Schwachstellen erarbeitet und diskutiert. In größeren Besprechungsrunden wird über die Umsetzung von umfangreichen und kostspieligen Änderungen entschieden. Die Dokumentation der Änderungsmaßnahmen findet parallel zu den einzelnen Postprocessing-Schritten statt.

Diese Arbeit zeigt durch die Bereitstellung von virtuellen, interaktiven Arbeitsumgebungen für jeden Analyseteilschritt potenzielle Zeit- und Kostenersparnisse im Postprocessing auf. Eingesetzt werden hierfür moderne, auf die Problemstellung adaptierte Visualisierungsalgorithmen und Techniken der virtuellen Realität.

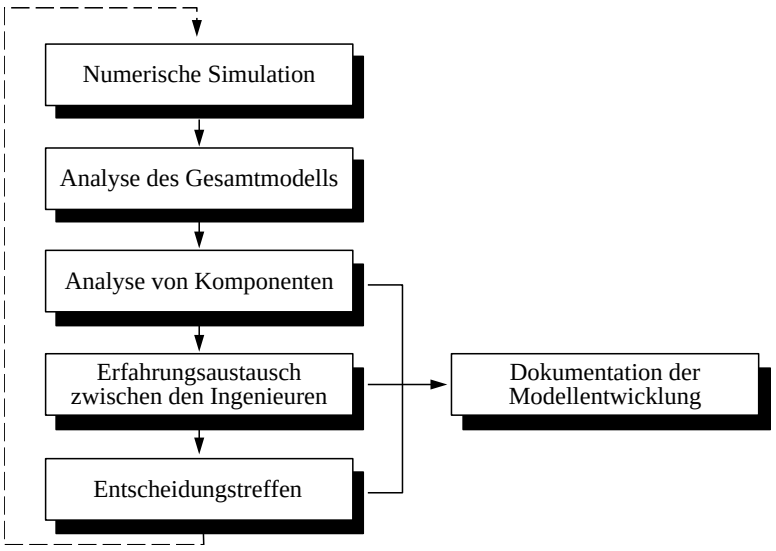

Abbildung 4.3: Teilschritte des Postprocessings.

4.2 Arbeitsumgebungen für das Postprocessing

Die einzelnen Teilschritte des Postprocessings (siehe Abbildung 4.3) stellen unterschiedliche Anforderungen an die Visualisierung der Berechnungsergebnisse. Die richtige Wahl der Hard- und Softwareumgebungen für die einzelnen Entwicklungsschritte ist für eine erfolgreiche Eingliederung eines Visualisierungssystems in den produktiven Prozess immens wichtig. In der Tabelle 4.2 sind den einzelnen Postprocessing-Teilschritten angepasste Hardwareumgebungen zugeordnet, auf die im Folgenden näher eingegangen wird.

4.2.1 Ingenieursarbeitsplatz

Der traditionelle Arbeitsplatz eines Berechnungsingenieurs ist ausgestattet mit einer Workstation der mittleren Leistungsklasse, an der die Eingabe über die Tastatur und die Standard 2D-Maus erfolgt. In einigen Fällen kommt bereits eine 3D-Maus, beispielsweise die DLR-Spacemouse, zum Einsatz. Für die Ausgabe werden handelsübliche Monitore verwendet, die teilweise zur stereoskopischen Darstellung mit Shutter-Brillen ausgestattet sind.

Die Hardware-Ausstattung begrenzt die maximale Teilnehmerzahl für einen Informationsaustausch am Arbeitsplatz auf 2-3 Personen. Die geringe Teilnehmerzahl beschränkt den Arbeitsplatz auf die Analysearbeit des Berechnungsingenieurs. Entscheidungen in größeren Runden können hier nicht gefällt werden, sondern müssen in speziellen Besprechnungsräumen stattfinden.

Je nach Rechnerleistung lassen sich Gesamtmodelle oder lediglich Teilmodelle in Standard-Postprocessing-Werkzeugen visualisieren. Am Arbeitsplatz kommen überwiegend die etablierten Postprocessing-Systeme, wie *Animator (GNS), PAM-VIEW (ESI)* oder *EnSight (CEI)*, zum Einsatz. Die Steigerung der Graphikleistung von Workstations erlaubt mittlerweile für die Visua-

Applikations-szenarien	Typ der virtuellen Umgebung	Anzahl der Anwender	Ein- und Aus-gabegeräte
Gesamtmodell-analyse	VR-Labor (immersiv)	1-2	CAVE BOOM Tracking-System CyberGlove
Komponenten-analyse	Ingenieurs-arbeitsplatz	1-3	Monitor Stereobrillen 2D-Maus 3D-Spacemouse
Wissensaustauch über das Intra-/Internet	*VRML*-Darstellung im Internet-Browser	2	Monitor 2D-Maus
Entscheidungs-treffen	Projektions-räume	10-30	Projektionsleinwand 2D-Maus 3D-Spacemouse Stereobrillen
Dokumentation der Modell-entwicklung	*VRML*-Darstellung im Internet-Browser	1-3	Monitor 2D-Maus

Tabelle 4.2: Die verschiedenen Applikationsszenarien mit korrespondierenden virtuellen Umgebungen.

lisierung auch den Einsatz von virtuellen Umgebungen mit einem höheren Grad an Interaktion, wie sie in dieser Arbeit vorgestellt werden.

Die Einführung von neuen Ein- und Ausgabegeräten wird schon bald zu einer Veränderung der Arbeitsbedingungen führen. Die stereoskopische Darstellung von geometrischen Objekten ist in einigen Bereichen bereits Standard und die Verwendung von haptischen Geräten, wie zum Beispiel dem *Phantom (SensAble)*, wird zu neuen Arbeitsformen und -abläufen führen, wie in Abbildung 4.4 exemplarisch dargestellt.

4.2.2 Projektionsraum

Wichtige und kostspielige Entscheidungen über die Weiterentwicklung von Fahrzeugen werden in Besprechungen mit 10-30 Teilnehmern gefällt. Besprechungsräume ohne multimediale Geräte reichen im traditionellen Entwicklungsprozess aus, da hier anhand von ausgedruckten Bildern und Plots entschieden wird. Die Ausnutzung von neuen Medien bietet im digital-unterstützen Entwicklungsprozess eine neue Stufe der Entscheidungsfindung an, da die Diskussionsgegenstände interaktiv auf den Leinwänden dargestellt werden können. Problematische Sachverhalte lassen sich so leichter präsentieren und regen die Diskussion unter den Teilnehmern

Abbildung 4.4: Das haptische Eingabegerät *Phantom* von *SensAble* im Einsatz am Arbeitsplatz.

an. In Projektionsräumen lassen sich Entscheidungen schneller herbeiführen und sie eignen sich ebenfalls für wichtige Managementpräsentationen.

Die neuentwickelten und kostengünstigen LCD- bzw. Röhrenbeamer führten in den letzten zwei Jahren zu starken Investitionen im Bereich der Projektionstechnik. In den meisten Fällen werden die vorhandenen Besprechungsräume durch mobile oder stationäre Auflichtprojektionsleinwände und Beamer aufgewertet. Ausgestattet mit einer mittelgroßen Workstation oder einem leistungsstarken Graphikrechner können in den umgerüsteten Besprechungsräumen interaktive Visualisierungen durchgeführt werden.

Die Ausstattung von Projektionsräumen kann sich in vielen Merkmalen unterscheiden. Zum einen ist die Graphikleistung des installierten Rechners für die Präsentationsmöglichkeiten entscheidend. Zum anderen ist für die Projektionsqualität die Art und Anzahl der Beamer wichtig. Eine sinnvolle Einteilung der Projektionsraumvarianten kann in mobile und stationäre Lösungen vorgenommen werden.

Eine mobile Projektionsanlage besteht aus einer tragbaren Kombination von LCD-Beamer und Leinwand. Die Kosten sind im Vergleich zur stationären Einrichtung mit lediglich 20.000 DM für den LCD-Beamer sehr gering. Die Anlage wird in einem beliebigen Besprechungsraum bei Bedarf aufgebaut und nach dem Einsatz wieder schnell entfernt, wodurch auch eine anderweitige Raumnutzung möglich ist. Der Preis für die hohe Flexibilität ist allerdings eine geringere Darstellungsqualität, bedingt durch die meist nicht ausreichend Raumabdunklung der provisorischen Aufbauten. Stereoskopische Darstellungen sind mit den handelsüblichen tragbaren Beamern ebenfalls noch nicht möglich, befinden sich aber gegenwärtig in der Entwicklung. Für Messen, Konferenzen und wichtige Managementvorführungen können auch große stereofähige Projektionsanlagen, bestehend aus zwei Röhrenbeamern, temporär aufgebaut werden. Der Aufbau und die Kalibrierung einer solchen Anlage ist jedoch sehr kosten- und zeitintensiv und erfüllt somit nicht die Anforderungen des täglichen Einsatzes.

Eine mobile Projektionsanlage erfordert auch einen mobilen Rechner, somit kommen hier meistens Laptops, PCs oder eventuell *SGI Octanes* bzw. *O2's* zum Einsatz. Die Leistungsfähig-

keit dieser Rechner ist beschränkt und es können meist nur Teilmodelle interaktiv dargestellt werden. Gut geeignet sind mobile Projektionsanlagen also für Demonstrationen von kleinen oder Teilmodellen mit eingeschränkter Interaktivität.

Die Installation einer stationären Projektionsanlage ist gekoppelt an umfangreiche und kostenintensive Baumaßnahmen. Sie betreffen den Einbau der Leinwand, der Beamer, des Rechners und einer Verdunkelungsvorrichtung. Der feste Einbau der Komponenten gewährleistet durch optimale Verdunkelung eine hohe Darstellungsqualität. Bei der Wahl der Projektionsart fällt die Entscheidung häufig auf die Rückprojektion, oder auch Durchlichtprojektion genannt. Sie bietet den Vorteil der hohen Leuchtkraft und der Begehbarkeit des Bereichs direkt vor der Leinwand ohne Schattenwurf des Benutzer auf die Projektionsfläche. Große Durchlichtanlagen mit Leinwandgrößen von 6 x 2 Meter können nur stationär installiert werden.

Eine stationäre Anlage kann je nach Leinwandgröße und Qualitätsanforderungen aus mehreren Beamern bestehen. Ein einzelner Beamer kann eine maximale Fläche von ca. 3 x 2 Metern mit einer stereoskopischen Darstellung bei einer Auflösung von 1280 x 1024 Bildpunkten beleuchten. Große hochaufgelöste Leinwände werden von mehreren Beamern bestrahlt, deren Lichtkegel durch den Einsatz des *Edge Blendings* zu einer gleichmäßig ausgeleuchteten Fläche verschmelzen. Eine durchschnittliche stationäre Anlage verfügt über eine 6 x 2 Meter große Leinwand mit drei Röhrenbeamern.

Der optimale Rechner für einen Projektionsraum mit mehreren Beamern ist momentan die *SGI ONYX*, da nur dieses System mehrere synchronisierte Graphiksubsysteme besitzt. Die Leistungsfähigkeit eines Rechners für eine Projektionsleinwand hängt von der Anzahl der Graphiksubsysteme ab. So lassen sich beispielsweise mit drei Graphiksystemen stereoskopische Projektionen mit einer hervorragenden Auflösung von 3240 x 1024 Bildpunkten erreichen.

Die volle Leistung einer Projektionsanlage kann allerdings nur mit Software erreicht werden, die mehrere Graphiksysteme unterstützt. Standardapplikationen beherrschen die Aufteilung einer graphischen Szene auf mehrere Graphiksysteme nicht und können so nur auf einem einzelnen Graphiksystem mit geringerer Auflösung und niedrigerer Bildwiederholungsfrequenz betrieben werden. Störend an einer Projektionsleinwand ist ebenfalls die 2D-Menüführung von Standardapplikationen, da diese überdimensional auf der Leinwand erscheint und die graphische Darstellung stört. Besser ist die Trennung von graphischer Darstellung und 2D-Menüführung. Die Steuerung der Applikation erfolgt dann über einen separaten Rechner, auf den die 2D-Menü-Oberfläche umgeleitet wird. Die Betrachter der Leinwand können sich so voll auf die dreidimensionale graphische Darstellung konzentrieren.

Applikationen von Forschungseinrichtungen oder kommerzialisierte Programme aus dem universitären Umfeld, beispielsweise *Lightning* vom *Fraunhofer Institut für Arbeitswirtschaft und Organisation* oder *COVISE* von *Vircinity*, bieten die Funktionalität zur Einbindung von mehreren Graphiksystemen an. Sie erlauben nicht nur die „Fernsteuerung“ von Applikationen, sondern auch die Verwendung von Tracking Geräten, wie beispielsweise den *Ascension MotionStar* oder optisches Tracking, und neue Interaktionsmechanismen für die Projektionsumgebung.

In Projektionsräumen kommen in den meisten Fällen angepasste Spezialanwendungen zum Einsatz, deren Arbeitsweise und Interface sich stark von den Standard Postprocessing-Werkzeugen des Arbeitsplatzes unterscheidet. Der Benutzer ist somit gezwungen beide Systeme zu beherrschen. Der Vorteil eines universalen Systems liegt auf der Hand, denn kann ein Anwen-

der ohne großen Aufwand beide Systeme bedienen, so wird er die gebotenen Möglichkeiten auch nutzen. Im Rückschluss ist eine mit großem Einarbeitungsaufwand verbundene Neueinführung einer speziellen Applikation meistens nicht vom Erfolg gekrönt.

Beobachtungen bei der *BMW Group* haben gezeigt, dass das Konzept der Projektionsräume sehr gut aufgenommen wird. Für den Design Review Prozess werden regelmäßig Besprechungen in Projektionsräumen durchgeführt und Entscheidungen getroffen. Ziel ist es nun, dieses Möglichkeiten auch für die Visualisierung von numerischen Simulationen anzubieten.

4.2.3 Immersive Arbeitsumgebungen

In immersiven Arbeitsumgebungen kommen die klassischen VR-Interaktionsgeräte, wie beispielsweise *HMD* und Datenhandschuh, zum Einsatz. Diese für den produktiven Prozess neuartige Arbeitsform besitzt in vielen Bereichen noch Kinderkrankheiten und ist somit im produktiven Umfeld nicht stark verbreitet, sondern wird von vielen Forschungsvorhaben auf Einsatzmöglichkeiten untersucht. Die geringen Absatzzahlen führen zusätzlich zu sehr hohen Anschaffungskosten und der zum Teil nicht ausgereifte Zustand zu hohen Unterhaltungskosten.

In das traditionelle Arbeitsumfeld eines Ingenieurs lassen sich immersive Arbeitsumgebungen aufgrund ihres enormen Platzbedarfes nur schwer integrieren. Die Aktionsradien von 2-3 Metern eines *HMD*s oder eines *BOOM*s können nur schwerlich im Bürobereich untergebracht werden. Auch die 6 x 6 Meter messende Grundfläche einer *CAVE* kann praktisch nicht in Arbeitsplatznähe bereitgestellt werden. Der Aufbau von immersiven Hardware-Umgebungen ist daher momentan nur in speziellen VR-Labors möglich, in die sich der Ingenieur für spezielle Untersuchungen begibt. Die Erfahrungen im Umgang mit dieser Technik sind jedoch vielversprechend.

Eine Vielzahl von immersiven Geräten befinden sich momentan auf dem Markt, von denen allerdings nur ein Bruchteil für die produktive Arbeit sinnvoll ist. Head-Mounted-Displays sind in Kombination mit einem Tracking-System und einem Datenhandschuh (siehe Abbildung 4.5) eine günstige und anteilsmäßig am weitest verbreitete Variante eines immersiven Arbeitsplatzes. Das hohe Gewicht der *HMD*s erzeugt beim Anwender jedoch schnell Kopf- und Genickschmerzen und verhindert ein längeres Arbeiten in dieser Umgebung. Die geringe Auflösung, das kleine Sichtfeld und die Ungenauigkeit des elektromagnetischen Trackings beeinträchtigen zusätzlich die Präzision im Umgang mit den digitalen Modellen. Head-Mounted-Displays finden hauptsächlich in Forschungslabors zur Demonstration von zukünftigen Arbeitsvorgängen Anwendung.

Einige Nachteile der Head-Mounted-Displays sind beim *Fakespace BOOM* beseitigt worden. So lastet das Gewicht nicht mehr auf dem Kopf des Benutzers, sondern schwebt an einer Galgenkonstruktion förmlich vor seinen Augen. Die Darstellung in den Röhrenbildschirmen ist durch eine höhere Auflösung und ein größeres Blickfeld auch für Detailuntersuchungen geeignet. Die Hände des Benutzer sind jedoch durch das Halten der Displays gebunden, wodurch weitere Interaktionen mit dem System behindert werden. Zusätzlich erschwert das Nachschwingen des Ausgleichsgewichtes die präzise Positionierung und mindert den immersiven Eindruck. Der hohe Anschaffungspreis eines *Fakespace BOOM* beschränkt seine Anwendung ebenfalls auf Forschungslabors. Der *Push-BOOM* als Tischvariante des *BOOM*s ist für den Arbeitsplatz geeig-

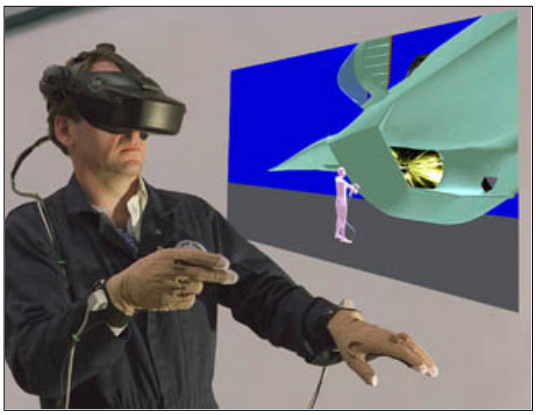

Abbildung 4.5: Immersiver Arbeitsplatz mit Head-Mounted-Display und *CyberGloves*.

net und bietet als reduziert immersives Gerät größere Einsatzmöglichkeiten. In Konkurrenz zum stereoskopischen Monitor hat sich der *Push-BOOM* am Arbeitsplatz jedoch nicht durchgesetzt.

CAVE-Installationen werden als einzige immersive Arbeitsumgebung im produktiven Einsatz genutzt. In ihnen werden Design-Reviews mit 1-4 Teilnehmern durchgeführt. Der Vorteil der Immersion erleichtert den Designern die Begutachtung des Fahrzeug-Interieurs, da die maßstabsgetreue stereoskopische Abbildung von virtuellen Fahrzeugkomponenten eine subjektive und objektive Bewertung von Schalteranordnungen und des Fahrersichtfeldes erlaubt. Hierfür wird in die *CAVE* eine Sitzkiste gerollt (siehe Abbilung 4.6) die der virtuellen Darstellung ein Stück des realen „Look and Feel“ verleiht. Praktisch alle großen Automobilhersteller besitzen einen CAVE-artigen Projektionsraum für die Begutachtung von Designentwürfen und CAD-Konstruktionen.

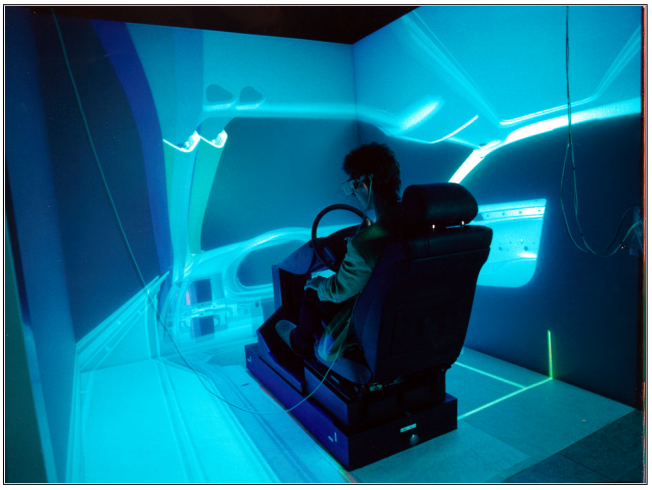

Abbildung 4.6: Die Sitzkiste zur Begutachtung eines virtuellen Interieurs in der CAVE.

Stärker noch als im Falle der Projektionsleinwände muss eine Applikation für den Betrieb in einer immersiven Lösung an die speziellen Ein- und Ausgabemechanismen angepasst werden. In den meisten Fällen ist eine Steuerung über Tastatur und 2D-Maus nicht möglich, sondern erfordert spezielle Treiber für die VR-Interaktionsgeräte. Die 2D-Menüsteuerung muss durch einen Menümechanismus ersetzt werden, der auch in stereoskopischer Darstellung und mit den VR-Eingabegeräten bedienbar ist.

Standardapplikationen, wie beispielsweise *4D-Navigator*, *PAM-VIEW* und *Animator*, sind nicht in der Lage, immersive Umgebungen zu betreiben. Stärker noch als in Projektionsanlagen kommen in immersiven Umgebungen daher speziell angepasste Applikationen zum Einsatz, die allerdings nicht das volle Datenformat-Spektrum des Entwicklungsprozesses abdecken. Meistens findet eine Beschränkung der Applikationen auf die Visualisierung von Konstruktiongeometrie statt.

Trotz der vielen Probleme und Kosten bieten die immersiven Arbeitsumgebungen für die Zukunft ein hohes Potenzial zur Verbesserung der Arbeitsumfeldes.

4.2.4 Netzwerkunterstützte Arbeitsumgebung

Die Globalisierung der Märkte führt zu neuen Anforderungen an moderne Arbeitsplätze, da weltweit verteilte Ingenieurteams gemeinsam Produkte entwickeln. Die neuen Arbeitsmethoden erfordern schnelle und globale Kommunikationswege, die über die reine Telekommunikation hinausgehen und zur Nutzung des Internets mit seinem schnellen und globalen Austausch von elektronischen Informationen führt. Die Kommunikation über das Internet, beispielsweise per email, hat bereits in vielen Bereichen zu einem Wandel der Arbeitsweisen geführt und die Vorteile des Internets sollen auch für die Visualisierung im Ingenieursumfeld genutzt werden.

Im Fahrzeugentwicklungsprozess untersuchen Ingenieure anhand von dreidimensionalen Berechnungsergebnissen mögliche Fehlerquellen und Verbesserungsmaßnahmen. Räumlich getrennt arbeitende Ingenieure müssen über einen vorliegenden Sachverhalt kommunizieren können. Der Austausch eines kompletten Ergebnisdatensatzes einer Simulation ist aufgrund der enormen Datenmengen meist nicht praktikabel und so rücken drei verschiedenen Varianten zum Informationsaustausch über das Internet in den Vordergrund:

- Austausch von Bildern oder *VRML*-Dateien
- Kooperatives Arbeiten in synchronisierten Applikationen
- Verwendung eines Bilderstroms für verteiltes Arbeiten

Der Austausch von Dateien in plattformunabhängigen Dateiformaten ist die einfachste Verwendung des Internets. Die Anwender erstellen in ihren Standardapplikationen beispielsweise Bilder oder *VRML*-Szenen und verschicken diese anschließend über das Internet. In Telefon- oder Konferenzschaltungen kann anhand der ausgetauschten Daten der Sachverhalt diskutiert werden. Das Fehlen einer interaktiven Analyse über das Internet lässt kein kooperatives Arbeiten zu und unterschiedliche Interpretationen der Bilder oder *VRML*-Darstellungen führen leicht zu Missverständnissen und zu einer problematischen Diskussionsführung.

Digitale Bilder und dreidimensionale Modelle in Standardformaten eignen sich jedoch sehr gut für die Dokumentation des Entwicklungsprozesses. Auf allgemein zugänglichen HTML-Seiten können sich die involvierten Anwender durch zusammengestellte Bilder und *VRML*-Szenen über den aktuellen Stand der Entwicklung informieren. Für die Bereitstellung von statischen oder animierten Szenen reicht der Datenaustausch des Internets aus, da kein interaktiver Wissensaustausch zwischen Ingenieuren erfolgt.

Kooperatives Arbeiten bietet durch die direkte Interaktion zwischen den Teilnehmern eine gute Diskussionsplattform. In einer virtuellen Umgebung werden die beteiligten Anwender durch die Synchronisation der Applikationen zusammengeführt. In gering immersiven Umgebungen sitzen die Teilnehmer an ihren Arbeitsplätzen und betrachten das dreidimensionale Modell aus synchronisierten Blickrichtungen. Zur Vermeidung von Inkonsistenzen ist es in einer solchen Umgebungen immer nur einem Benutzer gestattet, den Augpunkt und das Modell zu manipulieren. Unterstützt wird die Diskussion durch virtuelle Marker in der Szene und eine zusätzliche Telefon- oder Videoverbindung. In immersiven Umgebungen, beispielsweise in CAVEs, können die Teilnehmer auch über große Entfernungen gut miteinander arbeiten, da sie ihren Gegenüber als virtuellen *Avatar* sehen können.

Es gibt verschiedene Konzepte zur Synchronisation von Applikationen. Die Konfiguration der Verbindung hängt von der verfügbaren Hardware ab. Steht jedem Anwender ein leistungsstarker Graphikrechner zur Verfügung, so genügt der Austausch der Interaktionseingaben und der Transformationsmatrizen von virtuellen Objekten bzw. des Augpunktes für die Synchronisation. In diesem Konzept rendern beide Seite lokal den Vorgaben entsprechend die Szene. Die Auslastung des Netzwerkes ist sehr gering, so dass auch leistungsschwache 10 MBit Netzwerke ausreichen. Als plattformunabhängige Middleware kommt beispielsweise *CORBA* in Frage.

Um auch leistungsschwache Rechner für kooperatives Arbeiten zu Nutzen, empfiehlt sich die Organisation des Datenaustausches in einem Bilderstrom. Die Szene wird auf einem leistungsstarken Graphikrechner gerendert und der entstehende Bilderstrom komprimiert über das Internet zum leistungsschwachen Client verschickt. Vom Client aus werden die Eingaben von Maus und Tastatur zur Interaktion zum Graphikserver übermittelt. Diese Vorgehensweise wird beispielsweise von [34] erforscht. Interaktive Bildwiederholungsraten lassen sich auf Basis eines Bilderstroms allerdings nur mit einem schnellen Netzwerk erzielen.

Standard-Postprocessing-Systeme bieten bisher keinen Internet-Support an. So wird dieses Feld momentan von universitären oder universitätsnahen Lösungen, wie beispielsweise *COVISE*, bedient. Die internationalen Firmenzusammenschlüsse erzeugen jedoch einen enormen Bedarf an kooperativen Werkzeugen, die auch über große Entfernungen und leistungsschwache Netzwerke arbeiten. In absehbarer Zeit werden somit die heutigen Forschungsergebnisse in kommerzielle Produkte einfließen.

4.2.5 Fazit

Die Einführung eines neuen Postprocessing-Werkzeuges kann nur erfolgreich verlaufen, wenn es gegenüber den in den Arbeitsumgebungen etablierten Systemen signifikante Vorteile bietet. Standardpostprozessoren skalieren nur bedingt auf semi-immersive und immersive Hardware-Umgebungen, da sie einzig auf den Arbeitsplatz ausgelegt sind. Die in VR-Labors verwendeten

Applikationen sind häufig nur nach längerer Einarbeitungszeit von spezialisierten Anwendern bedienbar und besitzen für den Arbeitsplatz einen zu hohen Ressourcenbedarf. Ein neues Visualisierungssystem muss daher über eine intuitive, in allen Arbeitsumgebungen identische Benutzerführung verfügen und durch eine hohe Interaktivität mit schnellen Algorithmen bestechen.

Kapitel 5

Interaktion und Navigation in virtuellen Umgebungen

In nicht-immersiven Arbeitsplatzapplikationen erfolgt die Interaktion und Navigation über zweidimensionale graphische Menüs, die der Anwender über die Standardeingabegeräte 2D-Maus und Tastatur bedient. Standardisierte Programmierbibliotheken besitzen einen umfangreiche Satz an Bedienelementen, so genannte *Widgets*, aus denen der Programmierer die Benutzerschnittstelle (engl. graphical user interface *GUI*), leicht zusammensetzen kann.

Die Steuerung einer virtuellen Umgebung erfordert allerdings andere Interaktionsmechanismen, da bereits in stereoskopischen Darstellungen solche 2D-Menüs nur noch mit Einschränkungen eingesetzt werden können. Der ständige Wechsel zwischen der zweidimensionalen Menüführung und der dreidimensionaler Szenendarstellung führt zum Immersionsverlust und zur Verwirrung des Anwenders. In vollimmersiven Umgebungen erfolgen sämtliche Interaktionen und Navigationen über das Tracking-System in Verbindung mit einem Tastengerät, wie beispielsweise dem *Stylus*, und erfordern so eine spezielle Benutzerführung.

Die bekannten Graphik-APIs, wie zum Beispiel *Inventor*, *Performer*, *Optimizer* und *World-ToolKit*, bieten keine Lösung für dieses Problem an und es existiert auch keine unabhängige Standardsoftware. Der Bedarf einer Interaktionssteuerung für die zu entwickelnden Visualisierungssysteme führte in dieser Arbeit zur Betrachtung des Themas Interaktion und Navigation und resultierte schließlich in der Entwicklung eines 3D-Menüs zur Integration in die Szenengraphen der verschiedenen Graphik-APIs. Die Benutzerführung und Ergonomie spielen hierbei eine wichtige Rolle, da eine virtuelle Umgebung vom Anwender möglichst schnell und intuitiv bedient werden soll.

5.1 Interaktionselemente

Die Möglichkeiten des Benutzers, auf die virtuelle Welt einzuwirken und sie zu steuern, werdern im Allgemeinen als Interaktion bezeichnet. Die verschiedenen Interaktionsmöglichkeiten lassen sich nach Chris Hand [50] in drei Bereiche einteilen:

- Applikationskontrolle
- Manipulation von geometrischen Objekten
- Navigation des Benutzers durch die virtuelle Welt

5.1.1 Applikationskontrolle

Die Kontrolle der Applikation ist die Kommunikation zwischen dem Benutzer und Teilen des Systems, die nicht unbedingt Teil der visuellen Welt sind. Über sie kann der Anwender Prozesse starten und Betriebsparameter einstellen oder verändern. Die Benutzereingaben erfolgen in der zweidimensionalen Welt über die in 2D-Menüs zusammengefassten *Widgets*, wie beispielsweise Toggle-, Vario- und Radio-Buttons oder Schiebe- und Drehregler. *Motif* und *MFC* sind zwei GUI-Standards für Unix bzw. Windows Betriebssysteme.

Immersive VR-Systeme, die nicht über Spracheingabe verfügen, benötigen ebenfalls ein *GUI*, über das der Anwender mit der Applikation kommunizieren kann. Viele 2D-Interaktionselemente wie Buttons und Regler können für die virtuellen Welten nachempfunden werden. Dem Anwender sind ihre Form und Funktionalität auf Anhieb bekannt und er findet sich leichter zurecht. In vielen Beispielen werden die flachen Elemente durch dreidimensionale Boxen ersetzt, jedoch ist dabei auf einen vorsichtigen Umgang mit den Geometrieressourcen zu achten. Wird die Geometrie des Menüs zu aufwendig, führt dies automatisch zu geringeren Bildwiederholungsraten der Applikation.

Ein großes Problem stellt hierbei die Anzeige von Zahlen und Buchstaben dar. Sie können nicht einfach durch geometrische Objekte mit 20-50 Polygonen pro Zeichen ersetzt werden. Die Verwendung von Texturen hingegen ermöglicht die Abbildung von beliebig komplexen Zeichenkolonnen auf einem Polygon. Der Performanzverlust für moderne Graphiksysteme wird so auf ein Minimum reduziert.

Das von *SGI* entwickelte Programm *Buttonfly* besteht aus einem solchen einfachen dreidimensionalen Menü, in dem der Anwender mit der 2D-Maus Menüpunkte auswählen kann, die sich dann aus der Ebenen lösen und auf der Rückseite weitere Auswahlmöglichkeiten anbieten (siehe Abbildung 5.1). Dieses Menü ist zwar für virtuelle Umgebungen ungeeignet, zeigt aber eine erste Einsatzmöglichkeit von dreidimensionaler Graphik zur Benutzerführung.

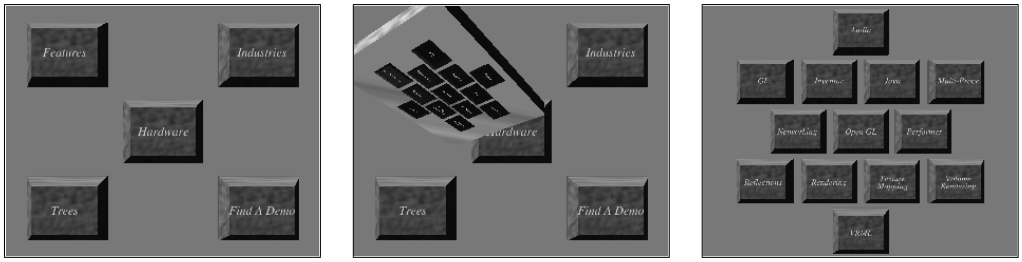

Abbildung 5.1: Auswahlprozedur des 3D-Menüs von *Buttonfly*.

Die Positionierung, Ausrichtung und Größe einer 3D-Interaktionsschnittstelle ist extrem wichtig. Das Menü muss für den Anwender immer leicht erreichbar und vor allem gut sichtbar sein. Andere Objekte der Szene dürfen es also nicht verdecken oder durchdringen. Zur Lösung des Problems sind viele 3D-Menüs frei beweglich und können beliebig im Raum platziert werden, wobei sie automatisch zur Blickrichtung des Benutzers ausgerichtet werden. In einigen Fällen wird das Menü zur optimalen Verfügbarkeit an die virtuelle Hand gekoppelt. Zum Erkennen der Aufschriften muss ein 3D-Menü ausreichend groß sein, darf das Blickfeld jedoch nicht zu stark einschränken.

Ein weiterer Ansatz ist die Kopplung einzelner Menüelemente an die Szenenobjekte, wie er beispielsweise beim Konstruktionsprogramm *Truespace4* von *Caligari* zur Erstellung von Szenen und Animationen realisiert wurde. Die verwendeten 3D-Widgets werden direkt an das betreffende Objekt angebracht und je nach Typ stehen entsprechende Bedienelemente mit eindeutiger Zuordnung zum Editieren der Eigenschaften bereit, siehe Abbildung 5.2. Allerdings sind die Probleme der Platzierung und Größe durch die hohe Anzahl der objektbezogenen Widgets größer und mit steigender Widgetzahl sinkt die Übersichtlichkeit und Lesbarkeit der Elemente. Eine kombinierte Lösung aus objektbezogenen Widgets und einem globalen 3D-Menü ist daher für den Anwender besser zu handhaben.

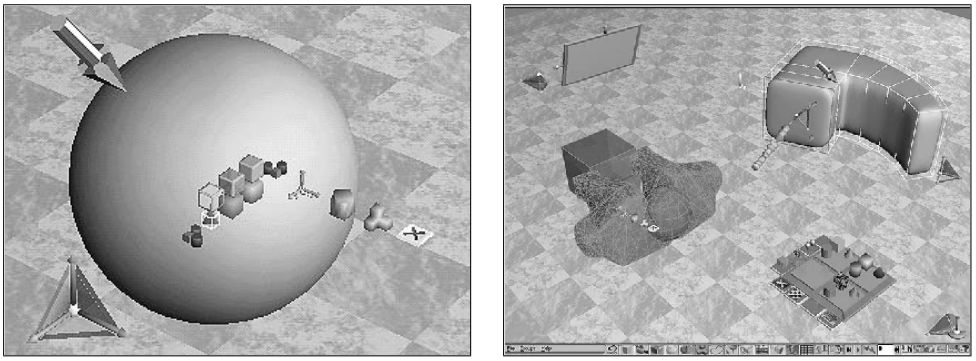

Abbildung 5.2: Die 3D-Widgets des Konstruktionsprogramms *Truespace4* werden direkt an den Szenenobjekten angebracht.

Die Auswahl einer Menükomponente in einem 2D-Menü erfolgt durch die einfache Positionierung der Maus auf dem Element und wird durch Klicken einer Maustaste ausgelöst. Dieser Mechanismus wird in immersiven Umgebungen durch einen virtuellen „Laserpointer“ nachgebildet. Ausgehend von der getrackten Position der Hand wird in Richtung des „Laserstrahls“ das erste geschnittene Objekt gesucht und durch Drücken einer Taste angewählt. Die einfache Bestimmung von Schnitten zwischen einer Linie und Polygonen ist nicht sehr aufwendig und kann somit interaktiv erfolgen.

Im virtuellen Windtunnel Projekt [14] des NASA Ames Research Center wurde ein virtueller Laserstrahl zur Eingabe von Betriebsparametern und zur Steuerung des Menüs realisiert. Wählt der Anwender beispielsweise die Kugel eines Schiebereglers (siehe Abbildung 5.3 links) aus, so kann er die entsprechenden Betriebsparameter durch Verschieben der Kugel innerhalb der

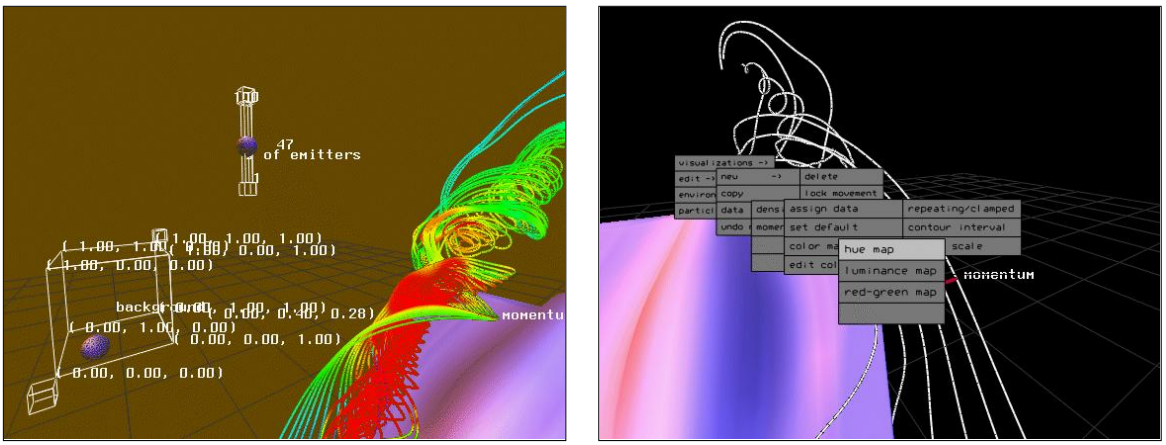

Abbildung 5.3: Das 3D-Menü des virtuellen Windtunnels.

Drahtgitterbox modifizieren. Die flachen Menüelemente (siehe Abbildung 5.3 rechts) können ebenfalls mit dem virtuellen Laser angewählt werden.

Vollständig von den Vorgaben aus dem zweidimensionalen Raum löst sich das Beispiel des Kugel-Menüs (siehe Abbildung 5.4) des *Fraunhofer Instituts für Arbeitswirtschaft und Organisation* ab. Das Kugel-Menü erscheint durch Drücken einer Taste am Tracking-Sensor direkt an der virtuellen Hand. Jedem Kugelsegment wird eine Funktion oder ein Untermenü zugeordnet, das der Benutzer durch Drehen der Hand nach vorne bringen und durch Loslassen des Knopfes auswählen kann. Diese Arbeitsweise erleichtert die Auswahl der Menüpunkte, da der Anwender nicht mit einem Laserpointer auf Buttons zielen muss. Allerdings ist nur eine geringe Menge an Segmenten möglich, wodurch die Auswahl stark eingeschränkt wird. Die Verwendung von vielen Untermenüs erweitert zwar die Anzahl von Menüpunkten, jedoch ist für den Anwender nicht sofort ersichtlich in welchem Untermenü er sich gerade befindet. Die Vorteile des Kugelmenüs können nur im Einsatz mit einem Tracking-System genutzt werden.

Abbildung 5.4: Das vom *Fraunhofer Institut für Arbeitswirtschaft und Organisation* entwickelte Kugel-Menü.

Die angeführten Beispiele zeigen eine Reihe von Lösungvorschlägen auf. Jede Variante ist aber auf die spezielle Applikation zugeschnitten und lässt sich nicht allgemein verwenden. Gegenwärtig gibt es lediglich einige Studien [117, 55, 51], die sich mit dem Erscheinungsbild und der Funktionalität von 3D-*GUI*s beschäftigen, aber noch keine Standardprogrammierbibliothek.

5.1.2 Manipulation von Objekten

Die Transformation und die Modellierung von virtuellen Objekten wird allgemein als Manipulation bezeichnet. Vom Anwender können sowohl komplexe Objektgeometrien, als auch die Ersatzgeometrien von Interaktionselementen manipuliert werden, wie sie in den Visualisierungssystemen dieser Arbeit vorkommen.

5.1.2.1 Transformation von Objekten

Die Transformation umfasst die Translation, Rotation und Skalierung von virtuellen Objekten. In vielen Anwendungen, wie zum Beispiel CAD-Systemen oder Szeneneditoren, ist gerade für ungeübte Anwender eine interaktive Manipulation von Objekten über 3D-Widgets leichter als eine umständliche Positionseingabe in zweidimensionalen *GUI*-Elemente.

Die Funktionsweise von 2D-Widgets (siehe Abbildung 5.5) aus bekannten 2D-Graphikapplikationen sind Vorbild für die dreidimensionale Variante. Im 2D-Transformationsmechanismus erscheinen nach der Auswahl eines Objektes kleine Interaktionsgriffe (engl. *Handles*), über die der Anwender mit Hilfe der 2D-Maus in einem Translations- und Rotationsmodus die Position und Orientierung des Objektes verändern kann.

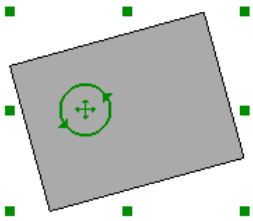

Abbildung 5.5: Handles zur Manipulation von zweidimensionalen Objekten, beispielsweise einem Rechteck.

In zweidimensionalen Anwendungen haben sich die Handles sehr gut bewährt und weit verbreitet. Die Ähnlichkeit der Problemstellung legt eine Umsetzung in den dreidimensionalen Raum nahe. Transformationsänderungen in den sechs Freiheitsgraden des dreidimensionalen Raumes können erst durch die klar definierten Bewegungsabläufe der 3D-Widgets vom Benutzer leicht auf die Objekte übertragen werden.

In dem Toolkit *OpenInventor* wurden so genannte *Manipulatoren* als 3D-Handles zur Transformation beliebiger Szenengraph-Objekte realisiert. Für die einzelnen Bewegungsabläufe Translation, Rotation und Skalierung gibt es, ähnlich wie im 2D-Fall, verschiedene Handles zur

Erfüllung der Aufgaben. In Abbildung 5.6 ist links der Manipulator für die Translation bzw. Skalierung und rechts für die Rotation zu sehen. Das Konzept des *OpenInventor* Szenengraphen erlaubt die einfache Integration von Eingabegeräten inklusive Event-Weiterleitung an die *Manipulatoren* und ist unter den Szenengraph-APIs einmalig.

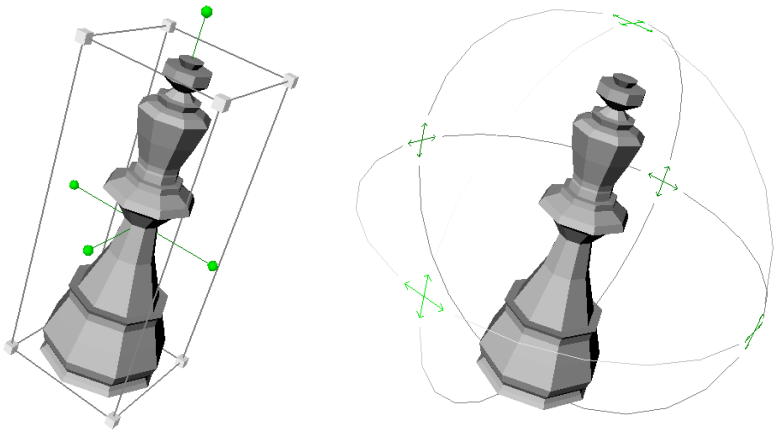

Abbildung 5.6: Die OpenInventor Manipulatoren *Transformer* und *Centerball*.

In den anderen bekannten Szenengraph-APIs, wie *Performer*, *Optimizer* und *WorldToolKit*, ist die Event-Verarbeitung nicht Teil des Szenengraphen und somit gibt es für sie auch keine Standard-Handles. Die meisten VR-Applikationen verfügen daher über proprietäre Lösungen zur Objekttransformation, da diese Funktionalität für virtuelle Umgebungen unerlässlich ist.

5.1.2.2 Modellierung von Objekten

Änderungen an der Objektoberfläche werden unter dem Begriff Modellierung zusammengefasst und besitzen gegenüber den Objekttransformationen eine höhere Komplexität. Objekttransformationen manipulieren das virtuelle Objekt als Ganzes, während die Modellierung die einzelnen Elemente der Oberflächenbeschreibung verändert. Entscheidend für die Modellierungstechniken ist die Oberflächendefinition als voxelbasierte, polygonale oder parametrische Beschreibung.

Voxelbasierte Oberflächen werden in dreidimensionalen kartesischen Gittern durch Markieren einzelner Zellen definiert. Der Vorteil für die Modellierung liegt in der einfachen Strukturbeschreibung. In einer virtuellen Umgebung beispielsweise können mit Hilfe eines Tracking-Systems Punkte gesetzt werden, die ohne weitere Verbindungsinformation Oberflächen beschreiben. Leicht zu realisieren sind additive und subtraktive Modellierungstechniken, die in virtuellen Werkzeugen wie Messer, Sandpapier, Pastatube und Bunsenbrenner zur Manipulation von Voxelobjekten umgesetzt werden [43]. Voxelbasierte Oberflächenbeschreibungen sind aufgrund des hohen Speicherbedarfs nicht weit verbreitet und wird hauptsächlich zur Visualisierung von Röntgentomographie-Daten eingesetzt.

Die Stütz- und Kontrollpunkte der parametrischen Oberflächenbeschreibungen ermöglichen eine sehr flexible Modellierung. Die Oberflächen führen durch die Stützpunkte und die dazu-

gehörigen Kontrollpunkte definieren den Flächenverlauf. Die Translation der Punkte, beispielsweise durch ein Interaktionsgerät, erlaubt eine einfache Manipulation der Oberfläche. Eingesetzt wird diese Art der Oberflächenbeschreibung, wie beispielsweise B-Splines und NURBS, vor allem in CAD-Systemen.

Polygonnetze sind die am weitesten verbreitet Beschreibung von Oberflächen. Die einzelnen Punkte des Netzes definieren durch Nachbarschaftsbeziehungen eine stückweise planare Fläche, die durch Translation der Punkte lokal manipuliert werden kann. Ein vielversprechender Ansatz zur globalen Modellierung ist die Subdivision-Technik [70]. Sie betrachtet eine polygonale Fläche in mehreren Auflösungsstufen und ermöglicht so eine lokale Verfeinerung oder Vergröberung von Polygonnetzen. Unter Verwendung von Handles können Polygonnetze fast wie Splines sehr flexibel editiert werden (siehe Abbildung 5.7).

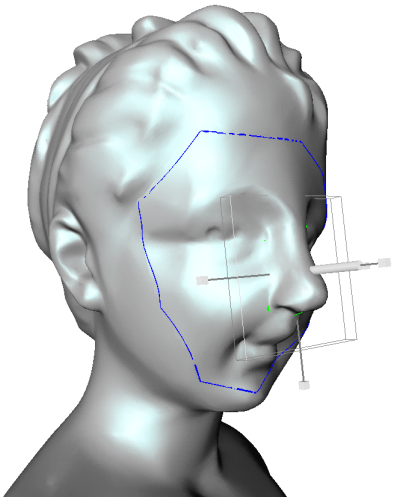

Abbildung 5.7: Manipulation einer polygonalen Objektoberfläche mit einem 3D-Handle.

Zur Darstellung am Bildschirm müssen alle Flächendarstellungen in Pixel umgewandelt werden. Die Optimierung der aktuellen Graphiksysteme für polygonale Flächenbeschreibungen, schließt den Einsatz von voxelbasierten und parametrisierten Oberflächen in Applikationen mit einem hohen Interaktionsgrad praktisch aus. Zukünftig kann aber gerade die voxelbasierte Objektdefinition sehr interessant werden.

Die Ungenauigkeit der Tracking-Systeme verhindert momentan einen produktiven Einsatz von interaktiver Objektmodellierung im Ingenieursumfeld. Die Einhaltung von Fehlertoleranzen im Millimeterbereich gerade im Automobilbau kann mit heutigen Tracking-Systemen nicht gewährleistet werden. Nur in Entwurfsphasen des Entwicklungsprozesses, wie beispielsweise dem Design und der Erstellung von Berechnungsvarianten, ist interaktive Objektmanipulation denkbar und wird bereits erprobt.

Die Entwicklung von genaueren Eingabegeräten wird langfristig zu einer Verbesserung dieser Situation führen. Sehr vielversprechend ist auch der Fortschritt im Bereich der haptischen Eingabegeräte, die in vielen Bereichen zu stark veränderten Arbeitsweisen führen werden.

5.1.3 Navigation

Die virtuelle Welt betrachtet der Anwender durch eine virtuelle Kamera, deren Position und Orientierung er mit Hilfe der Eingabegeräte transformieren und so durch die virtuelle Welt navigieren kann. Um den größtmöglichen Immersionseffekt zu erzielen, müssen die Eingaben des Benutzers möglichst intuitiv abgebildet werden. Die Art der Navigation hängt somit stark von der Hardware-Ausstattung ab.

In immersiven Umgebungen wird die Kopfposition des Anwenders mit Hilfe eines Tracking-Systems direkt auf den virtuellen Augpunkt abgebildet. Der Immersionseffekt ist in solchen Hardwareumgebungen natürlich am stärksten, da ohne lange Gewöhnungszeit dem Anwender eine bekannte Form der Navigation zur Verfügung steht. Allerdings sind solche Systeme nur in VR-Labors verfügbar.

Die Eingabegeräte am Arbeitsplatz besitzen meistens nur zwei bis drei Freiheitsgrade und reduzieren so die verfügbaren Bewegungsrichtungen für die Navigation. Die Übernahme von Fortbewegungsformen aus der realen Welt, wie Fahren und Fliegen, helfen trotz der Einschränkungen eine intuitive Navigation zu realisieren. Beispielsweise wird in der Fahr-Navigation die y-Position der Maus auf die Fortbewegungsgeschwindigkeit und die x-Position auf die horizontale Rotation abgebildet. Die Verwendung von verschiedenen Bewegungsmodi ermöglicht die Erschließung aller Freiheitsgrade. Je nachdem welcher Button der Maus gedrückt wird, findet eine Abbildung der Parameter auf Translation oder Rotation des Augpunktes statt. Eine geschickte Zusammenfassung der Freiheitsgrade in den Bewegungsmodi erlaubt dem Anwender nach kurzer Gewöhnungsphase eine intuitive Navigation durch den virtuellen Raum.

Eine komfortable Art der Navigation in Arbeitsplatzumgebungen bieten dreidimensionale Mäuse, da sie eine gleichzeitige Änderung in allen sechs Freiheitsgraden ermöglichen. Geübte Benutzer können mit ihnen sehr präzise, ähnlich einem Helikopterflug, durch den virtuellen Raum navigieren.

Eine zielorientierte Navigation bieten beispielsweise *VRML*-Browser an. Durch das Picken eines Punktes auf einem Polygon der Szene wird das Ziel einer Bewegung definiert, auf den die virtuelle Kamera automatisch zusteuert. Diese halbautomatische Form der Navigation ist durch ihre leichte Verständlichkeit sehr intuitiv.

Neben einer Treibersoftware als Schnittstelle zwischen Gerät und Applikation benötigen die virtuellen Welten auch sinnvolle Navigationsmechanismen, um die Benutzereingaben auf Augpunkttransformationen abzubilden.

5.2 3D-Interaktionsmenü *G3Menu*

Das Fehlen eines standardisierten 3D-Menüs erfordert eine eigenständige Lösung der Applikationskontrolle für jede neu entwickelte virtuelle Umgebung. Folglich besitzt jede virtuelle Welt ihre eigene Variante eines 3D-Menüs mit eigenem Bedienkonzept. Wünschenswert ist eine einfach zu benutzende Programmierbibliothek, mit der sich 3D-Menüs leicht und flexibel erstellen lassen und die dem Benutzer nach einmaliger Lernphase in der Bedienung den Zugang zu vielen virtuellen Umgebungen ermöglicht.

Anhand des in dieser Arbeit entwickelten Interaktionsmenüs *G3Menu* wurde das Konzept einer Szenengraph-unabhängigen Programmierbibliothek und deren Integration in die prototypischen Visualisierungssysteme untersucht. Das universelle *G3Menu* arbeitet gleichermaßen mit den Szenengraph-APIs *OpenInventor*, *WorldToolKit* und *Cosmo3D/OpenGL Optimizer* zusammen, indem es sich leicht in die Szenengraphstrukturen, die Hauptschleifen und das Event-Handling der unterschiedlichen APIs einbinden lässt und verschiedene Eingabegeräte unterstützt.

5.2.1 Abstraktionsschichten

Die Unabhängigkeit eines 3D-Menüs von den Szenengraph-APIs kann nur durch eine Abstraktionsschicht erreicht werden. Drei verschiedene Ansätze können zur Realisierung hierzu verfolgt werden:

- Für jede Szenengraph-API wird eine exklusiv Variante des 3D-Menüs implementiert
- Das 3D-Menü basiert auf *OpenGL*, dem kleinsten gemeinsamen Nenner der Szenengraph-APIs
- Eine einfache Szenengraphstruktur mit den nötigsten Grundfunktionen bildet eine Schnittstelle zwischen 3D-Menü und den Szenengraph-APIs

Im Falle der exklusiven Implementierung für jede Szenengraph-API sind durch den direkten Aufsatz auf die API-Schnittstelle hohe Bildwiederholungsraten zu erwarten. Jedoch erzeugt das synchrone Entwickeln für mehrere Schnittstellen einen unnötig hohen und vor allem redundanten Programmieraufwand. Zusätzlich müssen Erweiterungen konsequent in allen Versionen gepflegt werden, da ansonsten störende Variantenunterschiede auftreten können.

Ein direktes Aufsetzen auf *OpenGL* garantiert sowohl die Unabhängigkeit von den Szenengraph-APIs, als auch vom verwendeten Betriebssystem. Die Integration eines solchen Menüs würde über die *OpenGL*-Schnittstelle der APIs erfolgen, die jedoch nur zu einer unzureichenden Einbindung in die Szenengraphstruktur führt. Die Ausnutzung von vorhandenen, performanzsteigernden Algorithmen für die Menügeometrie ist somit nicht möglich. Zusätzlich ist die direkte *OpenGL*-Programmierung unkomfortabel und somit sehr aufwendig.

Im dritten Lösungsansatz bildet eine eigene kleine Szenengraphstruktur mit den nötigsten Grundfunktionen die Schnittstelle zwischen den Szenengraph-APIs und dem 3D-Menü. Die Menü-Objekte sind somit in die Graphstruktur integriert und trotzdem zum Großteil unabhängig von der verwendeten API. Vorteil dieses Ansatzes ist die leichte Erweiterung für alle Szenengraphen durch das Hinzufügen von unabhängigen Menü-Objekten. Die eindeutig definierte Schnittstelle erlaubt zusätzlich eine schnelle Portierung, da lediglich die kleine Szenengraphstruktur neu implementiert werden muss.

Die Forderungen nach Universalität und leichter Erweiterung führten in dieser Arbeit zur Verwendung des dritten Lösungsansatzes. Das entwickelte 3D-Menü besitzt, wie in Abbildung 5.8 skizziert, zwei Abstraktionsschichten, den *G3Scenegraph* und die *G3Components*. Der *G3Scenegraph* bildet die Schnittstelle zu der Szenengraph-API und ist daher in verschiedenen Varianten für *OpenInventor*, *Cosmo3D/OpenGL Optimizer* und *WorldToolKit* vorhanden. Die

einzelnen Objekte von *G3Scenegraph* besitzen die graphische Mindestfunktionalität, die für die Realisierung des 3D-Menüs erforderlich ist. Die Abstraktionsschicht *G3Components* hingegen besteht aus den einzelnen GUI-Elemente und setzt auf dem *G3Scenegraph* auf.

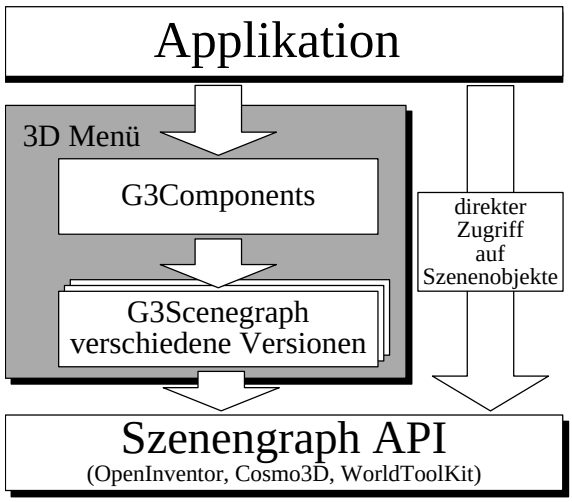

Abbildung 5.8: Die Abstraktionsschichten des *G3Menu*.

G3Menu wird in den Szenengraphen der Applikation eingehängt und von der Hauptschleife mit den Parametern der Eingabegeräten versorgt. Die Applikation kann, wie in Abbildung 5.8 dargestellt, sowohl auf das 3D-Menü zugreifen, als auch direkt den Szenengraphen manipulieren. Die Möglichkeit, das 3D-Menü nachträglich in Applikationen einzubauen, bleibt somit gewährleistet.

5.2.2 Szenengraph-Schnittstelle

Die *G3Scenegraph*-Schnittstelle stellt die Knoten bzw. Objekte für einen minimalen Szenengraphen zur Verfügung, mit dem sich dann 3D-Menüs kreieren lassen. Es wurden nur Knoten in die Schnittstelle aufgenommen, die für ein 3D-Menü unerlässlich sind. Globale Einstellungen, wie beispielsweise die Beleuchtung, Kameraeinstellungen etc., sind Aufgabe der Applikation und werden vom *G3Scenegraph* nicht beachtet. Die Anzahl der Objekttypen bleibt somit sehr gering und es entstand ein sehr kompakter Standard der sich leicht auf andere Szenengraph-APIs übertragen lässt.

Die *G3Scenegraph*-Abstraktionsschicht wurde als objektorientierte Programmierbibliothek in C++ für die drei Szenengraph-APIs realisiert. Die drei Derivate des *G3Scenegraph* bilden nach oben zu den *G3Components* eine einheitliche Schnittstelle und setzen nach unten direkt auf den Szenengraph-APIs auf.

In Abbildung 5.9 ist die Klassenhierarchie des *G3Scenegraph* dargestellt. Die Vererbung von Eigenschaften bei der Abarbeitung eines *G3Scenegraph* erfolgt, ähnlich wie bei *Cosmo3D*, nur an die Kinder und nicht an gleichgestellte Knoten. Der Szenengraph bleibt so übersichtlich und

flexibel. Szenengraph-APIs wie beispielsweise *OpenInventor* können eine andere Vererbungsreihenfolge besitzen. In diesem Fall musste durch das Hinzufügen von Separator-Knoten auf die Einhaltung der Vorgaben geachtet werden.

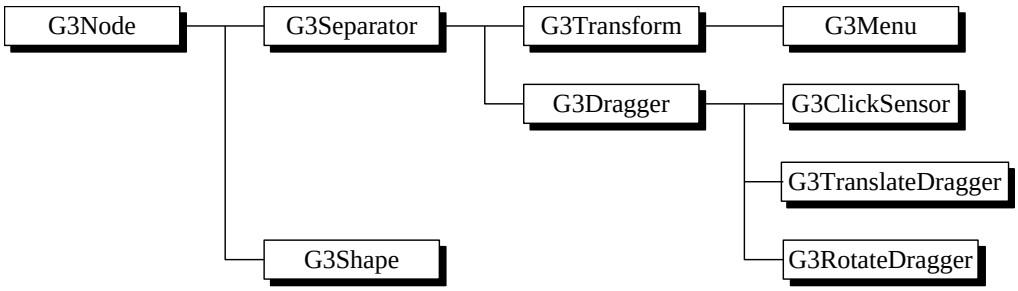

Abbildung 5.9: Die Klassenhierarchie der Abstraktionsschicht *G3Scenegraph*.

Die abstrakte Klasse *G3Node* bildet die Basis für die Gruppen-, Geometrie- und Interaktionsknoten des *G3Scenegraph*. Direkt davon abgeleitet wird der allgemeine Gruppenknoten *G3Separator*, dem alle Knotentypen als Kinder zugeordnet werden können. In den verwendeten Szenengraph-APIs unterstützen die Separator-Knoten häufig Caching und Culling Funktionalitäten, die zu einer Beschleunigung der Darstellung führen. Zusätzliche Methoden zur Umpositionierung besitzt der vom *G3Separator* abgeleitete Transformationsknoten *G3Transform*. Mit ihm kann ein Teilbaum des *G3Scenegraph* im lokalen Koordinatensystem transliert, rotiert und skaliert werden. Der *G3Transform* ist besonders für die Gruppierung und Anordnung von Untermenüs wichtig.

Das Bindeglied zwischen der Applikation und dem *G3Menu* bildet der *G3Menu*-Knoten, der als Wurzelknoten des *G3Scenegraph* eine Sonderstellung einnimmt. Seine Ableitung vom *G3Transform* erlaubt die Transformation des gesamten 3D-Menüs und als Bindeglied zur Applikation nimmt der *G3Menu*-Knoten alle auftretenden Events über die Schnittstelle entgegen und verarbeitet sie.

Interaktions-Events löst der Anwender durch einen virtuellen Degen aus, mit dem er auf die Objekte der virtuellen Umgebung zeigt. Die Positionierung des Degens erfolgt in einer CAVE beispielsweise durch ein Tracking-System, während an einem Arbeitsplatz die Richtung des Degens durch die Blickrichtung der virtuellen Kamera und die Position der 2D-Maus definiert wird. Die Verwendung des Deutungsstrahls erlaubt so eine geräteunabhängige Schnittstelle für die Event-Weitergabe zwischen der Applikation und dem *G3Menu*-Knoten. Es wird zwischen den drei Eventtypen „Taste gedrückt“ (*Press-Event*), „Eingabegerät wird bei gedrückter Taste bewegt“ (*Down-Event*) und „Taste wurde losgelassen“ (*Release-Event*) unterschieden.

In Abbildung 5.10 ist die Event-Verarbeitung des *G3Menu*-Knoten skizziert. Auftretende Events der Hauptschleife werden zuerst an den *G3Menu*-Knoten weitergeleitet. Nur wenn das *G3Menu* nicht von dem Event betroffen wird, tritt das Event-Handling der Applikation in Aktion. Wird eine Komponente des *G3Scenegraph* bei Auftreten eines *Press-Events* vom Degen geschnitten, geht der *G3Menu*-Knoten in den aktiven Zustand (menuActive = true) über. Die Überprüfung des *Press-Events* erfolgt durch die Methode *buttonPressed()*. Im aktiven Zustand

erwartet der G3Menu-Knoten nur *Down*- und *Release-Events*. Ein Ziehen des Anwenders an den Menükomponenten bei gedrückten Knopf erzeugt eine Reihe von *Down-Events*, die der G3Menu-Knoten kombiniert mit der Strahlinformation über die Methode *buttonDown()* an den entsprechenden Interaktionsknoten weiterleitet. Bei Auftreten eines *Release-Events* kehrt der G3Menu-Knoten nach Aufruf der Methode *buttonReleased()* in den inaktiven Zustand (menuActive = false) zurück.

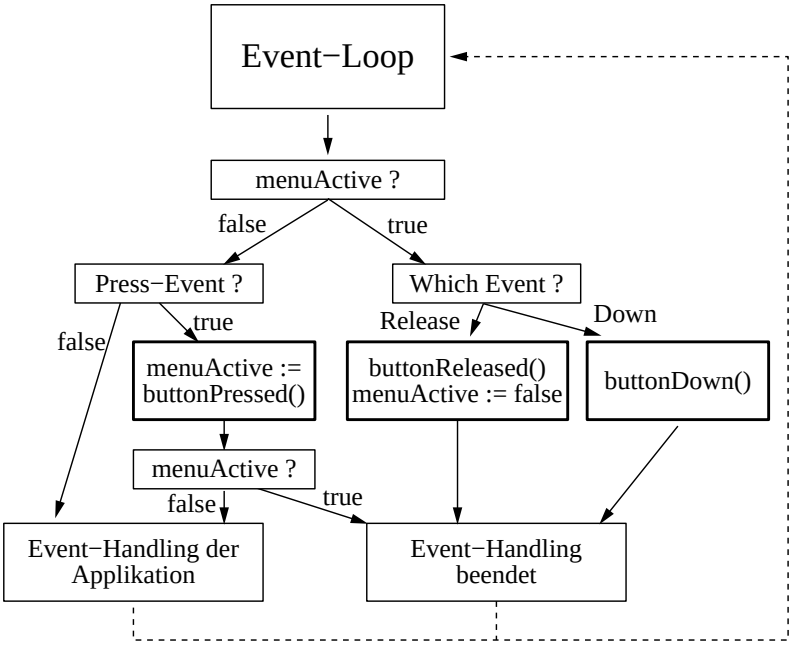

Abbildung 5.10: Die Verarbeitung von Eingabeereignissen im G3Menu-Knoten.

Die Events werden von den drei Szenengraph-APIs in unterschiedlichen Datenstrukturen gespeichert, deshalb besitzt der *G3Scenegraph* an dieser Stelle eine nicht einheitliche Schnittstelle. Für die APIs *Cosmo3D* und *WorldToolKit* unterscheiden sich die Übergabeparameter der Methoden und die *OpenInventor*-Version kommt sogar ohne die Methoden *buttonPressed*, *buttonDown* und *buttonReleased* aus, da der *OpenInventor* das Event-Handling intern abarbeitet.

Die Beschreibung sichtbarer Objekte setzt sich zusammen aus den Definitionen der Geometrie, der Texturierung und den Materialeigenschaften. Zur besseren Datenhaltung erfolgen die Definitionen in verschiedenen Objekt-Klassen und werden letztlich unter einem *G3Shape*-Knoten zusammengefasst.

Der *G3Geometry*-Knoten bietet die Definition der Objektoberfläche als einen Standardkörper (Würfel, Kugel, Zylinder, Kegel) oder als ein beliebig indiziertes Polygonnetz an. Die Materialeigenschaften für Transparenz und Farbeffekte (ambiente, diffuse, spekulare und emissive) werden in dem *G3Material*-Knoten definiert.

Texturen werden in *G3Scenegraph* zur Beschriftung der Buttons verwendet. Um einen Button mit einem Icon zu versehen, können bereits erstellte RGB-Bilder in ein *G3Texture*-Objekt geladen werden. Häufiger ist jedoch die Verwendung eines Schriftzuges auf den GUI-Elementen.

G3Texture kann hierfür mit Hilfe der *FreeType*-Algorithmen Zeichen in eine Textur rendern und auf Buttons kleben. So lassen sich die Menüelemente individuell und ohne Performanzverlust erstellen.

Die Dragger-Knoten bilden als Interaktionselemente des *G3Scenegraph* neben den Geometrie- und Gruppenknoten den dritten Knotentypen. Nach der Auswahl durch den Benutzer gehen sie in den aktiven Zustand über und nehmen solange Eingaben entgegen, bis sie durch das nächste Release-Event in den inaktiven Zustand zurückversetzt werden. Zur Markierung des aktuellen Zustandes besitzen die Dragger-Knoten zwei entsprechende *G3Geometry*-Knoten. Dragger eignen sich zur Eingabe von Skalarwerten oder zur Auslösung eines „Button-Click“ Events. Callback-Funktionen leiten die Eingabewerte an die entsprechenden Teile der Applikation weiter. Die drei auf der Basisklasse *G3Dragger* basierenden Typen *G3ClickSensor*, *G3TranslatorDragger* und *G3RotationDragger* wurden in *G3Scenegraph* realisiert.

Der *G3ClickSensor* ist die einfachste Form eines Interaktionselementes und dient der Erstellung von einfachen Knöpfen und Tastern zur Auslösung eines „Button-Click“ Events.

Der Translationsdragger *G3TranslateDragger* hingegen lässt sich durch Tastendruck aktivieren und bei gedrückter Taste entlang der X-Achse im lokalen Koordinatensystem verschieben. Die Callback-Funktion wird bei jeder Positionsänderung aufgerufen und liefert den aktuellen X-Wert. Die aktuelle Translation des Draggers kann über Methoden abgefragt oder gesetzt werden. Damit der Dragger im Geltungsbereich des Menüs bleibt, lässt sich der Bewegungsbereich einschränken. Besonders gut geeignet ist der *G3TranslateDragger* für die Erstellung von Schiebereglern.

Der Rotationsdragger *G3RotateDragger* lässt sich durch Benutzerinteraktion um seine lokale Z-Achse drehen. Sobald ein *G3RotateDragger* im aktiven Zustand gedreht wird, leitet er den Winkel der Rotation durch Aufruf der Callback-Funktion an die Applikation weiter. Beispielsweise lassen sich Drehknöpfe durch *G3RotateDragger* realisieren.

Die vorgestellten Objekte reichen aus, um darauf basierend die *G3Components*-Abstraktionsschicht aufzubauen.

5.2.3 Menükomponenten

Die zweite Abstraktionsschicht *G3Components*, siehe Abbildung 5.8, setzt als objektorientierte Programmierbibliothek auf dem *G3Scenegraph* auf und ermöglicht die Definition von Interaktionselementen und deren Zusammenstellung in einer graphischen Benutzerschnittstelle.

Für das Erscheinungsbild der Interaktionselemente standen in vielen Fällen zweidimensionale Beispiele Pate, mit deren Aussehen und Funktionalität der Anwender bereits vertraut ist und die er deshalb intuitiv bedienen kann. Neuartige dreidimensionale Komponenten wurden für komplexere Aufgaben entworfen. Beim Design der einzelnen Geometrieelemente wurde die Komplexität so gering wie möglich gehalten, um die Graphikperformanz nicht durch unnötige Polygone zu belasten. Beschriftet werden die einzelnen Interaktionselemente mit *G3Texture* durch automatisch generierte Texturen aus Textzeichen. Der direkte Vergleich in Abbildung 5.11 verdeutlicht an einem Beispielmenü die geringe Anzahl der benötigten Polygone bei der Verwendung von Texturen, die auf modernen Graphiksystemen ohne Performanzverlust dargestellt werden können.

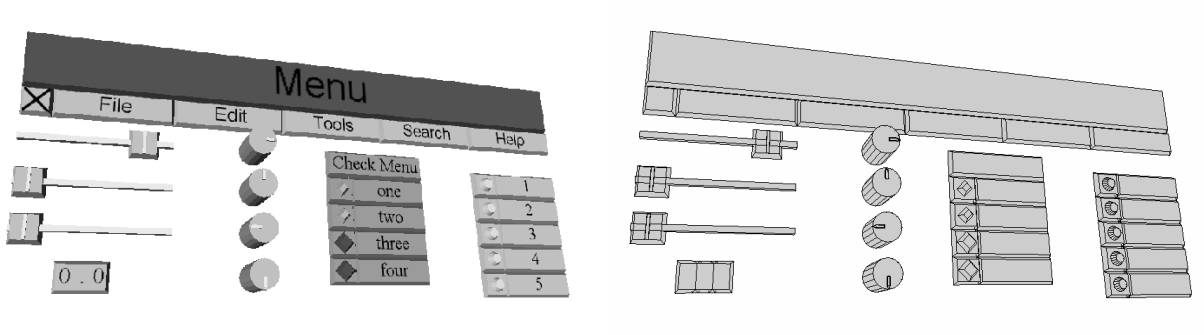

Abbildung 5.11: Vergleich eines *G3Menu* in texturierter und Drahtgitterdarstellung.

Die Elemente der *G3Components*-Abstraktionsschicht bilden eine Erweiterung der *G3Scenegraph*-Struktur und werden zur Integration in eine Applikation unterhalb des *G3Menu*-Knoten angeordnet. Entsprechend der Vererbungsstruktur aus Abbildung 5.12 untergliedern sich die Aufgaben der Komponenten ausgehend von der Basisklasse *G3Component* in Anzeige-, Interaktions- und Gruppenknoten. Die Basisklasse besitzt einen *G3Transform*-Knoten, unter den sich die vom *G3Dragger*- und *G3Shape*-Knoten abgeleiteten Klassen entsprechend ihrer Funktionalität anordnen. Der Transformationsknoten dient zur Definition eines lokalen Koordinatensystems für die Anordnungen der Geometrien einer Menükomponente und zur Positionierung im *G3Scenegraph*.

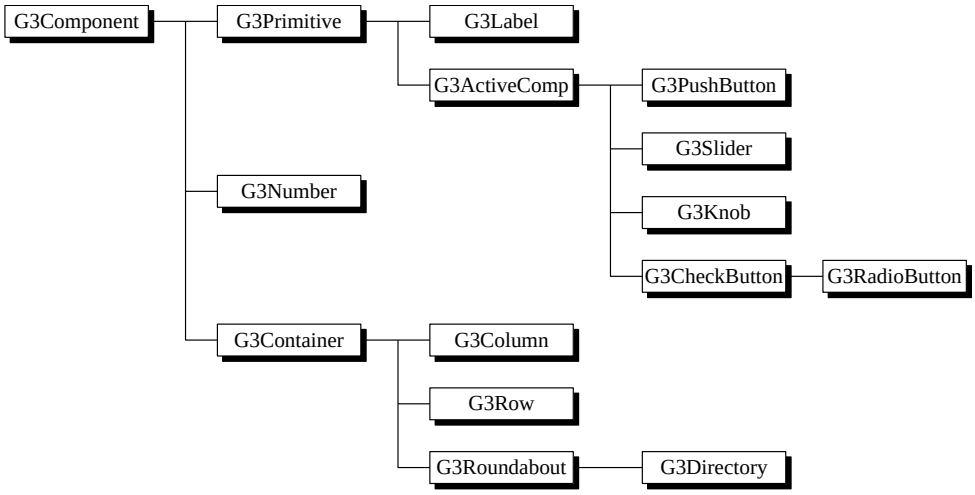

Abbildung 5.12: Die Vererbungshierarchie der 3D-Menü-Komponenten.

Die beiden Anzeige-Komponenten *G3Number* und *G3Label* dienen zur Darstellung von Zeichenkolonnen und bestehen aus einem *G3Shape* in Kombination mit einer *G3Texture*. Mit *G3Label* können lediglich statische Texte abgebildet werden, während sich die Textur von *G3Number* zur Darstellung von aktuellen Parametern zur Laufzeit ändern lässt. Anzeige-

Komponenten werden an vielen Stellen innerhalb eines 3D-Menüs benötigt.

Auswahlereignisse und Parametereinstellungen kann der Anwender über die Interaktionskomponenten vornehmen, die auf den *G3Draggern* basieren. Die Interaktionskomponenten verändern bei Benutzereingaben ihre Geometrien und rufen die gesetzte Callback-Funktion auf. *G3ActiveComp* ist die Basisklasse der unterschiedlichen Interaktionselemente.

Die Tasterfunktion der *G3Button*-Komponente ist die einfachste Form der Eingabe und wird beispielsweise für die Menüleiste benötigt (siehe Abbildung 5.13). Die Taster-Geometrie wird bei der Anwahl leicht nach hinten verschoben und die Oberflächenfarbe aufgehellt, um den Zustand „Angewählt“ darzustellen. Beim Loslassen der Komponente nimmt sie ihren ursprünglichen Zustand ein und ruft die Callback-Funktion auf.

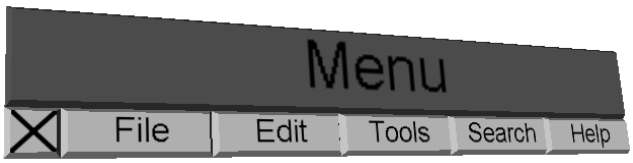

Abbildung 5.13: Eine aus *G3Buttons* zusammengesetzte Menüleiste.

Zustandsparameter kann der Anwender mit Hilfe der beiden binären Optionslisten *G3Checkbutton* und *G3Radiobutton* beeinflussen. Die Elemente der Optionslisten können durch Auswahl markiert werden und stellen ihren Zustand durch ein rotes Quadrat bzw. einen roten Kreis dar (siehe Abbildung 5.14). Aus der *G3Checkbutton*-Liste können mehrere Elemente angewählt werden, während von *G3Radiobutton* immer nur eines markiert werden kann.

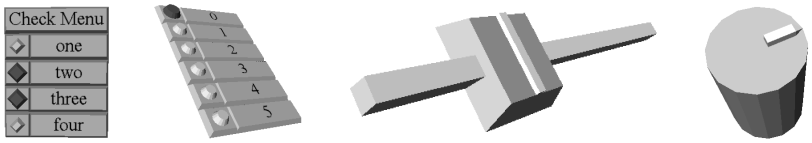

Abbildung 5.14: Die verschiedenen Interaktionselemente. Vlnr.: *CheckButton*, *Radio Button*, *Slider* und *Knob*

Die Interaktionskomponenten *G3Slider* und *G3Knob* dienen zur Definition von Zahlen. Der Schieberegler *G3Slider* erlaubt durch Translation des Knopfes Einstellungen in einem eingeschränkten Bereich, während der Drehknopf durch Rotation und ohne Bereichsgrenzen einen Parameter definiert (siehe Abbildung 5.14). Die Callback-Funktion wird bei jeder Translation bzw. Rotation aktiviert. In Kombination mit einem *G3Label* und einer *G3Number* können die Elemente beschriftet und der aktuelle Parameter dargestellt werden.

Die Aufgabe der *G3Container*-Objekte ist die Anordnung und Proportionierung von zugewiesenen Kind-Komponenten zur Erstellung komplexer Menüstrukturen. Als Kinder eines *G3Containers* sind alle Klasse von *G3Components*, also auch *G3Container*, zulässig.

Die Container-Klassen *G3Column* und *G3Row* ordnen ihre Kind-Komponenten vertikal bzw. horizontal zueinander an. Kombinationen der beiden Klassen erlauben die Anordnung von Komponenten in einer Ebene und somit die Konstruktion von beliebig komplexen Menüs, wie in Abbildung 5.11 an einem Beispielmenü demonstriert.

Die Container-Komponente *G3Rondell* kann es nur im dreidimensionalen Raum geben, da die Kind-Komponenten auf der Mantelfläche eines Zylinders angeordnet sind (siehe Abbildung 5.15). Die Mantelfläche des Zylinders besteht aus mehreren gleich großen Rechtecken, deren Anzahl und Größe automatisch durch die zugeordneten Kinder definiert wird. Die Drehachse des Rondells kann sowohl horizontal als auch vertikal liegen. Der Wechsel in ein Untermenü erfolgt durch Drehen des Rondells und unterscheidet sich so grundlegend von zweidimensionalen Techniken. Der Anwender hat so einen besseren Überblick über die Untermenüs und kann schneller und gezielter wechseln.

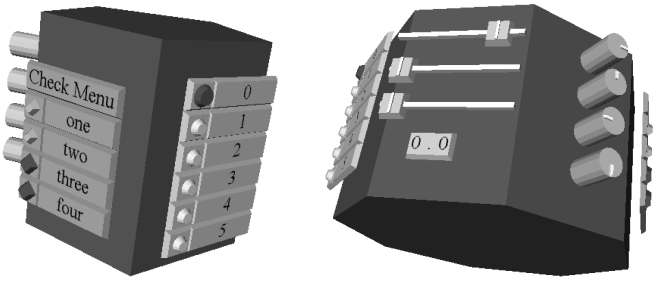

Abbildung 5.15: Anordnung von mehreren Untermenüs auf einem dreidimensionalen Rondell.

Die Selektion einer Datei ist eine wichtige Funktionalität, die auch in immersiven Umgebungen zur Verfügung stehen muss. Da die Adaption von zweidimensionalen Dateibrowsern zu keiner praktikablen Lösung führt, wurde in dieser Arbeit die dreidimensionale Interaktionskomponente *G3Directory* entwickelt. In ihr wird ein Verzeichnisbaum durch mehrere horizontale Rondells dargestellt (siehe Abbildung 5.16). Dateien werden durch Taster-Komponenten und Unterverzeichnisse durch Radio-Buttons repräsentiert. Um ein Verzeichnis zu durchsuchen, dreht der Anwender das entsprechende Rondell und kann durch Drücken eines Buttons eine

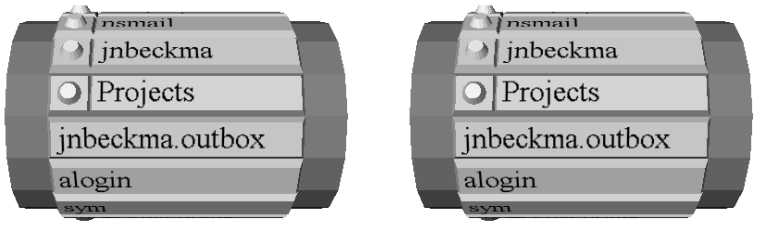

Abbildung 5.16: Die Darstellung eines einzelnen (links) und mehrerer Verzeichnisse in der *G3Directory*-Komponente.

Datei auswählen, deren Dateiname über die Callback-Funktion an die Applikation weitergeleitet wird. Betätigt der Benutzer einen Radio-Button, wird das entsprechende Unterverzeichnis geöffnet und rechts als weiteres Rondell dargestellt (siehe Abbildung 5.16 rechts). Zum Wechsel in ein anderes Unterverzeichnis, wird das Rondell gelöscht und durch ein neues ersetzt. Die *G3Directory*-Komponente kommt mit den minimalen Eingabemöglichkeiten einer immersiven Umgebung aus und ist somit auch für den Einsatz in einer CAVE geeignet.

5.2.4 Fazit

Das entwickelte 3D-Menü kann in Applikationen der drei Szenengraph-APIs *OpenInventor*, *Cosmo3D/OpenGL Optimizer* und *WorldToolKit* einfach und schnell integriert werden. Das angewandte Modell der Abstraktionsschichten garantiert eine leichte Erweiterung des 3D-Menüs auf andere Szenengraph-APIs und erlaubt auch das schnelle Hinzufügen von neuen Menükomponenten für alle unterstützten Szenengraphen. Der Einsatz eines unabhängigen Szenengraphen garantiert die System- und API-unabhängige Entwicklung von 3D-Menüs. Der bereitgestellte Grundstock an *G3Components* reicht bereits aus, um für vielfältige virtuelle Umgebungen eine dreidimensionale Steuerung zu implementieren.

Der Umgang mit den Systemressourcen ist ein wichtiger Punkt für die Akzeptanz beim Benutzer. Die Anzahl der Polygone des Menüs muss so gering wie möglich gehalten werden, da viele virtuelle Umgebungen die Graphikhardware bereits bis zur Leistungsgrenze ausnutzen. Eine einzelne *G3Menu* Komponente benötigt zwischen 6 (*G3Button*) und 30 (*G3Radio*) Polygone und ein durchschnittliches Menü somit deutlich unter 1000 Polygone. Die geringe Polygonanzahl wurde allerdings nur durch die Verwendung von Texturen zur Textdarstellung ermöglicht.

Die Graphikperformanz wird durch ein *G3Menu* somit nicht übermäßig belastet und gestattet den Einsatz auch in vollimmersiven Hardware-Umgebungen, zumal *G3Menu* auch mit VR-Eingabegeräten bedient werden kann. Praktisch eingesetzt wurden *G3Menus* in den beiden prototypischen Visualisierungssystemen dieser Arbeit.

Kapitel 6

Interaktive Visualisierung von geometriebezogenen Simulationen

Geometriebezogene Finite-Element-Simulationen stützen bereits in vielen Bereichen den Fahrzeugentwicklungsprozess und werden produktiv zur Absicherung von Crashtest-, Schwingungs-, Kinematik- und Blechumformuntersuchungen eingesetzt. Die Entwicklung neuer Graphikhardware kann jedoch nicht mit der steigenden Komplexität von Finite-Element-Modellen mithalten. Neue Visualisierungstechniken sollen daher das Auffinden von Schwachstellen und Verbesserungsmöglichkeiten erleichtern und den Auswertungsaufwand für die Berechnungsingenieure senken.

Die Aufbereitung der Berechnungsergebnisse in einer virtuellen Umgebung ist ein vielversprechender Ansatz, die Analyse zu erleichtern. Der Einsatz moderner VR-Eingabegeräte und immersiver Darstellungstechniken in virtuellen Umgebungen bietet im Gegensatz zu den konventionellen Postprocessing-Systemen nicht nur in aufwendigen und kostspieligen Laborlösungen mehr Interaktivität, sondern auch am VR-unterstützten Arbeitsplatz.

6.1 Visualisierungssystem *VtCrash*

In dieser Arbeit entstand das Programmpaket *VtCrash* zur Visualisierung von Berechnungsergebnissen geometriebezogener Simulationen in einer virtuellen Umgebung zur Unterstützung von vielfältigen Visualisierungsszenarien. Das Visualisierungssystem spaltet sich in einen High-End-VR- und einen *VRML*-basierten Teilbereich auf, um den Anforderungsprofilen der Visualisierungsszenarien gerecht zu werden. Die wichtigsten Strukturelemente von *VtCrash* sind in Abbildung 6.1 dargestellt.

Der High-End-VR-Teil besitzt Laderoutinen für die Dateiformate von *PAM-CRASH* (Crashtest), *NASTRAN* (Schwingung) und *MARC* (Kinematik), die den direkten Zugriff auf Ergebnisdaten ohne Zwischenspeicherung sicherstellen. Mit Hilfe einer Objektstruktur für effizientes Datenmanagement werden die einzelnen Geometrien im Ladevorgang optimiert und in einen *WorldToolKit*-Szenenbaum eingehängt. Modelle mit bis zu 200.000 Finiten-Elementen und 40 Zeitschritten können so in der virtuellen Welt visualisiert werden. Eine Reihe von Interaktions-

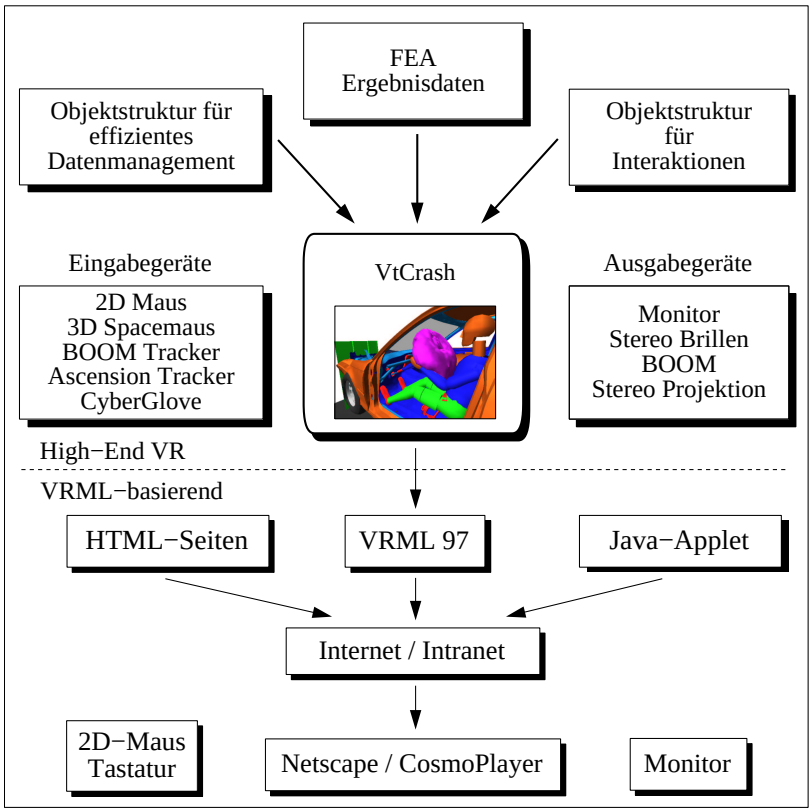

Abbildung 6.1: Systemstruktur des Visualisierungssystems *VtCrash*.

mechanismen zur Manipulation des Szenenbaumes werden von einer weiteren Objektstruktur bereitgestellt.

Die unterstützten Ein- und Ausgabegeräte erlauben den Einsatz von *VtCrash* als Arbeitsplatz- und VR-Laborwerkzeug. Am Arbeitsplatz wird *VtCrash* in einer „Fish-Tank“ Umgebung mit Stereodarstellung und 2D-Maus bzw. Spacemouse eingesetzt. Im VR-Labor werden sowohl immersive Lösungen, wie beispielsweise *Fakespace BOOM* und *CAVE*, als auch großflächige Stereoprojektionen unterstützt. Zufriedenstellendes Arbeiten ist mit dem High-End-VR-Teilbereich nur auf leistungsstarken Workstations, wie beispielsweise einer *SGI Octane* möglich.

Die Zielplattform des *VRML*-basierten Teilbereichs sind ausreichend ausgestattete Arbeitsplatz-PC’s, deren durchschnittliche Graphikleistung die Visualisierung von Gesamtmodellen mit 200.000 Polygonen und mehreren Zeitschritten nicht zulässt. Es bietet sich daher die Beschränkung auf Teilmodelle der Simulationsergebnisse an. In der virtuellen Umgebung von *VtCrash* können interessante Teilbereiche als *VRML97*-Datei gespeichert und anschließend auf HTML-Seiten kombiniert mit *JavaScript* in Internet-Browsern wie *Netscape* oder *Explorer* betrachtet werden. *JavaScript* bietet dem Anwender ähnliche Interaktionsmechanismen im *VRML*-Browser, wie sie der High-End-VR-Teilbereich besitzt.

6.2 Datenfluss

Ein Finite-Element-Modell besteht zur Beschreibung der Fahrzeuggeometrien aus den verschiedenen Elementtypen Beams, Shells und Solids. Die verschiedenen Elemente werden zur effizienten Speicherung in den Ergebnisdateien unsortiert in Konnektivitäts-, Knoten- und Parameterfeldern abgelegt (siehe Abbbildung 6.2 oben).

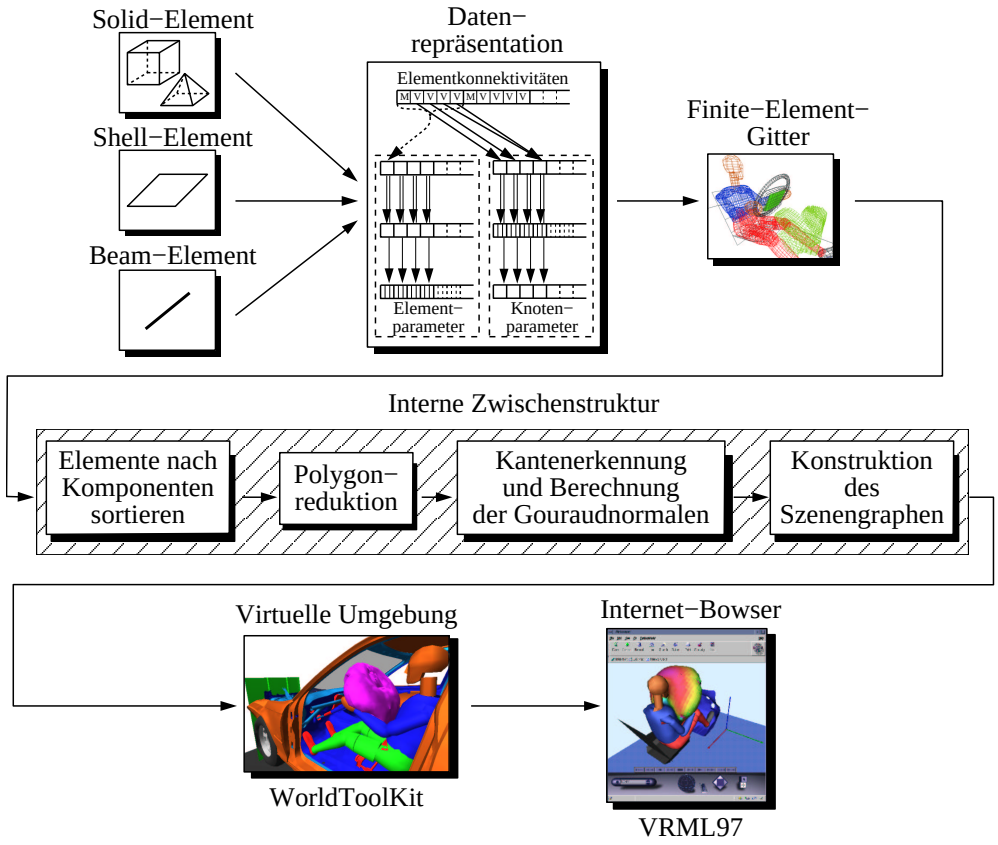

Abbildung 6.2: Transformation des Finite-Element-Modells über die virtuelle Umgebung in eine *VRML97*-Datei.

Im Datenfluss von *VtCrash* (siehe Abbildung 6.2) werden die Simulationsergebnisse schrittweise für eine interaktive Visualisierung aufbereitet und optimiert. Im ersten Schritt des Ladevorganges werden die Finiten-Elemente nach Fahrzeugkomponenten sortiert und in Polygone umgewandelt. Die eindimensionalen Beam- und die zweidimensionalen Shellelemente können direkt in polygonale Darstellungen konvertiert werden, wohingegen die Solid-Materialien eine Reduktion auf ihre äußere Hülle erfordern. Die generierten Polygone werden von *VtCrash* in einer internen Zwischenstruktur gespeichert.

Die Polygonanzahl eines durchschnittlichen Fahrzeugmodells ist zu groß für eine interaktive Visualisierung. Die Zwischenstruktur bietet deshalb im nächsten Schritt eine Reduktion der Geometriekomplexität durch das Zusammenfassen von Polygonen an (siehe Abbildung 6.3). Re-

duktionskriterium ist hierbei der Winkel zwischen den Polygonnormalen mit Beachtung der zeitlichen Geometrieveränderungen, um im Laufe der Simulation entstehende Kanten zu erkennen. Optimal wäre eine separate Polygonreduktion in jedem Zeitschritt, die allerdings einen hohen Zeitaufwand im Ladevorgang erzeugen würde. Eine Alternative für ausreichend genaue Ergebnisse ist es, die Polygonreduktion auf den zwei extremen Geometriezuständen, erster (undeformierter) und letzter (maximal deformierter) Zeitschritt, durchzuführen und ein ausgedünntes Gitter für alle Zeitschritte zu erzeugen. Eine solche Polygonausdünnung (siehe Abbildung 6.3) erzielt in der Darstellung höhere Bildwiederholungsraten und reduziert den Bedarf an Arbeitsspeicher.

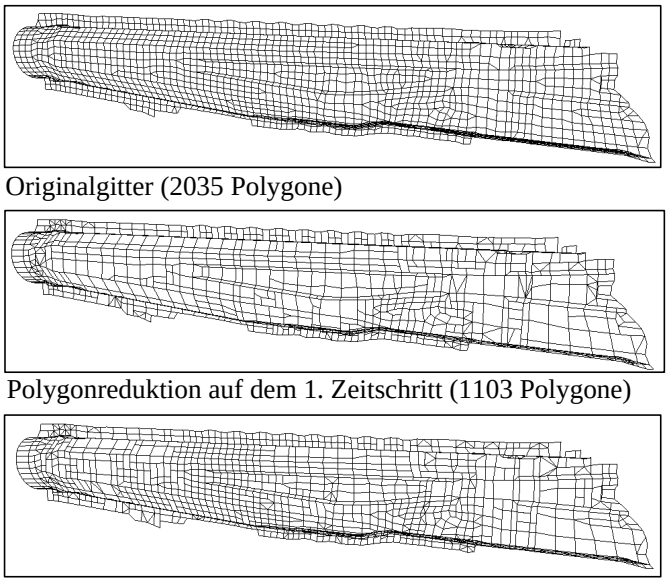

Abbildung 6.3: Anwendung des Reduktionsalgorithmus am Beispiel des Motorträgers.

Im folgenden Schritt wird das Flat- oder Gouraud-Shading der Fahrzeuggeometrie vorbereitet. Flat-Shading besitzt pro Polygon eine Normale und die Oberfläche weist somit viele Kanten auf. Das Gouraud-Shading hingegen erfordert die Definition einer Normalen an jedem Polygonknoten und erzeugt den Eindruck von geglätteten Oberflächen. Die Gouraud-Normalen in den Eckpunkten werden durch Mittelung der Normalen der angrenzenden Polygone berechnet. Allerdings entfernt die Mittelung bei Nichtbeachtung eines Grenzwinkels sämtliche Kanten aus einer Geometrie (siehe Abbildung 6.4). Während der Berechnung von Gouraud-Normalen wird daher in *VtCrash* der Winkel zwischen den Polygonnormalen beachtet und bei Überschreiten des Grenzwertes erzeugt das Hinzufügen von zusätzlichen Knoten und Normalen eine sichtbare Kante, wie in Abbildung 6.5 skizziert. Das Gouraud-Shading erzeugt in Kombination mit der Kantenerkennung eine glatte Oberfläche mit den erwünschten Kanten.

In der Ergebnisdatei sind für jedes einzelne Finite-Element simulierte skalare Größen gespeichert. Die Abbildung einer skalaren Größe mittels einer Farbtabelle kann zur Einfärbung der

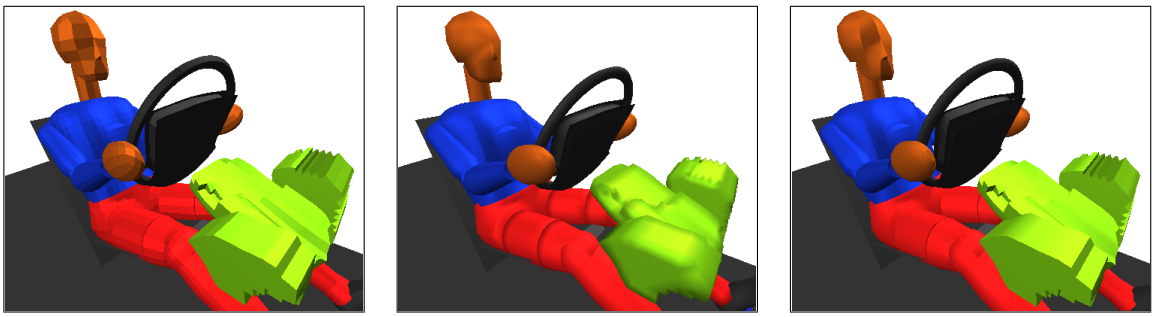

Abbildung 6.4: Vergleich der verschiedenen Shading-Verfahren: links Flat-Shading, Mitte Gouraud-Shading, rechts Gouraud-Shading mit Kantenerkennung.

Fahrzeuggeometrie verwendet werden. Die Zwischenstruktur besitzt ein zusätzliches Feld, in das eine skalare Größe geladen wird.

Aus der Zwischenstruktur kann nun der *WorldToolKit* Szenengraph erstellt und in der virtuellen Umgebung dargestellt werden. Im interaktiven Betrieb lassen sich durch Separation Teilmodelle des Gesamtfahrzeuges als *VRML97*-Datei speichern.

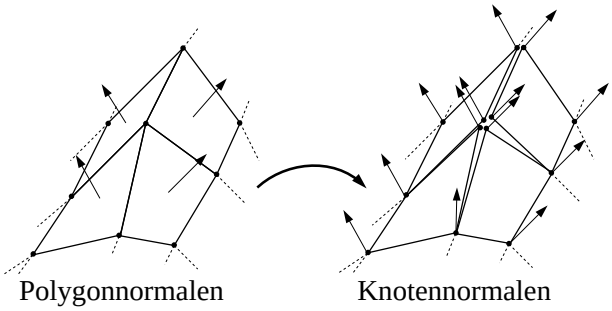

Abbildung 6.5: Kantenerkennung für das Gouraud-Shading durch das Hinzufügen von Knotennormalen.

6.3 Szenengraphstruktur

Die Struktur des Szenengraphen ist entscheidend für die Performanz und Flexibilität der virtuellen Umgebung und muss den Interaktionsanforderungen entsprechend ausgelegt werden. *WorldToolKit* ist eine leicht zu programmierende Szenengraph-API, die aufgrund ihrer hohen Zahl an Gerätetreibern als Basis für *VtCrash* gewählt wurde.

Die Visualisierung der Simulationsergebnisse erfordert die zeitabhängige Darstellung der Fahrzeuggeometrie, die sich in Fahrzeugkomponenten und Bauteile unterteilt. Abbildung 6.6 skizziert am Beispiel einer Tür den aus der *VtCrash* Zwischenstruktur aufgebauten Szenengraphen. Der zentrale *Switch*-Knoten besitzt in jedem seiner Kind-Knoten einen Teilgraphen mit der

vollständigen Fahrzeuggeometrie eines Zeitschrittes und animiert durch Umschalten des sichtbaren Kind-Knotens die Szene. Unterhalb des *Switch*-Knotens werden die Gruppen-Knoten der Fahrzeugkomponenten angeordnet, die wiederum die Bauteile als Geometrie-Knoten besitzen.

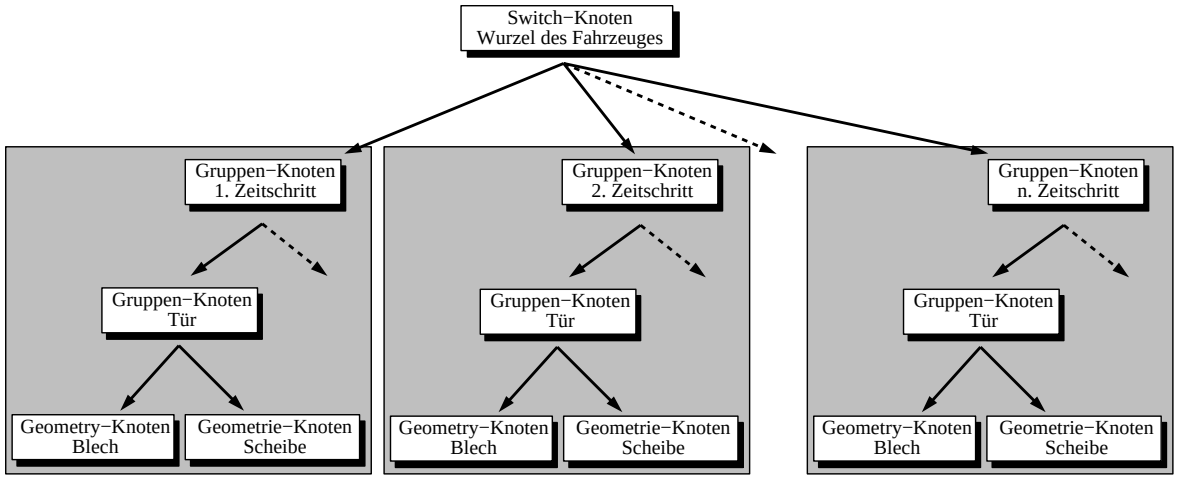

Abbildung 6.6: Der *WorldToolKit* Szenengraph von *VtCrash* am Beispiel einer Tür.

Die Interaktionen, wie beispielsweise das Picken und Bewegen von Komponenten oder die Berechnung von Geometrieschnitten, sollen für alle Zeitschritte erfolgen und müssen somit Teilgraph-übergreifend durchgeführt werden. Das Fehlen einer zeitschrittunabhängigen Zuordnung der Geometrien erschwert allerdings die Interaktion mit den einzelnen Fahrzeugkomponenten. Die Bereitstellung einer objektorientierten Interaktionsstruktur behebt in *VtCrash* dieses Manko durch erneutes Abbilden der Bauteilgruppen-Knoten mit Referenzen auf die Geometrien der einzelnen Zeitschritte des Szenengraphen (siehe Abbildung 6.7). Die Algorithmen der Interaktionsobjekte können so zeitschrittübergreifende Manipulationen an den Gruppen- und Geometrie-Knoten einer Fahrzeugkomponente oder eines Bauteils vornehmen.

Die starke Kapselung der *WorldToolKit*-Objekte erlaubt eine vergleichsweise einfache Erstellung von Szenengraphen, jedoch erwies sich im Laufe der Arbeit die fehlende Flexibilität als großes Hindernis. Das Fehlen eines direkten Zugriffs auf die Knoten- und Polygonlisten erhöhte den Aufwand zur Realisierung von Interaktionsmechanismen. So konnten Manipulationen an erstellten Geometrien nur langsam durch viele *WorldToolKit*-Funktionsaufrufe durchgeführt werden. Ein optimierter Szenengraph konnte in *WorldToolKit* aufgrund fehlender Indexfelder, Triangle-Strips und Mehrfachreferenzen auf Parameterfelder nicht erstellt werden.

6.4 Interaktionen

Der Schritt von der Betrachtung einer Szene hin zur Analyse erfolgt durch den Einsatz von Interaktionsmechanismen. Sie erlauben dem Anwender, direkt mit den virtuellen Objekten zu interagieren und ermöglichen so eine intuitive Auswertung der Simulationsergebnisse. Die folgenden Interaktionsmechanismen wurden im High-End-VR-Teilbereich von *VtCrash* realisiert.

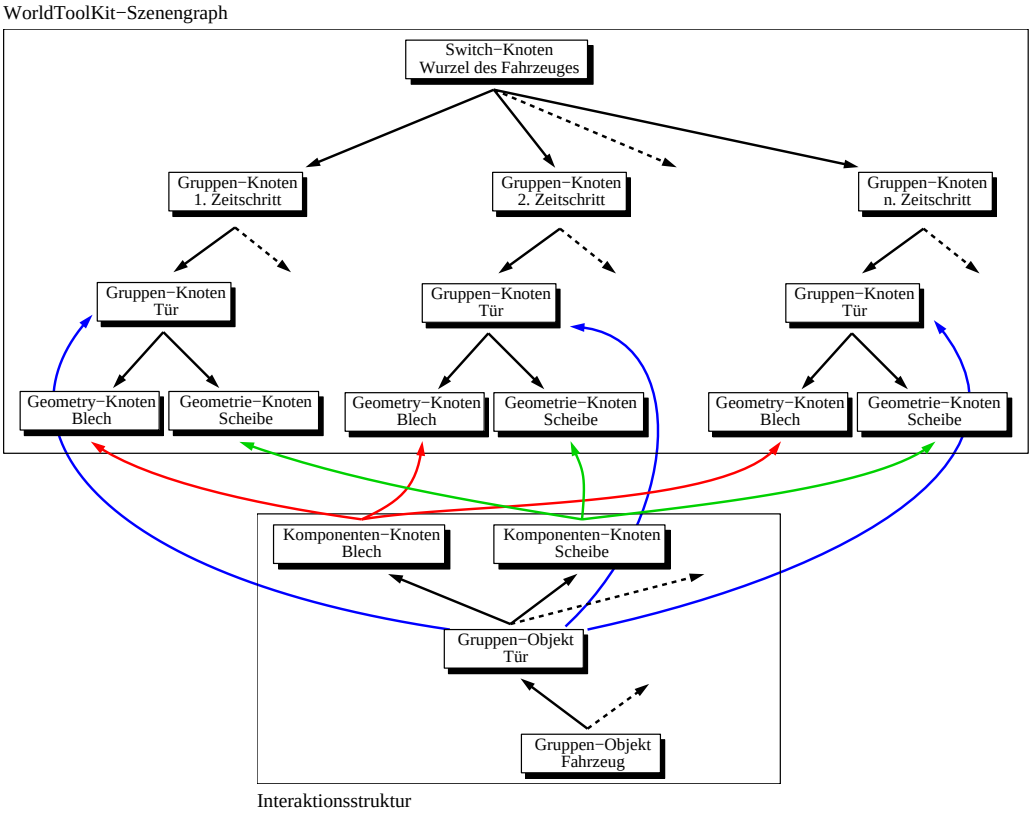

Abbildung 6.7: Das Zusammenspiel des *WorldToolKit* Szenengraphen und der Interaktionsstruktur von *VtCrash*.

6.4.1 Picken und Bewegen von animierten Fahrzeugkomponenten

In einer virtuellen Umgebung erfolgt die Auswahl von Fahrzeugkomponenten, anders als in konventionellen Postprocessing-Systemen nicht über 2D-Menüs, sondern direkt über die dreidimensionale graphische Darstellung. Am Arbeitsplatz werden Fahrzeugkomponenten mit der 2D-Maus ausgewählt, wohingegen in einer immersiven Umgebung die Auswahl durch Schneiden mit einem virtuellen Degen erfolgt, der mit einem Tracking-Device positioniert wird.

Eine ausgewählte Fahrzeugkomponente kann anschließend durch Benutzereingaben transformiert werden. Im Falle von stationären Geometrien reicht die Veränderung der Transformationsmatrix aus, um sie den Interaktionen entsprechend zu transformieren.

Die instationären Fahrzeugkomponenten sollen in *VtCrash* vom Fahrzeugkontext gelöst, zeitlich ortsfest zum Komponentenschwerpunkt bewegt werden, um die lokalen Deformationen besser analysieren zu können. Hierzu wird vom entsprechenden Objekt der Interaktionsstruktur die aktuelle Transformationsmatrix in den sichtbaren Geometrie-Knoten kopiert. Mit dieser Technik bleiben die Schwerpunkte der Fahrzeugkomponente aus den verschiedenen Zeitschritten in der Animation konstant an der transformierten Position und lassen so Detailanalysen zu (siehe Abbildung 6.8).

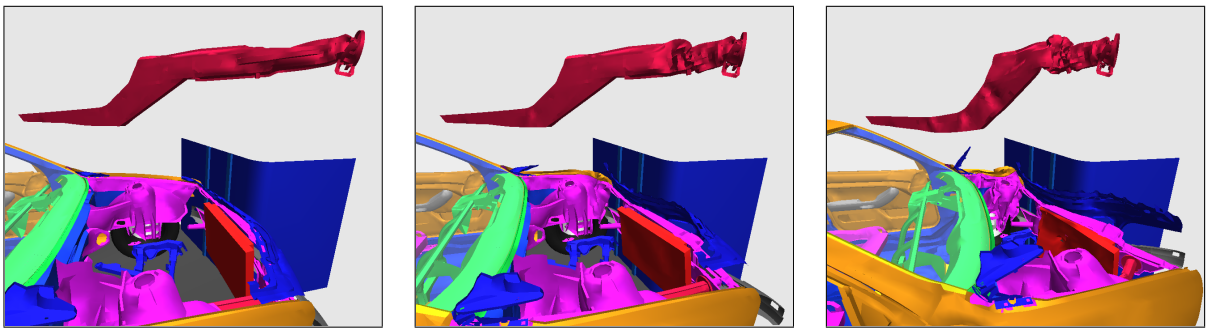

Abbildung 6.8: Detailanalyse der Deformation einer bewegten Fahrzeugkomponenten.

6.4.2 Andocken an Objekte

Die globale Bewegung des Fahrzeuges erschwert die Detailbetrachtung einer animierten Szene. Soll zum Beispiel im Motorraum das Knickverhalten des Motorträger bewertet werden, ist es sinnvoll, den Abstand zwischen Augpunkt des Betrachters und animierter Fahrzeugkomponente konstant zu halten.

In der virtuellen Umgebung von *VtCrash* kann sich der Anwender an eine Fahrzeugkomponente heften, wodurch der Abstand zwischen Komponentenschwerpunkt und Augpunkt während der Animation konstant bleibt (siehe Abbildung 6.9). Der Benutzer kann zusätzlich den Augpunkt im lokalen Koordinatensystem der Komponente bewegen und so das Deformationsverhalten im Umfeld betrachten. Realisiert wurde diese Funktionalität durch die kontinuierliche Anpassung der Augpunkt-Transformationsmatrix relativ zur Referenz-Komponente.

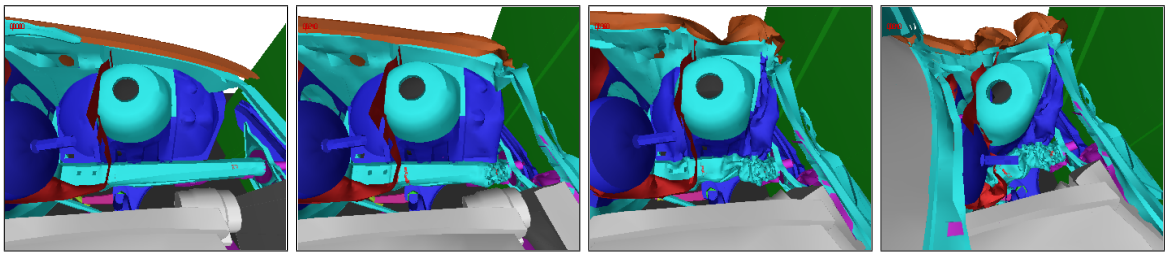

Abbildung 6.9: Der Augpunkt ist während der Animation an den Motorblock geheftet.

6.4.3 Schnittebenen

Schnitte durch Materialien sind ein wichtiges Werkzeug zur Auswertung von Simulationsergebnissen. Der Anwender legt an geeigneter Stelle eine Schnittebene durch die Fahrzeugkomponente und kann anhand der berechneten Schnittfläche die Deformation der Komponente bewerten. Ingenieure kennen Schnitte aus der realen Welt und aus den zweidimensionalen Plots der klassischen Postprocessing-Systeme.

Die zweidimensionale Darstellung einer Szene auf einem Monitor und die Steuerung mit der 2D-Maus erschweren die Positionierung einer freibeweglichen Schnittebenen, weshalb konventionelle Postprocessing-Systeme häufig nur achsenparallele Schnitte zulassen. Die stereoskopische Darstellung und der Einsatz von VR-Eingabegeräten hingegen erlauben in *VtCrash* eine intuitive Positionierung von frei definierbaren Schnitten. Der Anwender positioniert die Schnittebene in der Fahrzeuggeometrie und startet die Berechnung (siehe Abbildung 6.10).

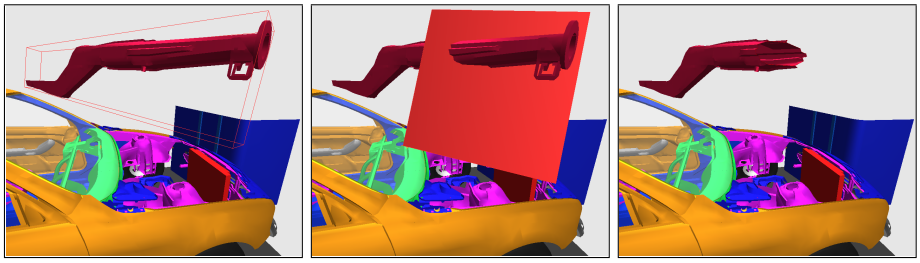

Abbildung 6.10: Durchschneiden des Motorträgers mit einer Schnittebene.

Für alle Zeitschritte der Simulation wird der zeitabhängige Schnitt mit einem der zwei unterstützten Algorithmen von Interaktionsstruktur von *VtCrash* berechnet. Der einfache Euler-Schnitt wird zu jedem Zeitschritt erneut bestimmt und durchtrennt aufgrund der Komponentenbewegung und -deformation im zeitlichen Verlauf verschiedene Polygone (siehe Abbildung 6.11). Der Lagrange-Schnitt hingegen schneidet die Fahrzeugkomponente in einem initialen Zeitschritt, der dann auf die anderen Zeitschritte übertragen wird. Der Bezug des Schnittes auf die Geometriestruktur kann mit dem Durchtrennen einer realen Komponente verglichen werden.

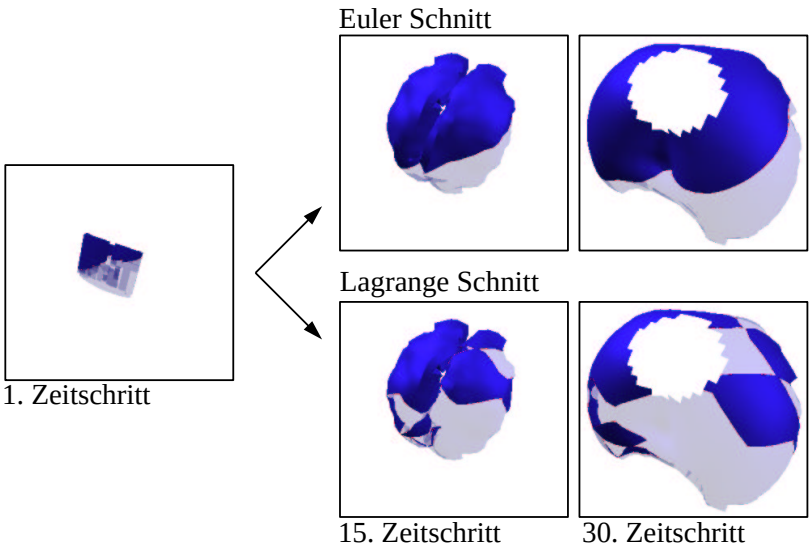

Abbildung 6.11: Vergleich des Euler- und des Lagrange-Schnittes am Beispiel des Airbags.

6.4.4 Intrusionsberechnung

Die Eindringtiefe von Fahrzeugkomponenten in die Fahrgastzelle ist ein wichtiges Kriterium zur Beurteilung der Konstruktionsgüte. Gerade bei seitlichen Crashtests hängt das Überleben des Fahrgastes von der Größe des verbleibenden Platzes ab.

In realen Crashtestversuchen kann die Intrusion am Fahrzeugwrack nachgemessen und beurteilt werden. Eine zeitabhängige Analyse des Eindringverhaltens ist nur anhand von Videoaufnahmen aus vorgegebenen Blickrichtungen möglich. Die interaktive Visualisierung von Berechnungsergebnissen in *VtCrash* bietet eine flexiblere Form der Analyse. Die Eindringtiefe wird als Abstand zu einer vom Benutzer definierten Referenzebenen berechnet, die durch das Anpicken von drei Polygonen im Modell aufgespannt wird (siehe Abbildung 6.12).

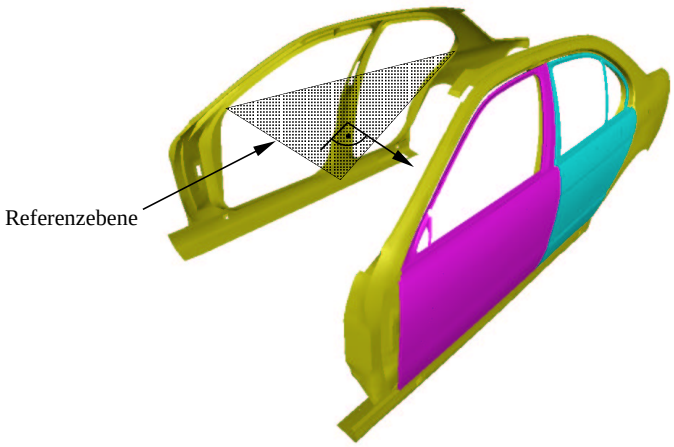

Abbildung 6.12: Definition der Referenzebene in der Fahrzeugkarosserie.

Der Anwender kann durch die interaktive Selektion von Fahrzeugkomponenten die Intrusionstiefen berechnen lassen, deren Oberflächen mit dem Betrag der Abstandsdifferenz eingefärbt werden. Unter Verwendung einer angepassten Skalierung sind Bereiche mit geringer Intrusion blau und mit großer entsprechend rot markiert (siehe Abbildung 6.13).

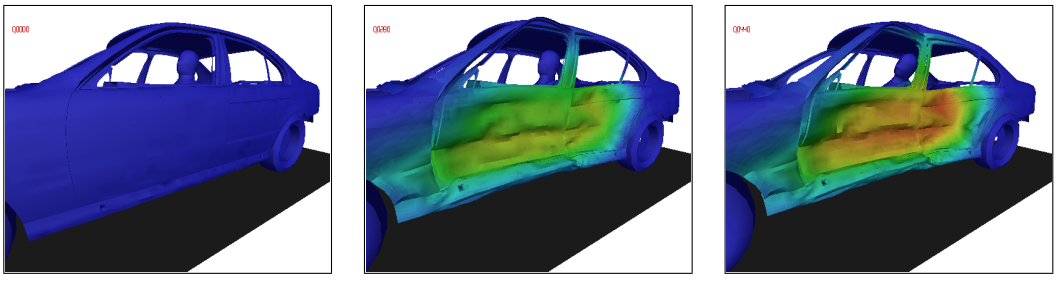

Abbildung 6.13: Visualisierung der Intrusionstiefe am Beispiel eines Seitenaufpralls.

Diese interaktive Form der Intrusionsberechnung erlaubt auch ungeübten Benutzern eine intuitive Analyse des Intrusionsverhaltens und gibt einen besseren Überblick als es einzelne Messungen nach einem realen Crashtest ermöglichen.

6.5 Integration in eine CAVE-Umgebung

CAVE-Installationen bieten mit 3-6 stereofähigen Projektionsleinwänden und einem Trackingsystem für Kopf und Hände einen hohen Grad an Immersion. Der Ingenieur bekommt in einer CAVE einen guten dreidimensionalen Eindruck des Sachverhaltes vermittelt und kann direkt mit der Visualisierung in Interaktion treten, wodurch auch ungeübte Anwender intuitiv arbeiten können. Die VR-Hardware muss von Seiten der Software durch Anpassungen und Erweiterungen unterstützt werden.

Die Projektionsleinwände der CAVE erfordern eine parallele Ansteuerung durch die Applikation. Der Viewer des Szenengraphen muss daher Multi-Pipe und Multi-Thread fähig sein, um die Leistung des angeschlossenen Rechners (im Allgemeinen eine *SGI ONYX*) voll auszunutzen. Die Bilder der Projektionsleinwände werden je nach Position des Kopfes kontinuierlich in Echtzeit berechnet und führen zur vollständigen Auslastung der Graphiksysteme. Die Position des Augpunktes erhält die Applikation vom Tracking-System und kann die Viewing-Frustums entsprechend bestimmen (siehe Abbildung 6.14).

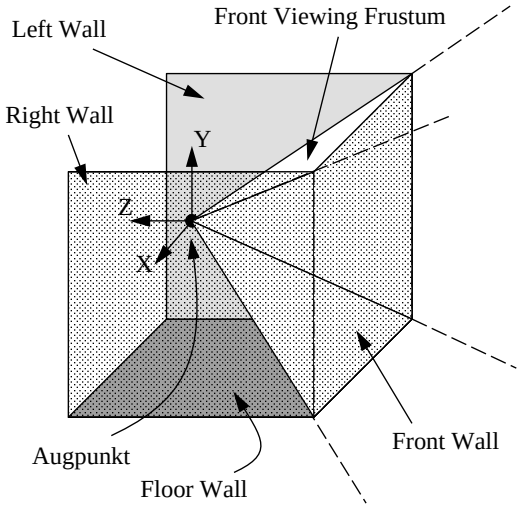

Abbildung 6.14: Die Definition des Front View Frustums in einer CAVE.

Nach [22, 24, 23] lässt sich die Berechnung der Viewing-Frustums direkt aus der Abbildung 6.15 ableiten.

Die Projektion P^* eines Punkte $P = (P_x, P_y, P_z)$ auf die *FrontWall* ist gegeben durch:

$$P_x^* = P_x + \frac{(D - P_z)(E_x - P_x)}{E_z - P_z} \tag{6.1}$$

$$P_y^* = P_y + \frac{(D - P_z)(E_y - P_y)}{E_z - P_z} \tag{6.2}$$

D ist der Abstand vom CAVE-Mittelpunkt zur Wand. Daraus ergibt sich die Projektionsmatrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{E_x}{E_z - D} & -\frac{E_y}{E_z - D} & 1 & -\frac{1}{E_z - D} \\ \frac{E_x D}{E_z - D} & \frac{E_y D}{E_z - D} & 0 & \frac{E_z}{E_z - D} \end{pmatrix} \tag{6.3}$$

Die Projektionsmatrizen der anderen Leinwände können mit den relativen Augpositionen $Leftwall : (E_z, E_y, E_x)$, $RightWall : (-E_z, E_y, E_x)$ und $FloorWall : (E_x, E_z, -E_y)$ entsprechend bestimmt werden.

Der Viewer von *WorldToolKit* bietet für parallel betriebene Viewer bereits Multi-Pipe- und Multi-Thread-Unterstützung, allerdings fehlt die direkte Ansteuerung einer CAVE. Für *VtCrash* wurde die CAVE-Projektionsmatrix in Kombination mit Tracking-Systemen in den *WorldToolKit*-Viewer integriert. Der Einsatz von *VtCrash* in VR-Hardware-Umgebungen wurde somit ermöglicht.

In immersiven Umgebung steht dem Anwender keine Standard-Benutzeroberfläche zur Verfügung. Zur Steuerung und Interaktion in CAVE-Umgebungen wurde die dreidimensionale Benutzerschnittstelle *G3Menu* in *VtCrash* integriert, über die der Benutzer mit Hilfe eines Tracking-Systems die *G3Menu*-Elemente auswählen und die Applikation steuern kann. Die Auswahl von Menü-Elementen und anderen virtuellen Objekten erfolgt mit einem virtuellen Degen, der an das Tracking-System gekoppelt ist. Angewählte Objekte kann der Benutzer mit dem Degen im Raum bewegen und so intuitiv numerische Simulationsergebnisse erforschen.

Die CAVE-Erweiterung von *VtCrash* eignet sich ebenfalls zur Ansteuerung einer mehrseitigen Projektionsleinwand und wurde im VR-Zentrum der *BMW Group* installiert.

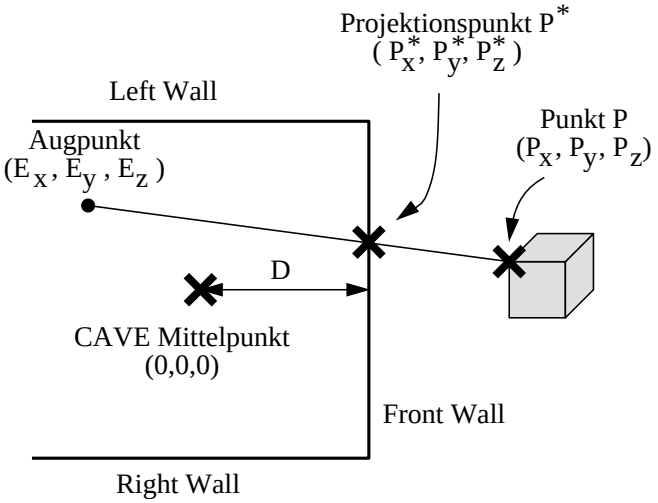

Abbildung 6.15: Skizze der CAVE-Projektion.

6.6 *VRML*-basierte Visualisierung

An die Gesamtmodellanalyse schließen sich im Fahrzeugentwicklungsprozess Detailuntersuchungen auf der Basis von Teilmodellen an. In dieser Entwicklungsphase ist der Informationsaustausch zwischen den Simulationsingenieuren und den Konstrukteuren zur Diskussion von Problemstellen extrem wichtig, die aufgrund der Globalisierung des Arbeitsumfeldes auch über das Intra- und Internet erfolgen muss. Wichtig ist in diesem Zusammenhang auch die Plattformunabhängigkeit der Visualisierung für heterogenen Rechnernetze.

Das Dateiformat *VRML97* (Virtual Reality Modelling Language) eignet sich zur Erfüllung dieser Anforderungen und wurde bereits in ähnlichen Projekten als Standardformat verwendet [83]. Internet-Browser wie beispielsweise der *Netscape Navigator* oder der *Microsoft Explorer* sind auf praktisch allen Plattformen verfügbar und bieten mit den *VRML*-PlugIns eine standardisierte, plattformunabhängige Graphikausgabe. Die Beschränkung in diesem Prozessschritt auf Teilmodelle ermöglicht eine Visualisierung nicht nur im High-End-VR, sondern auch im *VRML*-basierten Teilbereich von *VtCrash* (Abbildung 6.16).

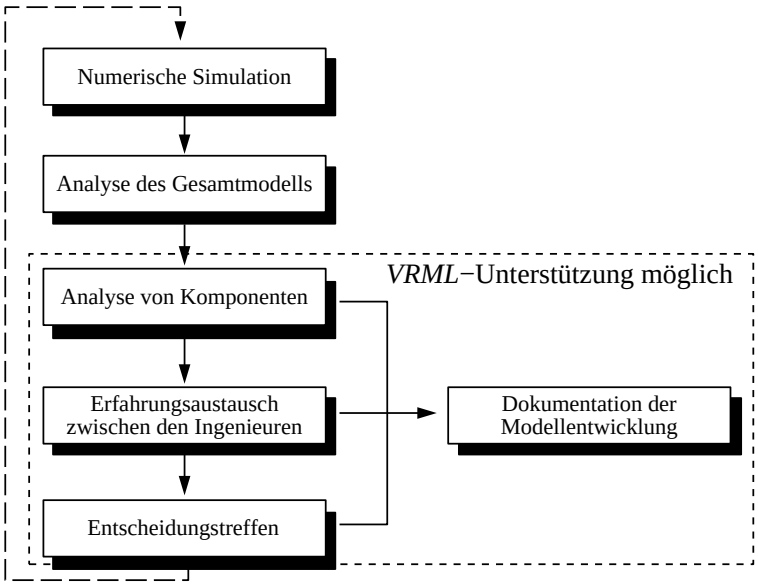

Abbildung 6.16: Einsatzmöglichkeiten von *VRML*-basierter Visualisierung im Postprocessing

Im High-End-VR-Bereich von *VtCrash* kann der Ingenieur eine Auswahl an problemrelevanten Fahrzeugkomponenten als Teilmodell in einer *VRML*-Datei speichern und die Analyse im *VRML*-basierten Teilbereich fortsetzen. Die *VRML*-Dateien können vom Berechnungsingenieur als email an den verantwortlichen Konstrukteur versendet und von ihm ohne spezielle Postprocessing-Software visualisiert werden. In kooperativen Visualisierungssitzungen via Internet können *VRML*-Dateien ebenfalls die Basis des Datenaustausches bilden.

6.6.1 *VRML*-Szenengraph für zeitabhängige Finite-Element-Modelle

Die Struktur der *VRML*-Szenengraphen verfügt neben den Standardelementen wie Gruppen-, Transformations- und Geometrie-Knoten zusätzlich über Sensor- und Multimediaobjekte. Die Sensorknoten leiten über so genannte *Routes* Benutzereingaben an Szenengraphobjekte weiter und die Multimediaobjekte dienen zur Einbindung von Bildern und digitalen Videosequenzen. Dreidimensionale Geometrien können in *VRML* als Standardobjekte und als beliebige Gitternetze definiert werden.

Die Szenengraph-Strukturen der beiden *VtCrash* Teilbereiche ähneln sich stark, wie aus den Abbildungen 6.6 und 6.17 hervorgeht. Auch der *VRML*-Szenengraph besitzt als Wurzel einen *Switch-Knoten* mit den darunter angebrachten Teilbäumen der Geometrie eines Zeitschrittes. Die einzelnen Fahrzeugkomponenten werden bei der Erstellung aus dem High-End-VR-Szenengraphen in *VRML* Face-Set-Geometrie-Knoten abgebildet. Die Bewegungen der Geometrien werden durch den Transformations-Knoten zwischen dem Gruppen- und dem Geometrie-Knoten ermöglicht.

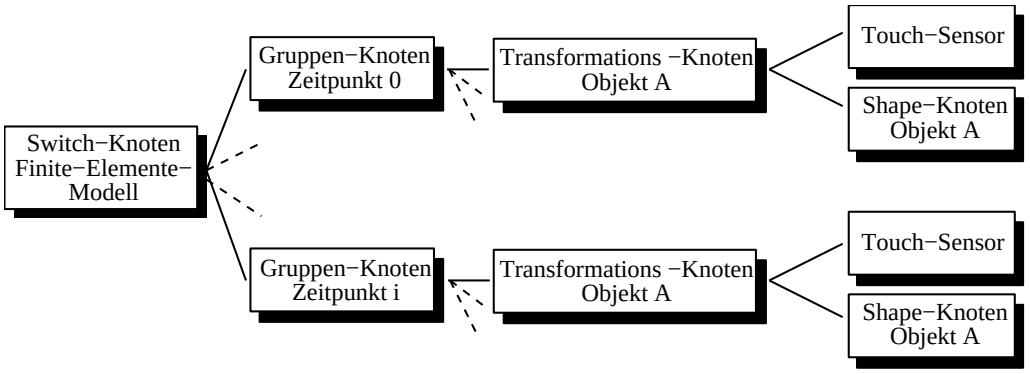

Abbildung 6.17: Der Szenengraph für die Fahrzeuggeometrie.

Das Event-Handling erfolgt innerhalb des *VRML* Szenengraphen über die Sensor-Knoten. So werden die Maus-Benutzereingaben von den parallel zur Geometrie angebrachten Touch-Sensoren abgefangen und über die definierten *Routes* an die Transformations-Knoten weitergeleitet.

6.6.2 Interaktionsmechanismen im Standard *VRML*-Browser

In der *VRML*-basierten Visualisierungsumgebung wurden Interaktionsmechanismen ähnlich dem High-End-Bereich realisiert, mit denen der Anwender Fahrzeugkomponenten interaktiv auswählen, transformieren und den Augpunkt koppeln kann. Zur Realisierung der Interaktionen standen nur die Sensorenklassen der Standard *VRML*-Browser zur Verfügung, deren Interaktionskonzept und die Beschränkung auf die 2D-Maus die Umsetzung erschwerten.

Die Einschränkungen der *VRML*-Sensoren lassen sich durch die Verwendung des External Authoring Interfaces (*EAI*) umgehen. Diese Schnittstelle des *VRML*-Viewers zu einem *Java*-Applet erlaubt zusätzliche Manipulationen des Szenengraphen. So können Maus-Events an das

Applet weitergereicht und durch *Java*-Funktionen Szenengraph-Knoten verändert werden.

Eine menügesteuerten Benutzerführung ist in einem *VRML*-PlugIn ebenfalls nicht vorgesehen. Das Abstract Windowing Toolkit (*AWT*) leistet hier mit seinen 2D-Menü Objekten Abhilfe, da es eingebettet in eine *HTML*-Seite Eingaben an ein *Java*-Applet ermöglicht. Auch erlaubt es Informationsausgaben in Textfeldern des Browsers.

In dieser Arbeit wurde die in Abbildung 6.18 dargestellte Arbeitsumgebung zur Visualisierung von *VRML*-Szenen realisiert. In dem *CosmoPlayer*-PlugIn wird der *VRML*-Szenengraph geladen und ein *AWT*-Buttonfeld erlaubt zusätzliche Steuerungen der Szene, wie beispielsweise die Auswahl des dargestellten Zeitschrittes. Ein *Java*-Applet reagiert im Hintergrund auf die Benutzereingaben des *AWT* und steuert über das *EAI* den *CosmoPlayer*. Im Textfeld erscheinen Statusmeldungen des *Java*-Applets.

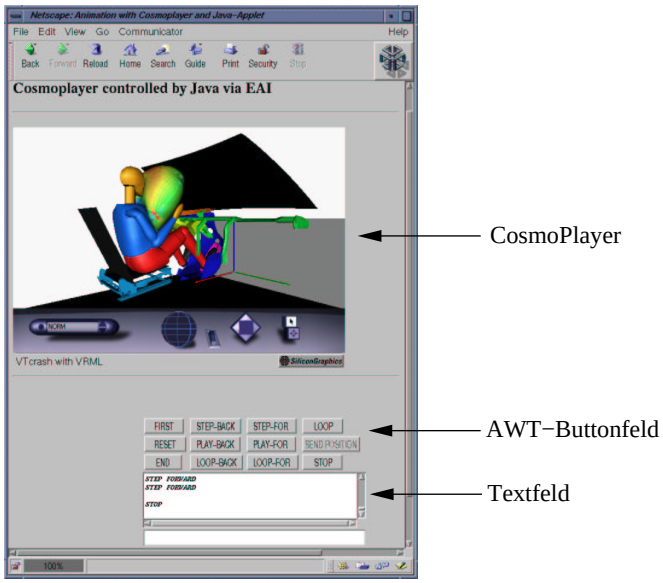

Abbildung 6.18: Arbeitsumgebung der *VRML*-basierten Visualisierung in einem Internet-Browser.

VRML97 besitzt für die Benutzerinteraktion die drei Sensor-Typen *Touch*, *Plane* und *Sphere*. Ein Sensor wird parallel zu einem Geometrie-Knoten im Szenengraphen angebracht und der Geometrie zugeordnet. Sobald die Geometrie vom Anwender durch Drücken der linken Maustaste ausgewählt wird, tritt ein zugeordneter *Touch* Sensor in Aktion, generiert ein Pick-Event für die entsprechende Geometrie und löst eine Methode des *Java*-Applets aus. Der *Plane* Sensor liefert entsprechend der Mausbewegung Parameter für eine zweidimensionale Translation der Geometrie in X- und Y- Richtung. Eine Rotation der Geometrie kann durch einen *Sphere* Sensor erfolgen. Die Kopplung von mehr als einem Sensor an eine Geometrie ist nicht empfehlenswert, da nur die separate Übertragung der Mauseingaben als Translation oder Rotation der Geometrie sinnvoll ist. Die gewünschte dreidimensionale Transformation einer Fahrzeugkomponente ist mit dem Standard-*VRML*-Szenengraphen nicht zu erzielen.

Zur Lösung des Problems wurde in dieser Arbeit eine Interaktionsgeometrie entwickelt, die eine sechsdimensionale Transformation einer Geometrie in einem *VRML*-Browser ermöglicht.

Die Interaktionsgeometrie besteht aus zwei senkrecht zueinander stehenden Ebenen und vier Würfeln (siehe Abbildung 6.19) und wird in den Szenengraphen hinzugefügt, sobald der Anwender eine Geometrie durch Picken mit der linken Maustaste auswählt. Anschließend lässt sich die Fahrzeugkomponente durch Picken der Ebenen translieren und mit den Würfeln rotieren.

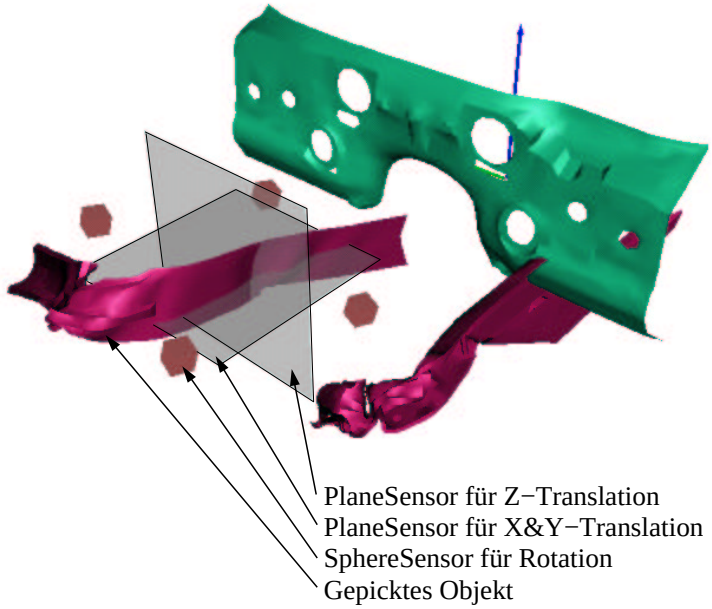

Abbildung 6.19: Die Interaktionsgeometrie zu einer gepickten Fahrzeugkomponente.

Der Szenengraph aus Abbildung 6.17 wird für die Interaktionsgeometrie um einen parallel angeordneten Teilbaum erweitert, wie in Abbildung 6.20 dargestellt. Der Interaktionsteilbaum besitzt für jede Fahrzeugkomponente eine Interaktionsgeometrie bestehend aus einem Gruppen-, einigen Translations-, den Geometrie-Knoten und den Sensoren. Durch die Anordnung von jeweils einem Paar aus Geometrie- und Sensor-Knoten unterhalb eines Translations-Knoten ist die eindeutige Zuordnung definiert (siehe Abbildung 6.20). Parallel zu jeder Ebene befindet sich ein *Plane*-Sensor für die Translation und parallel zu den Würfeln ein *Sphere*-Sensor für die Rotation. Die Interaktionsgeometrien werden nach einem Pick-Event sichtbar. Das *EAI* überträgt das Pick-Event an das *Java*-Applet, welches nun die Interaktionsgeometrie aktiviert und *Routes* zwischen den Objekten des Szenengraphen und der Interaktionsgeometrie erstellt. Sobald nun eine Interaktionsgeometrie von dem Anwender gepickt und die Maus bewegt wird, leiten die *Routes* die Events an die Transformations-Knoten der Fahrzeug- und der Interaktionsgeometrie weiter. Die Fahrzeugkomponente und ihre Interaktionsgeometrie bewegen sich somit synchron zu den Benutzereingaben.

Ein erneutes Picken der Fahrzeuggeometrie führt zu dessen Deaktivierung. Das *Java*-Applet entfernt über das *EAI* die Interaktionsgeometrie und die *Routes*. Die Fahrzeugkomponente befindet sich nach der Deaktivierung an der transformierten Position. Im *Java*-Applet wurde jedoch die ursprüngliche Position gespeichert und kann für ein Zurücksetzen der Geometrie verwendet werden.

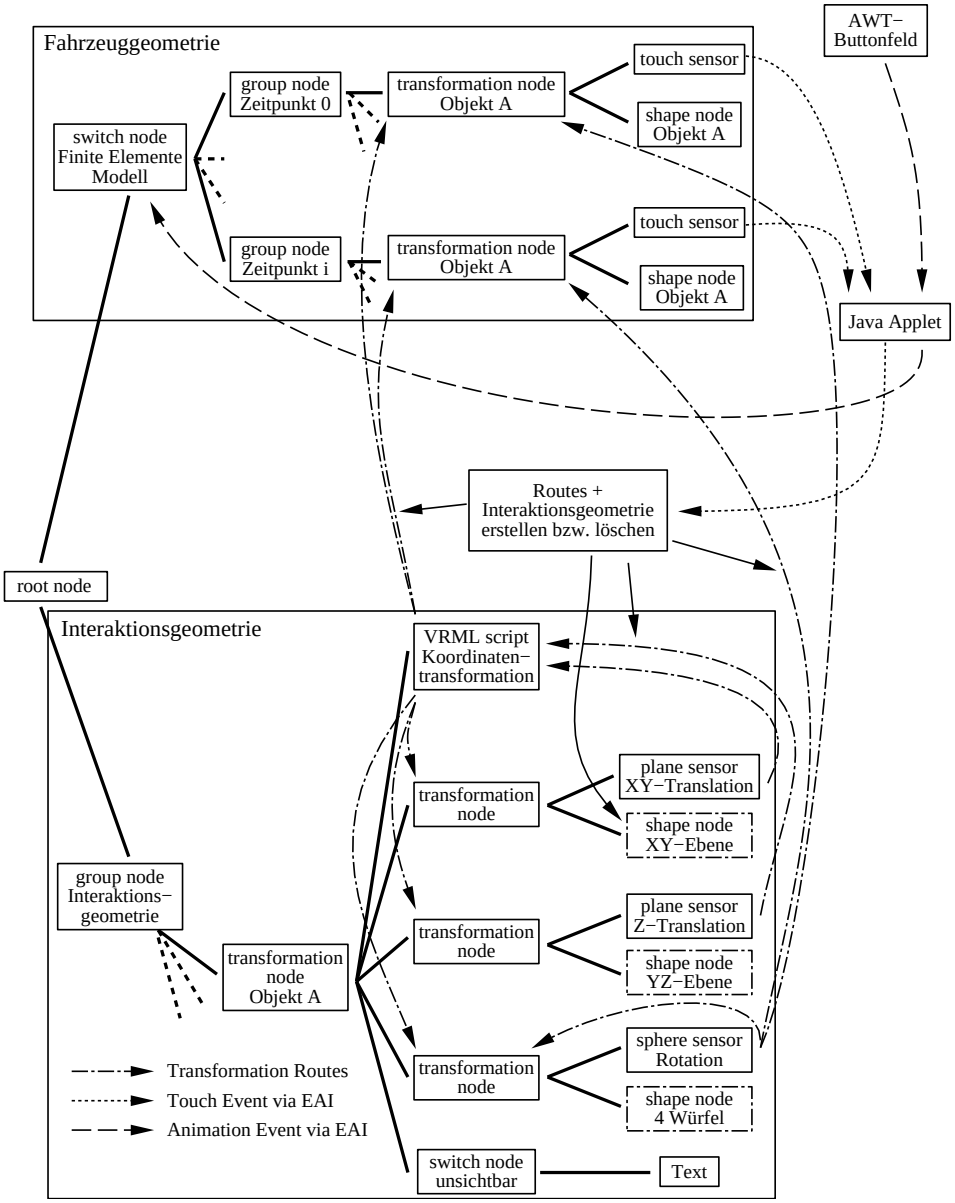

Abbildung 6.20: Integration der Interaktionsgeometrie in den *VRML* Szenengraph.

6.6.3 Variantenvergleich und kooperatives Arbeiten mit synchronisierten Browsern

Die Verbesserungsmaßnahmen am Fahrzeug werden durch erneute Simulationsläufe abgesichert und die entstehenden Varianten zur Bewertung miteinander verglichen. Da sich die Änderungen auch auf die Geometriestruktur beziehen können, ist ein direktes Mappen der verschiedenen Berechnungsergebnisse nicht ohne weiteres möglich. Somit ist das Wissen und die Erfahrung von Simulationsspezialisten erforderlich, um anhand der verschiedenen Varianten Verbesserungen

oder eventuelle Beeinträchtigungen zu erkennen. Eine synchronisierte Gegenüberstellung von Varianten kann den Ingenieur bei der Auswertung unterstützen.

Die Aufgabenteilung in die verschiedenen Elemente (*VRML*-PlugIn, *EAI*, *Java*-Applet und *AWT*) der *VRML*-basierten Visualisierung erlaubt eine simple Erweiterung zur gleichzeitigen Darstellung mehrerer Berechnungsvarianten. Die *HTML*-Seite enthält nun zwei *VRML*-Browser, die von einem *Java*-Applet synchron über die *EAI*-Schnittstelle angesteuert werden, siehe Abbildung 6.21.

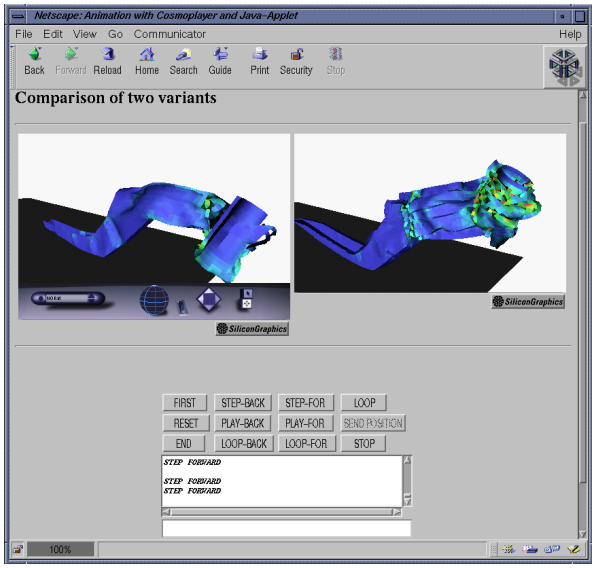

Abbildung 6.21: Synchronisation von Animation und Augpunkttransformation zweier VRML-Browser.

Das erweiterte *Java*-Applet leitet die Benutzereingaben des *AWT*-Buttonfeldes an die beiden angeschlossenen *VRML*-PlugIns weiter und synchronisiert so den dargestellten Zeitschritt und die Augpunktposition bzw. -orientierung. Die Benutzereingaben im linken *VRML*-PlugIn werden über das *Java*-Applet und die *EAI*-Schnittstellen an das rechte weitergeleitet. Alle Manipulationen im Szenengraphen werden so synchron in beiden *VRML*-PlugIns durchgeführt, wie beispielsweise der Einsatz der Interaktionsgeometrie. Im rechten PlugIn sind Benutzereingaben somit nicht erforderlich und werden durch Deaktivierung des Eingabe-Panels unterbunden. Dem Ingenieur wird auf diese Weise ein leicht bedienbares Werkzeug für den Vergleich von Varianten zur Verfügung gestellt.

In vielen Fällen sind die Arbeitsplätze von Ingenieur und Konstrukteur räumlich getrennt und die Diskussion über Modellschwachstellen kann mit Hilfe einer virtuellen, kooperativen Arbeitssitzung via Internet erleichtert werden. Die *VRML*-basierte Visualisierung bietet hierfür durch den Einsatz eines Internet-Browsers und der Verwendung von *Java*-Applets die besten Voraussetzungen.

Realisiert wurden kooperative Arbeitssitzungen durch ein Client-Server-Konzept. Mehrere Internet-Browser inklusive *VRML*-PlugIn und *Java*-Applet bilden die Client-Seite, deren Datenaustausch über einen durch Sockets verbundenen und in *Java* entwickelten Server koordiniert

wird (siehe Abbildung 6.22). Informationen über Augpunkttransformationen und die Animation der Szene tauschen die Clients über den Server aus. Eine nicht-synchronisierte Darstellung durch gleichzeitige Eingaben in verschiedenen Clients wird durch die Regulierung der Sendeberechtigung mit einem Token verhindert. Nur der Client mit dem Token darf die Szene manipulieren und die Änderungen an die anderen Clients weiterreichen. Die nicht-berechtigten Clients können über den Server die Zuweisung des Tokens anfordern.

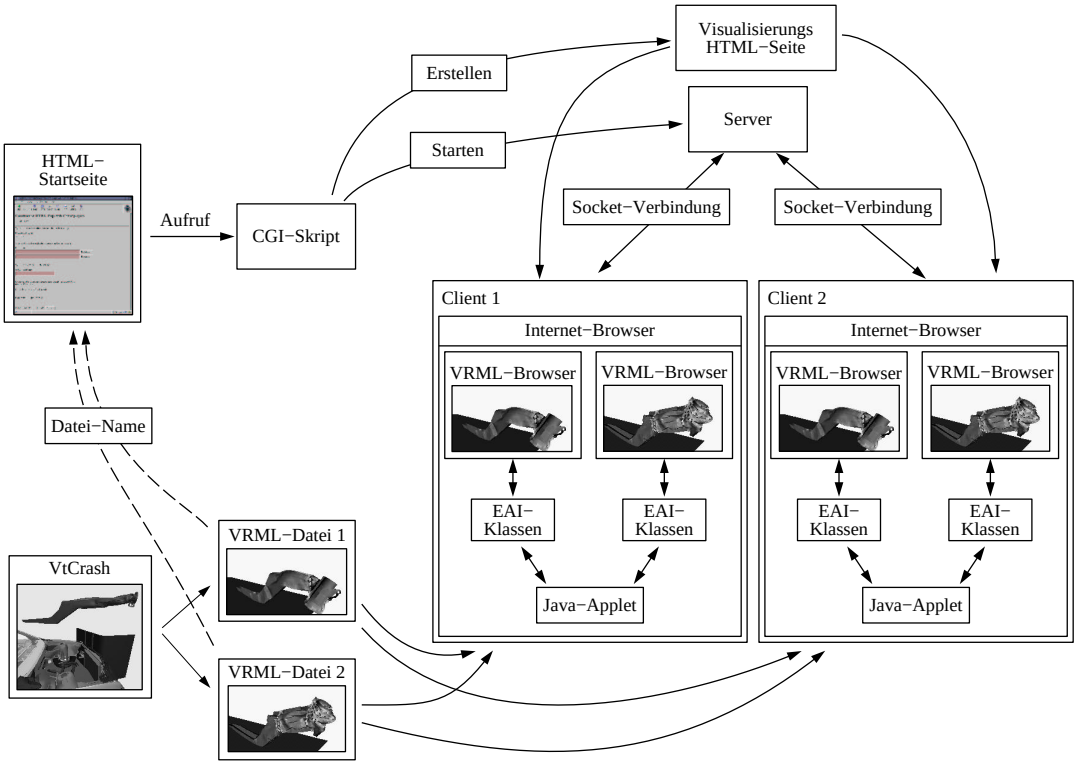

Abbildung 6.22: Aufbau einer kooperativen Sitzung mit zwei Clients.

Die Grundeinstellungen einer kooperativen Arbeitssitzung, wie beispielsweise die Namen der zu ladenden *VRML*-Dateien, werden in einer *HTML*-Startseite vorgenommen. Das anschließend ausgeführte *CGI*-Skript startet einen Server und erstellt eine Visualisierungs-*HTML*-Seite, über die sich Teilnehmer anmelden können. Der Anwender startet durch den Aufruf der Visualisierungs-*HTML*-Seite einen Client, dessen *Java*-Applet sich automatisch am Server anmeldet und die Datensätze lädt. Die etablierte Arbeitssitzung bleibt bis zur Abmeldung aller Clients bestehen und beendet sich selbstständig durch Terminierung des Servers.

In den kooperativen Arbeitssitzungen stehen alle Interaktionsmechanismen des *VRML*-basierten Teilbereichs von *VtCrash* zur Verfügung. Die verschiedenen *Java*-Implementierung auf den unterschiedlichen Plattformen verhalten sich jedoch noch sehr unterschiedlich bzw. instabil und lassen somit einen produktiven Einsatz dieser Techniken noch nicht zu.

6.7 Fazit

Der High-End-VR-Teilbereich von *VtCrash* zeigt ausbaufähige Ansätze für intuitive Interaktionsmechanismen auf. Ingenieure erkannten im Umgang mit *VtCrash* das Potenzial der virtuellen Realität für die Fahrzeugentwicklung und förderten die Integration der Interaktionsmechanismen in kommerzielle Produkte. Der hohe Ressourcenbedarf von *WorldToolKit* beschränkte das Einsatzprofil von *VtCrash* auf VR-Demozentren und Präsentationen. Trotz der Einschränkungen vertreibt der Hersteller des Simulationscodes *PAM-CRASH* eine kommerzielle Version von *VtCrash* als VR-Postprocessing-Werkzeug.

Der *VRML*-basierte Teilbereich zeigt ebenfalls ein großes Potenzial für zukünftige Entwicklungen auf. Die Struktur des *VRML*-Szenengraphen und die Kopplung mit *Java* hat gezeigt, dass selbst Standard-*VRML*-Viewer für die Analyse von Simulationsergebnissen geeignet sind. Die Plattformunabhängigkeit und der minimale Systembedarf ist ein großer Vorteil für kooperatives Arbeiten. Allerdings muss die Stabilität und die Standardisierung der *Java* Implementierungen noch verbessert werden.

Kapitel 7

Visualisierung von zeitabhängigen Akustikdatensätzen

Die Crashtest-Simulationen sind das klassische Beispiel für den Einsatz von Finite-Element-Analysen. Die Qualität eines Fahrzeuges hängt allerdings nicht nur von den sicherheitsrelevanten Aspekten ab, sondern auch vom Fahrkomfort. Ein niedriger Lärmpegel im Fahrzeuginneren ist beispielsweise für die Kaufentscheidung vergleichbar mit einer sicheren Fahrgastzelle. Störende Geräusche im Innenraum entstehen durch die von vibrierenden Fahrzeugkomponenten initiierten Schalldruckwellen. Bei der Konstruktion der Fahrgastzelle muss also auch das Schwingungsverhalten und dessen Auswirkungen auf den Lärmpegel beachtet werden.

Die Akustiksimulation wird zur komfortablen Auslegung des Fahrzeuges in Bezug auf Lärmerzeugung herangezogen. Im Gegensatz zu den geometriebezogenen Simulationen für Crashtest, Schwingung und Kinematik, wird in der Akustiksimulation mit einem zeitabhängigen Volumen gearbeitet. Die Basis für die Simulation des Luftdrucks bildet eine zuvor durchgeführte Schwingungsanalyse, dessen Finite-Element-Modell hierfür mit einem unstrukturierten Gitter ausgefüllt wird. Die simulierten Vibrationen initiieren in dem Volumen Luftdruckwellen, die sich durch das Fahrzeug bewegen.

In dieser Arbeit wurde ein prototypisches Visualisierungswerkzeug entwickelt, das die parallele Darstellung der vibrierenden Geometrie und des Volumens ermöglicht. Das bereits vorgestellte Visualisierungssystem *VtCrash* für geometriebezogene Simulationen dient als Ausgangspunkt, da es sehr gut für die Darstellung von Ergebnissen aus Schwingungsanalysen geeignet ist. Auch für die Volumenvisualisierung soll das System dem Anwender eine explorative Analyse der Berechnungsergebnisse in einer interaktiven virtuellen Umgebung bereitstellen.

7.1 Visualisierungstechniken für unstrukturierte Volumengitter

Volumendaten sind im Gegensatz zu geometrischen Objekten nur mit aufwendigen Algorithmen darzustellen. Ein bekanntes Verfahren ist das direkte Volumenrendering als Soft- oder Hardwarelösung, jedoch sind die erzielbaren Bildwiederholungsraten in Stereodarstellung mit momentan

weniger als 5 Hz für interaktives Arbeiten ungeeignet. Die indirekte Visualisierung des Volumens durch die Generierung von Geometrie ist hierfür ein besserer Ansatz, siehe Kapitel 2.2.2. Eine Anpassung der Basisalgorithmen an den vorliegenden Datentyp ist nötig, um eine interaktive Visualisierung zu erzielen.

7.1.1 Datensatzstruktur

Die Schwingung einer Fahrzeugkomponente resultiert aus einer linearen Simulation. Im Gegensatz zu der transienten Crashtest-Simulation kann die Bewegung einer vibrierenden Fahrzeugkomponente durch eine reell- oder komplexwertige Schwingungsgleichung abgebildet werden.

Die Ausgangsgeometrie (*Nullgeometrie*) besteht aus einem Feld von Knotenkoordinaten $[\vec{k_1}, \vec{k_2} \ldots, \vec{k_n}]$ und deren Elementzugehörigkeit. Die Position der einzelnen Knoten i zum Zeitschritt t ergibt sich aus der Addition von Ausgangsposition $\vec{k_i}$ und einem zeitabhängigen Displacement-Vektor $\vec{d_i}(t)$:

$$\vec{k_i}(t) = \vec{k_i} + \vec{d_i}(t) \quad \forall i \in 1, \ldots, n \tag{7.1}$$

Die Displacement-Vektoren $\vec{d_i}(t)$ werden nun durch die reell- oder komplexwertigen Schwingungsgleichungen (Gleichungen 7.2 bzw. 7.3) mit Parametern a_i und b_i für jeden Knoten i berechnet. Die zeitunabhängigen Parameter a_i und b_i werden in der Schwingungssimulation für jeden Knoten i bestimmt:

$$\vec{d_i}(t) = a_i * \cos t \tag{7.2}$$

$$\vec{d_i}(t) = a_i * \cos t + b_i * \sin t \tag{7.3}$$

Die Bewegung der Finiten-Elemente wird also von einer *sin*- und *cos*-Funktion beschrieben, da sich die Funktionswerte von *sin* und *cos* für Vielfache von π wiederholen, brauchen nur Displacement-Vektoren für den Bereich $[0.0, \ldots, \pi]$ berechnet und entsprechende Geometrien erstellt werden.

Die Fahrgastzelle ist aus einem stationären unstrukturierten Gitter bestehend aus Tetraedern, Pyramiden, Prismen und Hexaedern aufgebaut (siehe Abbildung 7.1). Anders als die vibrierende Geometrie der Fahrgastzelle ist die Volumenstruktur stationär und lediglich die an den Zellknoten simulierten Luftdruckparameter sind zeitabhängig. Die Ergebnisse der linearen Akustiksimulation können analog zu den Displacement-Vektoren der Schwingungssimulation berechnet werden. Der skalare Parameter s_i des Drucks wird aus dem Startwert $s_i(0)$ und der bekannten Gleichung mit den Parametern a_i und b_i bestimmt:

$$s_i(t) = s_i(0) + a_i * \cos t + b_i * \sin t \tag{7.4}$$

Für die Visualisierung werden n komplette Datensätze der Displacement-Vektoren und der skalaren Größen aus den Simulationsergebnissen benötigt, die mit den Funktionsgleichungen zu den diskreten Zeitschritten t (Gleichung 7.5) berechnet werden:

$$t = \frac{\pi}{n} * i \quad \forall i \in 0, \ldots, n-1 \tag{7.5}$$

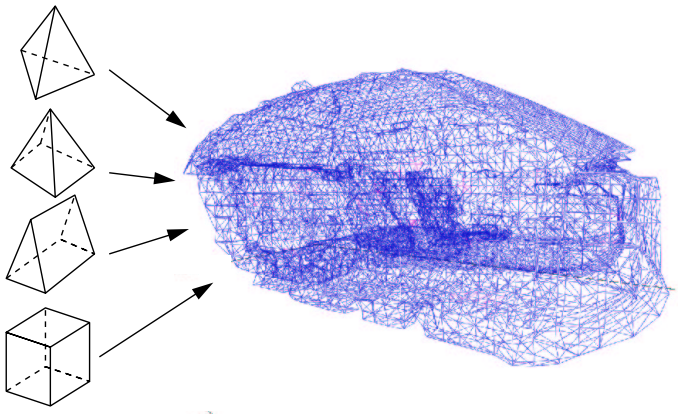

Abbildung 7.1: Volumenprimitive des Akustikvolumens.

Die Geometrien und das Volumen der diskreten Zeitschritte werden von dem Visualisierungssystem *VtCrash* im Ladevorgang berechnet. Zur besseren Visualisierung der Vibrationsbewegungen werden die Displacementvektoren um einen Faktor (beispielsweise 20.000) skaliert und sind so in der animierten Darstellung für den Anwender leicht erkennbar. In Abbildung 7.2 gibt die Visualisierung zum Beispiel einen guten Eindruck einer Verdrehung der Tür, die unskaliert nur unsichtbar um Millimeter zittert.

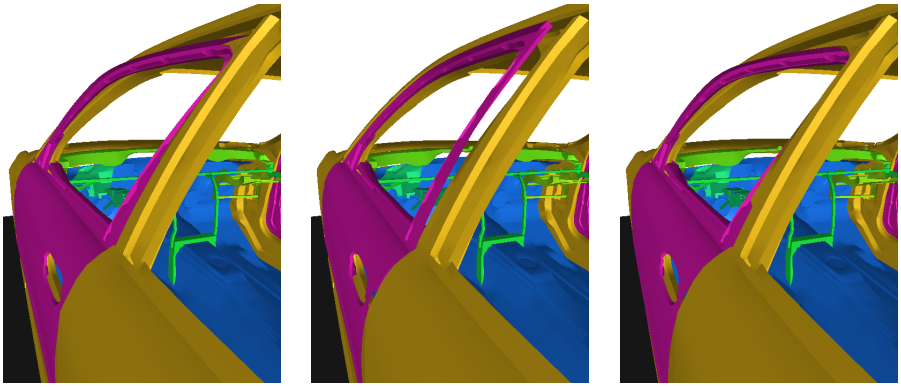

Abbildung 7.2: Visualisierung einer vibrierenden Tür.

Für eine erste Darstellung wird das Akustikvolumen auf seine Oberfläche reduziert und mit den skalaren Parametern eingefärbt. Die Abbildung 7.3 zeigt ein Akustikvolumen eingebettet in die Struktur der Fahrgastzelle. Die vibrierende Fahrzeuggeometrie und die Hülle des Volumens geben bereits erste Anhaltspunkte, an welchen Stellen Luftdruckwellen in das Volumen emittiert werden. Ein typischer Datensatz besteht aus ca. 100.000 Finiten-Elementen und ca. 20.000 Volumenzellen.

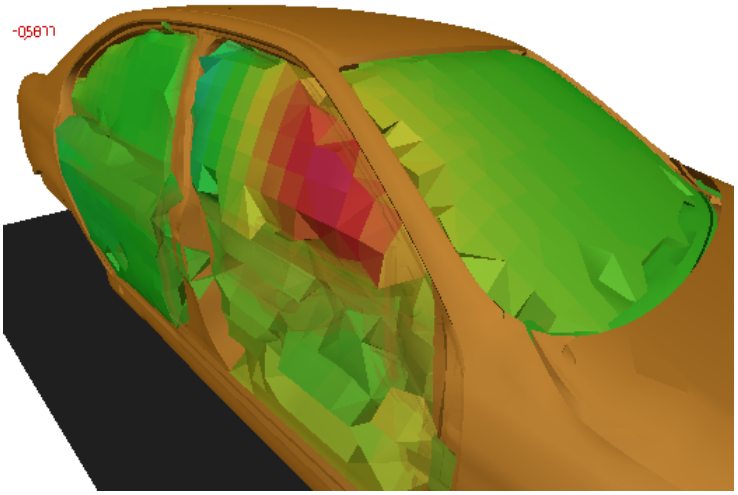

Abbildung 7.3: Ein Akustikvolumen eingebettet in die vibrierende Geometrie.

7.1.2 Zellsuche

Der Basisalgorithmus der Volumen-Visualisierung ist die Bestimmung der umschließenden Zelle zu einer gegebenen Position. Erst wenn die umschließende Zelle bekannt ist, können an der Position die Parameter des Volumens interpoliert werden. In strukturierten Gittern ist die Zellsuche durch die Vorgabe der Nachbarschaftsbeziehung vergleichsweise einfach und in uniformen Gittern beispielsweise implizit gegeben (siehe Kapitel 2.2.2.1). Das Auffinden der umschließenden Zelle eines Punktes in einem unstrukturierten Gitter erfordert hingegen komplexere Algorithmen.

Die naive Zellsuche in einem unstrukturierten Gitter, ohne Kenntnis über die Nachbarschaftsbeziehungen, besteht aus der Überprüfung jeder Zelle des Gitters. Sobald eine Zelle die gegebene Position umschließt, bricht der Algorithmus ab und liefert den Index der Zelle. Falls sich die Position allerdings außerhalb des Gitters befindet, müssen alle Zellen überprüft werden. Für eine interaktive Visualisierung ist der Aufwand der naiven Zellsuche zu hoch. Besser geeignet ist ein iteratives Verfahren, das sich Schritt für Schritt der gesuchten Zelle nähert. Hierbei wird ausgehend vom Mittelpunkt einer geratenen oder der zuletzt benutzten Zelle in Richtung der gegebenen Position die Nachbarzelle bestimmt, wie in Abbildung 7.4 skizziert.

Die Nachbarzelle wird durch Schneiden des Suchstrahls mit den Zellwänden bestimmt. Der Übergang zur gesuchten Nachbarzelle ist die Zellwand mit dem Schnittpunkt. Anhand der Beziehungen zwischen Knoten und Polygonen lässt sich die Nachbarzelle leicht bestimmen, da eine Zellwand nur von zwei Zellen referenziert werden kann. Verlässt der Suchstrahl das Volumen, wird die Suche abgebrochen. Die iterative Zellsuche lässt sich mit folgendem Pseudocode beschreiben:

```
neueZelle = geratene Zelle oder zuletzt benutzte Zelle;
DO{
  aktuelleZelle = neueZelle;
  IF(Position ist in aktuelleZelle)
```

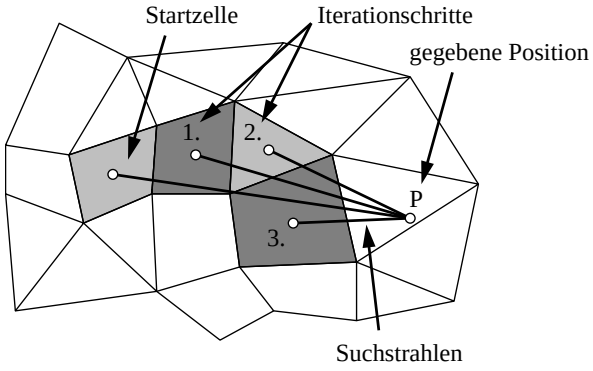

Abbildung 7.4: Iterative Suche der umschließenden Zelle zu einer gegebenen Position.

```
    RETURN Index von aktuelleZelle;
  suchRichtung = Position - Mittelpunkt(aktuelleZelle);
  Bestimme neueZelle in suchRichtung;
} WHILE (neueZelle gefunden)
RETURN keine Zelle gefunden
```

Im Gegensatz zum naiven Algorithmus, der für eine Menge von n Zellen eine Komplexität von $O(n)$ besitzt, ist der Aufwand des iterativen Suchalgorithmus durch $O(\sqrt[3]{n})$ beschränkt, denn im schlechtesten Fall muss das Volumen lediglich diagonal durchlaufen werden. In der Praxis sind die Positionen, an denen Skalarwerte ermittelt werden, nicht willkürlich innerhalb des Volumens verteilt, sondern liegen in einem lokalen Bereich nahe beieinander. Für die Zellbestimmung von Positionen mit geringem Abstand müssen weniger Zellen durchlaufen werden und der Algorithmus terminiert entsprechend schnell.

Der vorgestellt iterative Zellsuchalgorithmus hat allerdings mit den drei Spezialfällen „konkave Volumina“, „Löcher im Volumen“ und „mehrere separate Volumina“ Probleme. Der Suchstrahl führt in diesen Fällen, wie in Abbildung 7.5 skizziert, aus dem Volumen heraus und lässt den Algorithmus ohne Ergebnis terminieren. Alle drei Fälle treten in der Akustiksimulation auf, da eine ausgefüllte Fahrgastzelle in den seltensten Fällen konvex ist, Löcher beispielsweise durch die Sitze entstehen und die Türverkleidungen mit separaten Volumina ausgefüllt sind.

Mit dem naiven Ansatz lassen sich natürlich alle drei Problemfälle lösen. Zur Erzielung von interaktiven Raten wurde in dieser Arbeit jedoch ein heuristischer Ansatz verfolgt. Falls der erste Versuch einer Zellsuche fehlschlug, wird mit einer vorgegebenen Anzahl an weiteren Versuchen von heuristisch bestimmten Zellen aus gesucht. Bei diesem Ansatz kann es unter ungünstigen Voraussetzungen vorkommen, dass zu einzelnen Positionen die umschließende Zelle nicht gefunden wird. Dieser Nachteil wird allerdings in Hinblick auf die Interaktivität in Kauf genommen.

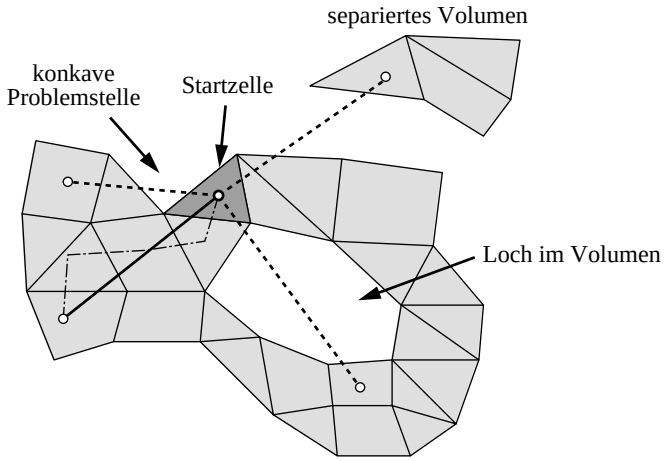

Abbildung 7.5: Problemfälle der iterativen Zellsuche.

7.1.3 Isoflächenberechnung

Die Berechnung von Isoflächen ist eine weitverbreitete Methode zur indirekten Visualisierung von skalaren Volumendatensätzen. Alle Raumpunkte eines Volumens, die dem Wert des gegebenen Isowertes entsprechen, definieren die Isofläche im dreidimensionalen Raum.

Der Marching Cubes Algorithmus (siehe Kapitel 2.2.2.3) extrahiert zellenweise aus einem uniformen Gitter Isoflächen. Die separate Betrachtung einzelner Zellen ist die Basis des einfachen Verfahrens, jedoch besitzt es durch die Verwendung von Hexaeder-Zellen die Gefahr der Entstehung von Löchern durch Zweideutigkeiten, wie im Beispiel von Abbildung 7.6 zu sehen ist. Tetraeder-Zellen hingegen besitzen keine Mehrdeutigkeiten und bieten durch die Tetraedisierung der Volumenzellen eine Möglichkeit solche Problemfälle zu verhindern.

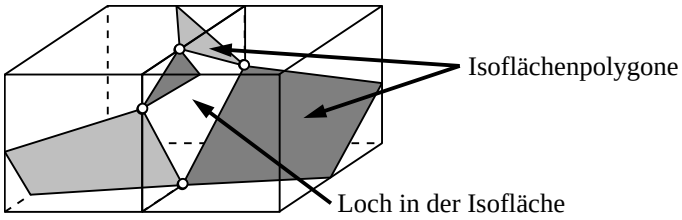

Abbildung 7.6: Entstehung von Löchern durch Mehrdeutigkeiten beim Marching Cube Algorithmus.

Die Klassifikation der Tetraeder-Zellen ergibt lediglich $2^4 = 16$ Fälle für die Triangulierung einer möglichen Isofläche. Unter Ausnutzung der Symmetrie-Eigenschaften sind letztlich nur noch drei Fälle zu unterscheiden (siehe Abbildung 7.7). An den Grenzflächen zweier Zellen können bei der Unterteilung in Tetraeder keine Zweideutigkeiten mehr auftreten und somit keine Löcher entstehen. Allerdings steigt durch die Tetraedisierung der Volumenzellen die Anzahl der zu betrachtenden Zellen und mit ihnen die Anzahl der entstehenden Polygone.

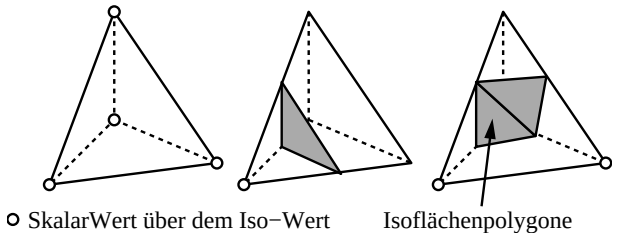

Abbildung 7.7: Die drei Fälle einer Isofläche in einem Tetraeder.

Die Tetraedisierung von Pyramiden, Prismen und Hexaedern in einem unstrukturierten Gitter muss zur Vermeidung von Spalten und Löchern konsistent erfolgen (siehe Abbildung 7.8). Die Beachtung aller Nachbarzellen führt zu einer konsistenten aber sehr aufwendigen Unterteilung der Zellen. Ein schnelleres Verfahren ist die Ausrichtung der Viereck-Unterteilungen anhand der niedrigsten Zahl einer eindeutigen, elementübergreifenden Nummerierung der Zellecken.

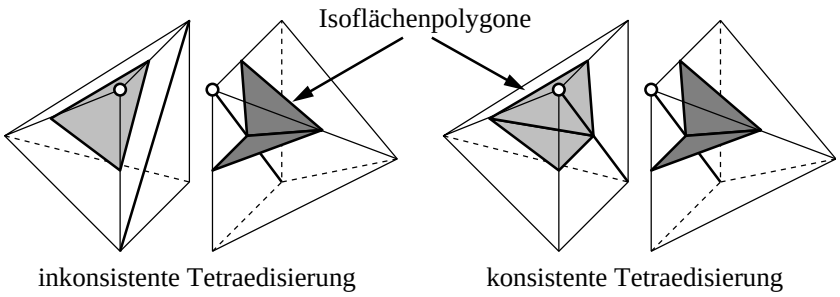

Abbildung 7.8: Entstehung von Löchern in der Isofläche durch eine inkonsistente Tetraedisierung.

Die konsistente Tetraedisierung eines unstrukturierten Gitters erfordert die unabhängige Orientierung der Unterteilung der viereckigen Zellseitenflächen, ansonsten lässt sich das Problem aus Abbildung 7.8 nicht für das gesamte Volumen lösen. Die Zellprimitiva Prisma und Hexaeder müssen daher mit einem entsprechenden Schema in Tetraeder unterteilt werden. Die Unterteilung eines Hexaeders in fünf oder sechs Tetraeder weist (siehe Abbildung 7.9 links und mitte) diese Eigenschaft nicht auf und so muss die Unterteilung in zwölf Tetraeder (siehe Abbildung 7.9 rechts) erfolgen.

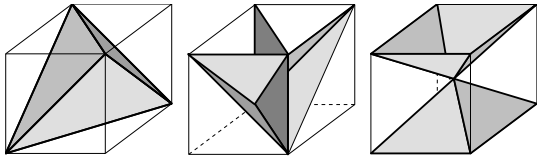

Abbildung 7.9: Tetraedisierung eines Hexaeders in 5, 6 oder 12 Tetraeder.

Die separate Abarbeitung der Zellen mit dem Standard-Marching-Cube Algorithmus birgt für die graphische Darstellung ein weiteres Problem. In jeder Zelle entstehen zusammenhanglose Polygonstücke, deren Knoten für benachbarte Zellen mehrfach im Gitter interpoliert und abgespeichert werden. Die Optimierung einer solchen Geometrie, beispielsweise durch Triangle-Strips, ist nur schwer realisierbar. Die zusätzlichen Eckpunkte belasten ebenfalls unnötig die Graphikhardware und führen zu einer geringeren Bildwiederholungsrate. Ein besseres Verfahren für die interaktive Visualisierung ist daher die propagative Isoflächenberechnung, deren Grundidee in [59] vorgestellt wurde.

Der Algorithmus startet in einer von der Isofläche geschnittenen Zelle und wird rekursiv in den Nachbarzellen fortgesetzt, deren Seitenflächen von der Isofläche geschnitten werden. Die zu untersuchenden Nachbarzellen werden zur Festlegung der Abarbeitungsreihenfolge in eine FIFO-Liste eingefügt. Die mehrfache Untersuchung einer Zelle wird durch das Setzen eines „Besucht“-Flags verhindert. Diese Vorgehensweise wird durch den folgenden Pseudocode beschrieben:

```
Initialisiere zellListeFIFO mit startZelle
WHILE (zellListeFIFO != NULL) {
   nimm aktuelleZelle aus zellListeFIFO
   IF ((aktuelleZelle noch nicht besucht) AND
       (Isofläche führt durch aktuelleZelle)){
           Berechne Isofläche in aktuelleZelle
           Markiere aktuelleZelle als besucht
           Bestimme Nachbarn, in denen sich die Isofläche fortsetzt
           Trage diese Nachbarn in zellListeFIFO ein
   }
   Entferne aktuelleZelle aus zellListeFIFO
}
```

Die berechnete Isofläche breitet sich ausgehend von der Startzelle konzentrisch im Volumen aus. Mit diesem Algorithmus werden jedoch nicht alle, sondern nur eine zusammenhängende Isofläche im Volumens bestimmt. Erst die Untersuchung aller Zellen stellt die Berechnung aller Isoflächen sicher. Der Aufwand hierfür ist mit dem propagativen Verfahren nicht höher als beim Standard-Marching-Cube, da durch das Setzen des „Besucht“-Flags jede Zelle nur einmal untersucht wird. Der Vorteil des Verfahrens ist das Wissen über die Zugehörigkeit der Polygone zu einzelnen Isoflächen. Allerdings liefert auch dieser Algorithmus noch nicht die optimale Isoflächen-Beschreibung, da die Nachbarschaftsbestimmung über die Zell-Seitenflächen keine Zuordnung der benachbarten Knoten einschließt.

Eine schnelle Optimierung der Geometrie ist möglich, wenn zu einem gegebenen Knoten die angrenzenden Isoflächenpolygone bekannt sind, wie es bei einer knotenorientierten Betrachtung der Isofläche der Fall ist. Als Basis für die knotenorientierte Bestimmung einer Isofläche werden die Zellkanten verwendet, da pro Kante nur ein Knoten der Isofläche existiert und da die Knoten einer Kante über eine eindeutige Nummerierung im Volumen leicht identifiziert werden können. Auf die Bestimmung des Bit-Musters und den Blick in die Lookup-Tabelle folgt bei der propagativen Isoflächenberechnung ein Vergleich der neuen Kanten mit den bereits berechneten Kanten.

Der Vergleich der Knotenindizes der Zellkanten ist schneller und genauer als der Vergleich von Knotenkoordinaten. Diese Vorgehensweise erlaubt die Generierung von Triangle-Strips und verhindert die Mehrfachberechnung von Isoflächenknoten.

Der daraus resultierende Algorithmus startet ebenfalls mit einer Saatzelle, die ein Isoflächenstück besitzt. In der Initialisierungsphase wird ein Schnittpunkt zwischen einer Zellkante und der Isofläche bestimmt und die Kante inklusive Schnittpunkt in einer Liste gespeichert. In den folgenden Schritten wird eine Kante aus der Liste genommen und für alle angrenzenden Zellen das Bit-Muster der Isofläche bestimmt. Jede von der Isofläche geschnittene Kante wird zuerst mit den Listeneinträgen verglichen und falls die Kante bereits in der Liste enthalten ist, kann anhand des gespeicherten Schnittpunktes die Isofläche ergänzt werden. Andernfalls wird die Kante mit dem neu berechneten Schnittpunkt in die Liste aufgenommen. Die betrachteten Zellen markiert der Algorithmus nun als besucht und die aus der Liste genommene Kante braucht nicht weiter behandelt werden, da alle angrenzenden Isoflächenstücke bestimmt wurden. Der Algorithmus fährt solange fort, bis in der Liste keine Kanten mehr enthalten sind. Der Pseudocode dieses kantenorientierten propagativen Isoflächenalgorithmus hat folgende Form:

```
Initialisiere kantenListe mit einer Kante der Startzelle
WHILE (kantenListe nicht leer){
   Nimm neueKante aus kantenListe
   FOR (alle Zellen, die neueKante enthalten) {
     IF (Zelle noch nicht besucht) {
        Markiere Zelle als besucht
        Berechne Isofläche in der Zelle
        Bestimme die schnittKanten zwischen Zelle und Isofläche
        FOR (alle schnittKanten)
          IF(schnittKante in kantenListe)
            Ergänze Isofläche um Schnittpunkt und verwerfe schnittKante
          ELSE
            Füge schnittKante in kantenListe ein
     }
   Entferne neueKante aus kantenListe
}
```

Die kantenorientierte Propagation einer Isofläche ist an einem Beispiel in Abbildung 7.10 skizziert. Dargestellt sind die Polygone der Isofläche und deren Schnittpunkte mit den Zellkanten. Die Zellkanten verlaufen senkrecht zur Bildebene und sind somit nur in den Eckpunkten sichtbar. Die Verwendung einer FIFO-Liste als Kantenliste erzeugt die spiralförmige Ausbreitung der Isofläche. Nach und nach wächst die Isofläche um die Saatzelle, bis das ganze Volumen ausgefüllt ist.

Sehr gut geeignet ist der kantenorientierte propagierende Algorithmus für die interaktive Visualisierung. Die Definition des Saatpunktes kann beispielsweise vom Anwender in einer virtuellen Umgebung ohne Verwendung einer Tastatur erfolgen. Hohe Bildwiederholungsraten können durch kurzes Stoppen der Berechnung für einen Renderingschritt garantiert werden. Durch das spiralförmige Wachsen bietet sich dem Anwender zu jedem Zeitpunkt der Berechnung eine zu-

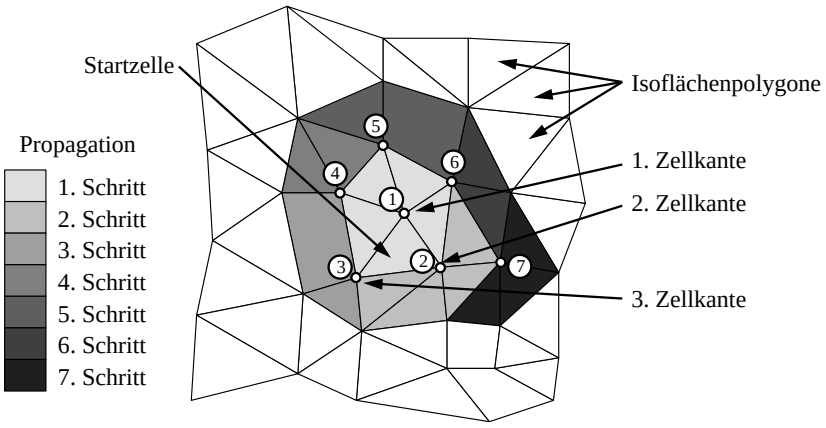

Abbildung 7.10: Kantenorientierte Isoflächenpropagation durch eine unstrukturiertes Volumengitter.

sammenhängende Isofläche, die nicht durch großflächige Überdeckungen unübersichtlich wird, wie es bei vollständig berechneten Isoflächen der Fall ist.

Der Vorteil der indizierten Dreiecksnetze beschränkt sich nicht nur auf eine Beschleunigung der graphischen Darstellung, sondern spart auch Speicherplatz. Neben den eigentlichen Koordinaten, kann auch bei den Knotennormalen des Gouraud-Shadings viel Speicher durch die Indizierung eingespart werden. Die Knotennormalen werden entweder durch Mittelung der Polygonnormalen oder durch Gradienten im Skalarfeld bestimmt.

7.1.4 Skalarwertinterpolation

Die Ergebnisdaten der Simulationen werden in den hier betrachteten Fällen knotenorientiert in den Eckpunkten des diskreten Gitters abgespeichert. Eine aufschlussreiche Visualisierung erfordert allerdings eine kontinuierliche Parameterverfügbarkeit im ganzen Volumen, die durch eine Interpolation erreicht werden kann.

Die reale Welt wird in der Simulation durch ein diskretes Gitter nachgebildet, dessen Datenwerte an den Gitterpunkten durch Interpolation bestimmt werden. In den wenigsten Fällen steht das Interpolationsverfahren der Simulation für die Visualisierung zur Verfügung, so wird mit einem geeignetem Verfahren für das Gitter versucht, eine bestmögliche Interpolation durchzuführen. Durch die Auswahl des Interpolationsverfahrens wird also schon ein schematischer Fehler in Kauf genommen.

Die trilineare Interpolation ist eine häufig eingesetzte Interpolationsmethode und eignet sich gut für uniforme Gitter. Für curvilineare und unstrukturierte Gitter ist die trilineare Interpolation allerdings nicht zu empfehlen, da der Interpolationsfehler für deformierte Zellen stark zunimmt.

Ein einfaches und robustes Verfahren für die hier verwendeten unstrukturierten Gitter ist das *Inverse Distance Weighting*. Es interpoliert den Parameter zu einer Position durch eine Gewichtung der Eckpunktdatenwerte der Zelle und ist im Gegensatz zu anderen Verfahren auf alle Zelltypen anwendbar (siehe Kapitel 2.2.2.2).

7.1.5 Gradientenberechnung

Gradienten geben in Skalarfeldern die Richtung der größten Werteänderung an. Die Gradienten werden für die Visualisierung und auch als Normalen des Gouraud-Shadings der Isoflächen benötigt. In den hier vorgestellten zeitabhängigen Luftdruckvolumen zeigen die Gradienten in die Bewegungsrichtung der Luftdruckwellen und helfen dem Ingenieur bei der Bewertung von Simulationsergebnissen.

Der Gradient ∇s eines Skalarfeldes s ist definiert als der Vektor der Richtungsableitungen entlang der Koordinatenachsen:

$$\nabla s = \begin{pmatrix} \frac{\partial s}{\partial x} \\ \frac{\partial s}{\partial y} \\ \frac{\partial s}{\partial z} \end{pmatrix} \tag{7.6}$$

Die Berechnung von Gradienten ist zu jeder Position im skalaren Feld erforderlich. Da die diskreten Gitter den Raum jedoch nur durch die Interpolation der Knotenwerte auffüllen, muss eine Annäherung an die partielle Ableitung im Skalarfeld durch eine Ableitung der Interpolationsmethode erfolgen.

In uniformen Gittern, mit zugrunde liegender trilinearer Interpolation, lassen sich Gradienten als die Differenzen zwischen benachbarten Knoten in Richtung der Koordinatenachsen berechnen. Verwendet werden können hierfür Vorwärts-, Rückwärts- oder zentrale Differenzen, bzw. Kombinationen daraus. Die Berechnung der Gradienten ist in diesem Fall konsistent zur Interpolation der skalaren Parameter. Auf unstrukturierte Gitter lässt sich dieses Verfahren jedoch nicht übertragen, da die Zellkanten im Allgemeinen nicht achsenparallel sind. Auch führt die Ableitung des *Inverse Distance Weighting* zu einer komplexen Funktion, deren Auswertung eine interaktive Visualisierung behindert.

Der hier verwendete Ansatz zur näherungsweisen Berechnung von Gradienten in unstrukturierten Skalarfeldern basiert auf der Annahme, dass die skalaren Werte an den Kanten und somit auch innerhalb der Zellen linear verteilt sind. In den linearen Bereichen eines Feldes ist der Gradient $\vec{g}$ konstant. Die Differenz zweier Skalarwerte $s(K_i) - s(P)$ lässt sich mit Hilfe des Gradientenvektors und der Differenz der Ortsvektoren der beiden Punkte $K_i - P$ ausdrücken:

$$\langle \vec{g}; K_i - P \rangle = s(K_i) - s(P) \tag{7.7}$$

An der Position P wird der Skalarwert $s(P)$ durch *Inverse Distance Weighting* zur Bestimmung des Gradienten $\vec{g}$ interpoliert. Die Skalarwerte $s(K_i)$ sind an den Gitterpunkten K_i durch die Simulationsergebnisse gegeben. Betrachtet man diese Beziehung für mehrere Knoten K_i, so erhält man ein im Allgemeinen überbestimmtes lineares Gleichungssystem für den Vektor $\vec{g}$. Durch eine Lösung dieses Gleichungssystems lässt sich der gesuchte Gradientenvektor $\vec{g}$ bestimmen.

Die Wahl der Punkte K_i hängt von der Position des Punktes P im unstrukturierten Gitter ab. Befindet sich P innerhalb einer Zelle, so werden alle Knoten der Zelle benutzt, um das Gleichungssystem aufzubauen, siehe Abbildung 7.11 links. Falls jedoch P mit einem Eckpunkt der

Zelle identisch ist, so müssen alle Knoten der vom Eckpunkt ausgehenden Kanten für die Berechnung des Gradienten benutzt werden, siehe auch Abbildung 7.11 rechts. Die Berechnung des Gradienten für die verschiedenen Zelltypen unterscheidet sich nur in der Größe des Gleichungssystems abhängig von der Anzahl der Zelleckpunkte.

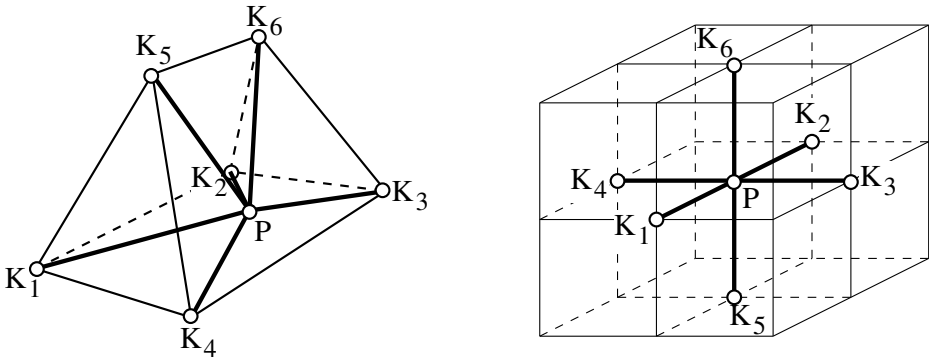

Abbildung 7.11: Knoten, mit denen das Gleichungssystem zur Berechnung des Gradienten aufgestellt wird.

Eine näherungsweise Lösung des Gleichungssystems bietet zum Beispiel das Standardverfahren des *General Least Square Fit*, das in der Standardliteratur ausführlich behandelt wird [91]. Dieses Verfahren überführt das Gleichungssystem durch Matrizenmultiplikationen in ein 3x3-Gleichungssystem, das dann mit einem beliebigen Verfahren, wie zum Beispiel der Gauß-Elimination oder LR-Zerlegung, gelöst werden kann. Ein beliebiger Gradient lässt sich also durch die Lösung eines 3x3-Gleichungssystems lösen. Der Aufwand hierfür ist verhältnismäßig gering und das Verfahren somit auch für interaktive Anwendungen geeignet.

7.2 Volumenproben als Interaktionselemente

Die direkte Interaktion mit den Fahrzeugkomponenten der geometriebezogenen Simulationen hat die Vorteile der intuitiven Analyse in einer virtuellen Umgebung gezeigt. Mit Hilfe der vorgestellten Visualisierungstechniken können auch zeitabhängige Luftdruckvolumen durch die Generierung von Geometrien interaktiv visualisiert werden. Zur Steuerung der Interaktionen für die Visualisierung wurden in dieser Arbeit die Volumenproben entwickelt.

7.2.1 Grundstruktur der Volumenproben

Volumenproben begrenzen den zu analysierenden Bereich des Volumens und erlauben somit den Einsatz von verschiedenen Visualisierungstechniken zur interaktiven Darstellung. Der momentane Geltungsbereich der Probe wird durch einen Drahtgitterrahmen markiert, in dessen Inneren die Visualisierung mittels einer zeitabhängigen Geometrie erfolgt. Die Probe lässt sich durch

angekoppelte Eingabegeräte transformieren und ermöglicht dem Benutzer so eine lokale Untersuchung des Volumens durch einfaches und intuitives Verschieben des Geltungsbereichs.

In einem *WorldToolKit*-Szenengraph besteht der Teilbaum einer Volumenprobe ebenfalls aus einem *Switch*-Knoten, unter dem zu jedem Zeitschritt die vom Visualisierungsalgorithmus generierte Geometrie hängt (siehe Abbildung 7.12). Im Gegensatz hierzu ist der Drahtgitterrahmen in allen Zeitschritten identisch und muss daher nur einmal als Geometrieobjekt instantiiert werden. Der Teilbaum einer Probe wird direkt dem Wurzelknoten zugeordnet und ein Wechsel des dargestellten Zeitschrittes muss daher synchron in den *Switch*-Knoten der Fahrzeuggeometrie und der Probe umgesetzt werden.

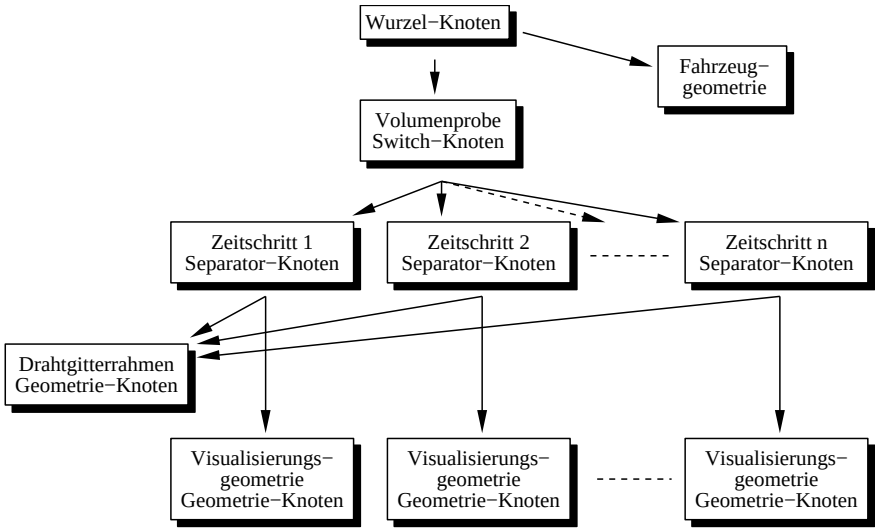

Abbildung 7.12: Szenengraph einer Volumenprobe.

Die Kopplung der Volumenprobe an ein Eingabegerät erfolgt durch die Integration in die *WorldToolKit*-Hauptschleife. Sobald eine Transformationsänderung für die Volumenprobe eintrifft, wird die Visualisierungsgeometrie in allen Zeitschritten aktualisiert. Zur Integration muss die Hauptschleife um die folgenden Punkte erweitert werden:

```
WHILE(1){
  Überprüfe Eingabegeräte
  IF(Neue Transformation für Volumenprobe)}
    Setze neue Positionsmatrix in der Volumenprobe
    Bestimme neue Visualisierungsgeometrie in allen Zeitschritten
  } Zeichne Szenengraphen
}
```

Die Aktualisierung der Visualisierungsgeometrie für alle Zeitschritte kann sehr aufwendig sein und in einer Single-Prozess-Applikation zum Stocken der Bildwiederholungsrate führen.

Eine Verteilung der Berechnungslast hingegen führt zu kontinuierlichen Frameraten. Beispielsweise ist die Aktualisierung aller Zeitschritte bei ständigen Benutzereingaben nicht erforderlich. Die Beschränkung der Berechnungen innerhalb eines Schleifenzyklus auf den aktuellen Zeitschritt resultiert bereits in einer merklich flüssigeren Darstellung. Die restlichen Zeitschritte werden in den folgenden Schleifenzyklen aktualisiert, sobald keine weiteren Benutzereingaben zur Transformation der Volumenprobe erfolgen. Ein abruptes Stocken der Visualisierung wird somit verhindert, allerdings sinkt die Framerate über einen längeren Zeitraum.

Die Aufteilung der Geometrieberechnung eines Zeitschrittes über mehrere Schleifenzyklen liefert noch adäquatere Frameraten und garantiert bei Vorgabe einer maximalen Rechenzeit für den Visualisierungsalgorithmus eine konstante Framerate. Insofern sich der Visualisierungsalgorithmus parallelisieren lässt, bietet sich auf Mehrprozessorrechnern auch die Verteilung der Arbeit auf verschiedene Prozesse an.

Für die Visualisierung von Akustiksimulationen wurden die vorgestellten Vorgehensweisen untersucht, in *VtCrash* integriert und im Fahrzeugentwicklungsprozess erprobt.

7.2.2 Typen von Volumenproben

Vier verschiedene Volumenprobe-Typen sind entsprechend der Dimensionalität der Unterräume des dreidimensionalen Raums möglich. An der Anzahl der Dimensionen orientieren sich die Möglichkeiten der Visualisierungsalgorithmen. Die verschiedenen Volumenprobe-Typen dieser Arbeit sind in Abbildung 7.13 dargestellt.

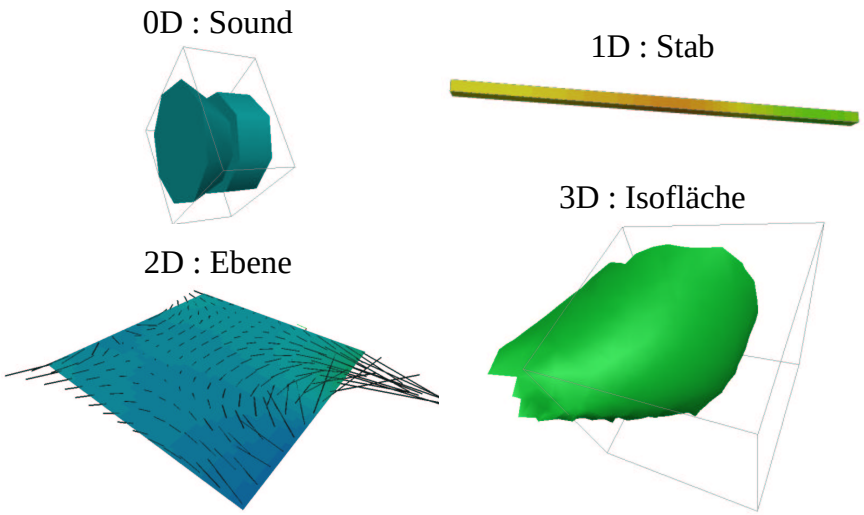

Abbildung 7.13: Null-, ein-, zwei- und drei-dimensionale Volumenproben.

7.2.2.1 Soundprobe

Die einfachste Variante eines Interaktionselementes ist die 0-dimensionale Soundprobe, bestehend aus einem einzelnen Abtastpunkt. Die Idee der Soundprobe ist die Ausnutzung des menschlichen Hörvermögens für die Visualisierung, denn ähnlich wie die Farberkennung des Auges, kann die Frequenz- und Lautstärkenwahrnehmung des Ohrs angeregt werden. Die Soundprobe eröffnet eine neue Variante der Analyse. Da sie dem Anwender über das Gehör Ergebnisse vermittelt, kann parallel dazu visuell ein völlig anderer Bereich analysiert werden. Es lassen sich somit schneller Zusammenhänge im Gesamtmodell erkennen.

Die umgebende Zelle des Soundproben-Mittelpunktes wird mit der iterativen Zellsuche bestimmt und der Skalarwert durch *Inverse Distance Weighting* interpoliert. Der so gewonnene Wert wird als Farbe auf die Geometrie der Probe und auf die Tonfrequenz gemappt. Die Bestimmung eines einzelnen Wertes im Akustikvolumen ist nicht aufwendig und hat somit keine starken Auswirkungen auf die Bildwiederholungsrate.

Die Erzeugung eines akustischen Signals ist jedoch durch die geringe Funktionalität von *WorldToolKit* stark eingeschränkt. Es hat sich herausgestellt, dass die Frequenzmodulation eines Tones nicht möglich ist. Die angestrebte Abbildung von verschiedenen Skalarwerten kann also nur über separat geladene Sounddateien erfolgen. In *WorldToolKit* können jedoch nur 16 verschiedene Tonstücke parallel gehalten werden. Die so zur Verfügung stehende Menge ist leider nicht ausreichend, um eine sinnvolle Skalarwert-Repräsentation für eine Analyse im produktiven Einsatz zu bieten, aber es demonstriert sehr gut die Möglichkeiten der Technik.

7.2.2.2 Stabprobe

Die Darstellung der Skalarwerte entlang einer Linie hilft dem Ingenieur beim Auffinden von stehenden Wellen, die Indizien für Lärmentwicklung sind. Solche lokalen Merkmale des Volumens lassen sich gut durch die Kombination von interaktiver Positionierung der Stabprobe und Ingenieurserfahrung bestimmen.

Die realisierte Stabprobe tastet in vorgegebenen Abständen das Volumen entlang einer Linie ab. Entsprechend den Abtastwerten wird die dreidimensionale Repräsentation des Stabes (siehe Abbildung 7.13) eingefärbt und zur besseren quantitativen Auswertung werden die Parameter zusätzlich in ein 2D-Diagramm, dem *VtCrash Sketch Pad*, abgebildet (siehe Abbildung 7.14). Im

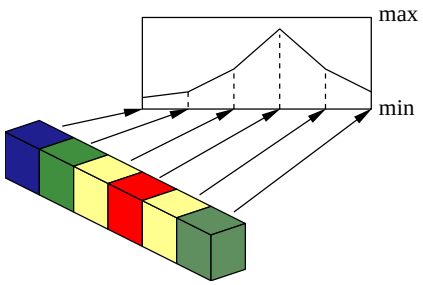

Abbildung 7.14: Abbildung der Skalarparameter als Farbe und im *Sketch Pad*.

Sketch Pad lässt sich auch sehr gut das zeitabhängige Verhalten entlang der Stabprobe bewerten.

Die Skalarwerte für die Stabprobe werden ebenfalls mit der iterativen Zellsuche und dem *Inverse Distance Weighting* Interpolationsverfahren bestimmt. Die Anordnung der Abtastpunkte auf einer Linie führt zu einer Beschleunigung der Suche, wenn die zuletzt bestimmte Zelle als Ausgangspunkt für die nächste Zellsuche verwendet wird. Der geringe Abstand zwischen den Punkten der Linie führt zu einer geringen Anzahl benötigter Iterationsschritte und somit zu einer schnellen Skalarwertbestimmung.

In Abbildung 7.15 ist der Einsatz der Stabprobe vor den Fahrzeugfenstern zum Auffinden einer stehenden Welle dargestellt. Anhand der Interaktion zwischen der schwingenden Geometrie und den Farbänderungen der Stabprobe in der kombinierten Visualisierung kann der Ingenieur die Entstehung der Welle analysieren.

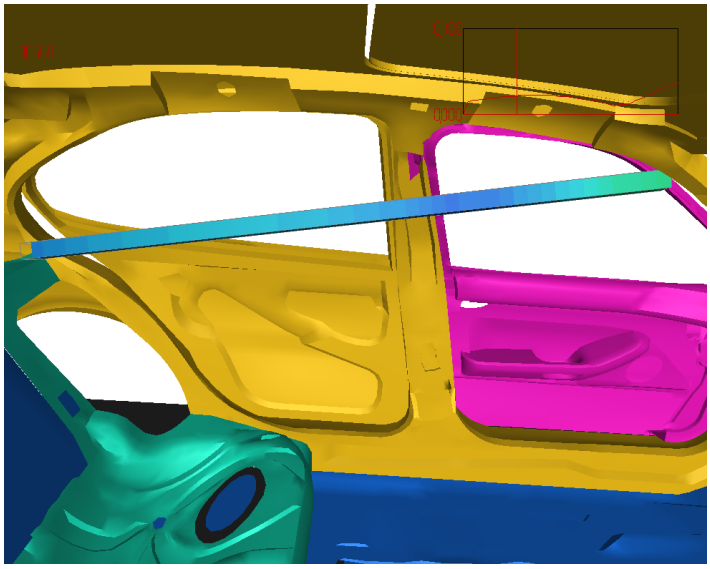

Abbildung 7.15: Positionierung der Stabprobe vor den Fahrzeugfenstern.

Die Stabprobe hat sich durch ihre leichte Positionierung und Handhabung als ein sehr hilfreiches Analyseinstrument erwiesen. Innerhalb eines Schleifenzyklus können aufgrund der geringen Anzahl von Abtastpunkten alle Skalarwerte berechnet und eine hohe Interaktivität zugesichert werden.

7.2.2.3 Ebenenprobe

Eine gängige Methode zur Analyse von simulierten Volumina ist die Betrachtung von zweidimensionalen Schnitten. Der Ingenieur analysiert anhand der zumeist achsenparallelen Schnitte die Intensität und Form der Luftdruckwellen. Allerdings können in traditionellen Postprozessoren die Schnittebenen nicht interaktiv transformiert werden.

In dieser Arbeit wurde eine freibewegliche Ebenenprobe entwickelt, die dem Ingenieur die gewohnte Analyse durch Schnitte bietet, aber darüber hinaus die Vorteile der Interaktivität aus-

nutzt. Die Ebenenprobe lässt sich schnell und präzise an den kritischen Stellen im Berechnungsmodell platzieren und durch die freie Beweglichkeit in jeder Dimension gewinnt der Anwender schnell einen räumlichen Eindruck vom Volumen.

Die Ebenenprobe besteht aus $n * m$ planaren Ebenenstücken, den so genannten Patches (siehe Abbildung 7.16). Der Anwender kann Größe und Anzahl der Patches definieren und somit die Ebenenprobe den Bedürfnissen der Analyse anpassen. Im Mittelpunkt eines jeden Patches wird der Skalarwert durch Zellsuche und Interpolation bestimmt und das Patch im anschließenden Mappingprozess konstant eingefärbt. Die Größe der Patches ist somit entscheidend für die Genauigkeit und Geschwindigkeit der Visualisierung, so ist bei kleiner Patchgröße zwar sichergestellt, dass jede von der Ebene geschnittene Volumenzelle mindestens einen Abtastpunkt besitzt, jedoch wird die Darstellung durch die Berechnung von vielen Abtastpunkten beliebig langsam. Zur Erzielung einer schnellen Visualisierung muss der Anwender durch Anpassungen die optimale Patchgröße herausfinden.

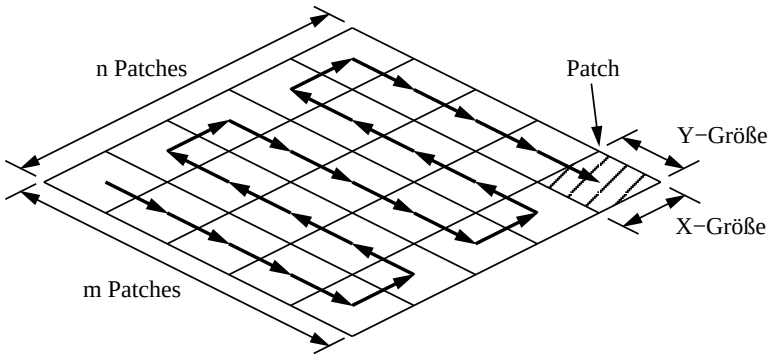

Abbildung 7.16: Optimale Abarbeitungsreihenfolge der Patches einer Ebenenprobe.

Die Anzahl der Abtastpunkte einer Ebenen ist gegenüber der Stabprobe um ein Vielfaches größer und erfordert eine schnelle Zellsuche für die interaktive Berechnung der Skalarwerte. Für die Ebene kann die Zellsuche durch die in Abbildung 7.16 dargestellte Suchreihenfolge beschleunigt werden. Die iterative Zellsuche startet immer von der zuletzt bestimmten Zelle und durch die Wahl der Reihenfolge wird der Abstand der aufeinander folgenden Abtastpunkte minimiert. Die Verwendung der Abtastreihenfolge spart somit einige Iterationen ein und führt zu einer merklichen Beschleunigung der Darstellung.

Ein zentraler Punkt der Ergebnisanalyse ist die Bestimmung der Ausbreitungsrichtung von Luftdruckwellen im Fahrzeuginneren. Der Ingenieur kann eine erste Aussage hierüber durch die animierte Darstellung der Skalarwerte als Farbkodierung auf der Ebenenprobe treffen. Anhand der zeitlichen Farbänderungen bekommt er einen Eindruck vom Verhalten des Luftdruckvolumens. Präzise Aussagen über die Richtung und Geschwindigkeit der Wellenausbreitung können auf Grundlage der Farben allerdings nicht getroffen werden. Die Berechnung von Gradienten, wie in Kapitel 7.1.5 beschrieben, bietet hier Abhilfe. Der Gradient eines Skalarfeldes zeigt immer in die Richtung der größten Steigung und steht somit senkrecht auf Isolinien bzw. Isoflächen. Im Luftdruckvolumen zeigt der Gradient in die Ausbreitungsrichtung der Luftdruckwellen, da der

Druck innerhalb einer Welle annähernd konstant ist.

An den Abtastpunkten der entwickelten Ebenenprobe können zusätzlich die Gradienten als Pfeile visualisiert werden. Zur Wahrung der Übersichtlichkeit wird nicht für jedes Patch ein Gradient berechnet, da der Anwender ansonsten durch mögliche Überlagerungen der Pfeile irritiert wird. Die Gradienten können sowohl als zwei- oder dreidimensionale Vektoren dargestellt werden, wobei die zweidimensionalen Gradientenvektoren durch Projektion in die Ebene der Interaktionsprobe entstehen und senkrecht zu den Isokonturen der Farbkodierung verlaufen. Die Betrachtung der dreidimensionalen Vektoren erfordert eine Stereodarstellung am Bildschirm, da nur so die Richtungsinformation der Gradienten vollständig vom Anwender erfasst werden kann. Am Standardarbeitsplatz ohne Stereowiedergabe kann die dritte Dimension der Gradienten nur erahnt werden und sie verlieren so an Aussagekraft. Der Unterschied zwischen den zwei- und dreidimensionalen Gradienten einer Ebenenprobe wird in Abbildung 7.17 deutlich. Die eigentliche Richtung der dreidimensionalen Gradienten kann in der Abbildung nur vermutet werden.

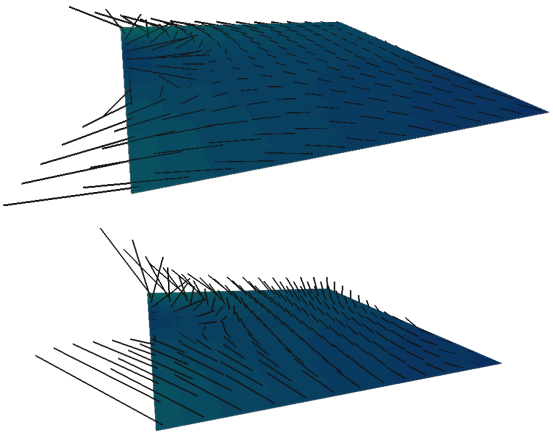

Abbildung 7.17: Gegenüberstellung von zwei- (oben) und dreidimensionalen (unten) Gradienten einer Ebenenprobe.

In Abbildung 7.18 ist die Ebenenprobe in einem typischen Einsatzszenario dargestellt. In dem Beispiel schwingt der Fensterrahmen der Fahrertür und initiiert im Akustikvolumen Druckwellen. Die Ebenenprobe ist so am Rahmen positioniert, dass sie senkrecht zu den Luftdruckwellen steht. Die interpolierten Skalarwerte werden als Farbkodierung und die Ausbreitungsrichtung mit Hilfe von zweidimensionalen Gradienten dargestellt. Anhand der Ebenenprobe erkennt man deutlich, wie Druckwellen ausgehend vom Rahmen in das Volumen initiiert werden. Die sichtbaren Löcher auf der Ebenenprobe markieren die Hohlräume der Sitze. Dank der verwendeten Algorithmen, lässt sich die Ebene interaktiv durch das Volumen bewegen.

7.2.2.4 Isoflächenprobe

Die Stab- und die Ebenenprobe können durch die Beschränkung auf eine bzw. zwei Dimensionen keinen echten dreidimensionalen Eindruck des Luftdruckvolumens vermitteln. Techniken

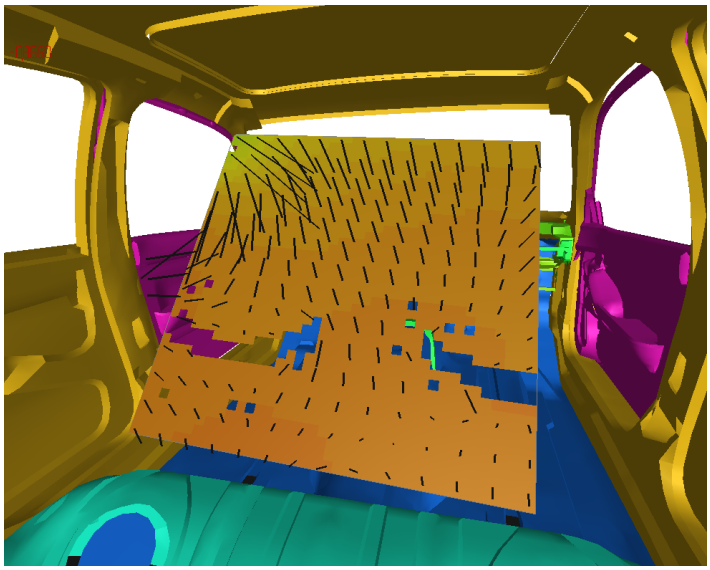

Abbildung 7.18: Visualisierung der Skalarwerte als Farbkodierung und der Gradienten als Pfeile auf der Ebenenprobe.

zur dreidimensionalen Visualisierung von Volumen sind beispielsweise das direkte Volumenrendering oder die Berechnung von Isoflächen. Direktes Volumenrendering ist momentan auch mit spezialisierter Hardware nur mit 2 bis 5 Bildern pro Sekunde in Stereodarstellung möglich. Da in dieser Arbeit die Interaktivität im Vordergrund steht, scheidet direktes Volumenrendering als Visualisierungstechnik aus.

Die Berechnung von Isoflächen in einem unstrukturierten Gitter kann durchaus mehr als eine Sekunde beanspruchen. Um interaktive Bildwiederholungsraten zu erreichen, wurde in *VtCrash* der in Kapitel 7.1.3 beschriebene Algorithmus für propagative Isoflächen verwendet. Die Berechnung einer Isofläche ist auf mehrere Schritte aufgeteilt, damit eine minimale Bildwiederholungsrate nicht unterschritten wird. Die vom Benutzer vorgegebene Zeitscheibe steht dem Algorithmus pro Schleifenzyklus zur Verfügung.

Der Saatpunkt des Algorithmus wird interaktiv durch den Mittelpunkt der Volumenprobe definiert. Optional kann die Ausbreitung der Isofläche durch den Drahtgitterrahmen der Volumenprobe begrenzt oder im gesamten Fahrzeuginnenraum berechnet werden. Für die Abbildung 7.19 wurde die Isoflächenprobe am Fensterrahmen positioniert und vermittelt dem Ingenieur die Form und Ausdehnung der initiierten Luftdruckwelle.

7.2.3 Beschleunigung der Visualisierung durch Multiprocessing

Die drei zeitaufwendigsten Schritte eines Schleifenzyklus sind die Berechnungen der Visualisierungsalgorithmen, die Generierung von Geometrie und das Rendering des Szenengraphen. Die vorgestellten Visualisierungsalgorithmen sind bereits auf Geschwindigkeit optimiert, jedoch müssen sie sich auf einem Single-Prozessor-System die CPU-Leistung mit dem Renderingalgo-

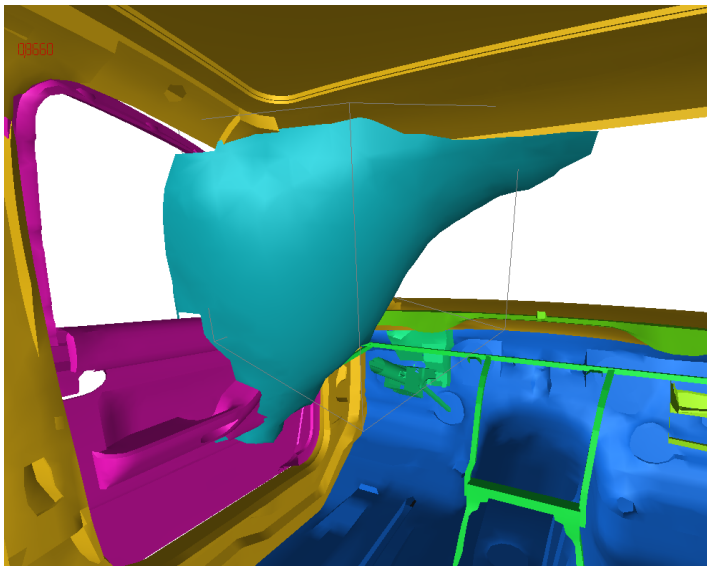

Abbildung 7.19: Aus dem Saatpunkt der Volumenprobe berechnete Isofläche.

rithmus teilen.

Auf Mehrprozessorsystemen kann durch die Aufteilung der Arbeit auf mehrere Prozesse eine Steigerung der Bildwiederholungsrate erzielt werden. Die Unabhängigkeit von Visualisierungsalgorithmus und Rendering ist eine sinnvolle Aufteilung der Arbeit, die asynchron in verschiedenen Prozessen abgearbeitet werden kann. Jedoch muss die Aktualisierung des Szenengraphen zum Abgleich der Prozesse in einem synchronisierten Arbeitsschritt stattfinden (siehe Abbildung 7.20). Je mehr Geometrie im Szenengraphen vorhanden ist und je mehr visualisiert wird, desto größer ist der Vorteil der Parallelisierung.

Implementiert wurde die Parallelisierung in *VtCrash* mit der *pThread*-Bibliothek für leichtge-

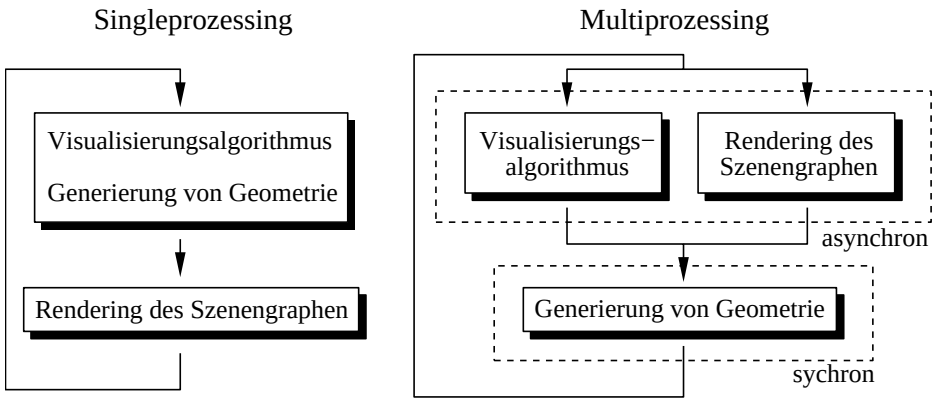

Abbildung 7.20: Single- und Multiprocessing im Visualisierungssystem.

wichtete Prozesse [87]. Das Rendering erfolgt im Hauptprozess, der mehrere Arbeitsprozesse zur Visualisierung startet und verwaltet. Den Arbeitsprozessen steht eine vorgeschriebene Zeitspanne zur Verfügung, die der gewünschten Bildwiederholungsrate entspricht und in der sie ungebremst arbeiten können. Sollen beispielsweise 10 Hz erzielt werden, so hat jeder Arbeitsprozess maximal eine Zehntelsekunde für den Visualisierungsalgorithmus.

Die verwendete Szenengraph-API *WorldToolKit* erlaubt leider keinen parallelen Zugriff mehrerer Prozesse auf den Szenengraph und es ist ferner auch nicht möglich, parallel Geometrie zu erstellen bzw. zu editieren. Diese Umstände behindern stark eine effiziente Parallelisierung. Zur Lösung des Problems ist die Erstellung von Geometrie und der Szenengraph-Zugriff auf den Hauptprozess beschränkt. Die Arbeitsprozesse legen die Ergebnisse der Visualisierungsalgorithmen in Datenlisten ab, aus denen der Hauptprozess in der synchronisierten Phase schnell Geometrie erzeugen kann.

Für die Stab- und Ebenenprobe wurde eine parallele Visualisierung durch die Auslagerung der Zellsuche, der Interpolation und der Gradientenberechnung in asynchrone Arbeitsprozesse realisiert. Die Berechnungsergebnisse der Skalarwerte und Gradienten werden in Datenlisten abgespeichert, deren Felder jeweils den Ergebnissen eines Abtastpunktes entsprechen (siehe Abbildung 7.21). Da jedem Arbeitsprozess nur eine begrenzte Zeit zur Verfügung steht, wird in einer weiteren Liste der Status des Abtastpunktes gespeichert. Ein Feld der Liste kann entweder *berechnet*, *nicht berechnet* oder *ungültig* sein. Für einen *ungültigen* Abtastpunkt wurde die Berechnung im Arbeitsprozess noch nicht gestartet, während ein gerade in der Berechnung befindlicher Abtastpunkt als *nicht berechnet* markiert ist. Der Hauptprozess generiert anhand der Status- und Datenlisten anschließend die entsprechende Geometrie, wobei *nicht berechnete* oder *ungültige* Abtastpunkte halbtransparent dargestellt werden und sich so deutlich von den *berechneten* Ergebnissen unterscheiden.

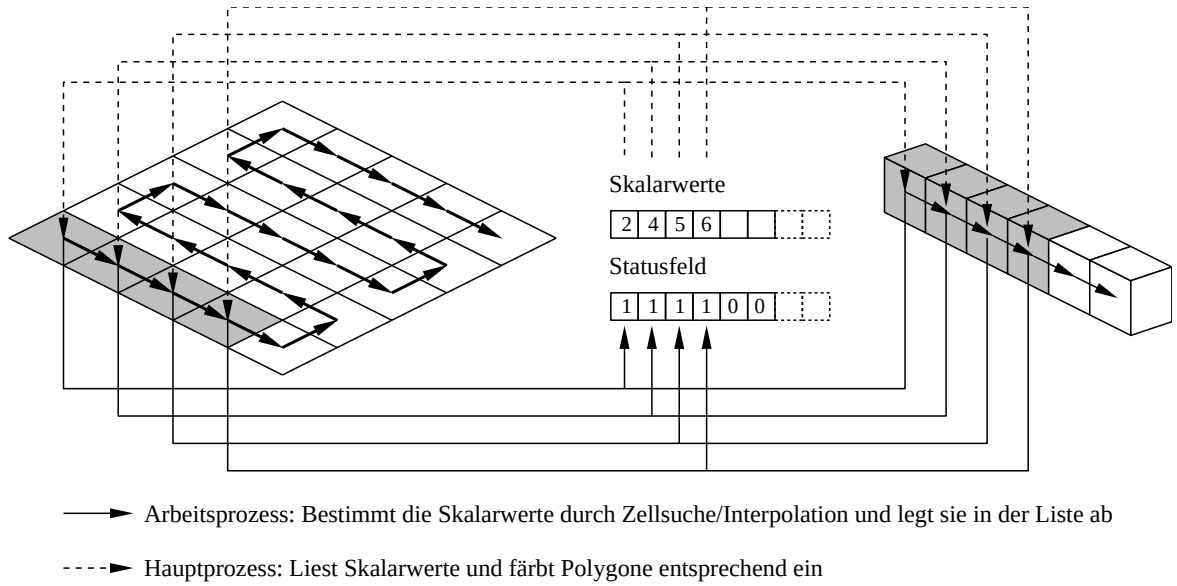

Abbildung 7.21: Arbeitsgebiete des Haupt- und der Arbeitsprozesse.

Bewegt der Anwender die Interaktionsprobe, so werden die Statuslisten durch den Hauptprozess mit *ungültig*-Einträgen initialisiert und alle Abtastpunkte von den Arbeitsprozessen erneut berechnet. Sobald die Statusliste nur noch *berechnet*-Einträge enthält, ist die Arbeit der asynchronen Prozesse beendet. Der folgende Pseudocode skizziert die Vorgehensweise der Arbeitsprozesse:

```
FUNCTION CHECK_BERECHNUNGSZEIT{
   IF(Berechnungszeit verbraucht){
     Warte auf Freigabe durch Hauptprozess
     IF(Statusliste initialisiert)
       GOTO Suche ersten ungültig Eintrag in der Statusliste
}  }

WHILE(Interaktionsprobe des Arbeitsprozesses aktiv){
   Suche ersten ungültig-Eintrag in der Statusliste
   IF(kein ungültig-Eintrag vorhanden)
     CHECK_BERECHNUNGSZEIT()
   Setze Statusfeld auf nicht berechnet
   Bestimme Koordinaten des Abtastpunktes
   Führe Zellsuche für den Abtastpunkt durch
   CHECK_BERECHNUNGSZEIT()
   Interpoliere Skalarwert
   CHECK_BERECHNUNGSZEIT()
   Berechne Gradienten
   Trage Ergebnisse in die entsprechenden Listen ein
   Setze Statusfeld des Abtastpunktes auf berechnet
   CHECK_BERECHNUNGSZEIT()
}
```

Zwischen den einzelnen Berechnungsschritten des Visualisierungsalgorithmus überprüft der Arbeitsprozess mehrfach die verbleibende Zeit. Falls seine Berechnungszeit verbraucht ist, wartet er bis der Hauptprozess den Szenengraphen gezeichnet hat und ihm neue Arbeitszeit zuordnet. Die geringe Anzahl an Arbeitszeittests birgt die Gefahr, dass ein Arbeitsprozess für eine Berechnungsschritt mehr Zeit benötigt als ihm zusteht. Die durchschnittliche Zeit der Berechnungsschritte ist jedoch so gering, dass eine eventuelle Zeitüberschreitung nicht ins Gewicht fällt.

Der zusätzliche Verwaltungsaufwand des Multiprocessings darf nicht größer sein als die Berechnungszeit der Arbeitsprozesse, da ansonsten die Bildwiederholungsrate sinken kann. Aus diesem Grund ist eine parallelisierte Visualisierung auch nur auf Mehrprozessorsystem effizient einsetzbar.

7.3 Ergebnisse

Die subjektive Beurteilung der interaktiven Volumenvisualisierung mit *VtCrash* durch Berechnungsingenieure war durchweg positiv, da sich die Volumenproben unter Verwendung der Space-

mouse mit interaktiven Bildwiederholungsraten leicht im Raum positionieren lassen. Um die Ergebnisse auch objektiv zu erfassen, wurde an einem geeigneten Beispieldatensatz eine Reihe von Performanzmessungen auf verschiedenen Rechnern durchgeführt.

7.3.1 Testumfeld für Performanzmessungen

Der verwendete Datensatz entstand aus einer Schwingungsanalyse eines Fahrzeuges mit anschließender Simulation des Luftdrucks im Fahrgastinnenraum, gerechnet mit der Simulationssoftware *NASTRAN*. Die Schwingung der Karosserie wurde durch eine konstante Motordrehzahl initiiert.

Das diskretisierte Gitter der Karosseriegeometrie enthält ca. 100.000 Finite-Elemente. Die Fahrgastzelle wurde automatisiert mit einem unstrukturierten Gitter bestehend aus ca. 20.000 Tetraedern, Pyramiden, Prismen und Hexaeder ausgefüllt. Alle angesprochenen Problemfälle für die Zellsuche (Löcher im Volumen etc.) kommen im Datensatz vor.

Die Performanzmessungen wurden auf drei leistungsfähigen *SGI*-Workstations durchgeführt: auf einer *Octane* SE mit einer Graphikleistung von 1.3 Millionen Polygone/sek dem typischen Arbeitsplatzrechner eines Berechnungsingenieurs entsprechend, auf einer Doppelprozessor *Octane* SE (1.5 Millionen Polygone/sek) für Projektionsräume und als Beispielsrechner für ein VR-Labor auf einer *ONYX2* mit BaseReality Graphiksystem (5 Millionen Polygone/sek). PCs und Rechner mit geringerer Graphikleistung wurden in dieser Untersuchung nicht berücksichtigt, da ihre Leistung für interaktives Arbeiten nicht ausreicht.

7.3.2 Performanzvergleich verschiedener Rechner

Zum Vergleich der Performanz wurden Untersuchungen mit einem Test-Setup auf den verschiedenen Rechnern durchgeführt. Das Visualisierungsszenario enthielt die äußere Hülle des Beispieldatensatzes und eine Ebenenprobe mit 30x30 Patches.

Für das Single-Prozess-Konzept wurden Messungen für die beiden Extremfälle der Ebenenprobe-Berechnung durchgeführt. Die minimale Bildwiederholungsrate ist bei einer Berechnung der Parameter für alle Abtastpunkte zu erwarten und die maximale, wenn sich die Probe außerhalb des Volumens befindet und so für alle Abtastpunkte der triviale Ablehnungsfall eintritt. Die Messungen erfolgten durch Positionierung der Ebenenprobe zum einen im Mittelpunkt und zum anderen außerhalb des Volumens.

Dem Prozess wurde für die Berechnung der Abtastpunkte eine Arbeitszeit von 20 ms zur Verfügung gestellt. Gemessen wurde neben der Bildwiederholungsrate für die beiden Positionen der Ebenenprobe auch die Anzahl der berechneten Abtastpunkte und die Geschwindigkeit ohne Darstellung der Ebenen. Da die Geschwindigkeit der Berechnung stark von der Anzahl der geladenen Zeitschritte abhängt, wurden zwei identische Untersuchungen für 20 und 40 geladene Zeitschritte durchgeführt. Die Ergebnisse der Performanzuntersuchung sind in Tabelle 7.1 aufgelistet. Wie erwartet hängt die Anzahl der in der vorgegebenen Zeit berechneten Abtastpunkte von der Leistung des Prozessors ab, somit erzielt die *Octane* (250 MHz) bei 20 geladenen Zeitschritten mit 225 berechneten Abtastpunkten das beste Ergebnis. Ohne den Einsatz der Ebenenprobe

Rechner	Anzahl berechneter Abtastpunkte		Bildwiederholungsraten min		max		ohne Ebene	
geladene Zeitschritte	20	40	20	40	20	40	20	40
Octane (175 MHz)	150	115	6,0	4,0	10	7,5	16	16
Octane (2x250 MHz)	225	180	6,4	4,2	13	10,0	18	18
ONYX2 (2x195 MHz)	202	172	8,0	5,0	22	12,5	36	36

Tabelle 7.1: Messergebnisse der Visualisierung mit der Ebenenprobe auf verschiedenen Rechnern.

ist die reine Graphikleistung ausschlaggebend für die Bildwiederholungsrate. In diesem Fall kann die *ONYX2* ihre doppelte Performanz gegenüber den *Octanes* klar umsetzen.

Sind 40 Zeitschritte geladen, so werden bei gleicher Anzahl an Zellsuchen doppelt so viele Interpolationen pro Abtastpunkt benötigt. Da die Zellsuche der zeitaufwendigste Schritt ist, findet keine Halbierung der Anzahl an berechneten Abtastpunkten pro Zyklus statt, sondern nur eine Reduktion um ca. 20%.

Ein sehr interessantes Ergebnis liefern die Werte der minimalen Bildwiederholungsrate, deren Bestimmung die maximale Leistung von Prozessor und Graphikhardware erfordern. Die ausgewogene Prozessorleistung der Rechnersysteme schmälert den Graphikleistungsvorteil der *ONYX2*. Die Bildwiederholungsrate der *ONYX2* liegt lediglich 25% über den *Octanes*. Erst wenn die Ebene nicht mehr bewegt wird, kann die *ONYX2* die doppelte Leistung bringen. Die Verwendung von mehreren Prozessen wird ebenfalls den Vorsprung der *ONYX2* verstärken.

7.3.3 Performanzverlust durch die Berechnung von Gradienten

Die Ebenenprobe kann zusätzlich zur Farbkodierung der Skalarwerte auch Gradienten als Pfeile visualisieren. Mit einer Testreihe wurde untersucht, in wieweit ein Performanzverlust durch die Berechnung der Gradienten auftritt. In Tabelle 7.2 sind die Ergebnisse dieser Untersuchung aufgeführt. Es wurden die Bildwiederholungsraten für eine 30x30 Ebenen mit 20 ms Berechnungszeit auf der *Octane* (2x250 MHz) gemessen. Die Ebene wurde hierfür mittig im Volumen platziert. Die Bildwiederholungsraten wurden gemessen für die Visualisierung bestehend aus der Farbkodierung der Skalarwerte ohne Gradienten- bzw. mit Gradientenberechnung und in einer weiteren Untersuchung nur mit Gradienten.

Geladene Zeitschritte	Anzahl berechneter Abtastpunkte	Bildwiederholungsraten ohne Gradienten	mit Gradienten	nur Gradienten
20	220	6,4	6,0	6,2
40	188	4,2	3,8	4,0

Tabelle 7.2: Performanzverlust durch Gradientenvisualisierung.

Wie die Messergebnisse deutlich zeigen, hält sich der Performanzverlust durch die zusätzliche Gradientberechnung in Grenzen. Die Bildwiederholungsrate sinkt bei 20 geladenen Zeitschritten um ca. 7% und bei 40 um ca. 10%. Im Vergleich zwischen Visualisierung ohne Gradienten und nur Gradienten wird der geringe Unterschied in der Bildwiederholungsrate deutlich. Beide Visualisierungstechniken benötigen annähernd gleich viel Zeit und somit wird auch hier wieder deutlich, dass die Zellsuche und das Rendern der Geometrie die zeitaufwendigsten Schritte sind.

7.3.4 Auswirkung der Berechnungszeit auf die Bildwiederholungsraten

Die Skalarwerte und Gradienten an den Abtastpunkten der Interaktionsproben werden in der Berechnungszeit durch Zellsuche und Interpolation im Akustikvolumen berechnet. Die Visualisierungsalgorithmen und der Renderingprozess müssen sich die Prozessorzeit teilen und so führt eine längere Berechnungszeit zu geringeren Bildwiederholungsraten. Der Grad der Interaktivität kann also vom Anwender über die Vorgabe der Berechnungszeit gesteuert werden. Diese Steuerung erfolgt allerdings nur indirekt, da ebenfalls die Komplexität der Szene und des Akustikvolumens eine entscheidende Rolle spielen. Auch die Anzahl der gezeichneten Polygone und die Position der Interaktionsprobe beeinflussen die Geschwindigkeit.

Das hier aufgeführte Beispiel kann also nur eine Richtung vorgeben, wie sich das Visualisierungssystem für vorgegebene Berechnungszeiten verhält. Für die Messungen wurde erneut die 30x30 Ebenenprobe ohne Gradientendarstellung mittig im Volumen positioniert und zum Vergleich 20 bzw. 40 Zeitschritte in den Speicher der *Octane* (2x250 MHz) geladen.

Tabelle 7.3 enthält die Messergebnisse der Untersuchungen. Wie erwartet, sinkt mit kürzerer Berechnungszeit die Anzahl der berechneten Abtastpunkte und die Bildwiederholungsrate steigt. Bei einer Berechnungszeit von 50 ms werden alle 900 Abtastpunkte der Ebenenprobe vollständig berechnet und sind somit immer aktuell zu der Position der Ebene. Allerdings kann bei 2,4 bzw. 1,2 Bildern pro Sekunde nicht mehr von interaktiver Positionierung gesprochen werden. Eine sehr gute Interaktivität mit 11,6 bzw. 8,0 Hz wird bei einer Berechnungszeit von 10 ms erreicht, jedoch dauert die vollständige Berechnung dann 1,5 bzw. 4,5 Sekunden. Ein guter Kompromiss ist eine Berechnungszeit von 20 ms. Mit einer Bildwiederholungsrate von 6,4 bzw. 4.2 kann die Ebene noch bewegt werden und es muss nicht allzulange auf die vollständige Aktualisierung ge-

Berechnungszeit ms	Anzahl berechneter Abtastpunkte		Bildwiederholungsrate	
Geladene Zeitschritte	20	40	20	40
50	900 (1260)	900 (1200)	2,4 (1,6)	1,2 (1,0)
40	590	550	3,3	1,8
20	225	180	6,4	4,2
10	50	25	11,6	8,0

Tabelle 7.3: Zusammenspiel von Berechnungszeit und Bildwiederholungsrate.

wartet werden. Sobald alle Abtastpunkte berechnet sind, schnellt die Bildwiederholungsrate auf 18 Bilder pro Sekunde hoch. Durch die Wahl der Berechnungszeit kann der Anwender also die Visualisierung den Bedürfnissen anpassen. Die hier aufgeführten Messergebnisse geben jedoch nur einen Anhaltspunkt und müssen für jeden Datensatz und Rechner erneut bestimmt werden.

Im Vergleich zur Ebenenprobe erfordert die Verwendung der Stabprobe eine deutlich geringere Rechnerleistung. Für eine sinnvolle Visualisierung mit der Stabprobe sollten mindestens 50 Abtastpunkte für die Linie berechnet werden. Auf jedem der verwendeten Rechner ist eine interaktive Visualisierung mit der Stabprobe möglich, auch wenn die *Octane* (175 MHz) mit 8,5 Hz knapp unter der 10 Hz-Grenze liegt. Die schnelleren Rechner erreichen mit einer Berechnungszeit von 10 ms die vollständige Aktualisierung der Stabprobe innerhalb eines Schleifenzyklus bei Bildwiederholungsraten von 10,5 Hz (*Octane* 2x250) bzw. 18,5 Hz (*ONYX2*). Durch die hohe Interaktivität eignet sich die Stabprobe somit sehr gut, um einen schnellen Überblick vom Datensatz zu gewinnen.

7.3.5 Performanzgewinn durch Multiprocessing

Das Multiprocessing wurde in *VtCrash* mit leichtgewichteten *pThreads* realisiert, die programmiertechnisch einfach zu handhaben sind. In der praktischen Umsetzung auf den verfügbaren Betriebssystemen *IRIX* 6.2 und 6.4 gab es jedoch Stabilitätsproblem mit der Graphikschnittstelle *OpenGL* 1.1, die für das Multiprocessing von *WorldToolKit* nur das *sproc*-Multiprocessingkonzept unterstützt. Die gleichzeitige Verwendung von *sprocs* und *pThread* in einem Programm führt zu grundsätzlichen Stabilitätsproblemen. *OpenGL* unterstützt ab der Version 1.2 die von *IRIX 6.5* zur Verfügung gestellten *pThreads*, wodurch die Stabilitätsprobleme behoben wurden. Allerdings stand ein *IRIX* 6.5 Betriebssystem erst sehr spät zur Verfügung, so dass nur eingeschränkt Erfahrungen gesammelt wurden.

Im Versuchsaufbau berechnete ein Arbeitsprozess parallel zum Hauptprozess die Parameter für die Ebenenprobe. Einfache Messungen der Prozessorauslastung haben den Performanzgewinn gegenüber einem einzelnen Prozess aufgezeigt. Die in Tabelle 7.4 aufgelisteten Ergebnisse wurden auf der Doppelprozessor *ONYX2* mit einer durchweg höheren Systemauslastung als 100% gemessen. Der zweite Prozessor wurde also nachweislich genutzt und trug zur Beschleunigung der Applikation bei.

Ebenen-größe	geladene Zeitschritte	Prozessor-auslastung [%]
900 Polygone	10	112
	40	105
3600 Polygone	10	132
	30	127
	40	113

Tabelle 7.4: Gesamtauslastung von zwei Prozessoren beim Multiprocessing.

Da die Programmbibliothek *WorldToolKit* keinen parallelen Zugriff auf den Szenengraphen oder die Geometrie erlaubt, gibt es in *VtCrash* eine synchronisierte Phase in der das Statusfeld für die Arbeitsprozesse gesperrt ist und die *WorldToolKit*-Geometrie vom Hauptprozess aktualisiert wird. In dieser Phase sind die Arbeitsprozesse gestoppt und es wird nur ein Prozessor für die Geometrieaktualisierung genutzt. Eine höhere Anzahl an Zeitschritten bedeutet mehr zu aktualisierende Geometrie und somit mehr Aufwand in der synchronisierten Phase. Die Standzeiten des Arbeitsprozesses spiegeln sich auch deutlich in den Ergebnissen der Tabelle 7.4 wieder. So sinkt die Prozessorauslastung bei einer Ebenen mit 3600 Polygonen von 132% bei 10 Zeitschritten auf 113% bei 40 Zeitschritten.

Interessant ist der Vergleich der Prozessorauslastung bei unterschiedlicher Größe der Ebenenprobe. Den 112% Auslastung für die kleine Ebene stehen 132% für die viermal größere gegenüber. Dieser Effekt ist auf die parallele Phase des Programmablaufes zurückzuführen, da der Hauptprozess für das Rendering der kleinen Ebenen länger benötigt als der Arbeitsprozess für die Parameterberechnung. Bei der großen Ebenen hingegen muss der Arbeitsprozess mehr berechnen, wodurch seine Wartezeit sinkt und die Gesamtauslastung steigt.

Die Messungen belegen den Performanzvorteil der Multiprozessorvariante von *VtCrash*. Die Auslastung der Prozessoren könnte durch eine Optimierung der parallelen und synchronen Phase noch erhöht werden. Eine bessere Lösung des Statusfeldes und eventuell eine Verbesserung von *WorldToolKit* bergen durchaus noch großes Potenzial.

7.4 Fazit

Die Erweiterung von *VtCrash* für Akustiksimulationen in der Strukturmechanik hat die parallele Echtzeitvisualisierung von zeitabhängigen Volumen und Geometrie in einer interaktiven Arbeitsumgebung demonstriert. Hohe Reaktionszeiten der Visualisierung wurden durch eine Beschleunigung der Algorithmen mittels Parallisierung und Optimierung auf unstrukturierte Gitter erzielt.

Der Einsatz des Probe-Konzepts ermöglicht eine intuitive Steuerung der Visualisierungstechniken und durch den sinnvollen Einsatz von Stereo-Projektion ist eine einfache Positionierung der Proben möglich. Die Entwicklung der propagativen Isoflächenberechnung hat auch dieser Visualisierungstechnik Interaktivität verliehen.

Kapitel 8

Interaktive Visualisierung von Strömungsberechnungen

Die Prozessreife der Strömungssimulation hinkt den geometriebezogenen Simulationen in vielen Bereichen um 5 Jahre hinterher. Beispielsweise ist die Crashtestsimulation mittlerweile ein fester Bestandteil des Fahrzeugentwicklungsprozesses, während sich viele Systeme der Strömungssimulation noch in der Evaluierungs- und Bewährungsphase gegenüber den realen Versuchen befinden.

Die Vergangenheit hat gezeigt, dass der Bedarf an effizienten Visualisierungsmethoden erst nach der Absicherung der Simulationstechnik entsteht. Im Falle des Simulationspaketes *PowerFLOW* von *Exa Corporation* erlangt das numerische Verfahren gerade die Akzeptanz gegenüber den realen Versuchen bei den Berechnungsingenieuren. Die verwendeten Datenstrukturen stellen allerdings neue Anforderungen an die bestehenden Visualisierungswerkzeuge, die meistens nur durch Geschwindigkeitseinbußen bewältigt werden können. Zusätzlich steigt auch in der Strömungsmechanik die Komplexität der Modelle und erschwert somit den Ingenieuren die Ergebnisanalyse. Der Vorteil eines interaktiven und intuitiven Visualisierungssystems zeichnet sich somit immer stärker ab.

In dem Kooperationsvorhaben *FORTWIHR III* entstand das Visualisierungssystem *PowerVIZ* zur Analyse von *PowerFLOW*-Berechnungsergebnissen. Kooperationspartner sind die *BMW Group,* die *Exa Deutschland GmbH* und die Universität Stuttgart. Ziel der Kooperation war die Entwicklung eines interaktiven Visualisierungssystems mit Schwerpunkt auf der Darstellungsgeschwindigkeit, der Anpassung von bekannten Visualisierungsalgorithmen an die hierarchische Datenstruktur der *PowerFLOW* Ergebnisse und der Erprobung neuer Visualisierungstechniken.

8.1 Handhabung großer hierarchischer Datensätze

Die Simulation von Strömungen wird numerisch häufig durch die Lösung der Navier-Stokes-Gleichung berechnet, deren partielle Differentialgleichnungen meistens durch den Finite-Volumen-Ansatz gelöst werden. Bei diesem Verfahren passt sich die Struktur des curvilinearen oder unstrukturierten Gitters der Fahrzeugoberfläche an (siehe Abbildung 8.1). Gerade in Hin-

blick auf die steigende Komplexität der Fahrzeugoberflächen ist der hohe Aufwand der Gittergenerierung sehr nachteilig, da auch für Variantenberechnungen die Gitter aufwendig angepasst werden müssen.

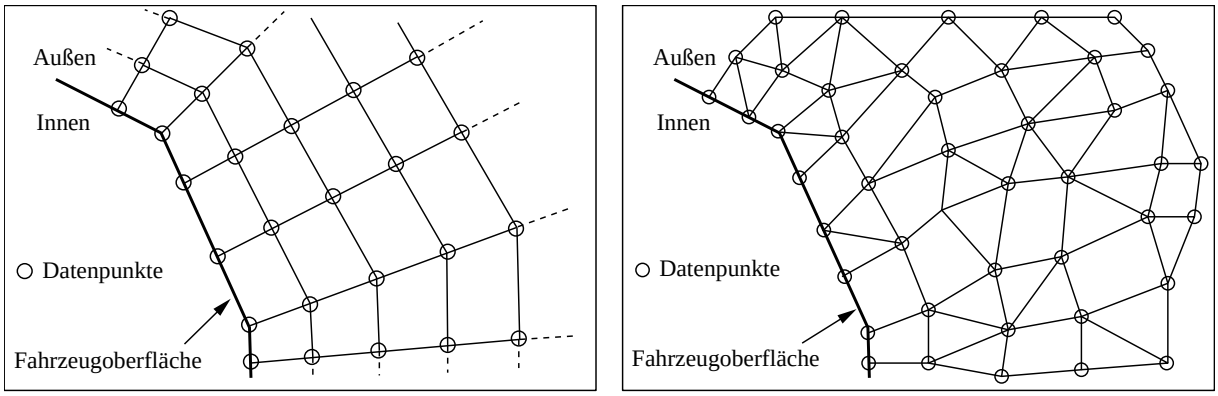

Abbildung 8.1: Vergleich der beiden Gittertypen: curvilineares Gitter (links) und unstrukturiertes Gitter (rechts).

Ein neuer vielversprechender Ansatz ist die Lattice-Gas-Methode, die beispielsweise von *PowerFLOW* genutzt wird. Durch die Trennung von Volumen- und Geometriebeschreibung arbeitet dieses Verfahren auch auf lokal verfeinerten kartesischen Gittern, dessen Struktur nicht an die Fahrzeugoberfläche angepasst ist (siehe Abbildung 8.2) und sich halbautomatisch generieren lässt.

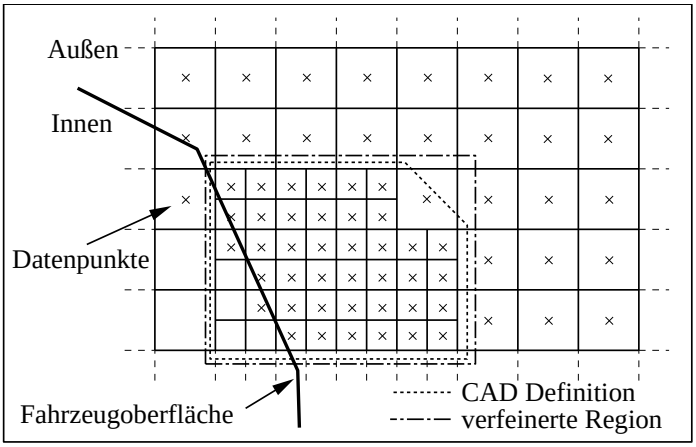

Abbildung 8.2: Gitterstruktur eines lokal verfeinerten kartesischen Gitters mit separater Geometrie.

Die lokal verfeinerte Gitterstruktur einer Lattice-Boltzmann-Simulation wird aus der Geometrie des Fahrzeuges und den CAD-Definitionen der Verfeinerungsregionen automatisch generiert.

Die Fahrzeuggeometrie entsteht aus der Diskretisierung der CAD-Modelle und muss durch eine geschlossene Oberfläche beschrieben sein. Eventuell vorhandene Löcher müssen nachträglich geschlossen werden. Bei kleinen Änderungen des Fahrzeugdesigns garantiert die Automatisierung eine schnelle Gittergenerierung und somit kurze Simulationszyklen. Im Beispieldatensatz der Abbildung 8.3 sind die Bereiche der Verfeinerungen durch Drahtgitterrahmen markiert.

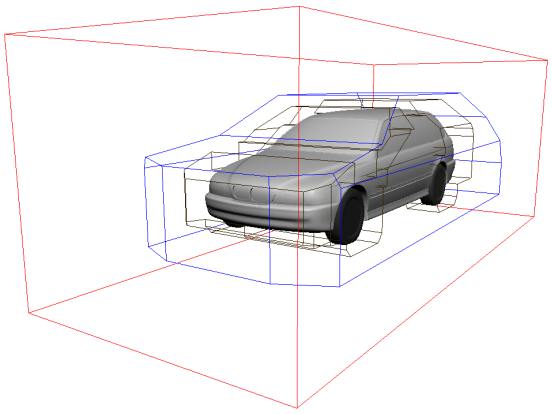

Abbildung 8.3: Die Verfeinerungsregionen eines Fahrzeugmodells.

Ein typischer Simulationsdatensatz besteht aus 8-9 Verfeinerungsstufen, den so genannten Voxel Refinement Levels (kurz *VRlevel*), deren Voxel in der feinsten Auflösungsstufe eine Kantenlänge von ca. 5 mm und in der gröbsten von ca. 50-70 cm besitzen. Die vom Benutzer definierten Verfeinerungsregionen, die so genannten Voxel Refinement Regions (kurz *VRregion*), enthalten Voxel einer einzigen Auflösungsstufe. Ein *VRlevel* hingegen kann in mehrere *VRregionen* unterteilt sein. Die *VRregionen* werden wie in Abbildung 8.3 erkennbar um das Fahrzeug herum angeordnet, um interessante Gebiete, wie zum Beispiel den Nachlauf, mit höherer Genauigkeit berechnen zu können. Der Quader des gesamten Simulationsvolumens hat typischerweise eine Kantenlänge von ca. 60 m und besitzt so ca. 20-30 Millionen Voxel verschiedener Auflösung. Ohne die hierarchische Struktur wäre bei der Verwendung eines einfachen kartesischen Gitters mit der feinsten Auflösungsstufe die Datenverwaltung durch die benötigen $2 * 10^{12}$ Voxel überfordert.

Zur Speicherung auf einem Massenmedium werden auch die hierarchischen Daten nochmals reduziert, so werden in jeder Auflösungsstufe jeweils acht Voxel zu einer *measurement cell* gemittelt. Für die Analyse ist nur der Bereich um das Fahrzeug interessant, so werden lediglich die 3-4 feinsten *VRlevel* gespeichert und ein typischer Datensatz besitzt somit 2-3 Millionen Voxel, die sich auf 3-4 *VRlevel* mit ca. 10 *VRregionen* verteilen.

Aus der Diskretisierung des CAD-Modells entsteht ein Polygongitter mit ca. 70.000 Dreiecken, das für den Simulationsprozess durch Schneiden mit den Voxelkanten in ca. 500.000 so genannte *surfel* unterteilt wird (siehe Skizze 8.4). Die *surfel* bilden die Basis der Simulation und für sie werden später ebenfalls Berechnungsergebnisse abgespeichert. Die Fahrzeuggeometrie durchdringt, wie in Abbildung 8.2 dargestellt, eine Vielzahl an Voxel, die dadurch eine Sonder-

behandlung bei der Simulation und Visualisierung erfordern. Die Voxel im Fahrzeuginneren sind nicht relevant für die Simulation und besitzen folglich auch keine Parameter.

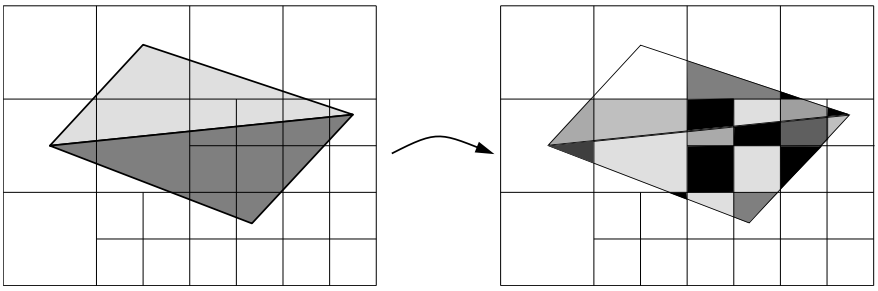

Abbildung 8.4: Generierung von 21 surfels (rechts) aus 2 Oberflächenpolygonen (links).

In der Tabelle 8.1 ist die Gitterstruktur des Beispieldatensatzes aus Abbildung 8.3 aufgeführt. Ein volles kartesisches Gitter der feinsten Auflösungsstufe würde ca. $6 * 10^7$ Voxel besitzen, während für das lokal verfeinerte Gitter nur $3 * 10^6$ Voxel gespeichert wurden. Die dazugehörige Geometrie besteht aus 70.820 Dreiecken.

VRregion Nr	VRlevel Nr	Kind-regionen	Dimension in Voxel (x,y,z)			Zellgröße in cm
0	2	1	309	44	69	5.0
1	1	2 - 8	287	80	125	2.5
2	0	-	87	72	49	1.25
3	0	-	87	72	49	1.25
4	0	-	43	71	189	1.25
5	0	-	191	125	189	1.25
6	0	-	111	67	189	1.25
7	0	-	87	72	49	1.25
8	0	-	87	72	49	1.25

Tabelle 8.1: Gitterstruktur des Beispieldatensatzes.

Die Größe und Struktur eines lokal verfeinerten Gitters erfordert für die interaktive Visualisierung eine effiziente Organisation und Speicherung der Daten. Verschiedene Konzepte kommen hierfür bereits in bestehenden Postprocessing-Systemen zum Einsatz. *ExaVIZ* beispielsweise verwendet eine Octree-Struktur und *EnSight* bildet die Verfeinerungsstruktur in ein unstrukturiertes Gitter ab. Beide Ansätze haben einen zu großen Speicherbedarf und sind für einen schnellen Zellzugriff nicht optimiert.

Kartesische Gitter sind als Basis für eine interaktive Visualisierung bestens geeignet. Ihr Speicherbedarf ist durch die implizite Nachbarschaftsinformation minimal und durch den schnellen Zellzugriff sind interaktive Visualisierungen möglich. Allerdings ist es von Seiten des

Speicherbedarfs nicht praktikabel ein lokal verfeinertes Gitter in ein rein kartesisches umzuwandeln.

Um die Vorteile eines kartesischen Gitters zu nutzen und trotzdem speichereffizient zu arbeiten, wurde in dieser Arbeit die in Abbildung 8.5 dargestellte Datenverwaltung entwickelt. In *VRregionen* sind durchschnittlich nur 60% der Zellen mit Parametern belegt und die anderen 40% sind entweder verfeinert oder befinden sich im Fahrzeuginneren bzw. außerhalb des Simulationsvolumens und sind deshalb nicht mit Daten besetzt. In der Datenverwaltung füllen ein Konditions- und ein Indexfeld die kartesische Struktur der einzelnen *VRregion* komplett aus, während die speicherintensiven Daten, wie Geschwindigkeit, Druck, Temperatur usw., in kleineren Feldern mit der Größe der belegten Voxel gespeichert werden. Das Konditionsbyte speichert zu einem Voxel die folgenden Informationen:

- Voxel ist mit Parametern belegt (Datenpräsenz)
- Voxel ist verfeinert und die Nummer der zugehörigen *VRregion*
- Voxel wird von Geometrie geschnitten

Markiert das Konditionsbyte eine Zelle als belegt, so zeigt der entsprechende Index auf den dazugehörigen Parametereintrag in den Datenfeldern (siehe Abbildung 8.5).

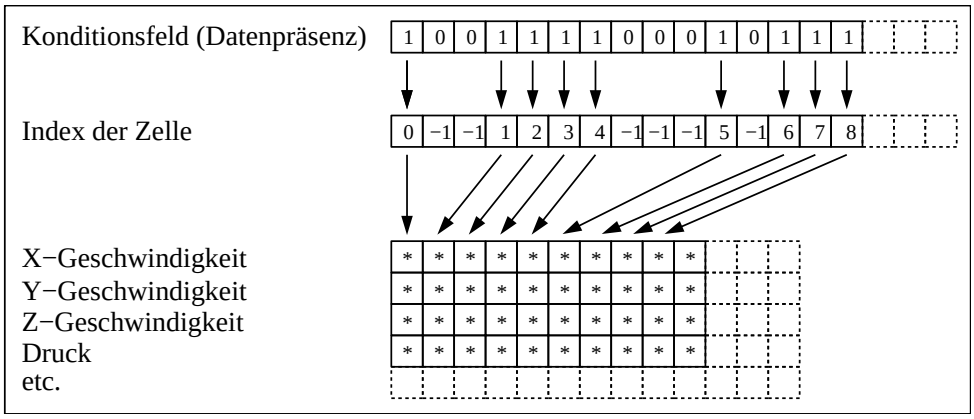

Abbildung 8.5: Die Felder einer *VRregion* in der Datenverwaltung.

Das Konditions- und das Indexfeld werden während des Ladevorganges aus dem *PowerFLOW* Dateiformat erstellt. In der Datei sind zu jedem Voxel die Position, Kantenlänge und die Ergebnisparameter gespeichert. Jedes Voxel wird einer *VRregion* zugeordnet und die Bits im Konditionsbyte entsprechend gesetzt. Die kompakte Datenverwaltung erlaubt eine Visualisierung auch auf Arbeitsplatzrechnern mit 256 MB Speicher.

Die Verwendung des Konditionsbyte-Feldes und der Vorteile des kartesischen Gitters erlauben nun die Anpassung des Algorithmus zur Zellbestimmung an lokal verfeinerte Gitter. Der angepasste Algorithmus lokalisiert zu einer gegebenen Position die umschließende Zelle in der entsprechenden *VRregion*:

```
beginne mit gröbster VRregion
WHILE(1){
    bestimme den Voxelindex zu der gegebenen Position
            im kartesischen Gitter:
        Index = (Position - untere Gittergrenze) / Voxelgröße
    // Überprüfe Konditionsbyte des Voxel
    IF (Voxel hat keine Parameter)
        return kein Voxel gefunden
    IF (Voxel ist verfeinert)
        fahre mit darunterliegender VRregion fort
        die Nummer der VRregion ist im Konditionsbyte gespeichert
    // Voxel gefunden
    RETURN Nummer der VRregion, Index des Voxels
}
```

Im Vergleich zu anderen Zellfindungsalgorithmen, wie beispielsweise dem *stencil walk* für curvilineare Gitter, ist dieser Algorithmus sehr viel einfacher aufgebaut und dadurch schneller. Anhand des Index können anschließend die Parameter des Voxels ausgelesen und visualisiert werden. Im lokal verfeinerten Gitter von *PowerFLOW* werden die Parameter als Zell-zentriert definiert und können somit als konstant über das gesamte Voxel angenommen werden. In Hinblick auf die Interpolationsmethode ist somit *nearest neighbour* ausreichend.

8.2 Octree-unterstützte Berechnung von Aufschlagpunkten

Die Fahrzeugoberfläche ist in curvilinearen und unstrukturierten Gittern implizit als Rand des Volumens definiert. Eine spezielle Berechnung von Aufschlagpunkten ist für diese Gittertypen nicht nötig, da ein Teilchen auf die Oberfläche schlägt, wenn es das Gitter an den entsprechenden Stellen verlässt. Die Bestimmung von Aufschlagpunkten wird also vom Standardalgorithmus zur Berechnung von Partikelbahnen erfüllt.

Die explizite Beschreibung der Fahrzeugoberfläche in lokal verfeinerten Gittern (siehe auch Abbildung 8.3) hingegen lässt in der Visualisierung das Durchdringen der Oberfläche und das erneute Heraustreten eines Partikels an einer anderen Stelle zu (siehe Abbildung 8.6 links). Direkt unterhalb der Fahrzeugoberfläche sind aufgrund der kartesischen Gitterstruktur auch Strömungsinformationen vorhanden (siehe Abbildung 8.2) und ermöglichen so eine Partikelbahnberechnung im Fahrzeuginneren. Dieser Effekt entsteht durch Fehler in der Interpolation und Integration der Teilchenbahn und durch die Vergröberung der Voxel zu *measurement cells*. Eine Kollisionserkennung der Partikel mit der Oberfläche verhindert diesen ungewollten Effekt und lässt die kollidierenden Partikelbahnen an der Geometrie enden (siehe Abbildung 8.6 rechts).

Der einfachste Algorithmus zur Kollisionserkennung schneidet in jedem Integrationsschritt die Partikelbahn gegen jedes Dreieck der Fahrzeuggeometrie. Eine typische Partikelbahn bestehend aus 1.000 Integrationsschritten würde bei einer Fahrzeuggeometrie von 70.000 Dreiecken und einer Berechnungsgeschwindigkeit von 100.000 Schnitten pro Sekunde mit 35.000.000 benötigten Schnittberechnungen innerhalb von 6 Minuten berechnet werden. Dieses kleine Bei-

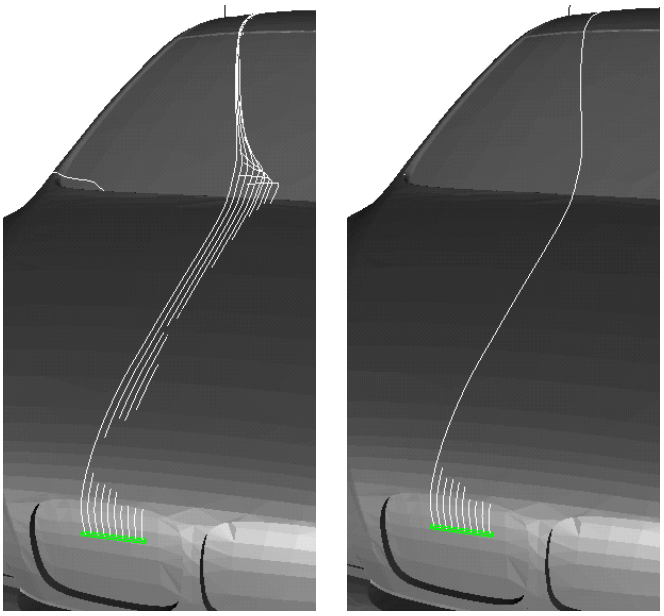

Abbildung 8.6: Partikelbahnen ohne (links) und mit (rechts) Kollisionserkennung an der Fahrzeugoberfläche.

spiel verdeutlicht den Bedarf einer effizienten Kollisionsberechnung für die interaktive Visualisierung.

Die Kollisionserkennung wurde in dieser Arbeit durch die Verwendung eines Octrees beschleunigt, mit dessen Hilfe sich in kürzester Zeit die in Frage kommenden Dreiecke einer Schnittberechnung bestimmen lassen. Der Octree besitzt eine Bounding Box Hierarchie, die das Strömungsvolumen schrittweise nach Geometrieinhalt unterteilt. Volumenbereiche ohne Geometrie werden von großen Boxen umschlossen, während um die Geometrie herum die Boxgröße abnimmt (siehe Abbildung 8.7). Falls eine Bounding Box nur noch ein Polygon oder ein Voxel enthält, wird die Unterteilung abgebrochen. Ein Voxel ist danach immer eindeutig einer Boun-

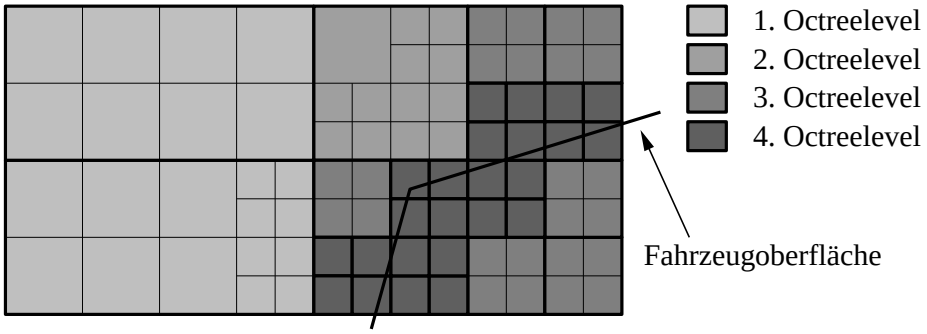

Abbildung 8.7: Beispiel der Octree-Unterteilung in Geometrienähe.

ding Box zugeordnet, da die Bounding Boxen an den Kanten der Gitterstruktur ausgerichtet wurden. Dreiecke hingegen können sich natürlich auf mehrere Bounding Boxen ausdehnen, da ihre Kanten im Allgemeinen nicht parallel zu den Gitterachsen verlaufen.

Für die Bestimmung einer Teilchenbahn müssen alle von der Bahn durchschrittenen Voxel auf Schnitte mit der Fahrzeuggeometrie überprüft werden. Ohne das Konditionsbyte-Feld muss hierfür jedes Voxel in der Octree-Struktur gesucht und überprüft werden. Die Verwendung des Konditionsbytes auf Geometriebesitz reduziert die benötigten Octree-Untersuchungen auf ein Minimum (siehe Abbildung 8.8). Nur von der Geometrie geschnittene Voxel werden im Octree gesucht und liefern die Dreiecke für die Schnittberechnungen. Die Berechnung der Schnitte wird mit Algorithmen aus dem Ray-Tracing-Umfeld [44] durchgeführt. Abbildung 8.6 stellt die Partikelbahnen ohne und mit Kollisionserkennung gegenüber.

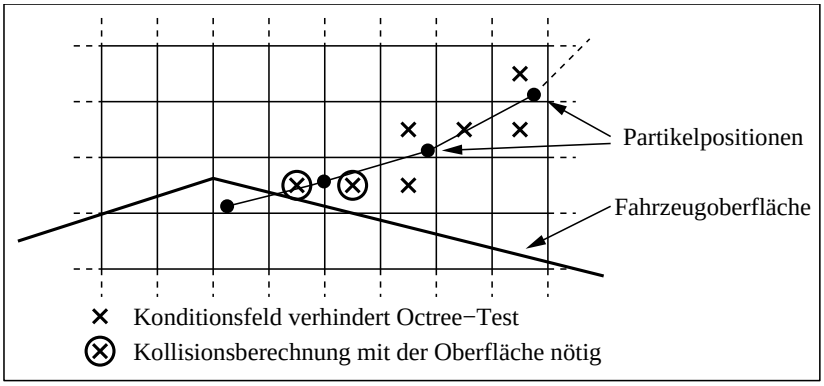

Abbildung 8.8: Die Verwendung des Konditionsbytes minimiert die Anzahl der benötigten Octree-Tests für die Berechnung einer Teilchenbahn.

Der Algorithmus zur Berechnung einer Partikelbahn hat mit Interpolation und Integration in dem Strömungsfeld, der Verwendung des Konditionsbyte-Feldes und des Octrees folgende Form:

```
WHILE(Partikel im Strömungsfeld und
      maximale Anzahl an Integrationsschritten nicht erreicht) {
   Bestimme umschließendes Voxel der Partikelposition
   Integriere einen Schritt
   Bestimme alle Voxel, die vom Integrationsschritt geschnitten werden
   FOR(i=0;i< Anzahl geschnittener Voxel;i++) {
      // Überprüfe Konditionsbyte
      IF(Voxel besitzt Geometrie) {
          Bestimme die Dreiecke des Voxels mit Hilfe des Octrees
          IF(Integrationsschritt schneidet ein Dreieck des Voxel) {
               Berechne den Aufschlagpunkt auf der Geometrie
               Beende Integration
} } } }
```

Durchschnittlich werden zehn Dreiecke für einen Integrationsschritt überprüft. In dem Beispiel von Abbildung 8.9 wurden die benötigten Dreiecke der Kollisionsberechnung markiert. Das Partikel fliegt zuerst dicht über die Fahrzeugoberfläche und schlägt schließlich am Radkasten auf. Lediglich ein Bruchteil der Fahrzeuggeometrie wird dank der Octree-Unterstützung gegen die Partikelbahn getestet.

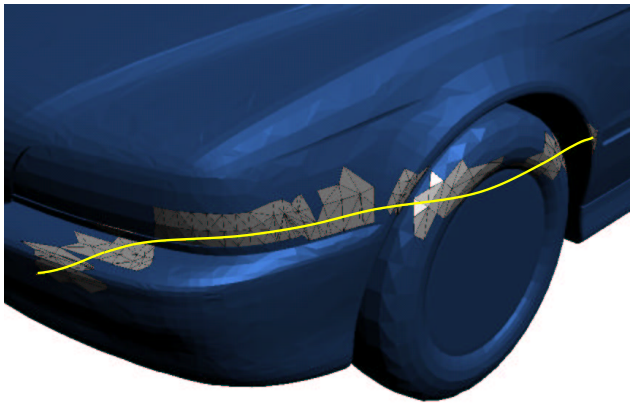

Abbildung 8.9: Die markierten Dreiecke wurden für die abgebildete Partikelbahn auf Kollision überprüft.

In Tabelle 8.2 sind zum quantitativen Vergleich die Zeiten der Bahnberechnung mit und ohne Kollisionsüberprüfung gegenübergestellt. Gemessen wurde die Zeit (in Sekunden) für Zellauffindung, Interpolation, Integration und Generierung der Visualisierungsgeometrie für 100 Partikelbahnen mit jeweils 1000 Integrationsschritten. Die Messungen erfolgten auf einer *SGI Octane SI* mit R10000 (175MHz) und 256 MB Arbeitsspeicher, die den typischen Arbeitsplatzrechner repräsentiert.

Ordnung des Runge-Kutta Schemas	Partikelbahn in Geometrienähe „worst case“			Partikelbahn mit Abstand zur Geometrie „best case“		
	mit Test	ohne Test	Verhältnis	mit Test	ohne Test	Verhältnis
2.	3.22	1.35	2.38	1.80	1.35	1.33
3.	3.56	1.72	2.07	2.16	1.69	1.27
4.	3.80	1.94	1.96	2.41	1.94	1.24

Tabelle 8.2: Zeit für die Teilchenbahnberechnung und Geometriegenerierung für 100 Partikel mit jeweils 1000 Integrationsschritten in Sekunden.

Die beiden Extremfälle der Kollisionsbestimmung liefern die aussagekräftigsten Ergebnisse (siehe Tabelle 8.2). Der schlechteste Fall („worst case“) ist eine Partikelbahn nahe der Oberfläche mit einer Kollisionsüberprüfung für jeden Integrationsschritt. Im optimalen Fall („best

case“) passiert das Teilchen die Fahrzeugoberfläche mit ausreichendem Abstand, so dass Kollisionen bereits durch das Konditionsbyte ausgeschlossen werden können. Die durchschnittliche Berechnungszeit beträgt für den „worst case“ 3,52 Sekunden und den „best case“ 2,12 Sekunden. Wird die Geometrie nicht beachtet, so ist die Berechnungszeit für 100 Partikelbahnen mit 1,67 Sekunden erwartungsgemäß in beiden Fällen identisch. Der zusätzliche Aufwand für die Kollisionsberechnung liegt somit zwischen 27% und 100% gegenüber der Berechnungszeit einer ungeprüften Teilchenbahn und ist noch akzeptabel.

In der praktischen Anwendung tritt der „worst case“ vergleichsweise selten auf, während viele Partikelbahnen in ausreichendem Abstand zur Fahrzeugoberfläche berechnet werden. Eine durchschnittliche Partikelbahn erfordert in 30% der Integrationsschritte eine Kollisionsüberprüfung und wird in der Berechnung somit um ca. 50% langsamer.

Die Anforderungen des aufgeführten Beispiel treten in der Praxis kaum auf, da der Anwender durch die auftretenden Überdeckungen bei Verwirbelungen von 100 Partikelbahnen die Übersicht verliert und die meisten Berechnungen weniger als 1000 Integrationsschritte benötigen. Visuell lassen sich noch 10 Teilchenbahnen voneinander unterscheiden, von denen viele mit weniger als 400 Integrationsschritten berechnet werden. Die Verwendung von adaptiven Integrationsverfahren führt zu einer weiteren Reduktion der Schrittanzahl. Die angeführten Zeiten sind somit für die interaktive Visualisierung ausreichend.

Die Berechnung von massebehafteten Partikelbahnen mit Gravitationseinfluss und Massenträgheit ist ein wichtiges Einsatzgebiet für die Kollisionserkennung, da erst die Berechnung von Aufschlagpunkten die Simulation der Fahrzeugverschmutzung ermöglicht. Ein aussagekräftiges Verschmutzungsbild benötigt die möglichst schnelle Berechnung einer hohen Anzahl von Teilchenbahnen mit entsprechend vielen Aufschlagpunkten. Im Forschungsvorhaben *FORTWIHR III* wurde die Modellbildung und der Einsatz des Octree zur Aufschlagpunktberechnung in einer weiterführenden Arbeit untersucht.

8.3 Volumenproben als Interaktionselemente

Die Schnittstelle zwischen Anwender und Visualisierungsalgorithmen zur einfachen und intuitiven Analyse von Ergebnisdatensätzen bilden die Volumenproben. Ihre Form wird über die Größe, Zellenzahl, Position und Orientierung definiert (siehe Abbildung 8.10). Die Volumenproben, bestehend aus einer Pickgeometrie, dem Drahtgitterrahmen des Teilraums und der eigentlichen Visualisierungsgeometrie, werden in einem eigenen Teilbaum im Szenengraphen eingegliedert und lassen sich so leicht durch Benutzereingaben über ein GUI, eine 2D-Maus oder VR-Eingabegeräte interaktiv transformieren. Der umschlossene Teilraum der Volumenproben wird zur interaktiven Analyse in diskrete Zellen unterteilt, die den verschiedenen Visualisierungsalgorithmen als Definition von Abtast- oder Startpunkten für Teilchenbahnen dienen. In *PowerVIZ* wurde zu jedem Visualisierungsalgorithmus eine spezielle Probe-Klasse von der Basisprobe-Klasse abgeleitet.

Das Multiprobe-Konzept von *PowerVIZ* erlaubt die Koordination beliebig vieler Proben in einer Visualisierungssitzung. Der Anwender hat so die Möglichkeit hochgradig komplexe Zusammenhänge mit den erforderlichen Mitteln zu visualisieren. Die Interaktionseingaben des An-

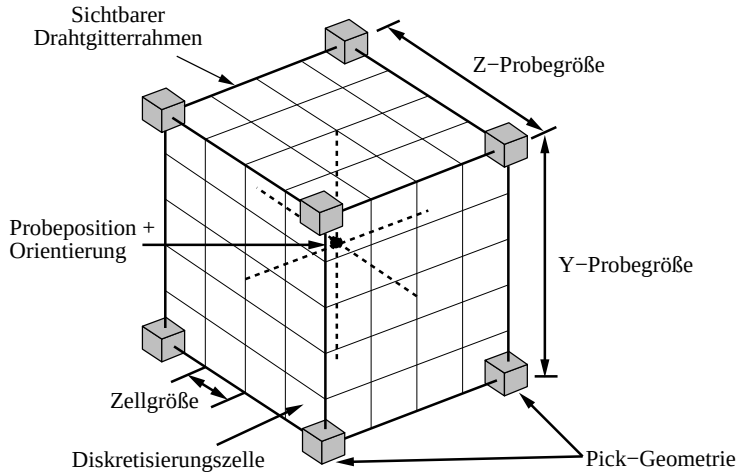

Abbildung 8.10: Definition der freibewegbaren Volumenprobe.

wenders werden von der Benutzerführung lediglich an die aktuell selektierte Probe weitergeleitet, die der Anwender durch Anwählen der „Pick“-Geometrie (siehe Abbildung 8.10) wechseln kann. Der Aktivierungszustand der Probe wird über die Farbe (blau: inaktiv, rot: aktiv) der Würfelgeometrie symbolisiert.

Die Bewegung einer Volumenprobe erfordert im Normalfall die Neuberechnung der Visualisierungsgeometrie und die Neuzeichnung der gesamten Szene einschließlich der Fahrzeugoberfläche. Da die Fahrzeuggeometrie bis zu 200.000 Dreiecke umfasst, kann die Bildwiederholungsrate sehr gering ausfallen. In diesem Fall bietet sich die Ausnutzung einer Optimierungsfunktion des *OpenGL/Optimizers* an. Der Szenengraph wird nicht vollständig gerendert, sondern in zwei Teilgraphen untergliedert. Der erste Teil, bestehend aus der Fahrzeuggeometrie und den inaktiven Proben, wird von der Graphikhardware gezeichnet und anschließend Frame- und Depthbuffer ausgelesen. Bei einem Update der aktivierten Probe werden nun zuerst der gespeicherte Frame- und Depthbuffer in die Graphikhardware geladen und dann lediglich die aktivierte Probe neugezeichnet (siehe Abbildung 8.11). Solange der Augpunkt nicht bewegt wird, erzielt dieses Verfahren selbst mit komplexen Geometrien sehr hohe Bildwiederholungsraten. Bei einer Augpunkttransformation muss jedoch die gesamte Szene neu gerendert werden, was zum Zusammenbruch der Performanz führt. Durch die Einschränkung auf einen konstanten Augpunkt kann dieses Verfahren nicht mit Headtracking, wie es beispielsweise in einer *CAVE* erforderlich ist, eingesetzt werden.

8.3.1 Partikelbahn-Probe

Die Partikelbahn-Probe ist ein virtuelles Interaktionselement, dessen reales Pendant die Rauchsonde ist. Konstruktions- und Versuchsingenieure ohne Erfahrungen im Umgang mit Visualisierungsystemen haben durch diese Verwandtschaft kaum Probleme in der Handhabbarkeit und im Verständnis dieses Werkzeuges.

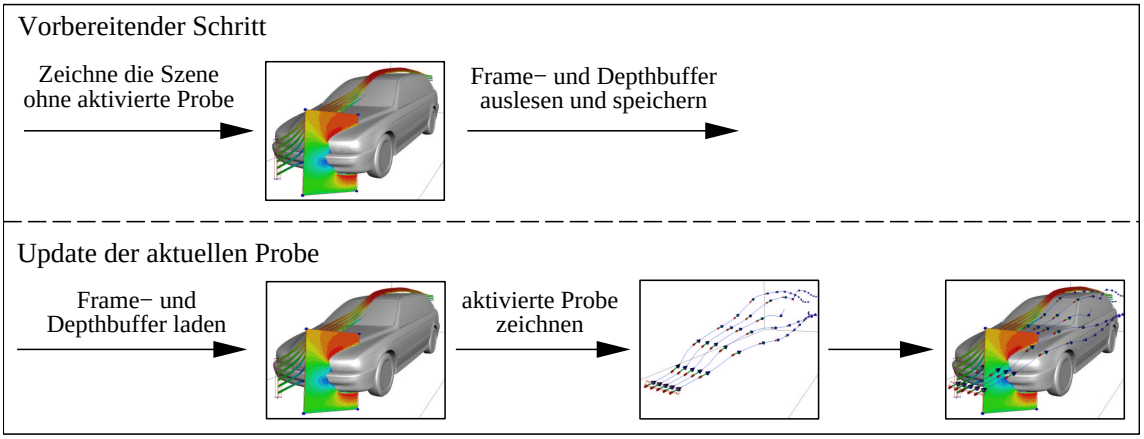

Abbildung 8.11: Lediglich die aktivierte Probe wird gezeichnet, während die restliche Szene als Bild in den Frame- und Depthbuffer geladen wird.

Die Mittelpunkte der Probe-Zellen werden von der Partikelbahn-Probe als Startpunkte für die Teilchenbahnen benutzt. Die Zellgröße definiert hierbei den Abstand der Startpunkte und die Anzahl der Zellen in jeder Dimension legt die Menge der Partikel fest. Eine eindimensionale Probe mit großen Zellen erzeugt beispielsweise einen Rechen, wie er auch in der realen Welt durch mehrere Rauchdüsen zur Gewinnung eines groben Überblicks eingesetzt wird. Eine sehr kleine Probe mit kleinen Zellen hingegen bildet die reale Rauchsonde nach, mit der sich lokale Effekte untersuchen lassen. Neben der individuellen Position und Orientierung können eine Reihe weiterer Parametereinstellungen für den Typ und die Berechnung der Teilchenbahnen vorgenommen und die Probe so den Erfordernissen angepasst werden.

Partikelbahnen werden in *PowerVIZ* durch die Generierung der entsprechenden Geometrie als Linien, Rotationsbänder und Glyphen visualisiert (siehe Abbildung 8.12). Im einfachsten Fall genügt eine Linie entlang der Teilchenbahn, die optional durch die Skalarwerte der Stützstellen eingefärbt wird. Die Polygone der Rotationsbänder werden ebenfalls entlang der Partikelbahn angeordnet und erlauben durch die Verdrehung um die Achse der Teilchenbahn die zusätzliche Visualisierung der lokalen Rotation innerhalb der Strömung, wie sie aus den Jacobi-Matrizen bestimmt werden kann. Weitere skalare Parameter können als Farbe oder Breite des Bandes visualisiert werden.

Auf die Ersatzgeometrie von Glyphen kann eine Reihe von skalaren Parametern gleichzeitig abgebildet werden. In dieser Arbeit kommen kleine Pfeile als Glyphen zum Einsatz, die parallel zur Strömung ausgerichtet sind und vier unabhängige skalare Größen visualisieren können (siehe Abbildung 8.13). Spitze, Schaft und Stumpf der Pfeile werden separat durch benutzerdefinierte Parameter eingefärbt und die Pfeillänge repräsentiert idealerweise den Betrag der Geschwindigkeit. Die im Vergleich zu anderen Arbeiten [28] einfache Glyph-Geometrie soll eine Überfrachtung der Darstellung und eine mögliche Verwirrung des Anwenders verhindern.

Im Gegensatz zu den Linien und Bändern besitzen die Glyphen durch die Ausrichtung der Pfeile auch in stark wirbelbehafteten Strömungen eine eindeutige Visualisierung der Stromrich-

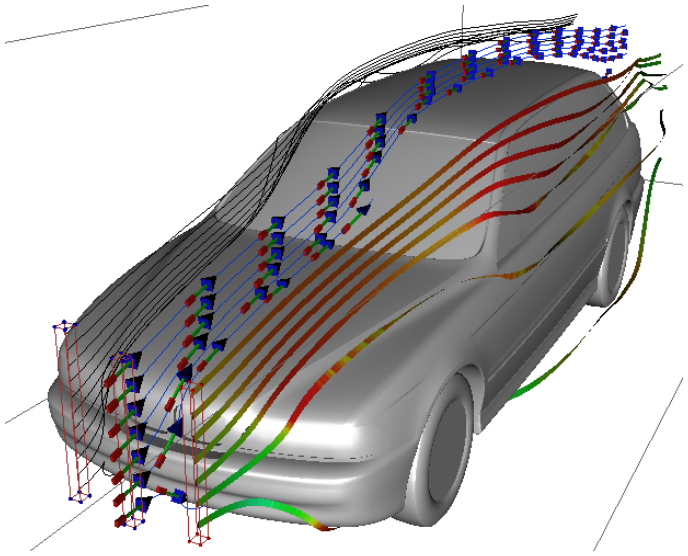

Abbildung 8.12: Visualisierung von Partikelbahnen als Linien, Rotationsbänder und Glyphen.

tung. Die Abbildung 8.14 zeigt deutlich das Versagen von Linien und Rotationsbändern in diesem Punkt.

Der Partikelbahnintegrator kann durch eine Vielzahl an Einstellungen optimal auf das zu visualisierende Strömungsfeld und die Hardwareumgebung angepasst werden. Die Wahl eines adaptiven Integrationsverfahrens mit großer Schrittweite und Fehlergrenze ermöglicht durch eine schnellere Berechnung eine höhere Interaktivität, wie sie zum Beispiel für eine CAVE-Anwendung wünschenswert ist. Der gewonnene Performanzvorteil kann in einem ersten Schritt zum Auffinden kritischer Regionen im Datensatz genutzt werden, dem anschließend eine Untersuchung mit höherer Genauigkeit folgt.

Der parallele Einsatz beliebig vieler Partikelbahn-Proben erlaubt in *PowerVIZ* die gleichzeitige Visualisierung von mehreren interessanten Gebieten.

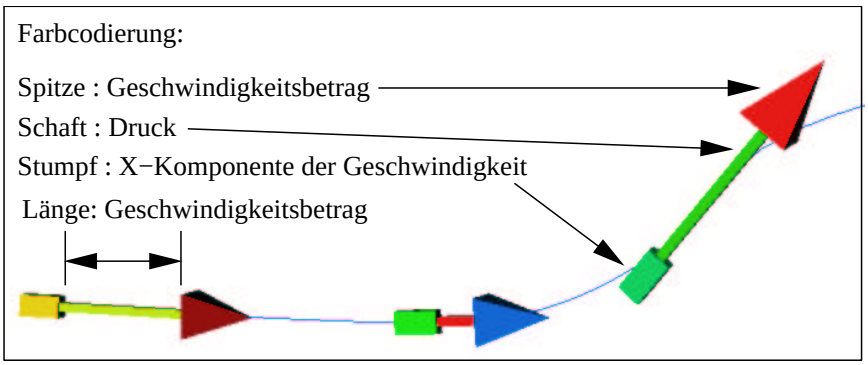

Abbildung 8.13: Abbildung von skalaren Parametern auf die Glyphgeometrie.

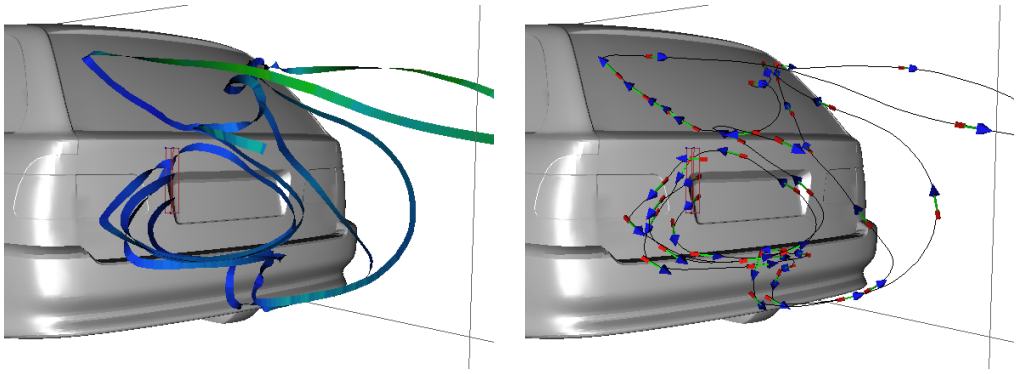

Abbildung 8.14: Vergleich von Rotationsbändern und Glyphen in einer wirbelbehafteten Strömung.

8.3.2 Schnittebenen-Probe

An den Stützstellen der Schnittebenen werden sowohl skalare als auch vektorielle Größen eines Strömungsfeldes abgetastet und visualisiert. Die Ingenieure können so anhand der Schnitte das globale und lokale Strömungsverhalten analysieren. In realen Windkanälen werden Schnittebenen aufwendig durch einen Hitzdraht bestimmt, der an einem Roboterarm befestigt ist. Diese Messmethode kann nur an leicht zugänglichen Positionen durchgeführt werden und ist zudem sehr zeit- und somit kostenintensiv. Virtuelle Schnitte hingegen sind in den Simulationsergebnissen beliebig im Raum positionierbar.

Die Abtastpunkte der Schnittebene definieren sich durch die diskreten Zellen der Volumenprobe. Die Dimensionen in X- und Y-Richtung geben die Anzahl der Abtastpunkte in der Ebene

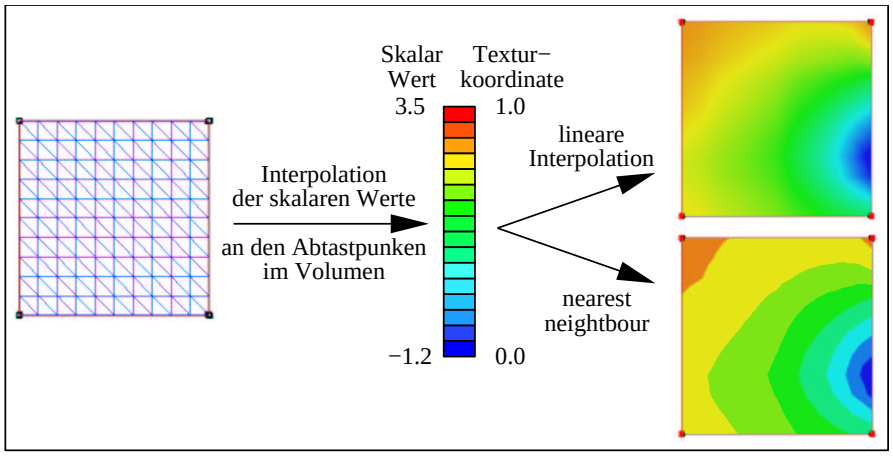

Abbildung 8.15: Mappen einer eindimensionalen Textur auf Polygone der gegebenen Gitterstruktur zur Visualisierung von skalaren Größen.

vor, während sich die Anzahl der Schnittebenen aus der Z-Dimension ergibt. An den Abtastpunkten wird das Strömungsvolumen interpoliert und die Werte auf den Ebenen visualisiert. Skalare Größen werden als Farbwerte auf die Polygone der Ebenengeometrie gemappt und vektorielle Größen beispielsweise als kleine Pfeile (Linien) dargestellt.

Die einfache konstante Einfärbung der Polygone für die Visualisierung der Skalarwerte besitzt deutlich sichtbare und somit störende Farbübergänge an den Polygongrenzen. Die Verwendung einer eindimensionalen Textur liefert hingegen eine gleichmäßige Darstellung der Farbverläufe, da die Skalarwerte stückweise auf die Fläche der Ebenen interpoliert werden. Die Texturverteilung wird durch Texturkoordinaten an den Stützstellen definiert und linear oder mit „nearest neighbour“ interpoliert. Die lineare Interpolation der Textur liefert fließende Farbverläufe, wohingegen „nearest neighbour“ die Graphikhardware zur Visualisierung von Isokonturen verwendet (siehe Abbildung 8.15).

Moderne Graphiksysteme können texturierte Polygone ohne Geschwindigkeitsverlust zeichnen und die verbesserte Visualisierung bedeutet somit keinen Performanzverlust. Die verschiedenen Visualisierungstechniken sind in Abbildung 8.16 gegenübergestellt.

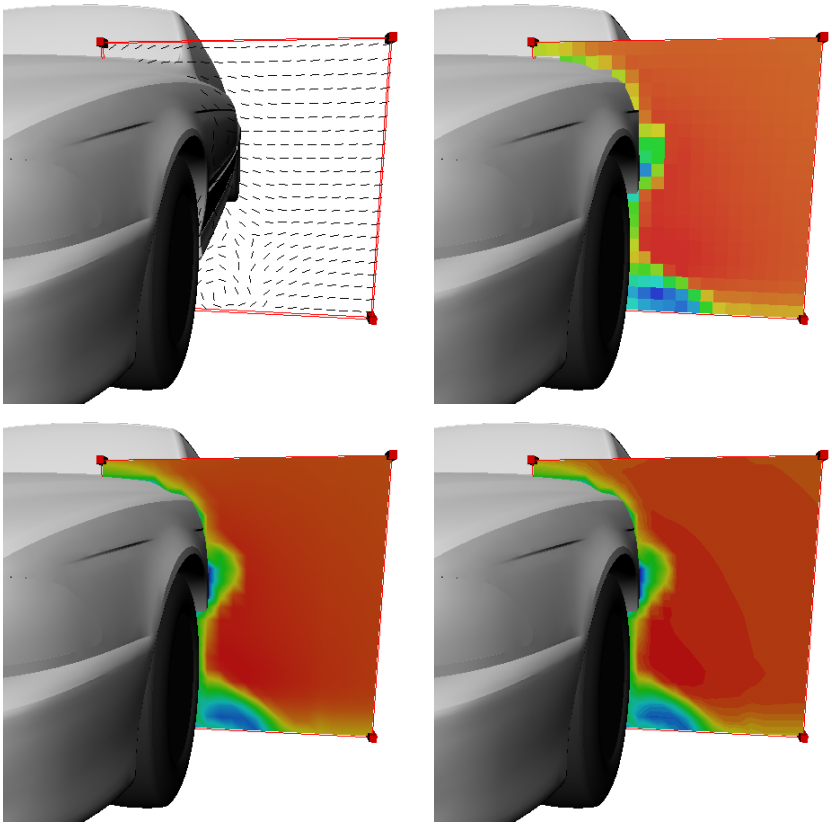

Abbildung 8.16: Die verschiedenen Visualisierungstechniken der Schnittebenen-Probe: o.l. Pfeile, o.r. eingefärbte Polygone, u.l. linear interpolierte 1D-Textur, u.r. Isokonturen durch „nearest neighbour“

Die Begrenzung der Schnittebenen auf zwei Dimensionen beschränkt die Analyse eines dreidimensionalen Volumens. Das direkte Volumenrendering beispielsweise ermöglicht einen dreidimensionalen Einblick in ein Volumen, ist aber momentan für große Datensätze nicht interaktiv verfügbar. In dieser Arbeit schaffen daher hintereinander angeordnete Schnittebenen Abhilfe. Die Verwendung von teiltransparenten Texturen verhindert im Ebenenstapel die Verdeckung der einzelnen Schnitte. Alle skalaren Parameter oberhalb oder unterhalb einer Grenze werden dabei transparent dargestellt. Beispielsweise werden in Abbildung 8.17 die Ebenen durch den Geschwindigkeitsbetrag eingefärbt und im Bereich hoher Geschwindigkeiten ist die Textur transparent, wodurch die Struktur des Radkastenwirbels sehr gut zu erkennen ist.

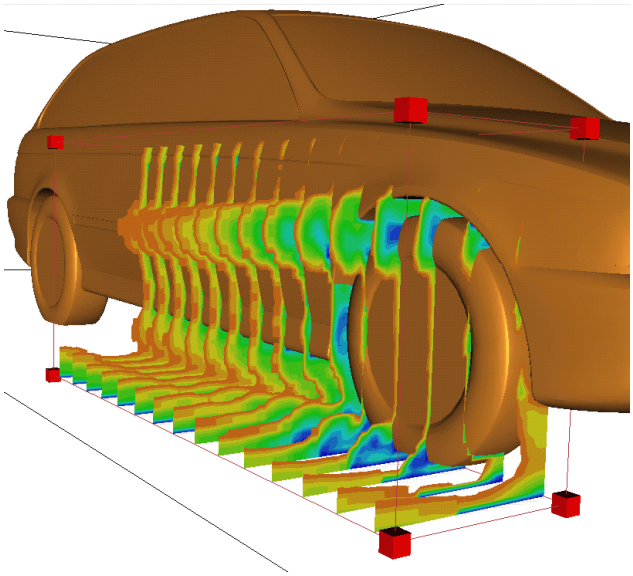

Abbildung 8.17: Einsatz von teiltransparenten Schnittebenen zur interaktiven Visualisierung von Verwirbelungen.

Die freie Positionierung und die leichte Handhabbarkeit zeichnen die Schnittebenen-Probe als ein sehr brauchbares Analysewerkzeug aus, mit dem sich innerhalb kurzer Zeit ein Eindruck vom Datensatz gewinnen lässt. Im Vergleich zu der realen Hitzdrahtmessmethode kann sie vielseitiger positioniert und auch in Bereiche eingesetzt werden, die für den Roboterarm unzugänglich sind.

8.4 Visualisierung von Skalarwerten auf der Fahrzeugoberfläche

Die Fahrzeugoberfläche einer *PowerFLOW* Simulation besteht aus zwei Auflösungsstufen, der Ausgangsgeometrie aus der Diskretisierung des CAD-Modells mit ca. 70.000 Polygonen und der *surfel*-Geometrie mit ca. 500.000 Elementen, die durch Schneiden der Ausgangsgeometrie mit

den Voxelgrenzen generiert wird (siehe Abbildung 8.4). Die Parameter werden lediglich auf Basis der *surfels* von *PowerFLOW* simuliert und abgespeichert. Für die interaktive Visualisierung stellt die hohe Anzahl der *surfel*-Elemente ein Problem dar, da die heutige Graphikhhardware sie nur mit einer geringen Bildwiederholungsrate rendern kann. In dieser Arbeit wurden verschiedene Techniken zur Optimierung der Visualisierung betrachtet, um eine hohe Interaktivität zu erzielen.

Die Verwendung von Texturen auf der Ausgangsgeometrie zur Abbildung der Parameter ist ein untersuchter Ansatz. Die Texturen werden pro Ausgangsgeometrie-Polygon aus den entsprechenden *surfels* bestimmt. Hierzu wird in vorgegebenen Abständen der skalare Wert zwischen den *surfel*-Werten interpoliert und als Farbe in die Texture eingetragen, wie in Abbildung 8.18 dargestellt.

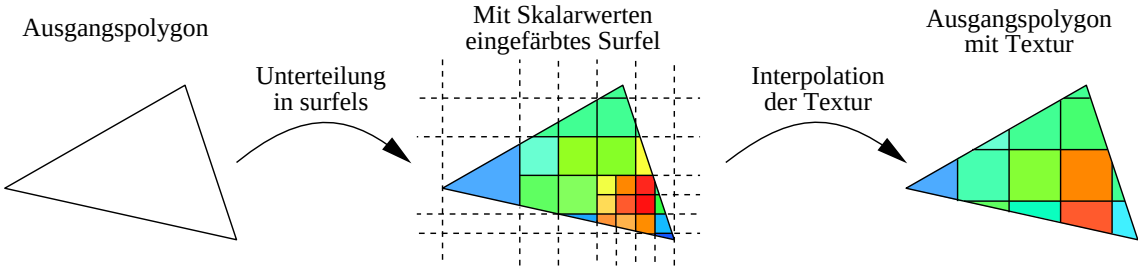

Abbildung 8.18: Generierung einer Textur aus den *Surfel*-Parametern.

Der geringe Geschwindigkeitsverlust durch die Verwendung von Texturen ist der Vorteil dieses Ansatzes. Allerdings muss die Textur in einem zeitaufwendigen Preprozessorschritt für jeden Parameter generiert und bei jeder Änderung der Abbildungsfunktion von Skalarwerten auf die Farben erneut berechnet werden. Ebenfalls muss die Texturgröße in einem zeitaufwendigen Schritt der Polygongröße angepasst werden, damit eine gleichmäßige Struktur der Farbverteilung gewährleistet und auch Details ausreichend gut aufgelöst sind. Dieser Ansatz liefert zwar hohe Bildwiederholungsraten, ist aber für interaktive Veränderungen der Visualisierung ungeeignet.

Vielversprechender ist die Verwendung eines „Level of Detail“-Ansatzes, bei dem je nach Geschwindigkeitsbedarf zwischen verschiedenen Auflösungsstufen der Geometrie gewechselt wird. Im vorliegenden Fall wird die Ausgangsgeometrie für hohe Geschwindigkeiten der Darstellung und die *surfel*-Geometrie für eine hohe Genauigkeit eingesetzt. Die Polygone der beiden Geometrien werden entsprechend den Skalarwerten entweder mit einer Farbe konstant oder durch eine eindimensionale Textur mit Farbverläufen eingefärbt. Die Skalarwertzuordnung erfolgt ohne Interpolation direkt aus den Ergebnisdaten und lässt so auch interaktive Änderungen zur Laufzeit der Visualisierung zu. Der Informationsgehalt der groben Ausgangsgeometrie ist natürlich geringer als von der *surfel*-Geometrie, er reicht jedoch für die Orientierung des Betrachters bei Bewegungen des Modells aus. Bei Bedarf wird dann auf die feine Auflösung umgeschaltet. Der Wechsel zwischen den Auflösungsstufen kann entweder durch die Entfernung zum Augpunkt oder die Bewegung des Modells automatisch gesteuert werden.

Das Verfahren des bewegungsgesteuerten Wechsels ist aber für Anwendungen mit „Headt-

racking“ unbrauchbar, da sich in diesem Fall die Position des Auges ständig ändert und somit nur die grobe Auflösung angezeigt wird. Hier bietet sich die adaptive Gitteranpassung der Geometrie an, die je nach Abstand des Augpunktes die Polygone vergröbert oder verfeinert (siehe Abbildung 8.19). Mit diesem Verfahren lassen sich sowohl hohe Bildwiederholungsraten erzielen als auch alle berechneten Details darstellen, jedoch muss die Geometrie ebenfalls aufwendig vorbereitet werden.

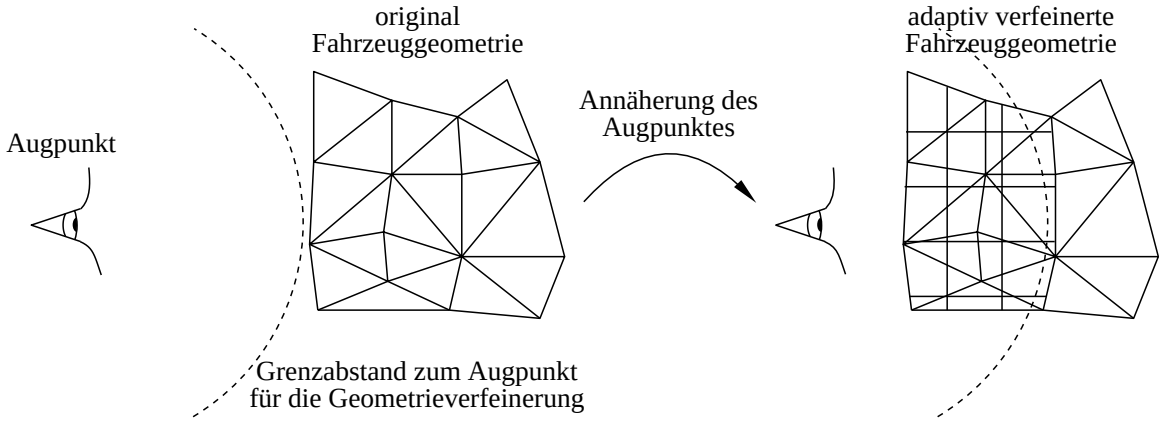

Abbildung 8.19: Abstandsgesteuerte adaptive Verfeinerung der Geometrie.

In der Praxis hat sich für die Standardanwendung der „Level of Detail“-Ansatz mit dem bewegungsgesteuerten Wechsel bewährt, da er während der Transformation des Modells hohe Bildwiederholungsraten ermöglicht, nach dem Stoppen der Bewegung jedes Detail mit der feinsten Auflösung visualisiert und interaktive Änderungen der Visualisierung zulässt.

8.5 Visualisierungssystem *PowerVIZ*

Ein modernes Visualisierungssystem muss viele Anforderungen erfüllen. So geht der Wunsch nach Flexibilität für Erweiterungen und Systemunabhängigkeit einher mit der konträren Forderung nach maximaler Performanz und Effizienz. Das Ausnutzen von vorhandenem Wissen aus bewährten Programmierbibliotheken führt im Allgemeinen zu Performanzvorteilen und unterstützt die Systemunabhängigkeit. Ebenfalls wird die Aktualität in vielen Systembereichen gesteigert und Entwicklungsaufwand für bereits gelöste Teilprobleme eingespart. Ein objektorientierter Ansatz bei der Implementierung lässt mehr Flexibilität in der Programmentwicklung zu.

Das Visualisierungssystem *PowerVIZ* für *PowerFLOW* Strömungssimulationen wurde in der objektorientierten Programmiersprache C++ entwickelt und basiert auf *SGIs* Szenengraph-API *Cosmo3D* inklusive den Erweiterungen von *OpenGL/Optimizer*. *Cosmo3D* und *Optimizer* bieten neben einer sehr flexiblen und schlanken Szenengraphstruktur viele Optimierungsalgorithmen für die Graphikkomplexität und Darstellungsgeschwindigkeit. Als GUI bietet sich eine *Motif/Viewkit*-Basis an, da die vorgesehenen Arbeitsplattformen UNIX-Systeme sind. Die Verwendung von klar definierten Schnittstellen und die Kapselung der Programmbibliotheken er-

lauben das einfache Hinzufügen von Funktionalitäten und wahren die Systemunabhängigkeit. Beispielsweise könnte so *Cosmo3D/Optimizer* vergleichsweise einfach durch den *IRIS Performer* ersetzt werden.

Das Visualisierungssystem *PowerVIZ* gliedert sich in fünf Aufgabengebiete, die über entsprechende Schnittstellen miteinander kommunizieren (siehe Abbildung 8.20). Der wichtigste Teilbereich ist die *Systemkontrolle* zur Instantiierung und Steuerung der anderen Bereiche. Die simulierten Ergebnisdaten des Strömungsvolumens und der Fahrzeugoberfläche werden in speziellen Objekten der *Strömungsdatenverwaltung* gespeichert und effizient angeordnet. Die *Graphikverwaltung* kapselt die Szenengraph-API und stellt ebenfalls den dreidimensionalen Viewer zur Verfügung. Die Benutzereingaben und die Systemausgaben der graphischen Benutzungsschnittstelle werden über die *Menüsteuerung* geleitet und die eigentlichen Visualisierungsalgorithmen sind in den *Interaktionselementen* untergebracht, die sowohl auf die Datenbank der Strömungsdaten zugreifen, als auch geometrische Objekte erzeugen und an die Graphikverwaltung weiterreichen. Die Aufgaben der Teilbereiche wurden in einer Reihe von Objektklassen umgesetzt.

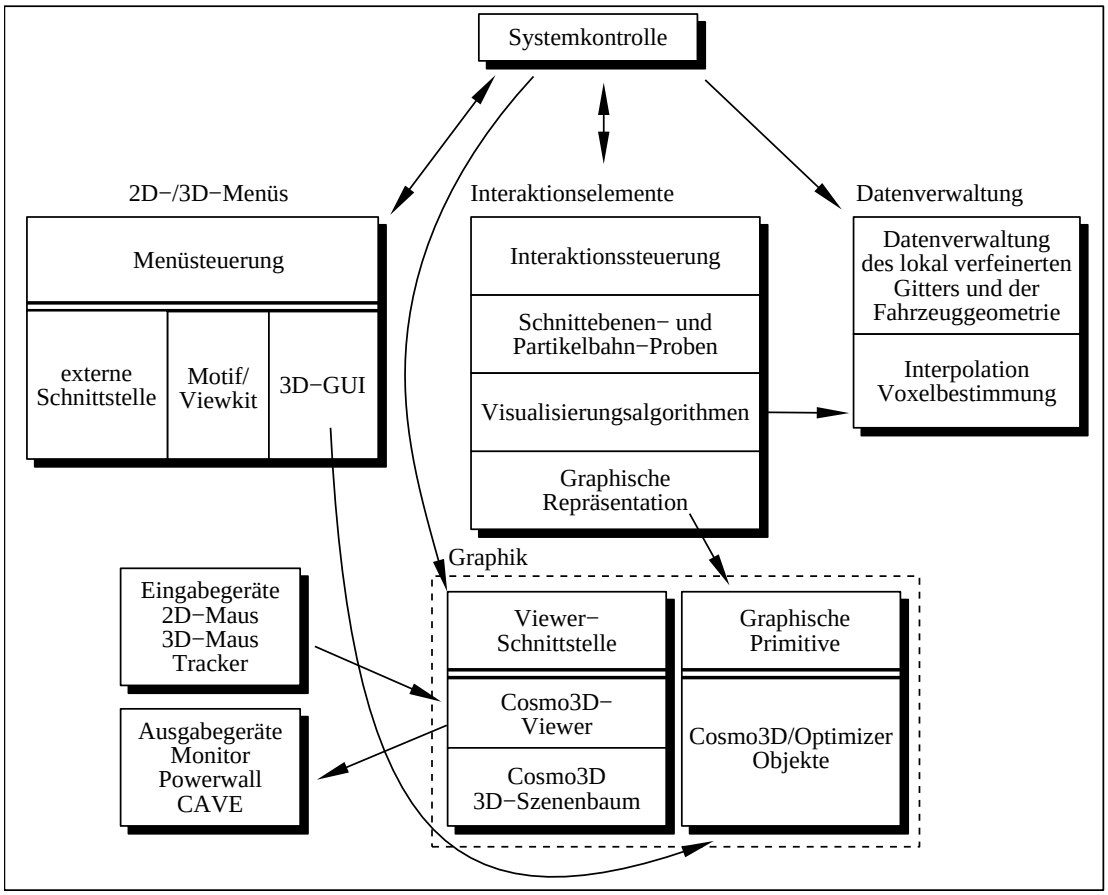

Abbildung 8.20: Skizze der Systemstruktur von *PowerVIZ*.

8.5.1 Systemkontrolle

Das einzelne Objekt der Systemkontrolle bildet mit seinen Klassenmethoden die Schnittstelle zwischen den Systembereichen von *PowerVIZ*. Sie instantiiert, verwaltet und steuert die Objekte der anderen Aufgabenbereiche. Von den Teilbereichen Menüsteuerung, Graphik-Viewer und Datenverwaltung wird jeweils eine Instanz zur Erfüllung der verschiedenen Aufgaben benötigt und nur für die Interaktionselemente verwaltet die Systemkontrolle eine Liste der Probe-Objekte, die je nach den Anforderungen der Visualisierung vergrößert bzw. verkleinert wird. Jegliche Kommunikation, wie zum Beispiel Benutzereingaben in das graphische Menü oder die graphischen Ausgaben der Interaktionselementen, leitet die Systemkontrolle an die entsprechenden Objekte weiter.

Die zentrale Struktur ermöglicht die vollständige Steuerung von *PowerVIZ* über eine externe Schnittstelle, wie sie beispielsweise für kooperatives Arbeiten, die Synchronisation mehrerer *PowerVIZ*-Instanzen oder die Steuerung mittels einer Skript-Sprache erforderlich ist.

8.5.2 Graphikverwaltung

Die Verwaltung der Graphik teilt sich in die Klassen der graphischen Primitive und dem eigentlichen Viewer zum Rendern des Szenengraphen auf. Die anderen *PowerVIZ*-Teilbereiche erstellen graphische Objekte und fügen diese über die Systemkontrolle in den Szenengraphen des Viewer ein, den der Benutzer über die angeschlossenen Eingabegeräte interaktiv manipulieren kann.

Die Klassen der graphischen Primitive kapseln die Elemente des *Cosmo3D*-Szenengraphen und ordnen sie in einer minimalen Szenengraphstruktur an. Damit eventuell auch andere Graphik-API's zum Einsatz kommen können, besitzen die graphischen Primitive nur die von *PowerVIZ* benötigte Funktionalität, die sich in einigen Fällen sogar aus mehreren *Cosmo3D*-Klassen zusammensetzt. Zur Wahrung der Schnittstelle können die anderen *PowerVIZ*-Bereiche nicht direkt auf die *Cosmo3D*-Objekte zugreifen.

Der Szenengraph wird von der Systemkontrolle durch Weitergabe der entstehenden Graphikobjekte im Viewer aufgebaut. Zwei spezialisierte Viewer für den Arbeitsplatz und für VR-Hardware-Umgebungen wurden vom *PowerVIZ*-Basisviewer abgeleitet. Der Arbeitsplatz-Viewer unterstützt lediglich ein einzelnes Graphiksystem und wird über ein 2D-*Motif/Viewkit*-GUI, die 2D- und 3D-Maus gesteuert. Die zweite Viewer-Variante hingegen ist auf das spezielle VR-Umfeld angepasst, unterstützt mehrere Graphiksysteme (*Multi-Pipe*) und besitzt spezielle VR-Gerätetreiber. Mittels einer Konfigurationsdatei können die einzelnen Windows des Viewers den verschiedenen Projektionsleinwänden einer *Powerwall* oder einer *CAVE* zugeordnet und das Blickfeld (engl. *Viewing Frustum*) inklusive weiterer Parameter (Stereo-Projektion usw.) definiert werden. Die Unterstützung des Tracking-Systems *Ascension MotionStar* in der *CAVE* ermöglicht das Head- und Hand-Tracking.

Für einen Wechsel von *PowerVIZ* auf eine andere Graphik-API müssen daher nur die graphischen Primitive und der Viewer neu implementiert werden.

8.5.3 Menüsteuerung

Die Steuerung von *PowerVIZ* am Arbeitsplatz erfolgt mittels einem zweidimensionalen GUI, während in VR-Umgebungen ein dreidimensionales GUI erforderlich ist. Zur Koordinierung der Ein- und Ausgaben über die Benutzerschnittstellen wurde die Menüsteuerung in einem Objekt gekapselt, das eine transparente Schnittstelle zwischen Systemkontrolle und dem *Motif/Viewkit*-basierten 2D GUI bzw. dem dreidimensionalen *G3Menu* bildet.

Alle Menüeingaben des Benutzer werden von der Menüsteuerung an die Systemkontrolle weitergeleitet. In die andere Richtung reicht die Systemkontrolle Parameteränderungen der *PowerVIZ*-Teilbereiche zur Aktualisierung an das Menükontrollobjekt weiter. Falls sowohl ein 2D- als auch ein 3D-Menü im Einsatz sind, stellt die Menüschnittstelle die synchrone Aktualisierung der Menüs sicher. Eine solche Konstellation tritt beispielsweise in einer *CAVE*-Umgebung auf, in der ein Benutzer in der dreidimensionalen Darstellung arbeitet und durch einen weiteren Anwender an einer Konsole betreut wird.

Die eindeutig definierte Menüschnittstelle erlaubt ebenfalls auf einfache Art und Weise das Hinzufügen von anderen Menükonzepten. Da alle Parameteränderungen über die Schnittstelle laufen, bietet sich nicht nur die Integration von neuartigen Menüs an, sondern es kann hier auch eine externe Anbindung stattfinden. Setzt beispielsweise ein Socket-Client auf der Menüschnittstelle auf, so können mehrere Applikationen synchronisiert werden.

8.5.4 Verwaltung der Berechnungsergebnisse

Die Objektstruktur zur Verwaltung der Strömungsergebnisse passt sich auf der Datenseite den Anforderungen der Gitterstrukturen von Volumen und Oberfläche an und bildet für die Visualisierungsalgorithmen eine transparente Schnittstelle (siehe Abbildung 8.21). Für zukünftige Anpassungen der Visualisierungsalgorithmen oder der Datenverwaltung ist somit sichergestellt, dass die jeweils andere Seite unverändert funktioniert.

Das lokal verfeinerte kartesische Gitter gibt die Objektstruktur zur Verwaltung der Volumendaten vor. Für jede *VRregion* wird ein Objekt erstellt, das neben den Datenfeldern für das Konditionsbyte und den skalaren und vektoriellen Größen auch die Zustandsinformationen, wie beispielsweise über den *VRlevel* oder die minimale und maximale Ausdehnung der *VRregion*, besitzt. Zu einer gegebenen Position kann somit jedes *VRregion*-Objekt die interpolierten Parameter oder die Nummer des Kind- oder Elterngitters, falls der Punkt außerhalb seines Wertebereichs liegt, liefern. Die Verwaltung der *VRregion*-Objekte erfolgt durch ein übergeordnetes Objekt, das die Interpolationsanfragen annimmt und an ein entsprechendes *VRregion*-Objekt weiterleitet.

Zu den Aufgaben der Objektstruktur der Fahrzeugoberfläche gehören die Verwaltung der graphischen Primitive und die Berechnung von Aufschlagpunkten der Partikelbahnen mittels einer Octree-Struktur. Die strikte Trennung von Darstellungsgeometrie und Octree-Struktur erlaubt die Optimierung der graphischen Primitive, beispielsweise durch Polygonreduktion zur Steigerung der Bildwiederholungsfrequenz ohne die Beeinflussung der Octree-Berechnungen. Die graphischen Primitive werden über die Systemkontrolle in den Szenengraphen eingehängt und im Viewer dargestellt. Die Octree-Struktur hingegen wird ohne graphische Repräsentation nur für die Berechnungen herangezogen.

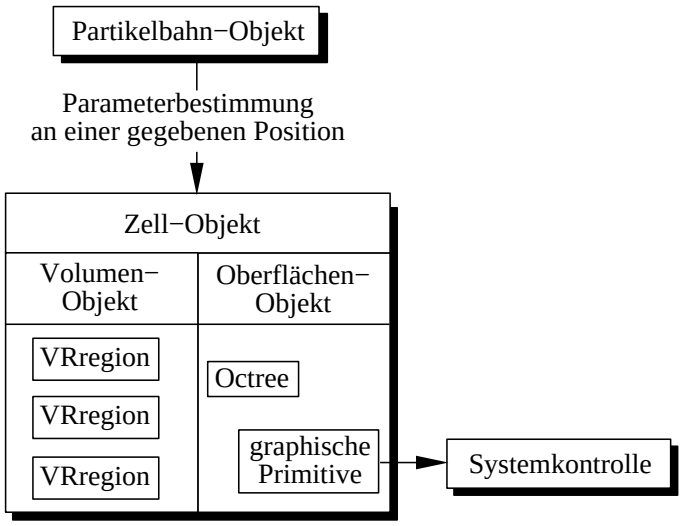

Abbildung 8.21: Objektstruktur zur Verwaltung der Berechnungsergebnisse des Strömungsvolumens und der Fahrzeugoberfläche.

Die Koordinierung der beiden Objektstrukturen für die Berechnung eines Integrationsschrittes ist Aufgabe der Zell-Objekte. Sie besitzen sowohl Zugang zu den einzelnen Volumen-Objekten als auch zum Octree. Zu einer gegebenen Position kann so ein Zell-Objekt in den *VRregion*-Objekten die Parameter bestimmen und anschließend einen Integrationschritt durch den Octree überprüfen lassen. Die Parameter werden für weitere Berechnungen an die Visualisierungsobjekte weitergereicht.

Letztendlich generieren die Visualisierungsobjekte aus den Parametern der Zell-Objekte graphische Primitive, die im Viewer visualisiert werden. Die Visualisierungsobjekte von *PowerVIZ* berechnen Partikelbahnen und Schnittebenen mit den unterschiedlichen Visualisierungsalgorithmen aus den vorherigen Kapiteln.

8.5.5 Interaktionselemente

Die Interaktionselemente von *PowerVIZ* entsprechen den Volumenproben aus Kapitel 8.3 und bilden die Schnittstelle zwischen den Benutzereingaben und den Visualisierungsalgorithmen. Sie legen die Ausgangslage und Parameter der Visualisierungsalgorithmen fest. Die Aufgaben der Volumenproben wurden in eine Basisprobe und davon abgeleitete Visualisierungsproben aufgeteilt (siehe Abbildung 8.22).

Die Basisprobe stellt Funktionalitäten für allgemeine Interaktionen zur Verfügung, die von allen Volumenproben benötigt werden. Die Positionierung und Orientierung der Probe gehört genauso zu den Aufgaben wie die Definition des Erscheinungsbildes (siehe Abbildung 8.10) und der Diskretisierung des Teilvolumens. Die allgemeine Pickgeometrie, bestehend aus acht Eckwürfeln und einem Drahtgitterrahmen, wird ebenfalls von der Basisprobe verwaltet.

Die unterschiedlichen Visualisierungsproben werden von der Basisprobe abgeleitet und er-

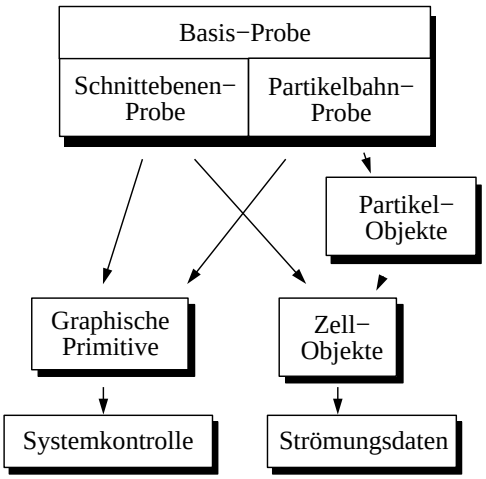

Abbildung 8.22: Objektstruktur der Volumenproben.

weitern den Funktionsumfang um die Visualisierungsalgorithmen inklusive der entstehenden Geometrie. Die Daten des Strömungsfeldes bestimmen die Volumenproben über die Zell-Objekte der Datenverwaltung und generieren graphische Primitive.

Die Systemkontrolle verwaltet die verschiedenen Visualisierungsproben in einem variablen Feld und leitet die graphischen Geometrien an den Viewer weiter. Zu den Verwaltungsaufgaben der Systemkontrolle gehört die Überprüfung der Auswahl von Proben durch den Benutzer und die Aktualisierung der Proben nach einer Transformation oder einer Änderung des Datensatzes.

Die verschiedenen Schnittstellen und das Konzept der Basisprobe stellen eine einfache Erweiterung von *PowerVIZ* durch neue Visualisierungsproben sicher.

8.6 Integration in eine CAVE-Umgebung

Die algorithmische Basis der Integration von *VtCrash* in eine CAVE-Umgebung (siehe Kapitel 6.5) lässt sich auf *PowerVIZ* übertragen. Die CAVE-Projektionsmatrizen kommen auch in der *PowerVIZ* CAVE-Erweiterung in einem Multi-Pipe- und Multi-Thread-Viewer zum Einsatz. Die Szenengraph-API *Cosmo3D/OpenGL Optimizer* besitzt im Gegensatz zu *WorldToolKit* keinen standardisierten Multi-Pipe-Viewer. Die Multi-Thread-Erweiterung ist jedoch mit Hilfe von Semaphoren, so genannten *Barriers*, möglich.

Die *Context*-Verwaltung von *Optimizer* erlaubt das Öffnen mehrerer Viewer auf einem oder verschiedenen Graphiksystemen (*Displays*) und stellt somit die grundlegende Funktionalität für Multi-Pipe-Viewer zur Verfügung. Um nun die Renderer auf den verschiedenen Graphiksystemen zu synchronisieren, muss die Zeitdifferenz zwischen den Updates der Displays minimal sein. Im Allgemeinen wird die dreidimensionale Szene in einem für den Benutzer unsichtbaren Speicherbereich, dem so genannten *Backbuffer*, gezeichnet und nach Vollendung mit einem einzigen Kommando (dem *Swapbuffer*-Kommando) sichtbar. Diese Vorgehensweise wird für die Synchronisation der Viewer ausgenutzt. Die Applikation startet die Viewer-Prozesse, die gleich

zu Beginn in der *Draw-Barrier* angehalten werden. Sobald die Applikation nach dem Aufbau des Szenengraphen ebenfalls in die *Draw-Barrier* gelangt, zeichnen die Viewer-Prozesse die Szene aus der entsprechenden Blickrichtung und der Applikationsprozess stoppt direkt in der *Swap-Barrier*. Haben alle Viewer-Prozesse die Szene gezeichnet und sind ebenfalls in der *Swap-Barrier* angekommen, wird ein praktisch synchrones *Swapbuffer*-Kommando in allen Viewern durchgeführt, wie in Abbildung 8.23 skizziert. Die Treiber für das Tracking-System sind nicht Teil der *Cosmo3D/OpenGL Optimizer*-API, sondern mussten ebenfalls durch Integration in die Hauptschleife von *PowerVIZ* hinzu gefügt werden.

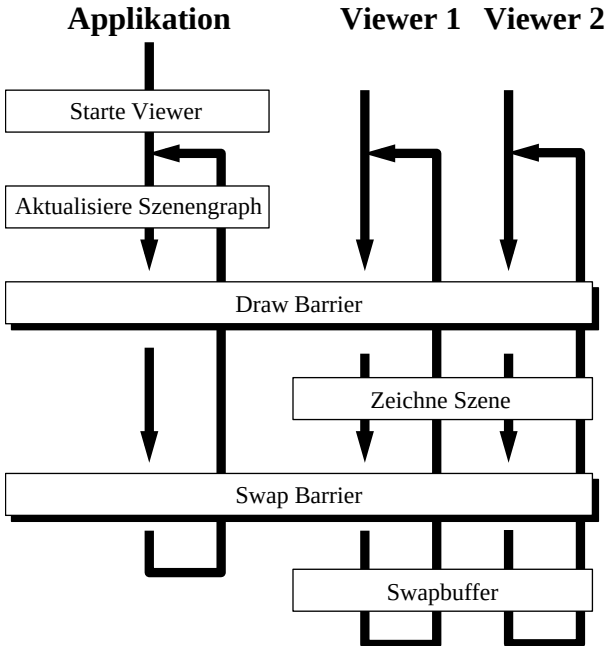

Abbildung 8.23: Synchronisation von Optimizer-Viewern mittels Barriers.

Die Interaktion des Benutzers erfolgt ähnlich wie in *VtCrash* über ein eingebundenes *G3Menu*, das der Benutzer mittels eines virtuellen Degens bedienen kann. Der Degen erlaubt ebenfalls die Transformation von Volumenproben. Diese Form der Strömungsanalyse gleicht stark dem realen Windkanal, in dem der Ingenieur eine Rauchsonde in den Luftstrom hält, die aus einem langen Rohr besteht. Die starke Ähnlichkeit zwischen der realen und der virtuellen Arbeitsweise eröffnet auch vollkommen ungeübten Anwendern die intuitive Interaktion mit den Simulationsergebnissen. Wichtig hierfür sind die schnellen Reaktionszeiten von *PowerVIZ*, da im Falle einer ruckenden Darstellung der Anwender den Eindruck der Immersion verliert.

Durch die Erweiterung um einen Multi-Pipe-Viewer skaliert *PowerVIZ* vom Ingenieurs-Arbeitsplatz über mehrseitige Projektionsanlagen bis hin zu vollimmersiven VR-Umgebungen. Die gleichbleibende Benutzerführung in allen Varianten reduziert den Lernaufwand des Anwenders auf ein Minimum und ebnet auf einfache Weise den Zugang zum VR-Labor. Eine Probeinstallation der VR-Erweiterung bei der *BMW Group* hat den sinnvollen Einsatz einer CAVE für die Analyse von Simulationsergebnissen demonstriert.

8.7 Kooperatives Arbeiten und Variantenvisualisierung

Die Globalisierung der Firmenstrukturen und das Outsourcing von Entwicklungsaufgaben stellen neue Anforderungen an die Arbeitsplatzumgebungen. Neben der klassischen Arbeitsplatzanwendung werden immer öfter kooperative Applikationen gefordert, die via Intra- und Internet mehrere Ingenieure in einer virtuellen Arbeitsbesprechung zusammenführen. Die Synchronisation mehrere Visualisierungssysteme über das Netzwerk kann diese Anforderungen erfüllen.

Die standardisierte Client-Server-Software *CORBA* („Common Object Request Broker Architecture“) ermöglicht den Datenaustausch zwischen Applikationen über Computernetzwerke. Plattformunabhängige Schnittstellen zur Kommunikation zwischen Applikationen lassen sich mit der *CORBA*-Programmiersprache IDL („Interface Definition Language“) für C++ und *Java* definieren. Mit *CORBA* können so auf einfache Weise plattformunabhängige Programme zum Datenaustausch entwickelt werden. In dieser Arbeit wurde *CORBA* in *PowerVIZ* für kooperatives Arbeiten und zur Visualisierung von Berechnungsvarianten eingesetzt.

Zur Koordination der synchronisierten Applikationen und zur Verwaltung des Tokens ist ein Session-Server erforderlich, an dem sich die Applikationen als Clients an- und abmelden. Für einen bidirektionalen Datenaustausch zwischen den einzelnen Komponenten sind *PowerVIZ* und der Session-Server sowohl *CORBA*-Server als auch *CORBA*-Client. Die Verknüpfung von drei *PowerVIZ*-Programmen mit dem Session-Server ist in Abbildung 8.24 skizziert.

Eine Arbeitssitzung wird von dem ersten Teilnehmer durch Starten des Session-Servers eta-

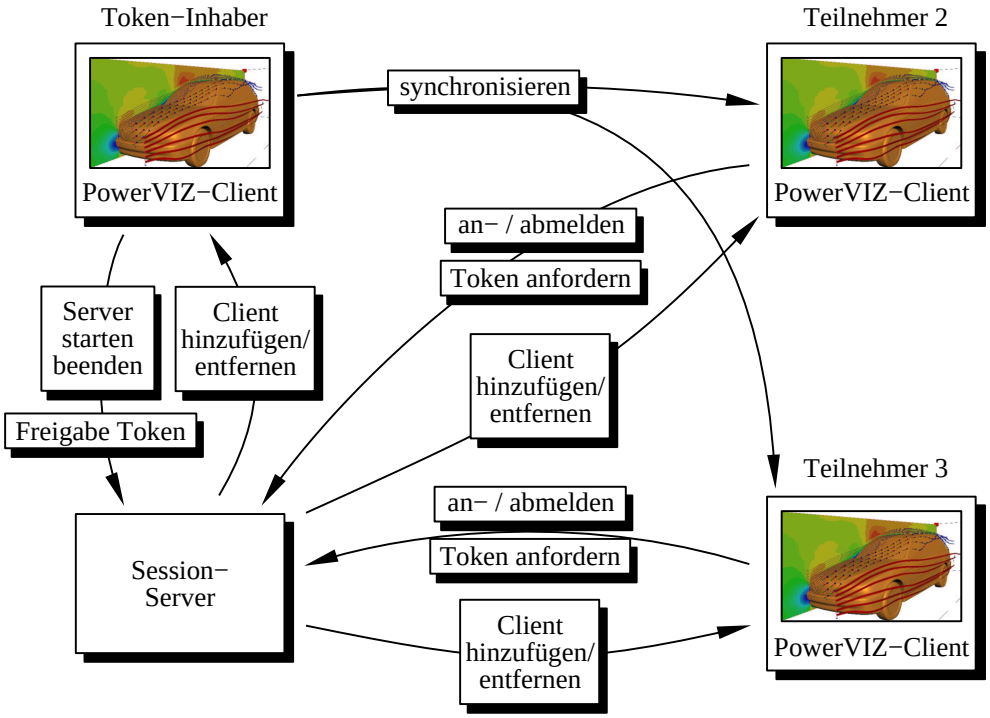

Abbildung 8.24: Verknüpfung von drei *PowerVIZ*-Anwendungen mit dem Session-Server.

bliert und es können sich anschließend weitere Teilnehmer anmelden. Die Verwendung eines Tokens, das initial der erste Teilnehmer erhält, beschränkt die Berechtigung für Interaktionseingaben in der synchronisierten Sitzung auf den Token-Inhaber. Die anderen Teilnehmer können am Session-Server das Token anfordern und erhalten es nach Freigabe durch den Inhaber. Der Token-Inhaber sendet an die angeschlossenen Teilnehmer Informationen über alle Benutzereingaben in das GUI bzw. in den Viewer und synchronisiert so den Augpunkt und alle Volumenproben der Teilnehmer. Zur Erzielung einer höheren Performanz sendet der Token-Inhaber ohne Umweg über den Session-Server direkt an die anderen Teilnehmer. Jeder *PowerVIZ*-Client besitzt daher eine eigene Liste der angeschlossenen Teilnehmer, die vom Session-Server durch Hinzufügen und Entfernen der Clients verwaltet wird.

Zusätzlich zum kooperativen Arbeiten ermöglicht das Client-Server-Konzept die direkte Visualisierung von Berechnungsvarianten. Zu diesem Zweck werden in synchronisierten *PowerVIZ*-Clients verschiedene Datensätze geladen und können somit vom Anwender im direkten Vergleich parallel ausgewertet werden. Diese Analyse kann sowohl auf einem Rechner oder zur Performanzsteigerung auf mehreren Rechnern erfolgen. Die synchrone Variantenvisualisierung erfordert keine Erweiterungen gegenüber dem kooperativen Arbeiten.

Das Client-Server-Konzept basierend auf *CORBA* eröffnet dem Visualisierungssystem *PowerVIZ* neue Einsatzgebiete.

8.8 Fazit

Mit *PowerVIZ* wurde ein interaktives Visualisierungswerkzeug entwickelt, das die speziellen Eigenschaften eines lokal verfeinerten Gitters zur Erzielung hoher Interaktions- und Bildwiederholungsraten ausnutzt. Die bekannten Visualisierungsalgorithmen zur Berechnung von Schnittebenen und Partikelbahnen wurden auf die Gitterstruktur angepasst und auf kurze Berechnungszeiten optimiert. Es lassen sich somit die Volumenproben mit hohen Bildwiederholungsraten auch in Stereo-Darstellung interaktiv positionieren. Die hohen Interaktionsraten qualifizieren *PowerVIZ* auch für VR-Hardware-Umgebungen, an die es durch die Einbindung von VR-Gerätetreibern und einen speziellen Viewer angepasst wurde.

Zukünftige Erweiterungen können in die Systemstruktur von *PowerVIZ* leicht eingegliedert werden. Die Auslegung der einzelnen Schnittstellen bietet hier Erweiterungsmöglichkeiten auf vielen Gebieten. Zusätzliche Visualisierungstechniken wie beispielsweise massebehaftete Teilchen, direktes Volumenrendering oder LIC („Line Integral Convolution“) auf der Oberfläche können so schnell integriert werden.

Der Einsatz von *PowerVIZ* im produktiven Fahrzeugentwicklungsprozess der *BMW Group* unterstützte den Entschluss zu einer Kommerzialisierung des Visualisierungssystems.

Kapitel 9

Resultate

Die vorliegende Arbeit hat in vielen Teilbereichen des Postprocessings im Fahrzeugentwicklungsprozess Einsatzszenarien für virtuelle Umgebungen und Techniken der virtuellen Realität aufgezeigt. Es wurden Visualisierungsalgorithmen für Struktur- und Volumen-basierte Simulationen demonstriert und in den beiden vorgestellten prototypischen Applikationen *VtCrash* und *PowerVIZ* implementiert. Neben den objektiv erfassbaren Resultaten in Bezug auf die Darstellungs- und Visualisierungsgeschwindigkeit hat diese Arbeit auch Erfolge in der Umsetzung der Verfahren im gesamten Arbeitsumfeld, von Workstations bishin zu VR-Labors, aufzuweisen.

9.1 VtCrash

Die Zielsetzung von *VtCrash* war die Untersuchung, in wieweit die Techniken der virtuellen Realität für das Tagesgeschäft der Analyse von Simulationsergebnissen geeignet sind. Zu Beginn des Projektes gab es keine Erfahrungswerte über die Handhabung der enormen Datenmengen eines zeitabhängigen Simulationslaufes in den verfügbaren Szenengraph-APIs. Auch war nicht bekannt, welche VR-Hardware zuverlässig in den produktiven Arbeitsablauf integriert werden kann. Anhand von *VtCrash* wurde in den Berechnungsabteilungen der *BMW Group* untersucht, mit welchen Visualisierungstechniken in virtuellen Umgebungen gearbeitet werden kann und wie die Interaktionsmechanismen ausgelegt werden müssen, damit auch Nicht-Spezialisten Zugang zu den Ergebnissen erhalten.

Ein wichtiger Punkt für eine erfolgreiche Eingliederung in den Prozessablauf war die direkte Unterstützung verschiedener Dateiformate. Sehr hilfreich in diesem Zusammenhang war die Kooperation mit den Herstellern der Simulationscodes. Diese Zusammenarbeit führte zur Untersuchung von zeitabhängigen Oberflächengeometrien mit den in Crashtest- und Schwingungssimulationen vorkommenden statischen Gitterstrukturen, so wie den aus der Umformtechnik bekannten zeitlich variablen Gitter. Eine spezielle Schnittstelle und Datenstruktur wurde für die unstrukturierten, zeitabhängigen Akustikdatensätze entwickelt, die kombiniert mit instationären Oberflächengeometrien visualisiert wurden.

Verschiedene Mechanismen für die Interaktion des Benutzers mit den zeitabhängigen Geo-

metrien wurden sowohl für den Arbeitsplatz als auch für immersive Umgebungen untersucht. Bewährt hat sich der Ansatz, die Interaktionen von den Eingabegeräten direkt an den Szenengraphen weiterzuleiten und nicht den Umweg über das GUI zu gehen, wie er von den traditionellen Postprocessing-Systemen bevorzugt wird. An Funktionalität für die Manipulation von zeitabhängiger Geometrie wurde speziell die Transformation animierter Komponenten, die Berechnung von Schnitten und die Bestimmung von Intrusionen betrachtet.

Effiziente Algorithmen sind für die kombinierte Visualisierung von instationären Geometrien und Volumen unerlässlich. Aus theoretischen Grundlagen wurden schnelle Algorithmen zur Berechnung von Schnittebenen und propagativen Isoflächen in unstrukturierten Gittern entwickelt, deren Reaktionszeiten ausreichen, um dem Benutzer eine interaktive Steuerung der Visualisierung über Manipulatoren zu ermöglichen. Die einfache, intuitive Positionierung der Manipulatoren im Volumen fördert das Verständnis der Simulationsergebnisse beim Anwender.

Die betrachteten Algorithmen und Verfahren wurden in dem Visualisierungssystem *VtCrash* implementiert. Die objektorientierte Systemstruktur wurde so ausgelegt, dass sowohl Geometrie- als auch Volumendaten von einer Datenverwaltung gehandhabt und in einen Szenengraphen integriert wurden. Die direkte Manipulation des Szenengraphen durch die Visualisierungsalgorithmen ermöglicht schnelle Reaktionszeiten für interaktives Arbeiten.

Die Unterstützung von semi-immersiven und immersiven Arbeitsumgebungen erforderte neben der Einbindung von Treiberbibliotheken spezieller VR-Geräten auch die Ausarbeitung von Interaktionsmechanismen für die Benutzer. Fragestellungen im Umgang des Benutzers mit virtuellen Objekten und der Benutzerführung in immersiven Umgebungen standen dabei im Vordergrund. Gelöst wurden die Aufgaben durch die Verwendung eines virtuellen Zeigestabes für die Benutzerinteraktionen und der Integration des dreidimensionalen *G3Menu* zur Benutzerführung.

Teilergebnisse der Arbeit mit *VtCrash* wurden auf einer Reihe von Konferenzen [75, 74, 106, 108, 109, 111, 110] veröffentlicht, die schließlich zu einem Gesamtüberblick in einem Zeitschriftenartikel [107] führten. Im industriellen Umfeld gab es zwei Messepräsentationen von *VtCrash* auf den *SGI*-Ständen der *AUTOFACT 96* in Detroit und der *High Performance Computing 98* in San Diego. Viele *BMW*-interne Demonstrationen für kleinere Gruppen und ganze Abteilungen sowie für Führungskräfte und auf internen Messen förderten den Bekanntheitsgrad von *VtCrash* und erhöhten die Akzeptanz der Techniken der virtuellen Realität.

Die Arbeiten an *VtCrash* haben zu der großen Verbreitung von VR-Hardware bei *BMW* mitbeigetragen. So gehört die Stereoprojektion in vielen Abteilungen mittlerweile zum Alltagsbetrieb und die Demonstration des Potenzials der interaktiven Arbeitsweisen für den Entwicklungsprozess führte auch zur Integration von VR-Techniken in traditionelle Applikationen.

9.2 PowerVIZ

Die Arbeiten mit *VtCrash* im Bereich der Akustikvisualisierung haben deutlich die Grenzen der Szenengraph-API *WorldToolKit* für die interaktive Volumenvisualisierung aufgezeigt. Die hierbei gewonnenen Erfahrungen sind in das anschließende Projekt zur Visualisierung von Simulationsergebnissen der Strömungsmechanik, *PowerVIZ*, eingeflossen. Zu den Arbeitsgebieten dieses Projektes gehörten sowohl die Erprobung der flexibleren Szenengraph-API *OpenGL/Optimizer*

als auch die Untersuchung von Visualisierungsalgorithmen für die interaktive Visualisierung von vektoriellen und skalaren Volumenfeldern.

Ziel war die Entwicklung einer virtuellen Umgebung zur Visualisierung von Berechnungsergebnissen des CFD-Simulationspaketes *PowerFLOW*. Im Gegensatz zu vielen anderen CFD-Solvern verwendet *PowerFLOW* einen Lattice-Boltzmann-Ansatz auf lokal verfeinerten Gittern. Die Auflösungsübergänge und die explizite Beschreibung der Fahrzeuggeometrie in dieser Gitterstruktur stellen Probleme für die traditionellen Visualisierungsalgorithmen dar. Die vorgestellte Datenstruktur zur effizienten Speicherung von lokal verfeinerten Gittern bietet durch die Verwendung eines Konditionsbytes pro Zelle in Kombination mit einer Octree-Struktur zur Handhabung der expliziten Fahrzeuggeometrie einen schnellen Zugriff auf die einzelnen Volumen- und Oberflächenparameter. Die Struktur der lokal verfeinerten Gitter wurde für einen schnellen Zellbestimmungsalgorithmus ausgenutzt. Auf dieser Basis wurden die Visualisierungsalgorithmen hinsichtlich ihrer Echtzeittauglichkeit untersucht. Im Mittelpunkt der Betrachtungen standen die Berechnungen von Partikelbahnen und Schnittebenen.

Die Berechnungszeiten von Partikelbahnen wurden für Integrationsmethoden mit statischen und adaptiven Schrittweiten bestimmt. In der praktischen Anwendung haben sich die adaptiven Integrationsalgorithmen trotz höherem Berechnungsaufwand, aber Dank größerer Schrittweiten, bewährt. Die besten Ergebnisse wurden mit dem adaptiven Runge-Kutta-Schema 3(4) erzielt.

In der lokal verfeinerten kartesischen Gitterstruktur stellt die explizite Geometriebehandlung ein Problem für die Partikelbahnberechnung dar. Um das Durchdringen der Geometrie von Partikelbahnen zu verhindern, muss jeder Integrationsschritt gegen die Fahrzeuggeometrie getestet werden. Der Einsatz des Octrees und der Zellen-Konditionsbytes reduzierte die benötigten Schnittuntersuchungen auf eine für Echtzeitberechnungen akzeptable Anzahl, wie anhand von Beispielen nachgewiesen wurde. Der Berechnungsaufwand stieg durch die Kollisionsberechnungen um durchschnittlich 50% an.

Besonders deutlich wurde die hohe Geschwindigkeit der Zellbestimmung bei der Untersuchung von freibeweglichen Schnittebenen. Skalare und vektorielle Parameter werden in jedem Berechnungszyklus an den diskreten Abtastpunkten im Volumen bestimmt und durch Einfärbung der Polygone bzw. durch Vektorpfeile visualisiert. In den Beispieldatensätzen konnten auf mittleren Workstations Ebenen mit 50^2 Abtastpunkten bei Frameraten $> 10Hz$ bewegt werden.

Die Berechnungen von Isolinien auf den Schnittebenen wurden in dieser Arbeit durch die Ausnutzung der Graphikhardware beschleunigt. Die Verwendung von Hardware-unterstützten eindimensionalen Texturen ermöglichte die interaktive Visualisierung von Konturen. Geleitet von den positiven Erfahrungen mit einzelnen Schnittebenen wurde ebenfalls die interaktive Berechnung von Schnittebenenstapeln analysiert. Teiltransparente Texturen wurden eingesetzt, um so eine dreidimensionale Visualisierung von Strömungsmerkmalen zu erzielen.

Implementiert wurden die vorgestellten Verfahren in dem objektorientierten Visualisierungssystem *PowerVIZ*. Die Steuerung der Visualisierungsmethoden erfolgte ähnlich wie in *VtCrash* über Interaktionsproben und ermöglichten so dem Anwender eine sichere Positionierung der Interaktionselemente. Verstärkt wurde die Intuitivität der Interaktion durch den Einsatz der Stereoprojektion, die es dem Anwender erlaubt, eine Probe sehr schnell und präzise zu platzieren. Die Interaktionsproben können über eine Reihe von Parametern, beispielsweise Größe, Anzahl der Abtastpunkte etc., an die Erfordernisse des Datensatzes angepasst werden. In immersiven Um-

gebungen werden die Proben an die Eingabegeräte gekoppelt und lassen den Benutzer auf diese Weise vollkommen in die virtuelle Welt eintauchen. Das Einsatzspektrum von *PowerVIZ* reicht somit vom Arbeitsplatz bis hin zum VR-Labor. Die Szenengraph-API *OpenGL/Optimizer* hat sich in diesem Projekt durch ihre hohe Flexibilität und Darstellungsgeschwindigkeit bewährt und wurde für den Einsatz in immersiven Umgebungen mit einer Multi-Pipe-/Multi-Thread-Variante des Viewers ergänzt.

Teilergebnisse dieses Projektes wurden ebenfalls auf Konferenzen veröffentlicht [105, 104] und in zahlreichen *BMW*-internen Demonstrationen einem breitgefächerten Publikum vorgestellt. Die Vorteile des geringen Speicherverbrauchs und der hohen Interaktivität gegenüber traditionellen Postprocessing-Systemen führten bereits zu einem produktiven Einsatz der prototypischen Applikation. *PowerVIZ* ersetzt bei der *BMW Group* in vielen Analyseschritten bereits die Standardwerkzeuge. Rückmeldungen von *BMW*-Mitarbeitern zeugen von einem regen Einsatz auf der Powerwall und am Arbeitsplatz.

Die Arbeiten an *PowerVIZ* bilden den Grundstock für das Transferprojekt FORTWIHR III der Bayerischen Forschungsstiftung, von der eine weitere Doktorandenstelle finanziert wird. Ein angestrebtes Ziel des Forschungsprojektes ist die Simulation von Schmutzpartikeln, einschließlich der Berechnung von Aufschlagpunkten und ihrer Verteilung auf der Fahrzeugoberfläche.

9.3 Kooperatives Arbeiten

Die Unterstützung von kooperativem Arbeiten wurde im Laufe der Zusammenarbeit von den Projektpartnern angeregt. Nach Analyse der Einsatzumgebungen wurden zwei unterschiedliche Ansätze für die praktische Anwendung im produktiven Prozess erprobt. Im ersten Ansatz wurde das *VRML*-Dateiformat als Informationsvehikel über das Internet betrachtet. Die Kopplung von mehreren Applikationen über die Middleware *CORBA* war Gegenstand der zweiten Untersuchung.

Das Arbeitsumfeld für das *VRML*-Dateiformat war die Visualisierung von Berechnungsergebnissen aus Struktursimulationen. Interessante Teilmodelle von Simulationsergebnissen können hierfür direkt aus der virtuellen Umgebung von *VtCrash* erstellt und als *VRML*-Datei gespeichert werden. Zur Darstellung der Dateien kamen Standard-*VRML*-PlugIns für Internet-Browser zum Einsatz, deren Funktionalität für die Anforderungen der Visualisierung erweitert werden musste. Um dem Anwender die gewohnte Funktionalität eines Postprocessing-Werkzeuges zur Verfügung zu stellen, wurden die Geometrie-Elemente kombiniert mit Sensor-Knoten im *VRML*-Szenengraphen angeordnet. Die interaktiven Auswahlverfahren und die Transformation der Geometrien konnten so über Pick- und Motion-Events der 2D-Maus im Standardbrowsern realisiert werden. Die Animation der zeitabhängigen Szene erfolgt durch einen Switch-Knoten und lässt sich vom Anwender über ein *VRML*- oder AWT-Buttonfeld steuern. Als sehr robust hat sich das in den *VRML*-Szenengraphen integrierte dreidimensionale Menü auf den eingesetzten Plattformen erwiesen, wohingegen die Animationssteuerung über das AWT-Buttonfeld deutliche Schwächen durch die Inkompatibilitäten der plattformabhängigen *Java*-Implementierungen zeigte.

Die Etablierung einer kooperativen Arbeitssitzung erfolgt über HTML-Seiten und CGI-

Skripte, über die sich mehrere Anwender via Internet anmelden können. Mittels einer Socket-Verbindung, *Java*-Applets und der EAI-Schnittstelle werden die *VRML*-Szenengraphen in den Browsern der Teilnehmer synchron verändert. Der als Master definierte Teilnehmer kann durch Benutzereingaben die Interaktionen und Animationen der angeschlossenen Browser steuern. Die Stabilitätsprobleme der *Java*-Implementierungen erwiesen sich allerdings auch für das kooperative Arbeiten als sehr nachteilig und verhinderten so vorerst den produktiven Einsatz dieser Visualisierungstechnik.

Erfolgversprechender konnte die Visualisierung mit *VRML*-Dateien im normalen Analysevorgang eingesetzt werden, da das einfache Versenden per email und die Plattformunabhängigkeit große Vorteile für das Tagesgeschäft bieten. Die synchronisierte Darstellung von zwei Berechnungsvarianten in einen Browser lässt sich schnell erzeugen und ist eine Hilfe bei der Beurteilung der durchgeführten Änderungen. Letztendlich können die Entwicklungsschritte von Teilmodellen als *VRML*-Dateien über das Intranet leicht auf *HTML*-Seiten den involvierten Ingenieuren zur Dokumentation bereitgestellt werden. Die Ansätze zum Einsatz von *VRML*-Dateien als Visualisierungsmedium wurden im Bereich der Dokumentation bei BMW erprobt und als Erfahrungsbasis zur Weiterentwicklung des *Crashbench*-Projektes genutzt. Es wurde jedoch deutlich, dass diese Technik noch nicht reif für den produktiven Einsatz in einem industriellen Umfeld ist.

Der zweite Ansatz für kooperatives Arbeiten wurde in dem Visualisierungssystem für Strömungssimulationen *PowerVIZ* untersucht. Mittels eines Client-Server Ansatzes können in einer Arbeitssitzung mehrere Applikationen über die Middleware *CORBA* synchronisiert werden. Nach der Übertragung eines Adress-Schlüssels erfolgt die Anmeldung beliebig vieler Teilnehmer an der kooperativen Sitzung, in der jedoch nur der Inhaber des Tokens Interaktionen in den synchronisierten Applikationen auslösen kann.

Die Umsetzung des *CORBA*-Ansatzes in *PowerVIZ* hat gezeigt, dass diese Form der Kopplung nur effizient implementiert werden kann, wenn eine klar definierte Schnittstelle für die Synchronisation vorhanden ist. *PowerVIZ* besitzt als Schnittstelle eine C++ Klasse, die alle Funktionalitäten für die graphische Benutzeroberfläche und auch *CORBA* bereitstellt. Die vollständige Funktionalität steht somit in den kooperativen Arbeitssitzungen zur Verfügung.

Im Gegensatz zum *VRML*-Ansatz müssen die Ergebnisdatensätze allen Teilnehmer lokal zugänglich sein, da die enormen Datenmengen einen schnellen Zugriff über das Netzwerk verhindern. Andererseits bieten sich durch den großen Umfang an Funktionalitäten mehr Analysemöglichkeiten, als sie eine Kopplung von *VRML*-Viewern bieten kann. In den Probeläufen zeichnete sich der *CORBA*-Ansatz ebenfalls durch eine höhere Stabilität aus, wodurch eine *CORBA*-Synchronisation von den Kooperationspartnern bevorzugt wurde.

9.4 Integration in den produktiven Fahrzeugentwicklungsprozess

Die vorliegende Arbeit war organisatorisch der Abteilung für Methodenentwicklungen der Außenhautkonstruktion der *BMW Group* zugeordnet. Neben der methodischen Arbeit sollten die

Einsatzmöglichkeiten der virtuellen Realität für den Fahrzeugentwicklungsprozess anhand von prototypischen Applikationen untersucht werden. In enger Zusammenarbeit mit den am Entwicklungsprozess beteiligten Abteilungen konnten die entstandenen Visualisierungssysteme direkt im produktiven Betrieb erprobt werden. Jede Abteilung besaß aufgrund der unterschiedlichen Simulationssysteme und deren Datenstrukturen abweichende Anforderungen an ein Visualisierungssystem, trotzdem entwickelte sich in allen Bereichen der Wunsch nach einem interaktiven Visualisierungswerkzeug. Die aus der Zusammenarbeit mit unterschiedlichen Abteilungen resultierenden Synergieeffekte wirkten sich positiv auf diese Arbeit aus.

Eine Zusammenarbeit mit Ingenieuren in produktnahen Abteilungen kann nur durch deren persönliches Engagement erfolgreich sein. Überzeugende Ergebnisse aus anderen Anwendungsfällen weckten schnell das Interesse der Anwender an den neuen Visualisierungsmethoden und ebneten den Weg für neue Aufgabenfelder. Ausgangspunkt dieser Arbeit waren die nichtlinearen numerischen Berechnungen in Form von Crashtestuntersuchungen der Abteilung für Finite-Element-Simulationen. Die große Erfahrung dieser Abteilung im Umgang mit numerischen Simulationen führt auch zu einem großen Erfahrungsschatz im Umgang mit Analysewerkzeugen. Die Berechnungsspezialisten stießen schnell an die Grenzen der traditionellen Postprocessing-Systeme und konnten somit sehr gute Denkanstöße und Funktionsvorgaben für die interaktive Visualisierung dieser Arbeit geben, um die alten Grenzen zu überwinden. In diesem Umfeld entstand das Visualisierungssystem *VtCrash* zur interaktiven Darstellung von Crashtest-Berechnungsergebnissen.

BMW-interne Demonstrationen von *VtCrash* führten zu einem hohen Bekanntheitsgrad der Ergebnisse. Das gestiegene Interesse spiegelte sich in Anfragen von Abteilungen mit ähnlichen Problemstellungen, wie beispielsweise Schwingungsanalysen und kinematischen Untersuchungen, wieder. Die Integration der linearen Simulationen war durch die starke Ähnlichkeit zu den bisherigen Datenstrukturen kein prinzipielles Problem, führte aber zu einer weiteren Steigerung der Akzeptanz von interaktivem Arbeiten in den entsprechenden Abteilungen.

An die Schwingungsanalysen der Karosserie schließt sich häufig die Simulation der Luftdruckverteilung innerhalb der Fahrzeugkabine an. Für die parallele Visualisierung von zeitabhängiger Geometrie und Volumen gab es kein zufriedenstellendes Werkzeug. Die prototypische Erweiterung von *VtCrash* zur Visualisierung von Akustiksimulationen eröffnete den Ingenieuren eine neue Sichtweise auf die Berechnungsergebnisse und zeigte neue Analysemethoden auf.

Die Unterstützung der zeitabhängigen Volumendatensätze aus der Akustiksimulation war der Einstieg zur Volumenvisualisierung von Vektor- und Skalarfeldern. So wurde in der Aerodynamik-Abteilung das Interesse an einer Zusammenarbeit für die Visualisierung der Außenhautumströmung von *PowerFLOW* geweckt, als deren Resultat das Visualisierungssystem *PowerVIZ* entstand.

Verteiltes Arbeiten ist für viele Abteilungen ein wichtiger Gesichtspunkt, da mit ausländischen Niederlassungen oder Zulieferfirmen zusammengearbeitet wird. In dieser Arbeit wurden zwei verschiedenen Ansätze für verteiltes Arbeiten untersucht. Zum einen die Kopplung von *VRML*-Browsern über das Intra-/Internet und zum anderen die Synchronisation mehrere Applikationen in einem *CORBA*-gestützten Client-Server-Ansatz. Beide Varianten sollten in den entsprechenden Abteilungen zum Einsatz kommen. Es hat sich jedoch schnell gezeigt, dass im pro-

duktiven Umfeld die Umsetzung dieser Techniken noch sehr schwierig ist. Im Falle der *VRML*-basierten Visualisierung erwiesen sich die verschiedenen *Java*-Implementationen als instabil und somit als zu unsicher, wohingegen die vielversprechende *CORBA*-Variante noch weiterer Erforschung bedarf.

Die beiden entstandenen interaktiven Visualisierungssysteme kamen in den Abteilungen sowohl am Arbeitsplatz als auch in semi-immersiven und voll-immersiven Hardware-Umgebungen zum Einsatz. Der hohe Ressourcenbedarf von *VtCrash* verschob dessen Einsatzschwerpunkt stärker zu den speziellen Anwendungen in Besprechungen und Demonstrationen, in denen *VtCrash* durch seine einzigartigen Funktionalitäten bestach. *PowerVIZ*, als zweites großes Projekt dieser Arbeit, profitierte stark aus den Erfahrungen mit *VtCrash* und den zu diesem Zeitpunkt verfügbaren APIs, wie beispielsweise dem *OpenGL/Optimizer*. Es konnte so der Arbeitsplatzrechner vom Projektbeginn an als Zielplattform von *PowerVIZ* festgelegt werden. Beide Prototypen haben in den Berechnungsabteilungen starke Impulse für die Anforderungen an ein Postprocessing-System gegeben.

Diese Arbeit hat neben den implementierten Applikationen auch die enge abteilungsübergreifende Zusammenarbeit als Ergebnis. Die daraus folgenden Synergieeffekte hatten sowohl auf die vorliegende Arbeit als auch auf die beteiligten Abteilungen einen positiven Einfluss. Ebenfalls wurden die Vorteile einer engen Verknüpfung von Industriestipendium und universitärer Forschung durch die Ergebnisse dieser Arbeit unterstrichen, auch wenn dies ein hohes Maß an persönlichem Engagement aller Beteiligten erforderte.

9.5 Erfahrungen aus dem Einsatz der Prototypen

Die durchweg positive Resonanz bei den Berechnungsingenieuren auf die neuen Interaktionsmechanismen der beiden Visualisierungssysteme wurde auf Messe- und Vortragsdemonstrationen unterstrichen. Es hat sich zum Teil aber auch gezeigt, dass die Anforderungen an ein Visualisierungswerkzeug zu umfangreich und hochgesteckt sind, wobei man zwischen Ingenieuren aus der Anwendung und der Methodenbeurteilung unterscheiden muss. Die Zusammenarbeit mit beiden Anwendergruppen hat im Laufe dieser Arbeit die unterschiedlichen Anforderungsprofile aufgezeigt.

Der Querschnittsbereich Methodenevaluierung stellt sehr hohe Anforderungen an die Flexibilität eines Visualisierungswerkzeugs. Im Vordergrund stehen Forderungen nach einer umfangreichen Unterstützung verschiedenster Dateiformate und der Lauffähigkeit der Software auf vielen Plattformen. Zielsetzung ist der Einsatz des Visualisierungssystems in möglichst vielen Abteilungen mit zum Teil stark abweichenden Aufgabenstellungen. Ingenieure aus der Methodenentwicklung müssen später nicht zwingend mit der Software im produktiven Umfeld arbeiten, sondern sie beschränken sich größtenteils auf die interne Demonstration der Techniken für Führungskräfte.

Im Blickfeld des eigentlichen Berechnungsingenieurs liegt stärker die leichte Bedienung und die optimale Unterstützung seiner Analysearbeit. Die Akzeptanz eines speziell ausgerichteten Visualisierungssystems in Bezug auf eingeschränkte Ladeflexibilität und Plattformunterstützung ist in produktnahen Abteilungen durch die Beschränkung auf wenige Simulationscodes sehr viel

größer. Hier wird das optimale Werkzeug für die tägliche Arbeit gefordert, auch wenn es nur ein Dateiformat und eine Rechnerplattform unterstützt. Solange die Visualisierungswerkzeuge ihre Aufgabe hervorragend erfüllen, werden auch unterschiedliche Applikationen für verschiedene Simulationsergebnisse in Kauf genommen.

Die verschiedenen Anforderungsprofile bestimmen schließlich auch die Auswahl der bevorzugten Visualisierungssysteme. Allgemeinere Systeme für die Methodenentwicklung, wie beispielsweise *EnSight* von *CEI*, *Muse* von *MUSE Technologies* oder *COVISE* von *Vircinity*, glänzen durch ihre hohe Flexibilität und weniger durch intuitive Bedienbarkeit, wohingegen man in Simulationsabteilungen hauptsächlich auf spezialisierte Postprocessing-Systeme, wie *Animator* von *GNS*, *PAM-VIEW* von *ESI* und *ExaVIZ* von *Exa*, trifft. Die vollständige Befriedigung beider Anforderungsprofile durch eine Applikation ist praktisch kaum umsetzbar, da die Flexibilität immer auf Kosten der Performanz und Bedienbarkeit geht.

Ein gemeinsamer Punkt, der von allen Seiten von einem Visualisierungssystem gefordert wurde, ist die Skalierbarkeit vom Arbeitsplatz bis hin zum VR-Labor mit einer einheitlichen Benutzerführung. Dieser Punkt kristallisierte sich im Laufe der Arbeit durch die Anwendung von *VtCrash* und *PowerVIZ* heraus. Der hohe Ressourcenbedarf von *VtCrash* für die Visualisierung von großen Modellen eignet sich gut für die Erläuterung der Problematik. Im Falle eines umfangreichen Modells kann *VtCrash* nur auf nicht arbeitsplatztypischen Großsystemen zum Einsatz kommen. Die Bereitschaft der Ingenieure für die Analyse eines Datensatzes den Arbeitsplatz zu verlassen, um am Großrechner zu arbeiten, war im Allgemeinen sehr gering und sank proportional zur zurückzulegenden Wegstrecke. Lediglich für Demonstrationen vor Entscheidungsträgern und für Besprechungen wurde dieser Aufwand akzeptiert. Das Tagesgeschäft muss jedoch mit derselben Anwendung am Arbeitsplatzrechner abgewickelt werden, um den Lernaufwand für verschiedene Applikationen so gering wie möglich zu halten. Dieser Umstand verhinderte letztendlich den breiten Einsatz von *VtCrash*.

Die prototypischen Applikationen dieser Arbeit orientieren sich stärker an den Forderungen der Berechnungsingenieure. Die Beschränkung auf ein spezielles Arbeitsgebiet ermöglicht hohe Performanz und angepasste Benutzerführung. Gerade *PowerVIZ* nutzt die Vorteile der Datenbasis extrem gut aus, um einen hohen Grad an Interaktion zu erzielen. Die beiden Visualisierungssysteme bieten die geforderte Skalierbarkeit bei identischer Benutzerführung in allen Anwendungsszenarien.

Beide Applikationen lieferten in der Zusammenarbeit mit den Anwendern weitere wichtige Anhaltspunkte für die Ausrichtung eines neuartigen Postprocessing-Systems. Um das Interesse der Ingenieure und ihr persönliches Engagement an einem neuen Analysewerkzeug zu wecken, müssen die vorgestellten Vorteile ihre gewohnten Arbeitsabläufe deutlich verbessern. Nur so sind sie bereit, den Umstieg auf ein anderes Visualisierungssystem mit dem damit verbundenen Arbeitsaufwand zu wagen.

Die Ladezeit für einen Datensatz stellte sich in diesem Zusammenhang als ein sehr wichtiges Kriterium heraus. Im Allgemeinen investiert der Ingenieur für den Ladevorgang im Tagesgeschäft nicht mehr als 2 Minuten. Dies ist im Falle eines Überschreitens ein klares Ausschlusskriterium für ein Visualisierungssystem. Ebenfalls ist ein Zwischenspeichern der Daten in einem anderen Datenformat nicht erwünscht, da die zusätzlich anfallenden Datenmengen die Verwaltung erheblich erschweren. Um ein Reengineering zu vermeiden, ist eine enge Zusammenarbeit

mit den Produzenten der Simulationscodes zwecks des Zugangs zu den Dateiformaten erforderlich.

Viele Forschungsprojekte finden nicht den Weg in die breite produktive Anwendung, da sie als Schnittstelle standardisierte Dateiformate, wie beispielsweise *Inventor*, benutzen. Demonstrationen der Leistungsfähigkeit von *VtCrash* und *PowerVIZ* gegenüber den Simulationsherstellern ermöglichten den Zugang zu den Dateiformatbeschreibungen bzw. Programmierbibliotheken und stellten somit eine gesteigerte Akzeptanz bei den Anwendern in Aussicht. Für große Modelle benötigt *VtCrash* trotz Verwendung der *DSY*-Bibliothek immer noch 5 Minuten, da der *WorldToolKit*-Szenengraph keinen schnelleren Aufbau einer so komplexen Geometrie zulässt. *PowerVIZ* hingegen erzielt, mit dem *OpenGL Optimizer* als Basis und den *Exa SDK*-Bibliothek, Ladezeiten von maximal 2 Minuten und fand so eine sehr hohe Akzeptanz bei den Ingenieuren.

Immersive Umgebungen bieten durch die realitätsbezogenen Interaktionen auch Nicht-Spezialisten einen leichten Zugang zu den Berechnungsergebnissen. Die unterschiedlichen Arbeitsweisen der Ingenieure in immersiven Umgebungen wurden auch im Umgang mit *VtCrash* und *PowerVIZ* erprobt. Mit *VtCrash* konnten ebenfalls Erfahrungen mit dem *BOOM* gesammelt werden. In größeren Demonstrationsrunden zeichnete sich der *BOOM* durch einen leichten und schnellen Wechsel des Betrachters aus, so dass jeder Teilnehmer schnell die immersive Umgebung erfahren konnte. Allerdings sind die restlichen Teilnehmer von der Immersion vollständig ausgeschlossen. Ein weiteres Manko des *BOOMs* ist der Verlust eines Teils des Immersionseffektes durch die Trägheit des Ausgleichsgewichtes, dass die Gefahr der „Cyber-Sickness“ erhöht. Die *CAVE* wird von beiden Visualisierungssystemen unterstützt. Zwar können in der *CAVE* bis zu 6 Personen an einer Präsentation teilnehmen, sinnvoll arbeiten können jedoch maximal 3 Personen. Das Eintauchen in die virtuelle Umgebung gelingt hier jedoch sehr gut, beispielsweise verlängerten CFD-Berechnungsingenieure ihren Besuchstermin von 5 Minuten auf über eine halbe Stunde, da sie von den Interaktionsmöglichkeiten stark beeindruckt waren. Die Erfahrungen bestätigen die Aussage, dass immersive Umgebungen sehr gut geeignet sind, auch ungeübten Anwendern die Berechnungsergebnisse zu vermitteln. Für große Besprechungsrunden ist man allerdings auf die semi-immersiven Leinwandprojektionen angewiesen, die sich mittlerweile weit verbreitet haben.

Die Ergebnisse und Erfahrungen mit *VtCrash* und *PowerVIZ* haben den großen Bedarf an interaktiven Visualisierungstechniken im Fahrzeugentwicklungsprozess aufgezeigt. Das große Interesse an den Techniken führte schließlich zu kommerziellen Versionen der beiden prototypischen Applikationen, außerdem wurden die Ergebnisse von Seiten der beteiligten Abteilungen als Impulsgeber für die traditionellen Postprocessing-Systeme verwendet, die mittlerweile einige Funktionalitäten übernommen haben.

9.6 Weiterführende wissenschaftliche Arbeiten

Die Erfolge der vorliegenden Arbeit führten zu staatlich- und industriell geförderten Folgeprojekten.

In der Abteilung für Crashtestsimulationen werden mit einem weiteren Doktorandenstipendium die Untersuchungen im Bereich interaktiver Visualisierung für Crashtestberechnun-

gen weitergeführt. Das neu entstandene Visualisierungssystem *crashViewer* greift viele Methoden von *VtCrash* auf, besitzt durch die Verwendung von *OpenGL Optimizer* jedoch geringere Ressourcenanforderungen. Ein weiterer Bestandteil dieses Projektes ist die Integration des Preprocessing-Schrittes in das Postprocessing-Werkzeug.

Einen positiven Einfluss hat die vorliegende Arbeit auch auf das BMBF Projekt *Autobench*. Zwei weitere Doktoranden erarbeiten in diesem Projekt Methoden für das Pre- und Postprocessing von Finite-Element-Simulationen in enger Zusammenarbeit mit der *BMW Group*. Im Rahmen dieses Projektes entstand bereits die Visualisierung von Schweiß- und Klebeverbindungen für das Preprocessing als auch eine *CORBA*-Anbindung des Postprocessings an den laufenden Simulationsprozess.

In dem FORTWIHR III Transfer Projekt der Bayerischen Forschungsstiftung erforscht ein Doktorand die Möglichkeiten der Simulation und Visualisierung von massebehafteten Partikeln in dem Visualisierungssystem *PowerVIZ*. Neben der Untersuchung der Theorie für die Berechnung von massebehafteten Partikelbahnen ist auch die Verteilung von Größe, Masse und Anfangsgeschwindigkeiten auf eine große Teilchenmenge Gegenstand der Forschung.

Kapitel 10

Zusammenfassung

Die Globalisierung des Wettbewerbs zwingt die Automobilhersteller parallel zur Ausweitung der Produktpalette zu immer stärkeren Zeit- und Kostenoptimierungen. Zur Wahrung der Konstruktionsqualität der Fahrzeugmodelle setzen die Hersteller verstärkt auf numerische Simulationen, um kostspielige, reale Prototypen einzusparen. Die Größe und Komplexität der digitalen Simulationsmodelle steigt durch die Forderung nach präzisen und aussagekräftigen Berechnungsergebnissen und somit auch der Aufwand der Berechnungsingenieure zur Erkennung von Konstruktionsschwachstellen. Die Analyseergebnisse werden in Diskussionsrunden genutzt, um frühzeitig Entscheidungen über Fahrzeugkonzepte und -konstruktionsänderungen zu fällen. In den Arbeitssitzungen müssen die Ergebnisse auch den Entscheidungsträgern, meistens Nicht-Berechnungsspezialisten, in einer leicht zugänglichen Art und Weise präsentiert werden. Ein gut verständliches und einfach zu bedienendes Analysewerkzeug ist dem Postprocessing somit sehr förderlich.

Die vorliegende Arbeit zeigt neue Methoden zur Beschleunigung der Analyse von numerischen Berechnungen auf. Ziel war es, die Arbeit der Berechnungsingenieure in jedem Teilschritt des Postprocessings durch die Bereitstellung von interaktiven und intuitiven Visualisierungswerkzeugen zu erleichtern und durch neue Möglichkeiten zu erweitern. Hierfür wurden die Arbeits- und Vorgehensweisen der Ingenieure betrachtet, darauf aufbauend Visualisierungsverfahren und -algorithmen entwickelt und im produktiven Prozess erprobt. Durch den Einsatz von Techniken der virtuellen Realität entstanden prototypische Applikationen, die im Gegensatz zu traditionellen Postprocessing-Systemen mit neuen Interaktionsmechanismen und mit der Unterstützung von neuartigen Ein- und Ausgabegeräten ausgestattet sind.

Die Betrachtung der einzelnen Teilschritte des Postprocessings ergaben verschiedene Einsatzszenarien für interaktive Visualisierungssysteme. An den Simulationslauf schließt direkt die Analyse des Gesamtmodells an, die die Berechnungsingenieure am Arbeitsplatz, oder in immersiven Umgebungen von VR-Labors durchführen. Die darauf folgende Diskussion der Ergebnisse und der Verbesserungsvorschläge findet im kleineren Rahmen von maximal 3 Teilnehmern ebenfalls am Arbeitsplatz, im VR-Labor oder in kooperativen Sitzungen über das Internet statt. Die abschließenden Entscheidungstreffen werden in Projektionsräumen mit bis zu 30 Teilnehmern durchgeführt. Interaktive Visualisierung kann im Fahrzeugentwicklungsprozess nur erfolgreich eingesetzt werden, wenn die virtuelle Umgebung und die Hardware-Ausstattung an das Einsatz-

profil der einzelnen Prozessschritte angepasst wurde.

Neue Visualisierungsalgorithmen, Interaktionsmechanismen und Arbeitsweisen wurden in dieser Arbeit auf den Gebieten der Finite-Element-Simulation und Strömungssimulation entwickelt und erprobt. Umgesetzt wurden die Methoden in den beiden Visualisierungssystemen *VtCrash* und *PowerVIZ*, die alle Teilschritte des Postprocessings unterstützen. Diese Vielseitigkeit reduziert den Einarbeitungsaufwand für die Berechnungsingenieure.

Im Bereich der Finite-Element-Simulationen wurden Methoden für den Umgang mit zeitabhängigen Geometrien und Volumen betrachtet, wie sie in der Crashtest-, Schwingungs-, Kinematik- und Akustiksimulation vorkommen. Verfahren zur Selektion, Transformation und die Schnittberechnung von animierten Objekten wurden für die Darstellung von zeitabhängigen Geometrien untersucht. Die kombinierte Analyse von Schwingungs- und Akustikergebnissen erforderte die Entwicklung von Volumenproben, mit denen die parallele Visualisierung von zeitabhängigen Geometrie- und Volumenelementen möglich wurde. Im Definitionsbereich der frei beweglichen Volumenproben wurden die Berechnungsergebnisse als farbcodierte Schnittebenen, Gradienten und propagierende Isoflächen visualisiert.

Die untersuchten Verfahren wurden in der prototypischen Applikation *VtCrash* implementiert, in dessen virtueller Umgebung der Anwender intuitiv über die Eingabegeräte mit den geometrischen Repräsentationen der Berechnungsergebnisse interagieren kann. Zur Lösung der auftretenden Speicherplatzprobleme wurden Algorithmen zur Optimierung der Geometrie und zum optimierten Aufbau des Szenengraphen betrachtet.

Die Arbeiten im Bereich der Strömungssimulation konzentrierte sich auf die Visualisierung von lokal verfeinerten kartesischen Gittern mit expliziter Geometriebeschreibung. Diese Art der Gitterstruktur kommt aufgrund ihrer automatisierten Generierung verstärkt im Entwicklungsprozess zur Anwendung. Für die interaktive Visualisierung wurde eine effiziente Datenspeicherung und -verwaltung der Volumen- und Geometrieelemente vorgestellt, auf deren Basis optimierte Algorithmen für die Zelllokalisierung, Partikelbahn- und Schnittebenenberechnung untersucht wurden. Die Verwendung von Textur-Hardware beschleunigte die Darstellung von Konturlinien auf den Schnittebenen und die Visualisierung von Strömungseigenschaften mittels halbtransparenter Texturen auf Schnittebenenstapeln. Der Einsatz eines Octrees in Kombination mit den Konditionsbytes der Volumenzellen ermöglichte für die Teilchenbahnintegration die interaktive Berechnung von Aufschlagpunkten auf der expliziten Geometrie. Die Partikelbahnen wurden als Linien, Bänder oder Glyphen visualisiert.

Praktisch umgesetzt wurden die vorgestellten Algorithmen in den Interaktionsproben des Visualisierungssystems *PowerVIZ*. Der Anwender kann in der virtuellen Umgebung die Parameter der Proben direkt manipulieren und erhält somit in einer immersiven Umgebung einen realitätsnahen Eindruck.

Die Nutzung von semi-immersiven und immersiven Multiprojektionsinstallationen für die beiden Visualisierungssysteme wurde erst durch Erweiterungen zu Multi-Thread- und Multi-Pipe-Applikationen möglich. Die Einbindung von Gerätetreibern und Interaktionsmechanismen erlaubte in Kombination mit der Integration des vorgestellten dreidimensionalen Interaktionsmenüs *G3Menu* die Steuerung der Visualisierung in vollimmersiven Hardware-Umgebungen. Gerade für die Visualisierung von Strömungsberechnungen hat sich der Einsatz von immersiven Umgebungen als ausbaufähig erwiesen.

Kooperatives Arbeiten wird im globalen Wettbewerb immer wichtiger und ist somit für Visualisierungssysteme unverzichtbar. Zwei kooperative Ansätze wurden in dieser Arbeit untersucht. Die synchronisierte Darstellung von Teilmodellen in *VRML*-Viewern wurde für die Ergebnisse von Finite-Element-Simulationen betrachtet. Die speziell für Interaktionen erweiterten *VRML*-Szenengraphen wurden mittels Socket-Verbindungen über das Internet synchronisiert und ermöglichten so das Zusammenarbeiten mehrerer Anwender. Im zweiten Ansatz wurde ein Client-Server-Konzept auf der Basis der Middleware *CORBA* für die Kopplung von *PowerVIZ* Applikationen realisiert. Alle Benutzereingaben wurden zwischen den angeschlossenen Client-Applikationen ausgetauscht. Der Funktionalitätenumfang dieser Umgebung ist größer als im ersten Ansatz. Er erfordert allerdings mehr Ressourcen, da in diesem Fall kein standardisierter *VRML*-Viewer auf einer beliebigen Plattform ausreicht. Der Einsatz im produktiven Umfeld hat jedoch gezeigt, dass Techniken zur Kopplung von Internet-Browsern noch nicht ausgereift sind und man daher eher auf die vielversprechenden Ergebnisse des *CORBA*-Ansatzes setzen wird.

Beide Applikationen kamen in den beteiligten Abteilungen der *BMW Group* zum Einsatz. Aufgrund des hohen Speicherverbrauches und der langen Ladezeiten, verwendeten die Ingenieure *VtCrash* fast ausschließlich in Projektionsräumen für Präsentationen. *PowerVIZ* hingegen wird mittlerweile sowohl am Arbeitsplatz als auch im Projektionsraum den Standardanalysewerkzeugen vorgezogen. Der Vorteil von *PowerVIZ* resultiert zum Teil aus der verwendeten Szenengraph-API *OpenGL Optimizer*, die sich durch ihre Flexibilität bewährt hat. *VtCrash* hingegen wurde durch den *WorldToolKit*-Szenengraphen in vielen Bereichen eingeschränkt. Die positive Resonanz aus dem Einsatz im produktiven Umfeld führte letztendlich zu kommerziellen Versionen der beiden prototypischen Anwendungen. Ebenfalls wurden traditionelle Postprocessing-Systeme von den Herstellern um einige der vorgestellten Interaktionsmechanismen erweitert.

Die vorliegende Arbeit hat mit der Untersuchung von Algorithmen, Methoden und der Implementierung von zwei Visualisierungssystemen Einsatzprofile und -felder für die VR-gestützte Analyse von Berechnungsergebnissen im Fahrzeugentwicklungsprozess aufgezeigt. Die Erfahrungen im produktiven Entwicklungsprozess der *BMW Group* und das Interesse an kommerziellen Varianten der prototypischen Applikationen haben die Einsatzreife dieser Techniken demonstriert. Das Potenzial dieser Techniken ist bei weitem noch nicht ausgeschöpft und so wird es zukünftig weitere Entwicklungen auf den Gebieten der interaktiven Visualisierung und den Techniken der virtuellen Realität geben. Neben der Entwicklung von neuen Visualisierungsmethoden, wie beispielsweise den Kraftflussröhren, wird es weitreichende Aktivitäten in der Anwendung von Hardware-unterstützten dreidimensionalen Texturen für die direkte Volumenvisualisierungen geben, deren Darstellung durch die rapide Entwicklung der Graphikkarten bald interaktive Raten erreichen sollte. Der Arbeitsplatz des Ingenieurs wird sich durch neue Interaktionsgeräte grundlegend ändern. Das *Phantom* beispielsweise ist das erste haptische Eingabegerät einer neuen Art von Devices, die verstärkt auf den Markt kommen werden. Grundlegend wird sich auch das Design der Visualisierungssysteme ändern. Während momentan der Pre- und Postprocessing-Schritt durch verschiedene Applikationen abgedeckt wird, zeichnet sich eine Verschmelzung der Aufgaben ab. Der Anwender wird in einer virtuellen Umgebung die neuen Berechnungsmodelle interaktiv erstellen können und parallel dazu die Visualisierung vorheriger Berechnungen zur Verfügung haben. Eine Anbindung an laufende Simulationsberechnungen, das *Interactive Steering*, ist ebenfalls absehbar.

Viele Forschungsbemühungen haben die automatisierte Analyse von Berechnungsergebnissen zum Ziel. Die Erfahrungen dieser Arbeit haben gezeigt, dass sich mittelfristig das Postprocessing nicht automatisieren lässt wird, vielmehr wird auch zukünftig die Erfahrung und Kompetenz der Berechnungsingenieure zur Beurteilung von Berechnungsergebnissen gefordert sein. Unter diesen Voraussetzungen führen Verbesserungen der Mensch-Maschine-Kommunikation auf absehbare Zeit sowohl zur Erleichterung der Ingenieursarbeit als auch zu besseren Ergbebnissen.

Abstract

During the concept phase of product development in automotive design, designers have to evaluate complex engineering scenarios. The problem facing them is two fold: typically the relevant information is only partially available, and the correctness of underlying assumptions regarding the viability of certain technical solutions is not yet proven. Numerical simulation has always played a key role in this context. Performance predictions and functional evaluations are possible long before real prototype test results become available. However, with the rise in model complexity, data set size, computing performance, and accuracy, we increasingly find ourselves lacking the tools, methods, and metaphors to deal with the information generated. At *BMW*, for example, experience shows that about 30 percent of the effort involved in a typical simulation is spent for preprocessing and about 10 percent for actual computation. Approximately 60 percent amounts for the analysis and communication of the results. Clearly, a strong incentive exists to reduce the latter percentage by implementing meaningful, intuitive visualization tools that allow effective communication among engineers.

This thesis points out new methods to accelerate the analysis of numerical simulations based on structure and volume. It was the goal to facilitate and extend the work of engineers in each step of the postprocessing by supplying interactive and intuitive visualization tools. Therefore the engineers' work procedures and techniques were considered to develop visualization algorithms and test them during the productive process. The use of virtual reality techniques and new interaction mechanisms results in prototype applications which in contrary to traditional postprocessing systems support new input and output devices. Apart from the objectively detectable results regarding the representation and interaction rate, this work was also successful in the integration of the procedures into the development process.

Interactive visualization can only be used successfully in the vehicle development process, if the virtual environment and the hardware configuration are adapted to the application profile of the individual process steps. Focussing on the individual steps of postprocessing led to different application scenarios for interactive visualization systems. Directly connected to the simulation run is the analysis of the entire model done by engineers on workstations, or in immersive environments of VR labs. The following discussion of results and design improvement suggestions takes place in smaller groups of max. 3 users likewise on the workstation, in VR labs, or in cooperative sessions over the Internet. The final decision meetings with up to 30 users are held in projection spaces.

New visualization algorithms, interaction mechanisms and modes of operation were developed and tested in this work in the areas of finite element and fluid dynamic simulation. The

methods were integrated into the two visualization systems *VtCrash* and *PowerVIZ*, which support each step of the postprocessing. This versatility also reduces the training required by the engineers.

Structural mechanics

Methods for handling time-dependent geometry and volumes as they occur in crash test, vibration, kinematic and acoustic simulations were considered within the area of finite element simulations. At the beginning of the project there was no experience in handling the enormous quantities of time-dependent simulation data in scene graph APIs. Algorithms for the optimization of geometry and for an optimized structure of the scene graph were investigated to solve the occurring storage space problems.

Different mechanisms for the user interactions with animated geometries were examined for standard work places and immersive environments with VR equipment. Procedures for selection, transformation, calculations of cuts and regulation of intrusions to analyze time-dependent components were especially inspected.

The combined analysis of vibration and acoustic results requires efficient algorithms for the combined visualization of time-dependent geometry and volumes. Fast algorithms were developed for the calculation of cutting planes and propagative iso-surfaces in unstructured grids. Their response times are sufficient to allow the user interactive control of the visualization. Volume probes were developed as manipulators of the visualization techniques for easy and intuitive positioning as well as to enhance the understanding of the simulation results by the user. The scalar and vector results were visualized in the definition range of the freely movable volume probes as color coded cutting planes, gradients and propagating iso-surfaces.

The investigated algorithms were implemented in the prototype application *VtCrash*. The object-oriented system structure was laid out in such a way that both geometry and volume data sets were handled by the data management and they were integrated into the scene graph *WorldToolKit*. The objective was to investigate the usability of virtual reality techniques and VR hardware in the productive work routine. In a virtual environment the user can intuitively interact through the input devices with the geometric representations of the simulation results. The direct manipulation of the scene graph by the visualization algorithms enabled fast response times for interactive operation. So even non-specialists gain direct access to the results.

An important aspect for a successful integration of *VtCrash* into the process cycle was the direct support of different file formats. Here the cooperation with the manufacturers of the simulation codes was very helpful and led to the investigation of time-dependent surface geometry with static grid structures of crash and vibration simulations. Additionally time-dependent grids of stamping simulations were considered. A special interface and data structure was developed for time-dependent acoustic volume data sets, which were simultaneously visualized with the corresponding surface geometry.

Partial results of the work with *VtCrash* were presented at several conferences [75, 74, 106, 108, 109, 111, 110] and were finally published in a journal article [107]. Two trade fair presentations of *VtCrash* took place at the *AUTOFACT 96* in Detroit and the *High performance Computing 98* in San Jose. Many internal demonstrations at *BMW* for smaller groups, entire departments, executives and at internal fairs promoted the publicity of *VtCrash*.

The work on *VtCrash* significantly supported the spreading of VR hardware equipment at *BMW*. Meanwhile large screen projections are used in many departments for everyday operations. The demonstration of the potential of interactive work for the development process also led to the integration of VR techniques into traditional applications.

Computational fluid dynamics

The work in the area of flow simulations concentrated on the visualization of locally refined Cartesian grids with explicit description of surface geometry as they are generated by the CFD solver *PowerFLOW*. Contrary to many other CFD solvers *PowerFLOW* uses a Lattice-Boltzmann approach on locally refined grids to simulate the flow. This type of grid structure is more and more used in the car development process due to its semi-automatic generation.

The areas of resolution transitions in the volume and the explicit description of the vehicle geometry in this grid structure result in problems for traditional visualization algorithms. The presented data structure for the efficient storage of locally refined grids offers fast access to the individual volume and surface parameters by the use of a condition byte per cell in combination with an octree structure for the handling of explicit geometry. Optimized algorithms for cell localization, particle tracing and cutting plane calculation were developed which are suitable for real time processing.

The high performance of the cell location algorithm became particularly clear in the investigation of freely movable cutting planes. Scalar and vectorial parameters are determined at discrete sampling points in the volume and they are visualized by coloring polygons or rendering vectors. For example, a cutting plane with 50^2 sample points can be moved in a typical data set with frame rates higher than $10Hz$ using a mid-size workstation. The calculation of iso-contours on the cutting planes was accelerated by using the graphics hardware for linear interpolation of textures to achieve interactive frame rates. Semi-transparent textures were also used for cutting plane stacks to obtain a three-dimensional visualization of characteristic flow features.

Integration methods with constant and adaptive step length were examined for the calculation of streamlines, which were visualized as lines, ribbons or glyphs. The adaptive integration algorithms have higher calculation requirements, but were the preferred algorithms in the practical application because of larger possible step length. The best results were obtained with the adaptive Runge-Kutta scheme 3(4).

In the locally refined Cartesian grid structures the explicit handling of geometry represents a problem for the streamline calculation. In order to prevent penetrations of the geometry by streamlines, each integration step must be tested against the vehicle surface. The octree structure and the cell condition bytes reduced the necessary intersection calculations to an acceptable number for real time performance. The computing cost for the collision calculations increased by an average of 50%, but is still fast enough for interactive visualization.

The presented algorithms were implemented in the object-oriented visualization system *PowerVIZ*. Similarly to *VtCrash* the visualization methods are managed by interactive volume probes, which give the user a precise way to position and control the visualization items. The user can manipulate the parameters of the probes directly in the virtual environment and he receives a close-to-reality impression in immersive environments. The interactive probes can be adapted by several parameters to the requirements of the data set.

In immersive hardware environments the probes are directly connected to the input devices which lets the user dive perfectly into the virtual world. Stereoscopic projection even strengthened the intuitive object handling. The usability spectrum of *PowerVIZ* ranges from the workstation up to the VR laboratory.

The work with *VtCrash* within the area of interactive volume visualization for acoustic simulations clearly pointed out the limits of the scene graph API *WorldToolKit*. Therefore *PowerVIZ* was based on the scene graph API *Cosmo3D/OpenGL Optimizer* which satisfied with its high flexibility and rendering speed. For immersive environments the *Optimizer*-viewer was extended for multi-pipe and multi-thread support.

Partial results of the *PowerVIZ* project were published in conference proceedings [105, 104] and presented in numerous internal demonstrations at *BMW*. The advantages of small memory consumption and the high interactivity in contrast to traditional postprocessing systems led to a productive usage of the prototype application. *PowerVIZ* already replaced the standard tools at *BMW Group* in many analysis steps. Reports of *BMW* engineers describe the intensive use at powerwall installations and workstations.

PowerVIZ also forms the base for the transfer project *FORTWIHR III* of the Bavarian research foundation. An objective of this research project was the simulation of dirt particles, including the calculation of impact points and its distribution on the vehicle surface.

In the future a commercial version of *PowerVIZ* will be developed by *science + computing ag* in cooperation with *Exa Corporation*.

Navigation and Interaction in Immersive Environments

The integration of driver libraries for special VR devices is only the precondition to support semi-immersive and immersive environments. More important are intuitive navigation and interaction mechanisms for the users within the virtual worlds. Questions in handling of virtual objects and in controlling the immersive environments by the user were also part of this work. We used a virtual three dimensional pointer to select and transform virtual objects. The input of the devices directly served as the position of the pointer. For controlling the application in fully immersive hardware environments we integrated the three-dimensional graphical user interface *G3Menu* which can be operated by the virtual pointer.

G3Menu was developed as universal GUI for applications using a scene graph API like *WorldToolKit* or *Cosmo3D/OpenGL Optimizer*. A minimal scene graph has been implemented to build the interface to the underlying scene graph API and to the basic GUI elements, like buttons and sliders. These two abstraction layers guarantee scene graph independency. Additional scene graph APIs can be used with *G3Menu* by easily porting the small internal scene graph. The basic set of GUI elements is sufficient for a large variety of virtual environments. New GUI elements can be designed on top of the internal scene graph for even more demanding applications and they are available for all supported scene graph APIs. We implemented *G3Menu* support for the scene graph APIs *WorldToolKit*, *OpenInventor* and, *Cosmo3D/OpenGL Optimizer* and various common and new GUI elements.

An important point for the acceptance by the user is the handling of system resources. The number of polygons needed by the GUI must be kept as small as possible, since many virtual environments already use the graphics hardware up to its limit. A particular *G3Menu* component

needs between 6 and 30 polygons and an average menu clearly under 1000 polygons. This small number has been reached by the use of textures for text representation and thus guarantees low system requirements. We integrated and tested *G3 Menu* in both visualization systems of this work and received positive feedback.

Cooperative work

Cooperative working sessions become more and more important in the global competition and thus also for visualization systems. We examined two different approaches in terms of functionality and usabilitiy.

The first approach was the synchronized presentation of partial results of finite element simulations in synchronized *VRML* viewers. Interesting components of the complete model were saved in the *VRML* file format during an interactive *VtCrash* session.

Standard Plug-ins of Internet Browsers are sufficient to visualize *VRML* files independent from the computer platform. The viewer functionality was extended to offer the user the used environment of a postprocessing tool. We arranged the geometry objects in combination with sensor nodes in a specialized *VMRL* scene graph structure and thereby achieved direct manipulation of geometry in *VRML* viewers. The interactive selection procedures and the transformation of geometries were implemented with pick and motion sensors which react on 2D-mouse events. The animation of the time-dependent simulation results was created by changing the visible child node of a switch node. The user controlled these extensions through a *VRML* or *AWT* based GUI.

The cooperative sessions were established for several users via the Internet by *HTML* pages in combination with *CGI* scripts. Socket connections, *Java* applets and the *external authoring interface (EAI)* were used to synchronously manipulate the *VRML* scene graphs of the connected browsers. A master slave concept allowed only one user to propagate user inputs. However the stability problems of the *Java* implementations clearly showed how unfavorable and incomplete this technique is for productive work right now.

A more promising operating area for *VRML* files is the common analysis process. A large advantage for the daily work is the easy dispatching by email and the platform independency. Also the possibility to connect two viewers in one *HTML*-page offers the direct comparison of different simulation variants. These synchronized representations in one Browser can accelerate and assist the evaluation of model changes.

The second approach was examined for the visualization system *PowerVIZ*. Several *PowerVIZ* applications were synchronized by a client server concept using the middleware *CORBA*. Such a session was easily established by *CORBA* after the transmission of an appropriate address key.

Experiences with the *CORBA* approach showed the requirement of a clearly defined interface to the application functions. The interface of *PowerVIZ* is set up by a C++ class, which offers all functionalities to the graphic user interface and to *CORBA*. All user inputs were distributed between the connected clients and thereby the whole functionality of *PowerVIZ* was available in a cooperative session.

In contrast to the *VRML* approach the result data sets of the simulations must be locally accessible for each user, since the enormous quantities of data prevent fast access over the network. On the other hand it offered more analysis possibilities by the large scope of functionalities than

a coupling of *VRML* viewers allows. In the test runs *CORBA* showed a higher stability and was preferred by the cooperation partners.

Result integration into the development process

Besides the methodical work the options of virtual reality in the vehicle development process were examined on the basis of prototypic applications. This thesis was assigned in close cooperation with the department for IT-support and application development in the car body development section of the *BMW Group*. Thereby the developed visualization systems could be directly tested in close cooperation with the departments in productive operation. We found out that each department demands varying requirements of a visualization system due to different simulation software and data structures. Nevertheless the desire for an interactive visualization tool grew within all these different areas and the synergies resulting from cooperation with the departments affected this work in a positive way. Only the personal engagement of the engineers made it possible to get new work procedures tested in the productive workflow. Therefore convincing results are necessary to awake the interest of the users for new visualization methods. New ideas and tasks were generated very quickly once their interests aroused.

The crash test simulation results of the department for finite element simulations were the starting point of this thesis. The large experience of this department in handling numerical simulations also resulted in a knowledge base about analysis tools. These specialists know the limits of traditional postprocessing systems and thus could give good hints and function specifications in order to overcome the old boundaries for the applications of this work. As one result of this cooperation we developed the visualization system *VtCrash*.

Internal demonstrations of *VtCrash* led to a high recognition of the new functionalities within the *BMW Group*. The rising interest reflected in requests of departments with similar problem definitions, e.g. vibration and kinematic simulations. The similar data structures of these simulation types made it easy to extend *VtCrash*.

More work had to be done for the integration of the simulation of air pressure distribution inside passenger compartments which frequently follows the vibration analysis. No satisfying tool existed for the parallel visualization of the resulting time-dependent geometry and volume. To handle these data sets we developed prototypic extensions of *VtCrash* to offer the engineers new analyzing techniques and views into the virtual model. The support of the time-dependent volume data sets from acoustic simulations was the beginning for the visualization of vector and scalar volumes. Thus the interest of the aerodynamics department in cooperating in the area of air flow visualization around vehicle bodies started. *PowerVIZ* was developed to visualize simulated results of the solver *PowerFLOW*.

Working in distributed sessions is an important issue since many departments cooperate with foreign partners and suppliers. We evaluated both presented approaches for cooperative working at *BMW* and it was shown that the use of these techniques in the productive workflow is still very difficult. In the case of the *VRML* based visualization the different *Java* implementations proved to be unstable and thus too uncertain. Whereas the promising *CORBA* approach requires still further investigations.

Experiences with *VtCrash* and *PowerVIZ* were collected on all kinds of workplaces like workstations, semi-immersive and full-immersive hardware environments. The high resource requi-

rements of *VtCrash* strongly shifted its use to special discussion sessions and demonstrations, in which *VtCrash* could convince by its unique functionalities. The second large project of this thesis, *PowerVIZ*, profited strongly from the experiences made with *VtCrash*. Also the available APIs, e.g. *Cosmo3D/OpenGL Optimizer*, at that time exhibit improved flexibility and functionality. Thereby the engineers working place could be determined as the target platform for *PowerVIZ* right from the launch of the project. Meanwhile *PowerVIZ* became the preferred analysis tool at the involved departments. The advantage of *PowerVIZ* partially resulted from the flexibility of the used scene graph API *Cosmo3D/OpenGL Optimizer*, whereas *VtCrash* was limited within many areas by the *WorldToolKit* scene graph.

The positive feed back from the use in the productive field finally led to commercial versions of both prototype applications. The prototypes gave strong impulses in the simulation departments for the request of interactive postprocessing systems. Likewise traditional postprocessing systems were extended by some of the presented interaction mechanisms.

The close inter-department cooperation is just another result of this thesis besides the implemented applications. The synergies had a positive influence on this work and on the departments. Likewise the advantages of a close linkage of industrial sponsorship and university research were underlined, because a high degree of personal engagement was required of all people involved.

Conclusion

This work pointed out fields in the productive vehicle development process for techniques of virtual reality by investigating algorithms, methods and by the implementation of two visualization systems. The experiences made at the *BMW Group* and the engineers' interest in commercial versions of the prototypic applications demonstrated their readiness for the productive workflow. The potential of these techniques is by far not yet exhausted and there will be further developments on the areas of interactive visualization and virtual reality to analyze simulation results.

In the near future there will be strong activities in the use of graphics hardware besides the development of new visualization methods. For example new graphics cards with three-dimensional textures for interactive direct volume rendering will become standard and will offer new possibilities. Also the working place of engineers will fundamentally change by new input devices. Device of new types, like the *phantom* with haptic feed back, will enter to the market.

Today the preprocessing and postprocessing steps are realized in different applications. But the fusion of these functions is approaching and will change the design of the visualization systems. The users will be able to interactively build up the simulation model simultaneous to the visualization of previous calculations in the same virtual environment. In this context, the integration of visualization and simulation, known as *Interactive Steering*, is likewise foreseeable.

Many research activities are targeted at the automated analysis of calculation results. The experiences of this thesis have demonstrated that in medium-term postprocessing can not be automated and also in the future the experience and authority of calculation engineers will be required for the evaluation of calculation results. Thus improvements in the area of man-machine communication will simplify the engineers' work and will lead to better results.

Literaturverzeichnis

[1] D. Allison, B. Wills, L. F. Hodges, and J. Winemann. Gorillas in the Bits. In *Proceedings of IEEE VRAIS '97*, pages 69–76. IEEE Computer Society, März 1997. 3.6

[2] K. W. Arthur. *Effects of Field of View on Performance with Head-Mounted Displays*. Ph.d. thesis, University of North Carolina, 2000. 3.4.1

[3] N. Asamura, N. Tomori, and H. Shinoda. A Tactile Feeling Display Based on Selective Stimulation to Skin Receptors. In *Proceedings of VRAIS '98*, pages 36–50. IEEE Computer Society, März 1998. 3.4.3

[4] J. W. Barrus, R. C. Waters, and D. B. Anderson. Locales: Supporting Large Multiuser Virtual Environments. *IEEE Computer Graphics and Applications*, pages 50–57, November 1996. 3.6

[5] J. J. Batter and Jr. Brooks, F. P. Grope-1: A Computer Display to the Sense of Feel. In *In Information Processing, Proceedings of IFIP Congress 71*, pages 759–763, 1971. 3.2

[6] J. Beckmann. 3D Interaktionselemente als Benutzerschnittstelle in virtuellen Umgebungen. Diplomarbeit, IMMD9 (Computergraphik), Universität Erlangen, März 1999. 1.2

[7] R. Blach, J. Landauer, A. Rösch, and A. Simon. A Flexible Prototyping Tool for 3D Real-Time User Interaction. In *User-Interaction, Proc. of Virtual Environments VE98*. Springer Wien, 1998. 3.5.2

[8] R. Blach, J. Landauer, A. Rösch, and A. Simon. A Highly Flexible Virtual Reality System. *Future Generation Computer Systems Special Issue on Virtual Environments*, 1998. Elsevier Amsterdam. 3.5.2

[9] F. P. Jr. Brooks, M. Ouh-Young, J. J. Batter, and P. J. Kilpatrick. Project GROPE - Haptic Displays for Scientific Visualization. In *Proceedings of SIGGRAPH'90*, volume 24. ACM, 1990. 3.2

[10] Jr. Brooks, F.P. What's Real About Virtual Reality. *IEEE Computer Graphics and Applications*, 19(6):16–27, November/Dezember 1999. 3.6

[11] S. Bryson. Approaches to the Successful Design and Implementation of VR Applications. In *Proc. SIGGRAPH'94*, 1994. 3.1, 3.5

[12] S. Bryson and S. Feiner. Virtual Environments in Scientific Visualization. In *Virtual Reality for Visualization, Course Notes of Tutorial 5 at Visualization 95*, 1995. 3.6

[13] S. Bryson, S. Johan, and L. Schlecht. An Extensible Interactive Visualization Framework for the Virtual Windtunnel. In *Proceedings of IEEE VRAIS '97*, pages 106–113. IEEE Computer Society, März 1997. 3.6

[14] S. Bryson and C. Levit. The Virtual Windtunnel. *IEEE Computer Graphics and Applications*, 12(4):25 – 34, 1992. 5.1.1

[15] H.-J. Bullinger, R. Breining, and W. Bauer. Virtual Prototyping - State of the Art in Product Design. In *Proceedings of the 26th International Conference on Computers & Industrial Engineering*, pages 103–107, Melbourne, Dezember 1999. 3.6

[16] G. K. Buning. Numerical Algorithms in CFD Post-Processing. In *Computer Graphics and Flow Visualization in Computational Fluid Dynamics*, Lecture Series 1989-07. Von Karman Institute for Fluid Dynamics, Brussels, Belgium, 1989. 2.2.2.1

[17] W. Buxton, G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. Large Displays in Automotive Design. *IEEE Computer Graphics and Applications*, 20(4):68–75, Juli/August 2000. 3.4.1

[18] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *ACM Symposium on Volume Visualization'94*, pages 91–98, 1994. 2.2.2.3

[19] E. Chen. Six Degree-of-Freedom Haptic System for Desktop Virtual Prototyping Applications. In *Proceedings of the First International Workshop on Virtual Reality and Prototyping*, pages 97–106, Juni 1999. 3.4.3

[20] S. Chen, G. D. Doolen, and K. Molvig. Lattice Boltzmann Method for Fluid Flows. In *Annual Reviews Fluid Mechanics*, volume 30, pages 329–364, 1998. 2.1.2

[21] A. Cohen and E. Chen. Six Degree-of-Freedom Haptic System for Desktop Virtual Prototyping Applications. In *Proceedings of the ASME Winter Annual Meeting, Dynamics Systems and Control, DSC-Vol67,*, pages 401–402, November 1999. 3.4.3

[22] C. Cruz-Neira and et al. The CAVE: Audio Visual Expirience Automatic Virtual Environment. *Communications of ACM*, 35(6):65, 1992. 3.4.1, 6.5

[23] C. Cruz-Neira, J. Leigh, C. Barnes, S. Cohen, S. Das, R. Englemann, R. Hudson, M. Papka, L. Siegel, C. Vasilakis, D. J. Sandin, and T. A. DeFanti. Scientists in Wonderland: A Report on Visualizaion Applications in the CAVE Virtual Reality Environment. In *Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality*, Oktober 1993. 3.2, 3.4.1, 6.5

[24] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *Proceedings of SIGGRAPH '93*, pages 135–142. ACM, August 1993. 3.2, 3.4.1, 6.5

[25] F. Dai. On the Role of Simulation and Visualization in Virtual Prototyping. In *International Workshop on Virtual Reality and Scientific Visualization*, Hangzhou, China, April 1995. 3.6

[26] F. Dai and M. Göbel. Virtual Prototyping - An Approach Using VR-Techniques. In *Proceedings of the 14th ASME International Computers in Engineering Conference*, 1994. 3.6

[27] J. Dauner, J. Landauer, E. Stimpfig, and D. Reuter. 3D Product Presentation Online: The Virtual Design Exhibition. In *Proceedings VRML 1998*, 1998. Monterey CA. 3.6

[28] W. C. de Leeuw and J. J. van Wijk. A Probe for Local Flow Field Visualization. In G. M. Nielson and Bergeron D., editors, *Visualization '93*, pages 39–45, Los Alamitos, CA, 1993. IEEE Computer Society. 8.3.1

[29] J. Deisinger, R. Breining, and A. Rößler. ERGONAUT: A Tool for Ergonomic Analyses in Virtual Environments. In Mulder J. D. and van Liere R., editors, *Proceedings of the 6th Eurographics Workshop on Virtual Environments*. Springer, Juni 2000. 3.6

[30] M. Dinsmore, N. Langrana, G. Burdea, and J. Ladeji. Virtual Reality Training Simulation for Palpation of Subsurface Tumors. In *Proceedings of IEEE VRAIS '97*, pages 54–60. IEEE Computer Society, März 1997. 3.6

[31] M. J. Diorinos. Concept of a Virtual Mall and Development of a Prototype Virtual Furniture Mall. Diplomarbeit, Informatik-Forschungsgruppe B "Betriebliche Anwendungen", Friedrich-Alexander-Universität Erlangen, Mai 1996. 3.6

[32] S. R. Ellis, B. D. Adelstein, S. Baumeler, G. J. Jense, and R. H. Jacoby. Sensor Spatial Distortion, Visual Latency, and Update Rate Effects on 3D Tracking in Virutal Environments. In *Proceedings of IEEE VRAIS '99*, pages 218–221. IEEE Computer Society, März 1999. 3.4.2

[33] L. M. Encarnacao and R. J. Barton. Walk-up VR: Virtual Reality beyond Projection Screens. *IEEE Computer Graphics and Applications*, 20(6):19–23, November/Dezember 2000. 3.1

[34] K. Engel, O. Sommer, C. Ernst, and T. Ertl. Remote 3D Visualization using Image-Streaming Techniques. In *Advances in Intelligent Computing and Multimedia Systems (ISIMADE '99)*, pages 91–96, 1999. 4.2.4

[35] S. Feiner and D. Thalmann. Virtual Reality. *IEEE Computer Graphics and Applications*, 20(6):24–25, November/Dezember 2000. 3.6

[36] S. S. Fisher, M. McGreevy, J Humphries, and W. Robinett. Virtual Environment Display System. In *Proc. 1986 ACM Workshop on Interactive 3D Graphics, Chapel Hill*, pages 77–87, Oktober 1986. 3.2

[37] B. Fröhlich, S. Barrass, B. Zehner, J. Plate, and M. Göbel. Exploring Geo-Scientific Data in Virtual Environments. In D. Ebert, M. Gross, and B. Hamann, editors, *Proceedings of IEEE Visualization 1999*, pages 169–173, San Fransisco (CA), Oktober 1999. 3.5.2, 3.6

[38] B. Fröhlich, W. Krueger, and et al. The Responsiv Workbench: A Virtual Working Environment for Architects, Designers, Physicians, and Scientists. In *The Edge, Visual Proceedings*, pages 200–201. ACM Siggraph, 1994. 3.4.1

[39] B. Fröhlich, J. Plate, J. Wind, G. Wesche, and M. Göbel. Cubic-Mouse-Based Interaction in Virtual Environments. *IEEE Computer Graphics and Applications*, 20(4):12–15, Juli/August 2000. 3.4.3

[40] A. Fuhrmann, H. Löffelmann, D. Schmalstieg, and M. Gervautz. Collaborative Visualization in Augmented Reality. *IEEE Computer Graphics and Applications*, 18(4):54–59, Juli/August 1998. 3.6

[41] T.A. Furness. The Super Cockpit and Human Factors Challenges. In M. Ung, editor, *Proceedings of Human Factors Society 30th Annual Meeting*, pages 48–52, 1986. 3.2

[42] R. Gallagher, R. Haber, G. Ferguson, D. Parker, W. Stillmann, and J. Winget. Applying 3D Visualization Techniques to Finite Element Analysis. In *IEEE Visualization '91*, pages 330–335, 1991. 3.6

[43] T. A. Galyean. Sculpting: an Interactive Volumetric Modeling Technique. In *ACM SIGGRAPH'91*, pages 267 – 274, Juli 1991. 5.1.2.2

[44] A. Glassner. *An Introduction to Ray Tracing*. Academic Press London, 1989. 8.2

[45] B Grant, A Helser, and R. M. Taylor. Adding Force Display to a Stereoscopic Head-Tracked Projection Display. In *Proceedings of VRAIS '98*, pages 81–88. IEEE Computer Society, März 1998. 3.4.3

[46] R. Grosso, M. Schulz, and T. Ertl. Fast and Accurate Visualization of Steady and Unsteady Flows. Technical Report 3, IMMD9, University of Erlangen, 1996. 2.2.2.4

[47] R. Grosso, M. Schulz, J. Kraheberger, and T. Ertl. Flow Visualization for Multiblock Multigrid Simulations. In *Virtual Environments and Scientific Visualisation'96*. Springer-Verlag Wien New York, 1996. 2.2.2.4

[48] C. G. Guan, L. Serra, R. Kockro, N. Hern, Nowinski W. L., and C. Chan. Volume-based Tumor Neurosurgery Planning in the Virtual Workbench. In *Proceedings of VRAIS '98*, pages 167–173. IEEE Computer Society, März 1998. 3.6

[49] H. Haase. Symbiosis of Virtual Reality and Scientific Vizualization System. In J. Rossignac and F. Sillion, editors, *Eurographics'96*, volume 15(3), pages 443–451. Eurographics Association, 1996. 3.6

[50] Ch. Hand. A Survey of 3D Interaction Techniques. *Computer Graphics*, 16 (5):269 – 281, 1997. 5.1

[51] R. Harmon, W. Patterson, W. Ribarsky, and J. Bolter. The Virtual Annotation System. In *Proceedings of VRAIS'96*, pages 239–245. IEEE, 1996. 5.1.1

[52] J. Hartman and J. Wernecke. *The VRML 2.0 Handbook*. Addison-Wesley, New York, 1996. 3.6

[53] P. Hastreiter. *Registrierung und Visualisierung medizinischer Bilddaten unterschiedlicher Modalitäten*. Doktorarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1999. 3.6

[54] M. Hereld, I. R. Judson, and R. L. Stevens. Introduction to Building Projection-based Tiled Display Systems. *IEEE Computer Graphics and Applications*, 20(4):22–28, Juli/August 2000. 3.4.1

[55] K. P. Herndon and T. Meyer. 3D Widgets for Exploratory Scientific Visualization. In *Proceedings of UIST'94*, pages 69–70. ACM SIGGRAPH, November 1994. 5.1.1

[56] D. Hix, J. E. Swan, J. L. Gabbard, M. McGee, J. Durbin, and T King. User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment. In *Proceedings of IEEE VRAIS '99*, pages 96–103. IEEE Computer Society, März 1999. 3.6

[57] L. F. Hodges and et. al. Virtual Environments for Treating the Fear of Heights. *IEEE Computer*, 28(7):27–34, Juli 1995. 3.6

[58] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization. In *SIGGRAPH '93*, pages 19–26. ACM, 1993. 2.2.1

[59] C. T. Howie and E. H. Blake. The Mesh Propagation Algorithm for Isosurface Construction. *EUROGRAPHICS*, 13 (3):65 – 74, 1994. 7.1.3

[60] T. J. R. Hughes. *The Finite Element Method*. Prentice-Hall, 1987. 2.1.1

[61] Y. Ikei, K. Wakamatsu, and S. Fukuda. Vibratory Tactile Display of Image-Based Textures. *IEEE Computer Graphics and Applications*, 17(6):53–61, November/Dezember 1997. 3.4.3

[62] V. Interrante and C. C. Grosch. Strategies for Effectively Visualizing 3D Flow with Volume LIC. In R. Yagel and H. Hagen, editors, *Proc. Visualization '97*. IEEE, 1997. 2.2.2.4

[63] V. Interrante and C. Grosch. Visualizing 3D Flow. *IEEE Computer Graphics and Applications*, 18(4):49–53, Juli/August 1998. 2.2.2.4

[64] H. Iwata. Walking About Virtual Environments on an Infinite Floor. In *Proceedings of IEEE VRAIS '99*, pages 286–293. IEEE Computer Society, März 1999. 3.4.3

[65] S. Jayaram, U. Jayaram, Y. Wand, H. Tirumali, K. Lyons, and P. Hart. VADE: A Virtual Assembly Design Environment. *IEEE Computer Graphics and Applications*, 19(6):44–50, November/Dezember 1999. 3.6

[66] A. Johnson, M. Roussos, J. Leigh, C. Vasilakis, C. Barnes, and T. Moher. The NICE Project: Learning Together in a Virtual World. In *Proceedings of VRAIS '98*, pages 176–183. IEEE Computer Society, März 1998. 3.6

[67] S. Julier, R. King, B. Colbert, J. Durbin, and L. Rosenblum. The Software Architecture of a Real-Time Battlefield Visualization Virtual Environment. In *Proceedings of IEEE VRAIS '99*, pages 29–36. IEEE Computer Society, März 1999. 3.6

[68] G. D. Kerlick and E. Kirby. Towards Interactive Steering, Visualization and Animation of Unsteady Finite Element Simulations. In *IEEE Visualization '93*, pages 374–377, San Jose (CA), Oktober 1993. 3.6

[69] G. D. Kessler, L. F. Hodges, and M. Ahamad. RAVEL, a Support System for the Development of Distributed, Multi-user VE Applications. In *Proceedings of VRAIS '98*, pages 260–267. IEEE Computer Society, März 1998. 3.6

[70] L. Kobbelt, S. Campagna, J. Vorsatz, and H. P. Seidel. Interactive Multiresolution Modeling on Arbitrary Meshes. In *SIGGRAPH'98*, pages 105–114. ACM, New York, Juli 1998. 5.1.2.2

[71] S. Kuschfeldt, T. Ertl, and M. Holzner. Hardwareunterstuetzte Visualisierung von Struktureigenschaften und physikalischen Belastungsgroessen in Crash-Simulationen. In *Proc. VisEng'97, Rechenzentrum Universitaet Stuttgart, Stuttgart, Germany*, 1997. 2.2.1

[72] S. Kuschfeldt, M. Holzner, and T. Ertl. Video Integration of PAM-VIEW Visualization Results. In *Proc. of PAM 95, Fifth European Workshop on Advanced Finite Element Simulation*, 1995. 3.6

[73] S. Kuschfeldt, M. Holzner, O. Sommer, and T. Ertl. Efficient Visualization of Crash-Worthiness Simulations. *IEEE Computer Graphics and Applications*, 18(4):60–65, 1998. 2.2.1

[74] S. Kuschfeldt, M. Schulz, T. Ertl, T. Reuding, and M. Holzner. The Use of a Virtual Environment for FE Analysis of Vehicle Crash Worthiness. In *Proc. IEEE 1997 Virtual Reality Annual International Symposium*, page 209. IEEE Computer Society Press, 1997. 1.2, 9.1, 10

[75] S. Kuschfeldt, M. Schulz, M. Holzner, T. Reuding, and T. Ertl. Advanced Visualization of Crashworthiness Simulations using Virtual Reality Techniques. In *Proceedings of*

the Conference on High Performance Computing in Automotive Design, Engineering and Manufacturing, pages 455–462. Silicon Graphics Inc. / Cray Research, 1996. 1.2, 9.1, 10

[76] D. A. Lane. Scientific Visualization of Large-Scale Unsteady Fluid Flows. In G. Nielson, H. Hagen, and H. Mueller, editors, *Scientific Visualization*, pages 125–145. IEEE Computer Society, 1997. 2.2.2.4

[77] R. Lea, K. Matsud, and K. Miyashita. *Java for 3D and VRML Worlds*. New Riders Publishing, Indianapolis, 1996. 3.6

[78] B. Leibe, T. Starner, W. Ribarsky, Z. Wartell, D. Krum, J. Weeks, B. Singletary, and L. Hodges. Toward Spananeous Interaction with the Perceptive Workbench. *IEEE Computer Graphics and Applications*, 20(6):54–65, November/Dezember 2000. 3.4.2

[79] K. Li and et. al. Building and Using A Scalable Display Wall System. *IEEE Computer Graphics and Applications*, 20(4):29–37, Juli/August 2000. 3.4.1

[80] C.-R. Lin, R. B. Loftin, and H. R. Jr. Nelson. Interaction with Geoscience Data in an Immersive Environment. In *Proceedings of IEEE Virtual Reality Conference 2000*, pages 55–62, New Brunswick, März 2000. IEEE. 3.6

[81] V. Lindorfer. Kopplung eines WWW-Browsers mit VRML-Plugin an eine verteilte Visualisierungsumgebung über eine CORBA-basierte Kommunikationsschale. Master's thesis, Rechenzentrum der Universität Stuttgart, Institut für Computeranwendungen II, Mai 1998. 3.6

[82] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Computer Graphics, SIGGRAPH '87*, volume 21(4), pages 163–169. ACM, Juli 1987. 2.2.2.3

[83] J. Marti. Viewing IGES Files Through VRML. In *Proc. IEEE Visualization'97, Phoenix, AZ*, pages 471–474, 1997. 6.6

[84] T. Massie. A Tangible Goal for 3D Modeling. *IEEE Computer Graphics and Applications*, 18(3):62–65, Mai-Juni 1998. 3.4.3

[85] U. Neumann and A. Majoros. Cognitive, Performance, and Systems Issues for Augmented Reality Applications in Manufacturing and Maintenance. In *Proceedings of IEEE VRAIS'98*, pages 4–11, März 1998. 3.1

[86] U. Neumann and J. Park. Extendible Object-Centric Tracking for Augmented Reality. In *Proceedings of VRAIS '98*, pages 148–155. IEEE Computer Society, März 1998. 3.1

[87] B. Nichols, D. Buttlar, and J. P. Farrel. *Pthreads Programming*. O'Reilly & Associates, Inc., 1996. 7.2.3

[88] D. Nishioka and M. Nagasawa. Reducing Polygonal Data by Structural Grouping Algorithm. In *Lecture Notes in Computer Science 1024, ICSC 95*, 1995. 2.2.1

[89] T. Ohshima, K. Satoh, H. Yamamoto, and H. Tamura. AR2Hockey: A Case Study of Collaborative Augmented Reality. In *Proceedings of VRAIS '98*, pages 268–275. IEEE Computer Society, März 1998. 3.6

[90] F. Post and T. van Walsum. Fluid Flow Visualization. In H. Hagen, H. Mueller, and G. Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer Berlin, 1997. 2.2.2.4

[91] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 1988. 7.1.5

[92] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones. Magnetic Position and Orientation Tracking System. In *IEEE Transaction on Aerospace and Electronic Systems*, volume AES-15,5, pages 709–717, September 1979. 3.2

[93] M. M. Rafferty, D. G. Aliaga, and A. A. Lastra. 3D Image Warping in Architectural Walkthroughs. In *Proceedings of VRAIS '98*, pages 228–233. IEEE Computer Society, März 1998. 3.6

[94] F. Reck. Strömungsvisualisierung in lokal verfeinerten kartesischen Gittern. Studienarbeit, IMMD9 (Computergraphik), Universität Erlangen, November 1998. 1.2

[95] J. D. Reid. Visualizing Cross Section Forces. In *Computer & Graphics*, volume 19(3), pages 475–480, 1995. 2.2.1

[96] K. J. Renze and J. H. Oliver. Generalized Surface and Volume Decimation for Unstructured Tesselated Domains. In *Proc. IEEE Conference VRAIS'96, Santa Clara, CA.*, pages 111–121, 1996. 2.2.1

[97] T. Reuding. Tightening the link between Design and Engineering Analysis - Real Possibilities for Virtual Reality? In *Second IFIP Workshop on Virtual Prototyping*. University of Texas, Mai 1996. 3.6

[98] O. Riedel and J. Deisinger. Displays für Virtual Reality. In *Proceedings of the 10. Electronic Displays 95*, pages 81–92. Network GmbH, 1995. 3.4.1

[99] Sadarjoen, A. AND van Walsum, T. AND Hin, A. AND Post, F. Particle Tracing Algorithms for 3-D Curvilinear Grids. In *Proc. 5th Eurographics Workshop on Visualization in Scientific Computing*, 1994. 2.2.2.4

[100] J. K. Salisbury and M. A. Srinivasan. Phantom-Based Haptic Interaction with Virtual Objects. *IEEE Computer Graphics and Applications*, 17(5):6–10, September-Oktober 1997. 3.4.3

[101] H. Scharm and R. Breining. How Automotive Industry uses Immersive Projection Technology. In H.-J. Bullinger and O. Riedel, editors, *Forschung und Praxis T52: Proceedings of the 3rd International Immersive Projection Technology Workshop*, pages 133–144, Stuttgart, Mai 1999. Springer Verlag. 3.6

[102] D. R. Schikore, R. A. Fischer, R. Frank, R. Gaunt, J. Hobson, and B. Whitlock. High-Resolution Multiprojector Display Walls. *IEEE Computer Graphics and Applications*, 20(4):38–44, Juli/August 2000. 3.4.1

[103] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of Triangle Meshes. In *Computer Graphics, SIGGRAPH '92*, volume 26(4), pages 65–70. ACM, Juli 1992. 2.2.1

[104] M. Schulz. Interactive Visualization-Systems for the Improved Analysis of Numerical Simulations in the Car Development Process. In *Graphiktag 2000*. Gesellschaft für Informatik, Fachausschuß 4.2 der Gesellschaft für Informatik Graphische Datenverarbeitung, September 2000. 1.2, 9.2, 10

[105] M. Schulz, F. Reck, W. Bartelheimer, and T. Ertl. Interactive Visualization of Fluid Dynamics Simulations in Locally Refined Cartesian Grids. In D. Ebert, M. Gross, and B. Hamann, editors, *Proc. of IEEE Visualization '99*, pages 413–416. IEEE Computer Society Press, Oct. 1999. 1.2, 9.2, 10

[106] M. Schulz and T. Reuding. Einsatz von Virtual Reality Techniken zur Darstellung von Ergebnissen der Karosseriestrukturberechnung. In *Proc. VisEng '97 Visualisierung im Ingenieurbereich*. Univ. Stuttgart, 1997. 1.2, 9.1, 10

[107] M. Schulz, T. Reuding, and T. Ertl. Analyzing Engineering Simulations in a Virtual Environment. *IEEE Computer Graphics and Applications*, 18(6):46–52, November/Dezember 1998. 1.2, 9.1, 10

[108] M. Schulz, T. Reuding, and T. Ertl. Crashing in Cyberspace - Evaluating Structural Behaviour of Car Bodies in a Virtual Environment. In R. S. Sipple, editor, *Proc. of IEEE Virtual Reality Annual International Symposium*, pages 160–166, 1998. 1.2, 9.1, 10

[109] M. Schulz, T. Reuding, and T. Ertl. From High-End VR to PC-based VRML Viewing: Supporting the Car Body Development Process by Adapted Virtual Environments. In M. H. Hamza, editor, *Proc. of IASTED Computer Graphics and Imaging*, pages 231–234. ACTA Press, 1998. 1.2, 9.1, 10

[110] M. Schulz, T. Reuding, E. Zimmermann, and T. Ertl. VRML-basierte Visualisierung von Finiten Elemente Berechnungen im Intranet. In O. Deussen and V. Hinz, editors, *Simulation und Visualisierung '99*, pages 33–42. SCS, P. Lorenz, März 1999. 1.2, 9.1, 10

[111] M. Schulz, M. Weiler, T. Reuding, and T. Ertl. Interactively Analysing Joint Simulations of Car Body Vibrations and Interior Acoustics. In *Proc. of Sixth SIAM Conference on Geometric Design*, page 42. SIAM, November 1999. 1.2, 9.1, 10

[112] C. Seiler and A. Schäfer. MUSyC: Scalable Multi-User Virtual Environments based on VRML. In *Proc. of Virtual Environments Conference and 4th Eurographics Workshop IEEE YUFORIC '98, Stuttgart, Germany*, pages 26.1–26.9, Juni 1998. 3.6

[113] E. Sharlin, P. Figueroa, M. Green, and B. Watson. A Wireless, Inexpensive Optical Tracker for the CAVE. In *Proceedings of IEEE Virtual Reality Conference 2000*, pages 271–278, New Brunswick, März 2000. 3.4.2

[114] D. Stalling and H.-C. Hege. Fast and Resolution Independent Line Integral Convolution. In *Computer Graphics, Proceedings of SIGGRAPH 95*, pages 249–256. ACM, August 1995. 2.2.2.4

[115] S. Stansfield, D. Shawver, and A. Sobel. MediSim: A Prototype VR System for Training Medical First Responders. In *Proceedings of VRAIS '98*, pages 198–205. IEEE Computer Society, März 1998. 3.6

[116] A Steed, M. Slater, A. Sadagic, A. Bullock, and J. Tromp. Leadership and Collaboration in Shared Virtual Environments. In *Proceedings of IEEE VRAIS '99*, pages 112–115. IEEE Computer Society, März 1999. 3.6

[117] M. P. Stevens, R. C. Zeleznik, and J.F. Hughes. An Architecture for an Extensible 3D Interface Toolkit. In *Proceedings of UIST'94*, pages 59–67. ACM SIGGRAPH, November 1994. 5.1.1

[118] I. E. Sutherland. Sketchpad – A Man-Machine Graphical Communication System. In *Proceedings of the Spring Joint Computer Conference, Detroit*, Mai 1963. 3.2

[119] I. E. Sutherland. The Ultimate Display. In *Proceedings of IFIPS Congress 1965, New York*, volume 2, pages 506–508, Mai 1965. 3.2

[120] I. E. Sutherland. A Head-Mounted Three-Dimensional Display. In *AFIPS Conference Proceedings*, volume 33/I, pages 757–764, 1968. 3.2

[121] A. Tanaka, K. Hirota, and K. Toyohisa. Virtual Cutting with Force Feedback. In *Proceedings of VRAIS '98*, pages 71–80. IEEE Computer Society, März 1998. 3.4.3

[122] D. L. Tate, L. Sibert, and T. King. Virtual Environments for Shipboard Firefighting Training. In *Proceedings of IEEE VRAIS '97*, pages 61–68. IEEE Computer Society, März 1997. 3.6

[123] C. Teitzel, R. Grosso, and T. Ertl. Efficient and Reliable Integration Methods for Particle Tracing in Unsteady Flows on Discrete Meshes. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 31–41, Wien, April 1997. Springer. Proceedings of the Eurographics Workshop in Boulogne-sur-Mer, France. 2.2.2.4

[124] C.-A. Thole. SIM-VR, Interactivity, Virtual Reality and Industrial Simulation. Final public report, GMD, September 1999. 3.5.2

[125] H. Tramberend. Avocado: A Distributed Virtual Reality Framework. In *Proceedings of IEEE VRAIS '99*, pages 14–21. IEEE Computer Society, März 1999. 3.5.2

[126] S. K. Ueng, C. Sikorski, and K. L. Ma. Efficient Construction of Streamlines, Streamribbons and Streamtubes on Unstructured Grids. *IEEE Transactions on Visualization and Graphics*, 2(2):100–109, Juni 1996. 2.2.2.4

[127] S. P. Uselton. exVis: Developing A Wind Tunnel Data Visualization Tool. In R. Yagel and H. Hagen, editors, *Proc. Visualization '97*. IEEE, 1997. 2.2.2.4

[128] A. van Dam, A. S. Forsberg, D. H. Laidlaw, J. J. Jr. LaViola, and R.M. Simpson. Immersive VR for Scientific Visualization: A Progrss Report. *IEEE Computer Graphics and Applications*, 20(6):26–52, November/Dezember 2000. 3.1, 3.6

[129] J. J. van Wijk. Spot Noise: Texture Synthesis for Data Visualization. In *Computer Graphics*, volume 25(4), pages 309–319, 1991. 2.2.2.4

[130] M. Wan, H. Qu, and A. Kaufmann. Virtual Flythrough over a Voxel-Based Terrain. In *Proceedings of IEEE VRAIS '99*, pages 53–60. IEEE Computer Society, März 1999. 3.6

[131] K. Watsen and M. Zyda. Bamboo - A Portable System for Dynamically Extensible, Realtime, Networked, Virtual Environments. In *Proceedings of IEEE VRAIS '98*, pages 252–259. IEEE Computer Society, März 1998. 3.6

[132] B. Wei, C. Silva, E. Koutsofios, S. Krishnan, and S. North. Visualization Research with Large Displays. *IEEE Computer Graphics and Applications*, 20(4):50–54, Juli/August 2000. 3.4.1

[133] M. Weiler. Visualisierung von Schwingungs- und Akustiksimulationen in einer Virtuellen-Realitäts-Umgebung. Studienarbeit, IMMD9 (Computergraphik), Universität Erlangen, November 1998. 1.2

[134] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *SIGGRAPH 98*. ACM, 1998. 2.2.2.3

[135] R. Westermann, C. Johnson, and T. Ertl. A Level-Set Method for Flow Visualization. In *Proceedings of IEEE Visualization '00*, page 9, 2000. 2.2.2.4

[136] A. Wierse. Performacne of the COVISE Visualization System Under Different Conditions. In *Proceedings of the Symposium on Electronic Imaging: Science & Technology*, Februar 1995. 3.5.2

[137] A. Wierse, U. Lang, and R. Rühle. A System Architecture for Data-oriented Visualization. In J. P. Lee and G. G. Grinstein, editors, *Lecture Notes in ComputerScience*, volume 871, pages 148–159. Springer, Berlin, 1994. 3.5.2

[138] W. Ye and J.M. Vance. Visualization of Structural Impact Problems in a Virtual Environment. In *Proc. SCS Simulation MultiConference, Atlanta, GA.*, pages 325–330, 1997. 3.6

[139] T.P. Yeh and J.M. Vance. Combining Sensitivity Methods, Finite Element Analysis, and Free-Form Deformation to Facilitate Structural Shape Design in a Virtual Environment. In *Proc. 23rd ASME Design Automation Conference, Sacramento, CA.*, 1997. 3.6

[140] E. Zimmermann. Virtual Reality Modeling Language (VRML) als entwicklungsbegleitendes Medium zur Kommunikation von Konstruktionsinformationen. Diplomarbeit, IMMD9 (Computergraphik), Universität Erlangen, September 1998. 1.2

Wissenschaftlicher Lebenslauf

Martin Schulz

geboren am 7. Dezember 1971 in Uelzen
verheiratet, eine Tochter, ein Sohn

Schulausbildung:

1978 - 1982	Grundschule in Oldenstadt
1982 - 1984	Orientierungsstufe an der Hermann-Löns-Schule in Uelzen
1984 - 1991	Lessing Gymnasium in Uelzen

Studium:

11/1991- 12/1996	Informatikstudium an der Friedrich-Alexander-Universität Erlangen-Nürnberg mit dem Schwerpunkt wissenschaftliche Visualisierung im Fachgebiet Graphische Datenverarbeitung
01/1997 - 06/1999	Promotion in Informatik am Lehrstuhl für Graphische Datenverarbeitung der Friedrich-Alexander-Universität Erlangen-Nürnberg
01/1997 - 12/1999	Promotionsstipendium der BMW Group München
07/1999 - 07/2000	Promotion in Informatik in der Abteilung Visualisierung und Interaktive Systeme des Instituts für Informatik der Universität Stuttgart
seit 08/2000	Projektleiter des Visualisierungssystems *PowerVIZ* bei science+computing Tübingen